Musical Instrument Classification using Democratic Liquid State Machines

Jornt R. de Gruijl JORNT.DEGRUIJL@PHIL.UU.NL
Marco A. Wiering MARCO@CS.UU.NL
Utrecht University, Department of Information and Computing Sciences, Padualaan 14, 3584 CH Utrecht

Abstract

The Liquid State Machine (LSM) is a relatively new recurrent neural network architecture, in which a static recurrent spiking neural network referred to as a 'liquid' and a trainable read-out network are combined to tackle time-series data. In this paper we describe the Democratic Liquid State Machine (DLSM) that uses an ensemble of single LSMs. We investigated the feasibility of the two LSM architectures as a complex spectrum analyzer over a broad frequency range using a musical instrument classification task in which bass guitar and flute had to be recognized by timbre. The experiments showed that single LSMs correctly classified 96% of all test samples, whereas the DLSMs classified 99% of all test samples correctly, improving overall performance to near perfection.

1. Introduction

Time-series data, such as sound, have long been a problematic type of input for classic feedforward neural networks. This is due to the fact that they lack the ability to retain information, which is a necessity for real-time processing of temporal patterns. Possible solutions for this problem are to use a time-delayed neural network (Waibel et al., 1988) or a recurrent neural network (RNN). The problem of using a time-delay neural network is that one has to decide in advance how many previous time-steps to take into account as useful input information. This is related to the (unknown) Markov order of the system. If one specifies too many inputs, the result is often slow learning and overfitting, whereas with too few inputs the task cannot be learned well. A problem with traditional recurrent neural networks trained with algorithms such as backpropagation through time (BPTT) (Rumelhart et al., 1986) or real-time recurrent learning (Williams & Zipser, 1989), is that learning long-term dependencies is made very difficult due to the problem of vanishing gradients (Hochreiter, 1998). This occurs when the error that is being backpropagated dilutes with every step, so that it cannot reliably learn from inputs that lie many (e.g. more than 10) steps in the past. A more advanced recurrent neural network architecture is Long-Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997). LSTM is ideally suited for remembering particular events, such as remembering whether some lights were on or off (Bakker et al., 2003), but in a previous comparison to evolving Spiking neural networks, LSTM was outperformed on certain toy problems (Koopman et al., 2003).

Two new recurrent neural network architectures are the echo state network (Jaeger, 2001), and the liquid state machine (Maass et al., 2002). These novel architectures share the principle of using a static filter (dubbed 'liquid') that transforms a temporal input stream to an activation pattern and a function approximator that learns to map the activation pattern to the desired output. Usually, a non-adaptive liquid is used, which makes it unnecessary to search in the space of recurrent neural networks. In this case, only a feedforward mapping has to be learned, which can be done by e.g. feedforward neural networks (Rumelhart et al., 1986).

In this paper we propose the Democratic Liquid State Machine (DLSM) that uses bagging and majority voting to enhance the capabilities of a single LSM. Since LSMs generally have low bias and high variance, an ensemble approach may turn out to be very fruitful. We compare the LSM and the DLSM on a musical instrument classification task. In this task we made our own bass guitar and flute samples and studied the classification accuracy of both algorithms.

Outline. In Section 2, we describe the Liquid State Machine approach and in Section 3 we describe the DLSM. Section 4 describes the experimental setup,

and Section 5 presents the results. In Section 6 we discuss the obtained results and sketch future possibilities.

2. Liquid State Machines

The Liquid State Machine (LSM) (Maass et al., 2002) utilizes two principles: the capacity of forward processing neural networks to work with high dimensional vectors, and the property of recurrent neural networks of retaining information. Since the latter occurs even without training, one could train a function approximator such as a linear or feedforward neural network on the perturbations in an untrained recurrent neural network. The difference between the LSM and the Echo State Network (ESN) (Jaeger, 2001) lies in the type of untrained recurrent neural network that is used. For the ESN, a recurrent neural network with neurons with a sigmoid activation function are used, whereas for the LSM a spiking neural network consisting of spiking neurons is used.

2.1. The Main Idea of the LSM

Input is fed into the liquid of an LSM M, which generates a corresponding liquid state. The liquid state can be regarded as the output of some operator or filter L^M that maps input functions $u(\cdot)$ onto functions $x^M(t)$:

$$x^M(t) = (L^M u)(t).$$

Because not all inputs last the same number of time steps and the amount of information that can be represented by a finite number of nodes is limited, an LSM is supposed to have a fading memory.

Fading memory (Boyd & Chua, 1985) is a continuity property of filters F that requires any output $(Fu)(T)$ of input function $u(\cdot) \in U^n$ to possibly be approximated by the output $(Fv)(T)$ of an input function $v(\cdot) \in U^n$ that approximates $u(\cdot)$ in a sufficiently long time interval $[0, T]$. With this requirement met, F has fading memory if for every $u \in U^n$ and for every $\epsilon > 0$ there exist $\delta > 0$ and $T > 0$ such that if $||(Fv)(T) - (Fu)(T)|| < \epsilon$ for all $v \in U^n$ it holds that $||u(t) - v(t)|| < \delta$ for all $t \in [0, T]$.

To illustrate this, one could view the liquid metaphorically as an actual liquid, e.g. a pond. Rain on the pond causes the surface to ripple, but due to friction the ripples fade. The most recent drops contribute the most to the current state of the pond's surface. Likewise, in a liquid, inputs decay as time goes by, causing more recent inputs to have a larger influence on the current liquid state than other ones.

In addition to the property of fading memory, the LSM needs to have the properties of approximation and separation. The property of separation means that that distinctly different input patterns should yield distinctly different liquid state representations. Any filter that has this characteristic can be used as a liquid. The property of approximation means that the read-out network can learn the target function until an arbitrary precision. Maass et al. have proven that LSMs satisfying the properties of separation and approximation have universal power for computations with fading memory on time-series (Maass et al., 2002).

Taking a liquid state $x^M(t)$ as input, the read-out network functions as a function f^M that transforms at any time t the liquid state into an output $y(t)$:

$$y(t) = f^M(x^M(t)).$$

The read-out network does not need some form of memory, since information about earlier states is retained by the liquid. Thus, the liquid is used to transform a temporal stream of inputs to an activation pattern of the neurons in the liquid at time-step t. The read-out network is now trained to map the activation pattern $x^M(t)$ to the desired output $y(t)$, which can be done using supervised learning algorithms such as feedforward neural networks.

2.2. Computations in the Liquid

We will describe the liquid we have used in our experiments, which can be seen as a simplified version of the integrate-and-fire model (Gerstner & Kistler, 2002). In the brain there are many kinds of neurons, but some things they all have in common. They all have means of sending a signal and receiving incoming signals: an axon and dendrites, respectively. When the sum of incoming signals exceeds a certain threshold level, the neuron generates an electro-chemical pulse along its axon. It then enters an absolute refractory period in which it cannot fire, followed by a relative refractory period in which it is hard to excite a response from the neuron. [1]

Due to the biological nature of the task, the decision was made to use a liquid consisting of spiking neurons, which are similar in many ways to organic neurons. Spiking neurons accumulate input signals until they exceed a certain threshold, at which point they reset their activation value and fire a pulse (usually a 1 in-

[1] Even this is greatly simplified. For more in-depth information about biological neurons, we refer you to books such as "Cognitive Neuroscience – the Biology of the Mind" by Gazzaniga, Ivry and Mangun (2002).

stead of the 0 indicating inactivity), as described by the following formula:

$$f(a) = \{\, 1, \text{if } \ a \geq \ \theta, \quad \text{and 0 otherwise}$$

In the function above, θ is the threshold value. For the experiments a value of $\theta = 1$ was used. If the threshold is not reached, the activation exponentially decreases back to 10% of its original level, which took 15 time steps in the experiments with the selected decay factor ϕ:

$$\phi^{15} = 0.1 \rightarrow \phi = e^{\frac{1}{15} \ln 0.1}$$

To calculate the activation value $a_t(i)$ at time t for neuron i, the following recursive formula is used:

$$a_t(i) = \phi \cdot a_{t-1}(i) + i_t(i)$$

Where $i_t(i)$ is the sum of incoming signals at time t for neuron i:

$$i_t(i) = \sum_j w_{ji} f(a_{t-1}(j))$$

The summation above is over the incoming connections with weights w_{ij}, and depends on the liquid structure. $f(a)$ is the threshold function as defined before.

When a node fires, the pulse travels along weighted connections to other neurons. After this, the neuron that fired enters an absolute refractory period in which it cannot fire. During this time it can still accumulate input signals, potentially causing the neuron to fire immediately once the refractory period is over. However, the decay factor possibly causes them to have extinguished enough to have little or no effect. An absolute refractory period of 15 time steps was selected. Unlike biological neurons, the spiking neurons used in the experiment had no relative refractory period.

2.3. Mapping the Liquid to the Read-out Network

There are multiple ways to map the liquid state to the read-out network. Some examples are to use activation patterns at a certain point in time of a randomly selected number of neurons in the liquid, or only those of a designated output layer, or even of the entire liquid. Another approach could be to use firing rates of neurons.

For the representation of the liquid states, it was decided to use a vector containing the activation values of all of the non-input nodes of the liquid. Thus, when the read-out network was called upon to classify the sample, it would be fed a list of activation values of that particular point in time. This way, no additional factors have to be introduced (e.g. firing rates) and the chances of leaving crucial liquid nodes out of the read-out network's input are eliminated.

3. Democratic LSMs

There exist a number of general algorithms that learn multiple models (classifiers) and combine them to produce the final result. One method is *stacked generalization* (Wolpert, 1992) which combines induced models from the bottom layer to the top-layer, where independent model errors are used to select models for predicting the answer to a query. Stacked generalization can be seen as a meta-theory for combining models. An example of this theory is the hierarchical mixtures of experts that uses gating networks to divide the input-space into regions where experts are responsible for giving the outputs (Jordan & Jacobs, 1992). Another ensemble algorithm is *bagging* (Breiman, 1996) which learns a set of independent models by first bootstrapping the data to get a training set and then trains a new classifier on this data set. This is subsequently repeated a number of times. The models are then combined by using majority voting of the predicted classes. Another method which receives a lot of attention is *boosting* (Freund & Schapire, 1996; Schapire et al., 1997) which sequentially trains a set of models where the data is reweighted after learning each new classifier. This is done so that misclassified examples get higher weight in the training data for the next classifier. By combining multiple classifiers through voting, individual errors are corrected by the other classifiers.

Here we introduce democratic Liquid State Machines which are similar to using a bagging method with liquid state machines. However, since a liquid state machine consists of a liquid and a read-out network, there are two options to use bagging; (1) A single liquid is used, and multiple read-out networks are trained using this liquid; (2) Multiple liquids and read-out networks are independently trained and used together with majority voting to produce the final result.

The first option we discarded, since we are using randomly initialized liquids. That could cause a liquid unable to represent all frequency bands equally well to be used for all read-out networks, leading to poor performance if a characteristic frequency should fall in such a range. Also, the diversity found in the read-out networks trained on a single liquid could be relatively small. We used the second method, and thereby initialized multiple liquids for which separate read-out networks are used.

It is well-known that bagging methods perform best if a classifier has a small bias and a large variance. Due to the large dimensionality of inputs (based on the many neurons in the liquid) and the use of a feedforward neural network as read-out network, we expect LSMs to have a large variance and thus to profit from using the ensemble method. We will see in the experiments whether our expectations turn out to be right.

4. Experimental Set-up

LSMs are relatively new and have been shown to perform well on toy problems. Little is still known about the feasibility of using them for more practical tasks, though promising research in this field has been done (Vreeken, 2004). We decided to investigate the feasibility of utilizing the LSM principle for a practical task that – to our knowledge – had not been formally investigated yet: complex spectrum analysis over a broad frequency range.

To this end, a classification task is used in which two musical instruments, the bass guitar and the flute, have to be identified by frequency analysis. Audio classification tasks using neural networks can be done by selecting key characteristics of the samples, analyzing a sample to extract these characteristics and then feeding the resulting information vector into a forward processing network (Malheiro et al., 2004). The hybrid network autonomously performs the first two steps, providing an improvement in terms of engineering.

The research focused on the tonal overlap of the two instruments, since the set of common tones is the only set where the distinction between the two musical instruments must be made by timbre alone. An advantage was the fact that this set is relatively small, thus creating natural boundaries for the range of sound fragments.

4.1. Processing the Input-streams

The information fed to the read-out network in this project came directly from the liquid. The liquid's input was pre-processed: a Fast Fourier Transform (FFT) algorithm with a fixed time window of 5.8 ms. was used to preprocess the sound samples, transforming the wave pattern into a vector of frequencies and their respective amplitudes. This is similar to the preprocessing that occurs in the human cochlea and was considered to be highly likely to boost the performance of the LSM. Feeding the liquid raw wave data was therefore not attempted.

The vectors resulting from the FFT were inserted into the liquid one by one every 5.8 ms. For every sample, 5 'snapshots' (or fragments) of liquid activations at different time steps were made, which were in turn passed on to the read-out network. These results of the read-out network after each snapshot are then combined using a majority voting procedure. The idea behind the snapshots procedure was that some part of a sound clip is likely to correspond with some of the data learned by the network, facilitating classification, while the chances of wrongly classifying a musical instrument because of an uncharacteristic activation pattern are decreased.

4.2. The Liquid Structure

For every frequency band that the FFT algorithm supplied there was one input neuron (for a total of 64). The input neuron added the FFT-value (between 0 and 1) for this frequency band to its discounted previous activation. It fired when its activity exceeded the firing threshold 1. Note that the only information used about the volume of a frequency band this way is whether the activity of the neuron (computed using FFT-values over multiple steps) exceeded its firing threshold or not. The volume levels of the recordings also differed greatly, making volume not much of a characteristic by which to determine what musical instrument was heard. The LSMs have to use crucial information about the timbre (the auditive fingerprint) of an instrument, for which the complete frequency spectrum plays a central role for the classification task.

For every input neuron there was a four-neuron column, thus creating a grid with 64 columns and five (5) rows, of which one row consisted solely of input neurons. The neurons were randomly connected with the chance of two neurons connecting given by the formula:

$$P_c(a,b) = C \cdot e^{-\frac{D(a,b)^2}{\lambda^2}}$$

Here $D(a,b)$ denotes the Hamming distance between nodes a and b. C is a constant between 0 and 1 and regulates the balance between local and global connections together with λ. For the experiments we used $C = 0.9$ and $\lambda = 2$.

Throughout the liquid all weights for the connections were set to 0.2, causing at least 5 simultaneous activations to be needed for a neuron to fire. A total of 10% of the connections was inhibitory as opposed to excitatory. These connections were randomly distributed through the liquid.

4.3. The Read-out Network

The read-out network was a feedforward neural network with two output units trained using the back-

propagation algorithm with a learning speed of 0.1. Training ended when the read-out network gave only correct answers that had an activity at least a factor 1.5 bigger than the activity of the wrong output unit on the training set, or when 2000 iterations had been done.

Though normally a network with a small number of hidden neurons works best for classification tasks – since this forces the network to generalize over the data – we found through trial and error that using 50 hidden neurons worked best for this task. This is possibly due to the way the network was trained. Presumably, when combined with the conditions for ending training, this large number of hidden neurons allowed the network to represent many possible properties a sample could have with a small weight attached, thus creating a good generalization performance.

4.4. The Sound Files

The sound files used were 16-bit mono Wave-files with a sample rate of 22 kHz.

The total set of 234 samples was equally divided in 117 recordings of bass guitar sounds and 117 of flute sounds. Per run, the system was trained on a random set of 100 bass and 100 flute samples, after which it was tested on the remaining samples. For the experimental results we repeated this 10 times using different training and test sets.

Most of the sound files were made especially for this research. For those recordings, three bass guitars and three flutes that were significantly different in timbre were used. To further prevent overfitting some audio snippets from compact discs of several performing artists were used. These samples did not necessarily only use tones that the bass guitar and flute have in common. But since they were relatively few (approximately 15% of the total number of samples), they were expected to give the system a bias at most.

The set of samples consisted of fragments of scales or melodies, two-tone intervals, and single tones. On most of the recordings only one instrument (bass or flute) could be heard. But to test the LSM's robustness, other recordings could also feature 'noise' ranging from softly playing instruments to roaring crowds. Care was exercised so as not to select recordings of both bass guitar and flute, though.

Although some were longer, most sound files were roughly one second in length, meaning they would be fed into the liquid in about 170 time steps. Combined with the five snapshot approach, that would boil down to 34 time steps between snapshots.

The snapshot method can also be regarded as providing the LSM with five times the number samples that was originally recorded. In the rest of this article, we will refer to such very short fragments of original samples as 'fragments' while using the term 'samples' to indicate original complete sound files. The multiple classifications of the single fragments of a sample are also used for majority voting over the classification of the complete sample.

5. Experimental Results

The experiments were done with software written in Java, run on a system with a 1.92 GHz AMD Athlon XP 2600+ processor with 512 MB of RAM, under Microsoft Windows XP Professional. A total of 50 single liquid state machines were tested on 10 different test sets (and training sets). For the DLSMs, 10 runs were performed, each with a different test set (and training set), with 10 LSMs to vote.

There is a structural difference between the tables with the results for the single LSMs and the DLSMs: those for the single LSMs have no 'undecided' category, whereas those for the democratic LSMs do. This is because single LSMs get a fragment or sample either wrong or right. Fragments could theoretically be undecided with the read-out network giving equal output for both, but in practice this never happens. Furthermore, because of the 5 snapshots procedure, either bass or flute had to be chosen in the end, with no middle way possible. Because we used 10 LSMs in the DLSM setup, things were different there. It was possible for 5 LSMs to vote for one instrument while the other half voted for another. In this case the classification is "undecided".

5.1. Results with Single LSMs

The experimental results with single LSMs are shown in Tables 1, 2, and 3. As can be deduced from Tables 1 and 2, the bass and flute were about equally difficult for the single LSMs. Apparently no easy means of identification was found to circumvent an in-depth analysis of the input, otherwise either flute or bass would have been classified correctly significantly more. Table 3 shows that the total accuracy of the single LSMs on fragments is 88.4% and on the samples it is 95.9% which are rather good results.

Simple tones made up for most of the training and test sets, and unsurprisingly these were usually correctly classified. They did not score highest, though, probably due to the fact that a single tone does not hold as much information for classification as several

Table 1. Classification accuracy of the single LSMs for the bass guitar samples.

	Average	Std. Dev.
Fragments Wrong	12.7 %	7.6 %
Samples Wrong	3.2 %	6.8 %

Table 2. Classification accuracy of the single LSMs for the flute samples.

	Average	Std. Dev.
Fragments Wrong	10.6 %	5.3 %
Samples Wrong	5.0 %	6.8 %

tones do. On top of that, it was observed that from run to run errors could concentrate in certain parts of the frequency range. This is probably due to the randomness of the liquid, causing it to react in different ways in different parts of the frequency range. A single tone in a part of the frequency range that the liquid does not respond well to, makes it hard to classify the sample without additional tones in other parts of the frequency range. As a result two-tone intervals and fragments of scales scored relatively higher.

Despite the few samples taken from compact discs, those sample fragments were classified correctly remarkably often. This is possibly due to professional mixing, enhancing salient features. There was a difference between the scores for bass and flute in this category, however. The system performed slightly better on the flute CD samples, which was probably caused by the fact that the average bass sample had more background noise than the average flute sample. A live performance for flute is not often accompanied by roaring crowds and a flute is often recorded with a very clean sound in the studio. A bass, on the other hand, is often found playing in front of noisy people when performing live and adding some amount of electronic effects when recording in a studio. All in all, the LSMs performed well and proved to handle large amounts of noise with relative ease.

The performance of the system regarding bass overtones is worth special mention. Overtones are basically the same as normal tones, minus the lowest frequency. As an effect, they are among the rare bass tones that can be as high as some of the higher flute tones. Only four training samples for these overtones

Table 3. Total classification accuracy of the single LSMs.

	Average	Std. Dev.
Fragments Wrong	11.6 %	5.2 %
Samples Wrong	4.1 %	4.1 %

existed, yet 85% of the *fragments* was classified correctly. The most spectacular result was in one run where there were no training samples for overtones in the training set, since they were all part of the test set. Despite that, one all-overtone sample was classified correctly 100% of the time, and the other samples normally. Apparently the system learned to do a decent frequency analysis, since flute CD samples were the only other ones to get into the same frequency range, which could have easily led the LSM to be mistaken.

Another interesting observation worth mentioning is the fact that flute and bass samples with a vibrato tone in it were almost always classified correctly. Possibly the system used this as a salient feature, since most flute tones have some vibrato, but the fact that the bass also benefited from this suggests otherwise. Most likely it is due to the fact that vibrato is not necessarily only a change in volume, but also a slight pitch shift, going up and down periodically. Because of this continuous sweeping across frequency bands, neighboring input neurons are sequentially activated through time, causing new neurons in the liquid to fire while other neurons sit through their refractory periods. This way, a pattern that defines the tone can be held almost continuously by the liquid, making it easier for the read-out network to classify.

5.2. Results with Democratic LSMs

The results of the DLSMs are shown in Tables 4, 5, and 6. The performance of the DLSMs is significantly better than that of single LSMs. The only samples that were incorrectly classified were flute samples. Most of these mistakes occurred during a run in which there were hardly any training samples similar to the samples that were incorrectly identified.

Performance increased overall with the greatest increase lying in the classification of single-tone samples. Most errors in identifying these were caused by random liquid structures incapable of adequately representing certain frequency-bound characteristics. Such individual errors are now canceled out. For the other files, majority voting in general causes a decrease

Table 4. Classification accuracies of the Democratic LSMs for the bass guitar samples.

	Average	Std. Dev.
Fragments Wrong	8.6 %	5.5 %
Fragments Undecided	1.5 %	2.5 %
Samples Wrong	0.0 %	0.0 %
Samples Undecided	0.6 %	0.5 %

Table 5. Classification accuracies of the Democratic LSMs for the flute samples.

	Average	Std. Dev.
Fragments Wrong	7.6 %	6.3 %
Fragments Undecided	1.2 %	3.5 %
Samples Wrong	1.8 %	6.1 %
Samples Undecided	0.0 %	0.0 %

in misidentified fragments, which in turn leads to improved performance in classifying samples.

The performance on all samples of the Democratic LSMs is 99.1% with 0.3% of the samples undecided. This is an excellent performance, but some fragments were structurally mistakenly identified, seemingly without reason for an experienced human ear.

The only general trend found in problematic cases is that some bass samples had tones without a clear onset, swelling like a flute tone. However, since the LSMs that were used are presumed to be insensitive to this feature due to the structure of the spiking liquid, this seems unlikely. For flute tones, no satisfying explanation was found either. It would seem that despite the positive results overall, there remains the question of which features the LSMs deem salient for classification and whether these are at all similar to the properties the human brain primarily takes into account.

6. Discussion

In this paper we described the Democratic Liquid State Machine that extends the normal LSM by using an ensemble method and majority voting using multiple liquids and read-out networks. The experimental results on a musical instrument classification task showed an excellent performance of the DLSM, getting an accuracy of about 99% on the testing samples.

Another musical instrument classification experiment using flute, eight other wind instruments and a piano in a one-versus-one classification paradigm yielded 98% correctly identified samples in the piano-versus-other cases.[2] (Essid et al., 2004) The samples used were recordings of solo musical phrases taken from CDs of classical music and jazz, and included both live and studio performances. Identification was done by doing an extensive and advanced feature analysis of the entire sample, rendering real-time classification impossible. It would appear that the DLSM yields similar results without a lot of a priori knowledge and assumptions about the input. Furthermore, it enables real-time classification.

Table 6. Total classification accuracies of the Democratic LSMs.

	Average	Std. Dev.
Fragments Wrong	8.1 %	4.8 %
Fragments Undecided	1.4 %	2.6 %
Samples Wrong	0.9 %	2.6 %
Samples Undecided	0.3 %	0.3 %

The matter of which features of the samples were used by the DLSMs for classification is one that raises questions. Though the system performs extremely well, little is known about the actual processes by which a decision is made. We do not know whether a DLSM scrutinizes the same qualities of the sounds that the human brain does. Thus we cannot tell to what measure a DLSM is biologically plausible.

For future research, it would be interesting to see how such a system would perform on similar classification tasks with more than two categories, e.g. the musical instruments used by Essid et al. (2004). Furthermore, it is worth investigating how the DLSM would perform on a more complex frequency analysis task, such as discriminating between a hobo and a clarinet by timbre, thus taxing the separation property of the liquids more. This would also shed more light on the performance of the DLSM when compared to earlier research in musical instrument classification.

Research as to the feasibility of a DLSM as an on-line music genre classifier may also prove fruitful. Many downloadable music files on computer networks are often found lacking such information, causing inefficiencies in processes such as search commands. Should a DLSM prove capable of classifying music genres (which

[2]Since they used no bass guitar in this experiment, we selected the piano, which is the instrument that approximates the sound of a bass the most out of the instruments they used.

may also involve such characteristics as rhythm), it could be used as an on-line learning system monitoring such files and adding informative labels in case they are missing and learning from them if they are present.

The excellent performance of the DLSM in our experiments bodes well for future application. In those cases where individual LSMs perform well in general, but exhibit a high variance, a DLSM stabilizes and increases performance.

Acknowledgments

We would like to thank Harm Aarts, Hado van Hasselt, Wilco Moerman and Leo Pape for graciously permitting us to use their code. Our thanks also go out to Maaike van den Broek who provided us with approximately 100 samples worth of flute snippets.

References

Bakker, B., Zhumatiy, V., Gruener, G., & Schmidhuber, J. (2003). A robot that reinforcement-learns to identify and memorize important previous observations. *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2003)* (pp. 430–435).

Boyd, S., & Chua, L. O. (1985). Fading memory and the problem of approximating nonlinear operators with Volterra series. *IEEE Transactions on Circuits and Systems*, 1150–1161.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*, 123–140.

Essid, S., Richard, G., & David, B. (2004). Musical instrument recognition based on class pairwise feature selection. *ISMIR Proceedings 2004*.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Proceedings of the thirteenth International Conference on Machine Learning* (pp. 148–156). Morgan Kaufmann.

Gazzaniga, M. S., Ivry, R. B., & Mangun, G. R. (2002). *Cognitive neuroscience, second edition*. W. W. Norton and Company.

Gerstner, W., & Kistler, W. (2002). *Spiking neural models*. Cambridge University Press.

Hochreiter, S. (1998). Recurrent neural net learning and vanishing gradient. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(2)*, 107–116.

Hochreiter, S., & Schmidhuber, J. (1997). Long-short term memory. *Neural Computation, 9(8)*, 1735–1780.

Jaeger, H. (2001). The 'echo state' approach to analyzing and training recurrent neural networks. *GMD report 148*.

Jordan, M. I., & Jacobs, R. A. (1992). Hierarchies of adaptive experts. In J. E. Moody, S. J. Hanson and R. P. Lippmann (Eds.), *Advances in neural information processing systems 4*, 985–993. Morgan Kauffmann.

Koopman, A., van Leeuwen, M., & Vreeken, J. (2003). *Dynamic neural networks, comparing spiking circuits and LSTM* (Technical Report UU-CS-2003-007). Institute of Information and Computing Sciences, Utrecht University.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation, 14*, 2531–2560.

Malheiro, R., Paiva, R. P., Mendes, A. J., Mendes, T., & Cardoso, A. (2004). Classification of recorded classical music using neural networks. Centro de Informática e Sistemas da Universidade de Coimbra.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel distributed processing*, vol. 1, 318–362. MIT Press.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1997). Boosting the margin: a new explanation for the effectiveness of voting methods. *Proceedings of the fourteenth International Conference on Machine Learning* (pp. 322–330). Morgan Kaufmann.

Vreeken, J. (2004). On real-world temporal pattern recognition using liquid state machines. Unpublished master's thesis, Institute of Information and Computing Sciences, Utrecht University.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. (1988). Phoneme recognition using time-delay neural networks. *Proceedings of the IEEE International Conference on Acoustic, Speech and Signal Processing* (pp. 107–110).

Williams, R., & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation, 1*, 270–280.

Wolpert, D. (1992). Stacked generalization. *Neural Networks, 5*, 241–259.