Musical Instrument Mapping Design with Echo State Networks

Chris Kiefer
EAVI, Department of Computing,
Goldsmiths, University of London,
New Cross, London, UK.
c.kiefer@gold.ac.uk

ABSTRACT

Echo State Networks (ESNs), a form of recurrent neural network developed in the field of Reservoir Computing, show significant potential for use as a tool in the design of mappings for digital musical instruments. They have, however, seldom been used in this area, so this paper explores their possible applications. This project contributes a new open source library, which was developed to allow ESNs to run in the Pure Data dataflow environment. Several use cases were explored, focusing on addressing current issues in mapping research. ESNs were found to work successfully in scenarios of pattern classification, multiparametric control, explorative mapping and the design of nonlinearities and uncontrol. *Un-trained* behaviours are proposed, as augmentations to the conventional reservoir system that allow the player to introduce potentially interesting non-linearities and uncontrol into the reservoir. Interactive evolution style controls are proposed as strategies to help design these behaviours, which are otherwise dependent on arbitrary values and coarse global controls. A study on sound classification showed that ESNs could reliably differentiate between two drum sounds, and also generalise to other similar input. Following evaluation of the use cases, heuristics are proposed to aid the use of ESNs in computer music scenarios.

Keywords

mapping, echo state networks, machine learning, software tools

1. INTRODUCTION

1.1 Motivations

The goal of this study is to explore the potential of ESNs as mapping tools, and in turn address wider issues in mapping research, with the aim of adding to the toolset which musicians and digital luthiers can use to create expressive mappings. Of particular interest, are ESNs' possibilities for further research in the design of non-linear mappings, multiparametric control, exploratory design and gestural control.

ESNs offer some interesting possibilities to computer musicians, their key property being that they are operate using time series; training data is constructed from sets of corresponding input and output sequences, and a trained ESN can be run alongside any other signal processing unit, both at control rate and audio rate. ESNs exhibit short-term memory, and can be trained to operate on temporal aspects of their input, as opposed to conventional neural networks which operate instantaneously on data. Furthermore, and as shall be explained later in this paper, ESNs are versatile and can be adapted to offer extra features which are potentially useful for mapping design. They also offer several challenges in use, concerning the design of training data, and the choice of network parameters and sub-algorithms. Section 2 serves as an introduction to the workings of these networks.

Jordà[18] proposes that nonlinearity is essential in a musical instrument, and can create the path to virtuosity for the player, allowing expressive control and expert gestures[1]. Furthermore, unpredictability can be desirable in creative systems[9]. Non-linearity and unpredictability are sources of uncontrol; they cause the behaviour of an instrument to move away from (and probably return to) the direct influence of the player. Uncontrol can be embodied in an instrument in a number of ways; in the sound synthesis engine, in the physical design and materials, and also in mappings. By implementing nonlinearity at the mapping stage, it can be modularised and separated from other elements of the system; eliminating this dependency can be an advantage to the designer. ESNs' signal processing abilities can range from linear to highly nonlinear behaviours, both instantaneously and temporally, making these systems interesting candidates for research into uncontrol. The ESN approach bears similarity to projects which have employed physically modelled dynamical systems as nonlinear mapping engines, such as the mass-spring systems by Momemi and Henry [20] and Johnson et. al. [17]. As universal approximators of dynamical systems [15], ESNs may have the power to model this class of system and more.

Hunt and Kirk [12] also explored nonlinearity, through the paradigm of multiparametric mappings. They proposed that complex networks of multiparametric control could emulate the expressivity of real-world mappings such as in acoustic instruments. Following from this, [19] used ESNs as multiparametric mapping tools for a malleable interface. The conclusions of this study showed that ESNs have further potential in this area that warrants exploration.

1.2 Related Work

In the array of mapping tools available to the digital musical instrument designer, machine learning based systems show a significant presence and have a long history in NIME, both with conventional tools and in more esoteric designs. This can be seen in systems ranging from earlier projects such as Fels' and Hinton's Glove-Talk [7] to more recent designs such as Fiebrink's Wekinator [8], Caramiaux's Gesture Variation Follower [3] and Gillian et. al's SEC [10]. Machine learning mapping tools offer the advantages of allowing the musician to train by example, and to create complex black-box mapping systems that would be awkward or impossible to design manually. These systems also allow composition through manipulation of parameters and processes specific to the machine learning algorithms.

Reservoir computing[23] systems have yet to be fully explored in the context of mapping. Reservoir computing (RC) describes classes of recurrent neural network which follow an architecture typically consisting of a high-dimensional dynamical system that is perturbed by its inputs, and monitored by a simple readout layer which is trained by a fast and relatively straight-forward algorithm such as least squares regression. The two most common classes of RC network are liquid state machines and echo state networks[13], and this paper focuses on the latter.

ESNs have a history of use in computer music. Holzmann and Hauser[11] ran a series of successful experiments using ESNs in audio signal processing, for modelling nonlinear processes and for audio prediction. Jaeger and Eck [16] used ESNs for cyclic melody generation, and Tidemann and Demiris[22] showed that ESNs could be used for generative drumming. Echo State Networks have yet to be trialled for gesture recognition, however Jaeger demonstrates their potential for dynamical pattern detection in [14].

The body of research on ESNs points to them having interesting possibilities as mapping tools that could work in a wide range of scenarios, but they have seldom been explored in this context. This paper is a study of ESNs and their potential uses in mapping and musical instrument design. It begins with a short introduction to the topology and parameterisation of these networks. A new Pure Data (PD) external is then presented, an implementation of ESNs specialised for use in PD's dataflow environment. Several use cases studies are explored, demonstrating and evaluating possible functions for this system. These case studies are then evaluated to show a general picture of ESNs as mapping tools, and finally heuristics are proposed to guide and aid the use of the toolset.

2. ECHO STATE NETWORKS

This section gives an overview of ESNs; for more in-depth treatments beyond the scope of this paper, see [23, 15].

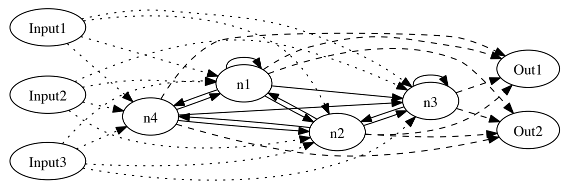


Figure 1: An Example ESN

Figure 1 shows a simplified example of ESN topology. The interconnected nodes n_i are the *reservoir*. A set of inputs is connected to the reservoir nodes, and these nodes are connected to one or more output nodes in a readout layer. The network is updated as follows:

$$\mathbf{x}[k+1] = f(\mathbf{W}_{res}\mathbf{x}[k] + \mathbf{W}_{in}\mathbf{u}[k] + \mathbf{W}_{fb}\mathbf{y}[k]) \quad (1)$$

$\mathbf{x}[k]$ is the network state at the current time step, $\mathbf{u}[k]$ is the current input matrix, $\mathbf{W}_{in}$ is the matrix of input weights, $\mathbf{W}_{res}$ denotes the reservoir weights and $\mathbf{x}[k]$ is a vector of activation levels. $\mathbf{W}_{fb}$ describes the feedback weights between the output and the reservoir, and $\mathbf{y}[k]$ is the value of the readout layer. f is a smoothing function, commonly either a linear mapping, $tanh$ or the sigmoid function, $f(x) = 1/(1 + exp(-x))$.

ESNs are trained with a set of equal length sequences, one for each input and output node, describing the input and desired output of the system. The key to the success of ESNs is the method of training; only the output weights are modified during this process. All other weights, in the input, reservoir and feedback matrices, are initialised with (typically) random constants. The range and scale of these initial values can be adjusted to change the global behaviour of the reservoir, in order to give the training phase a better chance of success. The output layer is adjusted to exploit the dynamics of the reservoir and achieve the desired behaviour. Only one layer of weights is trained, so training is a relatively straightforward linear problem that can be solved quickly using linear regression.

To function effectively, ESNs should possess the Echo State Property (ESP), meaning that the network has a slowly fading memory of its inputs. The presence of the ESP is dependent on the spectral radius of the network, a global scaling factor of the reservoir weights. This variable controls the richness of the dynamics and the non-linear modelling power of the network, at a trade off with its memory capacity [2].

Table 1 summarises the key parameters that describe ESNs.

Parameter	**Description**
Layer sizes	The number of nodes in the input, reservoir and readout layers.
Weight ranges	The minimum and maximum values for the random distribution of weight values, for the input, reservoir, readout and feedback weights.
Connectivity	The percentage of weights that are connected on each layer.
Spectral radius	The scaling factor for the reservoir weights. This should generally be less than one, but for musical purposes we may use a higher value.
Activation functions	Functions used in the calculation of reservoir node activations and outputs.
Washout	An initial number of frames that are ignored during training, to allow the dynamics of the reservoir to settle.
Training algorithm	The process used to solve the weights on the readout layer.
Simulation algorithm	The algorithm used to update the reservoir. The PD external offers the standard ESN algorithm, and one using leaky integration to slow down reservoir dynamics.

Table 1: Key Parameters for Echo State Networks

3. ECHO STATE NETWORKS IN PD

Pure Data [21] has been shown as a useful environment for neural network mapping [4]; it also provides libraries of mapping tools (SMLib, mapping) and a graphical environment for collecting and processing training data. A PD external was created, building on *Fecho*, an open source C++ ESN library. The external allows for full training and simulation within the PD environment, and is pictured in figure 2. The external offers facilities including three types of

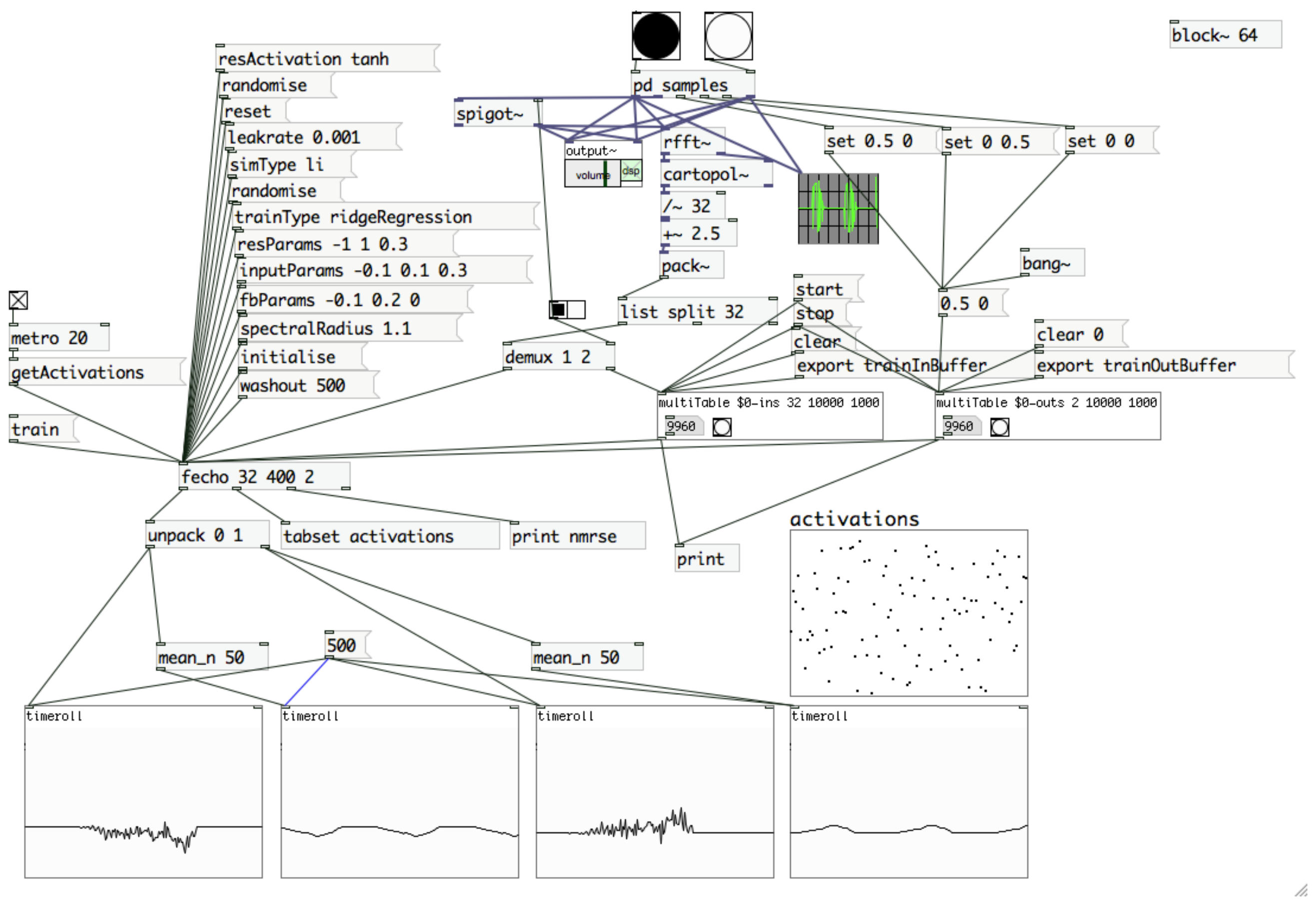


Figure 2: A PD External

training algorithm (least squares, pseudo inverse and ridge regression), two classes of simulation (standard and leaky integration), three types of activation function (linear, hyperbolic tangent and sigmoid) for the reservoir and readout layers, and configuration of weight ranges for the reservoir, readout and feedback connections. It was designed to integrate fully into the Pure Data environment. The external runs at control rate, and can be trained using data stored in PD tables and arrays. During exploration of the use cases in this paper, further non-standard features were added, and these shall be described in the relevant sections of the paper. Two additional tools were built to facilitate training and monitoring. A new abstraction, *multiTable*, collects multichannel time series data for constructing training sets. A new object, *csvlog*, collects network output in comma separated variable files, for analysis in other software applications. The system is freely available and open source under an MIT license [1].

4. EXPLORATIONS

This section presents a set of small case studies which explore the use of ESNs in a range of mapping scenarios.

4.1 Nonlinear Mapping and Uncontrol

There are a number of ways to introduce nonlinear behaviour into a trained network: scaling of the spectral radius of the network, scaling and biasing the input signal, scaling the feedback weights, and injecting noise into the network. The modulation of these parameters in realtime is not a conventional practice with ESNs, and has been added as an augmentation to the Fecho library.

Figure 3 demonstrates an example of this use case. A 100 node reservoir, with spectral radius 0.9, was trained to approximate the nonlinear function

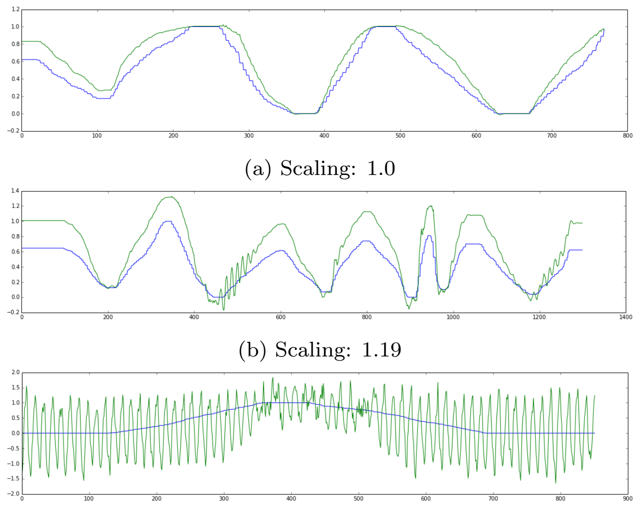
$$f(x) = sin(x * PI * 0.5), 0 < x < 1 \qquad (2)$$

The readout trained successfully with NRMSE 0.0128963. Figure 3a shows an example of input and the corresponding output from this network. The reservoir was scaled by a factor of 1.19, which introduced nonlinear behaviour into the reservoir as demonstrated in figure 3b; as the input rises, an oscillation occurs in the output. Figure 3c shows the scale factor at 2.06, where the system is driven into a highly unpredictable state, and can be seen to be self-oscillating.

This type of mapping has the possibility to create interesting musical interaction, furthermore the reservoir scaling can be explicitly mapped to modulate the level of nonlinearity in a system. For example, the rate of change in the input can be monitored with a high pass filter, and mapped to reservoir scaling so that an increase in energy by the player drives the system into nonlinear zones.

This behaviour can be classified as *un-trained* behaviour, as it is introduced independently of the training stage. The challenge here is the specification of the un-trained behaviour, which will largely depend on the course global parameters that guide the initial randomisation of the reservoir weights, and the data the reservoir was trained with. While this behaviour is difficult to specify, there are strategies which can aid the user in the development of useful mappings, falling under the banner of explorative mapping design.

[1]Available for download from `https://github.com/chriskiefer/Fecho`



(a) Scaling: 1.0

(b) Scaling: 1.19

(c) Scaling: 2.06

Figure 3: Introducing uncontrol by scaling the reservoir

4.2 Explorative Mapping Design

ESNs offer a variety of ways in which un-trained behaviours can be shaped, through variation of global network parameters, and through fine tuning of weights. The PD external exposes these parameters for the user to interact with, however, especially in the case of tuning weights, the parameter space can be large and nonlinear, and therefore difficult to explore intuitively. To aid navigation through this parameter space, a PD abstraction was designed to allow interactive evolution[5] of network parameters. Figure 4 shows the interface for this abstraction, which allows the user to randomly mutate arrays of numbers and send them back to the ESN external. This can be used to vary and explore the input, output, feedback and reservoir weights. PD can also be used to visualise the reservoir activations, which can help to isolate issues with parameter choice.

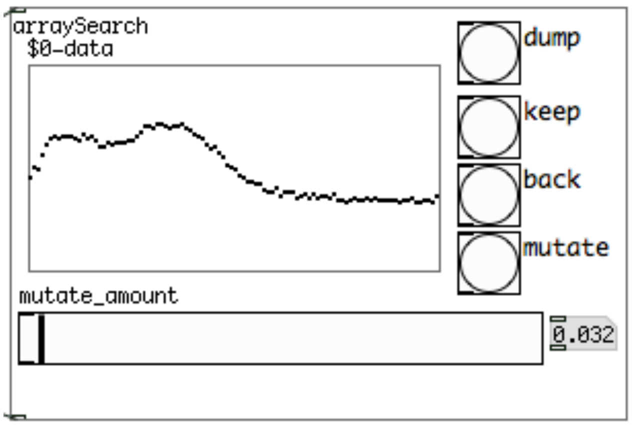


Figure 4: Parameter Space Search

To explore this facility, a task was devised where an ESN was trained to create the continuous synthesis controls for drum sounds. ESNs have an impulse response; a spike of noise will cause a fading response as the network dynamics settle after being perturbed. A mapping was set up where outputs of an ESN controlled the synthesis parameters of a drum synthesiser. As input, a velocity sensitive key was mapped to create an impulse to send to the network. The exploration tools were used to find a suitable mappings between the key and the drum sound. Figure 5 shows a sample of eight of the envelopes that were produced from fine tuning of the weights in this way. The system is capable of a variety of outputs, and interesting results were obtained from fine tuning the balance between feedback strength and output weights. While these control streams could be designed manually, the particular value of generating them in this way is that the ESN represents a model which can be modulated in realtime to vary the output. Furthermore, the system can be designed to behave in a nonlinear manner for different inputs. Figure 6 shows normalised outputs from an ESN perturbed by an impulse scaled to (a) 0.1 and (b) 5.0. The outputs have subtle differences, showing the nonlinear response to simple input variation.

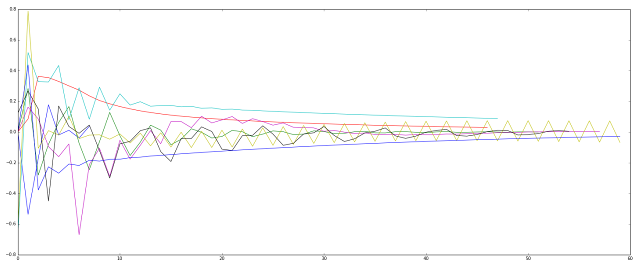


Figure 5: Percussive Envelopes from Output Weight Exploration

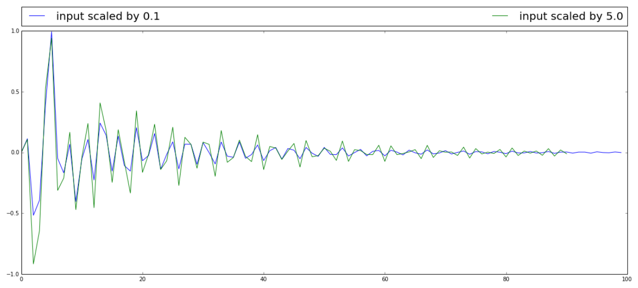


Figure 6: Percussive Envelopes with Scaled Inputs

4.3 Pattern Recognition

An experiment was set up to test the ESNs power of pattern recognition, using audio patterns as input. A training set was constructed as follows: two drum samples, a kick and a snare, were loaded into PD sample players, and a signal chain was constructed where the audio was analysed using a 64 point FFT, with the results recorded into 32 vectors. As this was a classification task, a bias of 2.5 was added to the FFT outputs, to push the reservoir into non-linear zones of behaviour. Two output vectors were set up for the teacher signal, and set as follows: with no sample playing, both were set to 0, when sample A playing, vector 1 was set to 0.5, and with sample B playing, vector 2 was set to 0.5. A training set was recorded, consisting of 4 independent repetitions of each sample, lasting a total of 10000 audio frames. Following from [14], gaussian noise was passed through the signal chain when samples were not playing. A reservoir was set up with 32 inputs, 400 reservoir nodes, and 2 outputs, with a *tanh* reservoir activation function, and linear readout. The simulation was run using leaky integration, with a leak rate of 0.001. The reservoir and input layers had 30% connectivity, and no feedback was used between output and reservoir. A spectral radius of 1.1 was chosen. After training, it could be observed that the two outputs showed opposite polarities, depending on the sample being played. A moving average filter simplified this output, show clearly separable responses for the 3 states (silence, sample A and sample B) (see figure 7). This result was consistent over 200 repetitions of the samples. The trained network also showed that it could generalise to untrained data in similar classes of sound. Following this, a test was carried out with two similar kick drum sounds, which the trained ESN could also consistently differentiate between. While the limitations of this small study are acknowledged and a deeper treatment of the results is outside the scope of this paper, the results show an encouraging outcome for further research, using a greater number of classes and systematic measurement of similarity between training samples.

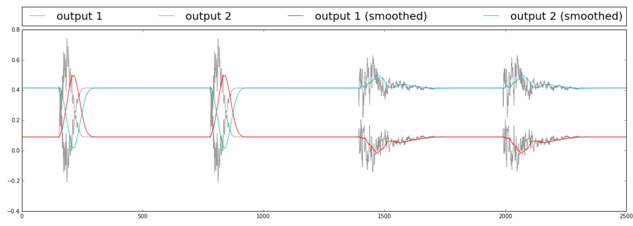
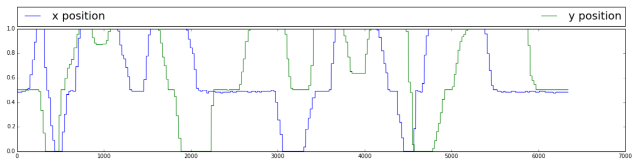


Figure 7: Recognition of Drum Samples

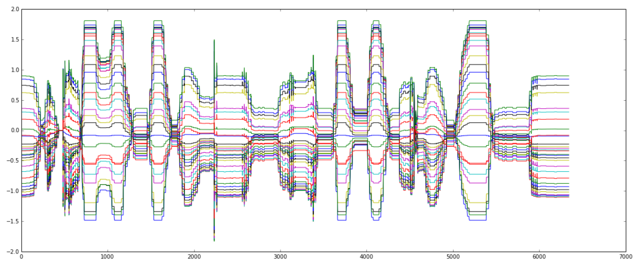
4.4 Multiparametric Control

This experiment explored the ESN as a trainable mapping processor for controlling 32 parameters of a synthesis engine with a two parameter joystick on a gamepad. Two 32 point arrays were hand-drawn in PD, and a training set was built that associated the array data with different positions of the joystick. 10000 frames of training data were recorded, and used to train a 400 node ESN. The ESN used a *tanh* reservoir activation and a linear readout, with 30% connectivity on each layer. No feedback was used. The spectral radius was 0.3, and a washout of 500 frames was used in training. The reservoir used the standard simulation algorithm, and ridge regression was used for training.

A recording of input and corresponding output of the trained network is shown in figure 8. The result was a network which interpolated between the two training arrays according to the position of the joystick, providing a musically interesting mapping system. The network responded to training successfully, and at key joystick positions, it recreated the exact data from the original arrays. To enhance the mapping, the second joystick on the gamepad was mapped to reservoir scaling, allowing variable nonlinearity to be introduced into the system.



(a) Joystick Movement



(b) ESN Output

Figure 8: Recording from a 2 to 32 Parameter Mapping

5. DISCUSSION

PD is a promising environment for the design and use of ESNs; it provides useful facilities for creating training sets, including hand drawn data in graphical arrays, procedurally generated mappings, and the ability to record sensor data from a range of sources such as cameras, game controllers and Arduinos. PD's environment also helps with training by providing realtime monitoring and visualisation of network states, helping the user to understand how the network is functioning. As Jaeger states[15], monitoring internal states can be very helpful for creating successful networks, and it also helps the user to find an intuitive understanding of how ESNs operate. Furthermore, observation of realtime output signals can be invaluable. The pattern classification use case is a strong example of this; the trained network did not successfully recreate the teacher signal, however observation of the outputs showed that the result still consistently classified the inputs; further post-processing of the output signals using moving average filters clarified this further. This also highlights a wider issue with ESNs, that training can be a complex process; there is a wide choice of parameters with which to create and train a network, and the way in which the training sequence is put together can have a significant effect on the result, for example the addition of gaussian noise in the pattern classification experiment. Training is also aided by an understanding of the inner workings of the algorithm, and achieving a good result can require some expertise and involve an extended process of experimentation. This situation can be helped by providing guidelines for solving common problems (see section 6), and by providing additional interactive tools to support training.

Outside of the training process, ESNs offer a variety of options for processing input. Almost any part of the process can be modulated by external signals to achieve interesting creative results. The process of perturbing a dynamical system with external input can give compelling, unpredictable and strangely lifelike behaviours for music and interaction. The main challenge is in specifying these behaviours; the explorative approach proposed here aid the discovery of interesting mappings, but there are still many more possibilities to enhance the tools provided by this system.

6. CONCLUSIONS

The use cases shown in this paper demonstrate that the Echo State Network shows promise as a useful and flexible tool for mapping in digital musical instrument design. Good results were achieved in scenarios of pattern classification, multiparametric control, explorative mapping and the design of nonlinearities and uncontrol. The use cases also highlight several areas where difficulties may occur when using this system. Jaeger[15] offers some very useful 'tricks of the trade' for effective training and general use of ESNs. Following the studies in this paper, some more 'tricks' can be added to this list, for the specific context of designing musical mappings.

6.1 Heuristics for Mapping with ESNs

1. Push inputs through the ESN at a constant sample rate. If used with controls that only update when changed (e.g. the PD sliders), slower dynamics in the reservoir may mean that the output is inconsistent.

2. Uncontrol can be introduced into a network by scaling the spectral radius, scaling and biasing the input and

feedback weights, and adding noise to the activation function.

3. Interactive evolution strategies can aid the design of un-trained behaviours of the reservoir.

4. For classification functions, adding input bias can improve the results, by pushing the reservoir into nonlinear zones of operation.

5. ESNs are easier to train if appropriately tuned or preprocessed information is provided in the input stage. For example, frequency analysis of a time series may give better results than the time series itself.

6. Following from this, post-processing is also helpful at the output stage. For example, smoothing the output of a classifier may improve the classification rate.

7. To reiterate Jaeger's guidelines, it's best to avoid symmetry in the input. This is especially important when working with audio or control signals; try adding a bias to symmetrical input.

7. FUTURE WORK

The purpose of this study has been to establish efficacy and outline potential use cases for ESNs within the field of mapping design. The next stage in this research address issues surrounding wider use by computer musicians, focusing on HCI aspects; usability of the software, and user experience. In terms of development of the ESN external, it would be compelling to investigate the efficacy of ESN topology algorithms such as scale-free small world networks[6] for mapping tasks, and to expand the explorative mapping tools. The uncontrol study leads to questions concerning the classes of uncontrol behaviour which the ESN can model, and how these behaviours can be represented in the training process and in the modulation of network parameters. The study highlights the design of nonlinear mappings and uncontrol features as an interesting area in mapping research, in which ESNs may be able to contribute significantly.

8. ACKNOWLEDGEMENTS

The work reported in this paper is part of the PRAISE (Practice and Performance Analysis Inspiring Social Education) project which is funded under the EU FP7 Technology Enhanced Learning programme, grant agreement number 318770.

9. REFERENCES

[1] D. Arfib, J. M. Couturier, L. Kessous, and V. Verfaille. Strategies of mapping between gesture data and synthesis model parameters using perceptual spaces. *Organised Sound*, 7(02):127–144, 2002.

[2] J. Butcher, D. Verstraeten, B. Schrauwen, C. Day, and P. Haycock. Extending reservoir computing with random static projections: a hybrid between extreme learning and rc. In *European Symposium on Artificial Neural Networks*, 2010.

[3] B. Caramiaux. *Studies on the Relationship between Gesture and Sound in Musical Performance.* PhD thesis, University of Pierre et Marie Curie (Paris 6) and Ircam, 2012.

[4] A. Cont, T. Coduys, and C. Henry. Real-time gesture mapping in pd environment using neural networks. In *NIME '04: Proceedings of the 2004 conference on New Interfaces for Musical Expression*, 2004.

[5] P. Dahlstedt. Evolution in creative sound design. In *Evolutionary Computer Music.* Springer London, 2007.

[6] Z. Deng and Y. Zhang. Complex systems modeling using scale-free highly-clustered echo state network. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 3128–3135. IEEE, 2006.

[7] G. E. Fels, S. S. Hinton. Glove-talk: A neural network interface between a data-glove and a speech synthesizer. *IEEE Transactions on Neural Networks*, 4(1), 1993.

[8] R. Fiebrink. *Real-time Human Interaction with Supervised Learning Algorithms for Music Composition and Performance.* PhD thesis, Princeton University, 2011.

[9] S. Gelineck and S. Serafin. From idea to realization - understanding the compositional processes of electronic musicians. In *Audio Mostly*, 2009.

[10] N. Gillian, R. Knapp, and S. O'Modhrain. A machine learning toolbox for musician computer interaction. In *NIME '11: Proceedings of the 11th international conference on New interfaces for musical expression*, 2011.

[11] G. Holzmann. Echo state networks with filter neurons and a delay and sum readout. *Neural Networks*, 2009.

[12] A. Hunt and R. Kirk. Mapping strategies for musical performance. In M. Wanderley and M. Battier, editors, *Trends in Gestural Control of Music.* Ircam - Centre Pompidou, 2000.

[13] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. 2001.

[14] H. Jaeger. Short term memory in echo state networks. Technical report, Fraunhofer Institute for Autonomous Intelligent Systems, 2002.

[15] H. Jaeger. A tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach. Technical report, Fraunhofer Institute for Autonomous Intelligent Systems, 2005.

[16] H. Jaeger and D. Eck. Can't get you out of my head: A connectionist model of cyclic rehearsal. In *ZiF Workshop*, pages 310–335, 2006.

[17] A. Johnston, B. Marks, L. Candy, et al. Sound controlled musical instruments based on physical models. In *Proceedings of the 2007 International Computer Music Conference, pp. vol1*, pages 232–239, 2007.

[18] S. Jordà. *Digital Lutherie: Crafting musical computers for new musics' performance and improvisation.* PhD thesis, Universitat Pompeu Fabra, 2005.

[19] C. Kiefer. *Multiparametric Interfaces For Fine-Grained Control of Digital Music.* PhD thesis, University of Sussex, 2012.

[20] A. Momeni and C. Henry. Dynamic independent mapping layers for concurrent control of audio and video synthesis. *Computer Music Journal*, 30(1), 2006.

[21] M. S. Puckette. Pure data. In *Proceedings, International Computer Music Conference*, pages 37–41, 1996.

[22] A. Tidemann and Y. Demiris. Groovy neural networks. In *Proceeding of the 2008 conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, 2008.

[23] D. Verstraeten. *Reservoir Computing: computation with dynamical systems.* PhD thesis, Ghent University, 2009.