JACOBS
UNIVERSITY

Eco State Networks - Music Generation for Human Accompaniment

by

Tomas Pllaha

Bachelor Thesis Proposal in Computer Science

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

Date of Submission: December 7, 2013

Jacobs University — 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 document introduces a project that aims to demonstrate the suitability of Echo State Networks (ESNs) for a music generation task. The task is simple: After “listening“ to melodies with at least two tracks (one leading track and one accompanying track), the system should be able to follow a melody in real time, by generating an accompanying track that fits with the leading melody.

The use of such a system could be in helping composers create music. While a composer may have created a main melody with their favorite instrument, the song might still be incomplete because of the lack of other instruments. This project aims to produce a software that is able to play along with human musicians, discuss its performance, and analyze the music generation task. Such a system can also be used during improvising sessions, to make them more lively and enjoyable. Moreover, tackling music generation tasks can be useful in furthering research about creative AI.

Recurrent Neural Networks (RNNs) and other Machine Learning techniques have been previously used for music generation tasks. The novelty of this approach lies simply in using ESNs for this particular kind of task. ESNs 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.

This guided research will not focus on actually capturing the music from a microphone and playing simultaneously, but rather on the computational process (Computing a suitable melody, while listening to a leading track). For this purpose, midi files will be used as training data and as output.

Contents

1 Introduction

Echo State Networks form a type of Recurrent Neural Network. As shown in [1], 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, which need to be learned (See Figure 1). The reservoir weights are random and fixed.

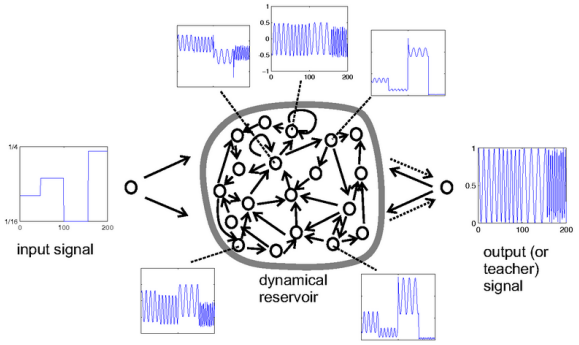

Figure 1: The basic schema of an ESN, illustrated with a tuneable frequency generator task. Solid arrows indicate fixed, random connections; dotted arrows trainable connections source: http://www.scholarpedia.org/article/Echo_state_network

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 [2]. The resevoir random weights can be easily scaled to ensure the convergence of the system [3]. 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).

For the task of music generation several methods have been explored. Some of the methods first used to produce software that is able to generate music relied on algorithmic composition [4]. While these methods sometimes produced satisfactory results, they suffered from a lack of orignality. Markov Chains have also been exploited for composition of music [5]. While they produce melodies that are more random/original than those achieved by algorithmic composition, they 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 unconvational computing approach to tackle the music generation problem was using Memristors [6], 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 determistic or non-deterministic in nature. RNNs have also been used several times in the past. CONCERT [7] is an example of a network that achieved good performance. The structure of this particular RNN was very complicated and based on human perception.

Therefore, the advantages of ESNs towards 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 recent past affect the output at the present step more than those that occurred earlier. Intuitively, this is very similar to the musical structure.

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

However, the music generation task remains a difficult task to tackle and previous results show that while software is able to generate correct music, it is difficult to build a system that produces melodies that are both original and musically pleasing. Nonetheless, satisfying results have been achieved in accompaniment systems (i.e: 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. This project aims to find out whether ESNs can be trained to accompany human musicians in a way that is both original and musically satisfying. Section 2 contains detailed information about the research problem and motivation of research. Section 3 describes planned investigations that are to be pursued during the implementation phase of this guided research. Section 4 describes the criteria used to evaluate the behavior of the system that will be created. Section 5 lists the planned time-line of the project and section 6 contains a summary of the main aspects of this document.

2 Statement and Motivation of Research

Recurrent Neural Networks form a biologically inspired model. Each unit is analogous to neurons and the connections between neurons 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, which partially explains the attention they have received in the field. This section starts with a formal description of Echo State Networks. Subsections 2 and 3 give more detailed information about the research. Subsection 4 describes preliminary experiments that have been carried out and subsection 5 concludes this section with a list of the research questions that will be addressed in this project.

2.1 Echo State Networks

As mentioned in the introduction, ESNs consist of an input vector **u** of size K, a reservoir (random vector) **x** of size N, and an output vector **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 [1]

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

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, 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.[11] Tikhonov Regularization is the recommended standard, because it yields stable solutions.[2]

2.2 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 will be crucial, because of the difficulty of the task we are dealing with. We will extract information about notes from midi files and we will represent notes with two main features: pitch and duration.

As mentioned in [7], pitch does not seem to be a good predictor for duration and vice-vrsa. Therefore we can treat pitch and duration separately and use two different networks, one to predict the next pitch, and one to predict the next duration. The nth output of the first network can correspond to the nth output of the second one. In a real life application, this can be easily accomplished by running two seperate synchronized threads, one for pitches and one for durations.

2.2.1 Representation of Pitch

The songs that will be used to train the network may be in different keys. If we just supply raw data, the expected performance inadequate, because the network will not be able to find similarities between similar songs that are played in different keys. Even if the same song is used twice, but played in different keys, the network will try to find an approximation of both, even though both sequences are essentially the same. To avoid this undesired behavior, all major songs will be transposed to C major, and all minor ones to C minor. Moreover, to allow for a succesful representation of most songs, four octaves will be used (C1,C1#, D1, .. C5, where Xn stands for the note X on the nth octave). Initially only single notes will be represented. Time permitting, the network will be extended to also support chords. This system can ideally be further extended to play several instruments at a time.

One of the ways pf representing pitch could be through a vector of size $49 = 4 \times (7+5)+1$, 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). For this reason, pitch will be represented by one single value, an integer from 0 to 49, 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. If time permits it, the representation used by CONCERT [7] (described in section 5.1 of [7]), will also be attempted and results will be compared.

2.2.2 Representation of Duration

Similarly to our representation of pitch, duration will be represented as an integer from 1 to 6. A value of n will be used to represent a length of $\frac{1}{2^{n-1}}$ i.e: 1 represents a full length, 2 represents a half-length and so on (6 represents a $\frac{1}{32}$).

2.3 Statement of Research

In this subsection I will briefly outline the expectations and challenges of this project, argue the properties of Echo State Networks that make them suitable for this particular task and try to justify the research questions, which follow in section 2.5.

Through this guided research, the suitability of ESNs for the music accompaniment task will be tested. Additionally, an attempt will be made to find the network parameters that yield the best performance, by manual experimentation. The current state-of-the-art systems that tackle the same task use either HMMs or classical RNNs. Ideally, the presentation of ESNs will present solutions that are faster than other RNNs and more “creative“ than HMMs. 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.

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 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 have to “fit“ for the melody to be musically pleasing. Moreover, this property should be especially useful for the instrumental accompaniment task, because in 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]).

This Short Term Memory [3] of ESNs is expected to be of great use for the music generation task. However, further challenges are presented by the instrumental accompaniment

task that are not faced by the music composition one. If we were trying to train an Echo State Network to create music from scratch, we would have the option of post-processing the results to eliminate detectable inconsistencies. This is not an option for the task at hand, because we want the network to be able to play along with a leading track. Therefore any restrictions need to be imposed on every note as it is generated. For some restrictions this is easy. For example, the possible durations have already been restricted to negative powers of 2 between 0 and 5. One other restriction that can be easily imposed as the music is being generated is the measure restriction. For example, if a song in $\frac{4}{4}$ is being played, we want to make sure that any duration, that makes the length of the any measure exceed $\frac{4}{4}$, is never generated. Other restrictions may be more difficult to impose in real time. Nonetheless, an instumental accompaniment system can be more performant than a music composition one, because it is driven with human input. Through this guided research I will also try to estimate the performance an ESN that can have no prior knowledge of the future.

As stated in the previous section, the investigation will start with only one line of melody per track. Provided this goal is attained successfully, I will also try to extend the network to support more dimensions for the input and output and thus support chords. This will of course introduce new challenges which will need to be dealt with. (i.e: making sure that only correct chords are generated). Further research can be carried out on building a system that accompanies a leading track with more than 1 instrument.

2.4 Preliminary Experiments

As a preliminary experiment for this guided research, I re-implemented the “Little Red Riding Hood“ text generation example from [3]. Instead of “Little Red Riding Hood“, a 2031 characters long text from “Snow White“ was used as training data. The experiment was carried out almost exactly as described in [3] with minor changes in the network parameters. Below I present a brief summary of the experiment, the network parameters, and a few results. In the end of this section I describe how this experiment is relevant to the music generation task and also point out the differences between the tasks.

Task: Given a text we want to train an ESN to produce a text with a similar distribution of letters as the original text. Moreover we also want sequences of letters to be similarly distributed along the text (The network should produce “correct“ sequences of letters).

For this experiment, a reservoir of 400 units was used. The reservoir weight matrix was scaled to have a spectral radius of 0.75. Bias to reservoir weights were set randomly to values between -1 and 1. Input to Reservoir weights were set randomly to either 1 or -1. There were no feedback weights or input to output weights. For simplicity the set of characters was reduced to lowercase letters of the English alphabet, the full stop (.), the comma (,), and the white space (_). This left a total of 29 characters to represent.

The input was a vector with 30 dimensions. The first dimension was used to feed a bias input into the network (equal to 1 at every step). Each of the other 29 elements was used to represent a character. At each step, these 29 elements were all equal to 0, except for the one corresponding to the character being represented, which was equal to 1. (i.e: The letter ‘a‘ was represented as [1 0 0 0 ...0], the letter ‘b‘ as [0 1 0 0 0 ...0] etc).

The output was a vector of size 29, with each dimension corresponding to a character.

A 2030 teacher sequence was used to train the network. At every step n the nth character of the teacher sequence was fed into the network as input and the $n+1$th was forced as output, so the network was trained to predict the next character. Tikhonov Reuglarization with $\alpha^2 = 10^{-3}$ was used to compute the output weights.

After training, the network was run, following these steps:

1. Generate a random valid input and feed it to the network. Compute the output vector $(y_1(1), y_2(1)..., y_{29}(1))$
2. Turn the output vector 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. ($F = 3$ produced the best behavior. This "favor factor" is used to favor higher output values over proportionality [3]. 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.)
3. Select the next character from $\mathbf{y}(1)$ by a weighted random draw where the weights are given by the updated $y_1(1)...y_{29}(1)$.
4. Feed the selected character back to the network as input $\mathbf{u}(2)$ and compute the next output $\mathbf{y}(2)$
5. Repeat the previous steps (starting from step 2) until an output of desired length is generated.

Figure 2 shows 50-step plots of 4 randomly selected reservoir units, whereas Figure 3 shows 50-step plots of proabilities of selecting each of the first 10 characters of the english alphabet (a through j).

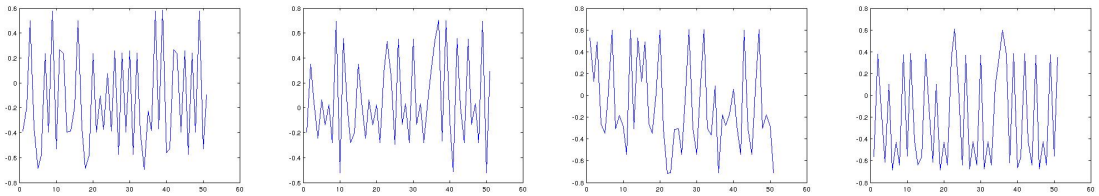

Fig 2: 50 step plot of 4 randomly selected reservoir neurons

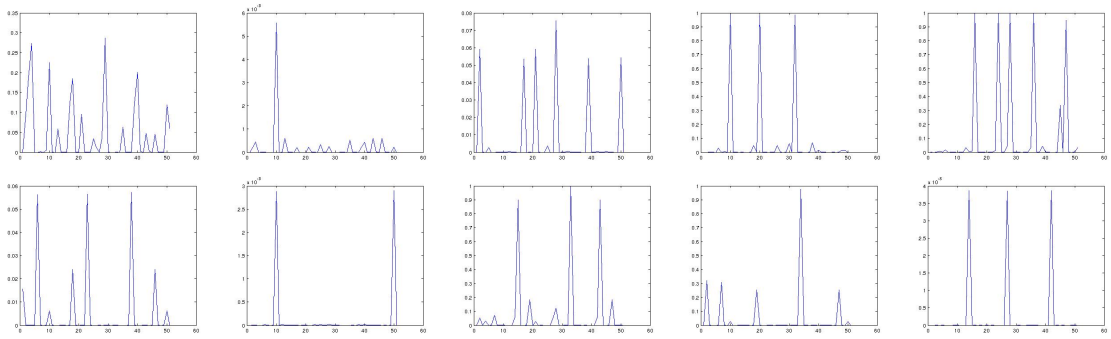

Fig 3: 50 step plot of of proabilities of selecting each of the first 10 characters of the

English alphabet (a through j).

As seen in Figure 3, letters have spikes in the right time steps.

Here are two 100 step outputs of the network:

1. `and_snow_white_the_cottage_kicked_snow_white_white_the_dwarfs_was_and_snow_white_the_wite_and_snow_w`
2. `for_saw_white_cottage_and_she_was_white_white_the_white_for_white_dwarfs he_was_dwarfs_was_and_cottag`

The text generation task is similar to the music generation task in that they are both time series prediction tasks. Analogously to the text generation task, we intend to teach the network to predict the next note (We will teach one network to predict the next pitch, and another one to predict the next duration). We will also expect the network to generate a vector of probabilities for all possible values, rather then letting it produce a single output, in which case the system would be deterministic (given the leading track).

However, the tasks differ in several aspects. Firstly, music has more dimensions than text. A musical note has a pitch, a duration, a timbre, a volume etc. We will reduce the dimensions to just pitch and duration and deal with them seperately. This should not introduce any problems, because duration and pitch seem to be indepentent of each other and therefore they can be treated independently. Another difference between the tasks stands in the distribution of symbols. In the case of English text, some transitions are impossible (i.e: they never occur) and often the next character can be determined with almost certainty. This is not the case with music, where almost all transitions are possible, in principle. The introduction of a cultural bias (explained better in the following sections) may limit this high variability of notes. In fact, the choice of training data will play a crucial role in the behavior of the network (unlike the text generation example, where almost any text that is long enough would yield similar performance).

2.5 Research Questions

Throughout this guided research the following research questions will be addressed:

1. How can ESNs be used as an instrumental accompaniment system?
2. How well can such a system perform without having any knowledge of the future?
3. What restrictions can be imposed on the generated melody while it is being generated?
4. What restrictions cannot be imposed while the melody is being generated?
5. How does cultural bias affect the performance of the network? (A “cultural bias“ can be imposed on the network by training it only with data that belongs to a particular class, i.e: the same artist, the same genre etc.)
6. What are the network parameters that yield the best results?
7. (Optional) What are the additional challenges that are introduced when extending the network to also support chords?
8. (Optional) How can these challenges be overcome?

9. (Optional) What way of representing input yields the best performance?
10. (Optional) Are there significant differences in performance between using one single network to produce both pitches and durations and using 2 networks (1 for each)?
11. (Optional) How can this system be extended to play several instruments together?

3 Planned Investigation

Most of the questions of this guided research will be approached by building an ESN that works as a human accompaniment system. This section will begin with a formal description of the network. In the next subsection I will list the sources that will be used to build and train the network. Finally the last subsection will list some of the most crucial challenges that are expected and methods that can be applied to overcome these challenges.

3.1 Structure of the network

As previously mentioned in this document, in musical pieces, pitches and durations of notes are not good predictors of each other. Therefore the system we will build can have two different ESNs, each of which deals with one of these features of musical notes. This will reduce the complexity of each network and hopefully make the learning tasks easier. I will start with the description of the network that predicts the pitches and then move on to the one that predicts the durations.

The network will support four octaves (C1 .. C5). The investigation will begin by only representing single notes. Time permitting, the dimensions of input and output will be adjusted to allow for representation of chords.

- The input will consist of two units ($\mathbf{u} = (u_1, u_2)$), which will take integer values between 0 and 49 (inclusive), and possibly one bias neuron that will always be set to one. 0 stands for rest, while the numbers 1..49 each correspond to one of the notes between C1 and C5 (inclusive). u_1 will be used for the leading track, whereas u_2 will be used for the supporting track.
- The output will consist of 50 units, with the first one corresponding to rest, and each of the others corresponding to one of the notes between C1 and C5.
- The bias to reservoir weights will be set randomly between -1 and 1, while input to reservoir weights will be set at random to either 1 or -1.
- The state update function for the reservoir states will be $tanh$.
- The output activation function will be identity. (Same as: there will be no output activation function).
- There will be no feedback connections.

Training the network

During training, the pitches from the training data will be read into the network, one per

update cycle. Each pitch is an integer between 0 and 49. The pitches of the main track will be fed into the network through u_1. Those that belong to the accompanying melody will be fed through u_2. Recall that all major keys will be transposed to C major and all minor ones will be transposed to C minor.

The teacher signal will be an encoding of the *next* pitch on the accompanying track:

$$y_{teach} = (0, 0, ..., 0, 1, 0, ..., 0, 0) \tag{3}$$

where the "1" occurs in the unit corresponding to the *next* note on the accompanying track. The state update function for this network:

$$\mathbf{x}(n+1) = tanh(W\mathbf{x}(n) + W_{in}\mathbf{u}(n+1)) \tag{4}$$

The output weights will then be computed using Ridge Regression.

Running the network

The procedure of running the network after training will be very similar run as described in section 2.4, when talking about the preliminary experiment. The output activation equation:

$$\mathbf{y}(n) = W_{out}\mathbf{x}(n) \tag{5}$$

The output will be converted to a probability vector, by nulling all the negative entries, raising the remaining entries to the power of F (where F is the "favor factor"), and normalizing them so that they sum up to 1. The next supporting pitch is then chosen by a random weighted draw, where the weights are given by the constructed probability vector, and the selected pitch is fed back to u_2 for the next update cycle. Note that u_1 will be supplied by the leading track as input to the network. This will be repeated until an output of desired length is generated.

The "durations" network

The network that generates durations will be very similar to the one generating pitches. The only differences will be that the input units will take integer values between 1 and 6. A value of n represents a duration of length $\frac{1}{2^{n-1}}$.

When generating durations we will also have to impose restrictions to make sure that the produced music is correct. MIDI files contain information about the time signature of the piece being played. We will use this information to ensure that the network never generates a measure that violates the restriction imposed by the time signature. For example, suppose the time signature of the piece being played is $\frac{3}{4}$ and the sum of durations of notes in the current measure is $\frac{1}{2}$. In this case we should not allow the network to generate a duration greater than $\frac{1}{4}$. This can be easily achieved by keeping track of the total length of the measure that is being generated in a special variable. Once this length reaches the time signature, the variable can be reset to 0, indicating that we have moved on to a new measure. When constructing the probability vector out of the network's output, we don't only set the negative values to 0, but also those that violate the restriction imposed by the time measure.

3.2 The Cultural Bias effect

The effect of the cultural bias on the performance on the system will be tested by training the network with 3 different types of training data and comparing the results. The network will be trained once only with Beatles' songs, once only with songs that belong to the same genre (here we can test the performance with different genres), and once with songs from different genres. The expectation is that the network will be more precise when a cultural bias is in effect, but it can respond to more styles when trained without a cultural bias. If this is indeed the case, more experiments will follow, to find the best balance. These experiments will be carried out manually.

In order to identify the best network parameters, experiments based on iterative manual tuning and evaluation will be performed.

3.3 Selecting and Manipulating the Training data

Midi files will be downloaded from `http://mididb.com/`, a large midi database with songs from various disciplines. The database contains a large collection of Beatles' songs, which is why the band was selected to test the cultural bias effect.

The tools midicsv and csvmidi (`http://www.fourmilab.ch/webtools/midicsv/`) will be used to convert files from midi format to Comma-Separated Value (CSV) format and vice-versa. These tools were selected because they extract more information than the Matlab Midi Toolbox and they have very good documentation. Moreover, it is quite easy to read and write to CSV files from Matlab, using the functions csvread and csvwrite.

The Matlab toolbox from `http://organic.elis.ugent.be/node/129` will be used to program the ESNs and to carry out the experiments.

3.4 Challenges

Besides the challenges that have already been mentioned, one very important challenge will be selecting suitable training data. All of the songs that will be used as training data need to have at least 2 common instruments, one of which will be the main instrument and the other, the accompanying one. This challenge is relatively easy to solve and it can be solved programmatically. MIDI files contain information about the instrument playing each track and that information is preserved by the chosen (midicsv) conversion tool.

A more interesting challenge lies in detecting the main track and the accompanying track. This information cannot be found in midi files. One practical approach to overcome this challenge would be to chose an instrument that frequently plays a leading track and one that frequently plays an accompanying track (i.e Piano and Bass, Guitar and Bass etc). However, reliable detection is not always guaranteed, and the only way to be completely sure will remain listening to the songs. This is not unfeasable, but it is significantly more time consuming.

Last, but not least, evaluating the performance of the system is not trivial, because of the subjective nature of music. It is not self evident whether a given output is good or bad, as even the most random combinations of sounds may be considered musically pleasing by some audiences. There is no absolute measure of the quality of music. However, some

relative measures that should give some meaningful estimates about the performance of the system are discussed in the following section.

4 Evaluation Criteria

This section describes the criteria that will be used to evaluate the performance of the system:

- The log of the product of probabilities of selecting the "right input" at the "right time". This is a relative evaluation criterion and it is computed as follows:
  1. Divide the training data in two groups, one to be used for training and the other for cross-validation.
  2. Train the network using the training group.
  3. Let a_i be the correct expected output of the network at time i (i.e: The pitch/duration of the accompanying track of the sequence used for cross-validation at time i)
  4. Let $P_i(a_j)$ be the probability that the network assigns to the pitch/duration a_j at time i.
  5. Let n be the length of the cross-validation sequence.
  6. Then the value of $log(\prod_{i=1}^{n} P_i(a_i))$ is a relative indicator of the performance of the system. The higher this value is, the better the system. This criterion gives us an approximation of how close the networks choices were to the actual accompanying track. However it cannot be treated as an absolute estimate, because accompanying tracks that suit a given leaing track are not unique.
- Conducting a survey. While the answers are expected to vary a lot among different participants (because the evaluation will depend on personal taste and there is no absolute measure of quality of music), the following should normalize the results and give another relative measure of the performance of the system:(method taken from [9]).
  1. Ask participants to rate the accompaniments produced by the system and some original accompaniments created by human musicians.
  2. Let them listen to all the pieces before starting to rate and ask them to rate the pieces on a scale from 1 to 10, relative to one-another.
  3. Perform statistical tests on the results. Since different raters are asked to perform ratings under different conditions (the network generated results and the ones created by humans), a two-way ANOVA can be used to isolate the effect of condition on rating.

5 Timeline

01.02.2014 - 25.02.2014	Integration phase (System to translate input from CSV to network format and vice versa)
01.02.2014 - 30.02.2014	Gathering and analysing of all training data
25.03.2014	Final implementation of the ESNs
25.03.2014 - 04.04.2014	Experimentation phase
05.04.2014 - 30.04.2014	Extension of the system to support at least 4 lines of melody and chords
1.05.2014	Evaluation of the system's performance
11.05.2014	Thesis submission

6 Conclusions

This section concludes this document and summarizes the main aspects of this proposal.

The main focus of this guided research project is the use of Echo State Networks as an Instrumental Accompaniment system. There is a strong argument that Echo State Networks are very suitable for this particular task. After the network is trained with several pieces of music, it can be expected to learn to follow a leading melody in real time. In other words, the network should learn how to play a song with a human. This guided research project will not focus on actually capturing the sounds from a microphone, but rather on the computation of melodies. In fact, at the start of the investigation, the system will be oversimplified and able to only play one note at a time. Should time allow it, the system will be extended to allow multiple lines of melody and thus support chords. If this guided research project is successful, further research can be carried to build a system that can follow a leading melody with more than 1 instrument.

References

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

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

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

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

[5] 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.

[6] 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.

[7] 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.

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