Classification of Musical Styles Using Liquid State Machines

Han Ju, Jian-Xin Xu, and Antonius M.J. VanDongen

***Abstract*— Music Information Retrieval (MIR) is an interdisciplinary field that facilitates indexing and content-based organization of music databases. Music classification and clustering is one of the major topics in MIR. Music can be defined as 'organized sound'. The highly ordered temporal structure of music suggests it should be amendable to analysis by a novel spiking neural network paradigm: the liquid state machine (LSM). Unlike conventional statistical approaches that require the presence of static input data, the LSM has a unique ability to classify music in real-time, due to its dynamics and fading-memory. This paper investigates the performance of an LSM in classifying musical styles (ragtime vs. classical), as well as its ability to distinguish music from note sequences without temporal structure. The results show that the LSM performs admirably in this task.**

I. Introduction

MUSIC as an art form can carry diverse types of information, and may elicit in the listener different moods or emotions. Such information in computer music can be retrieved (known as Music Information Retrieval) through machine learning algorithms and pattern recognition techniques to automate the process of classification and indexing in large digital music libraries. Computers after training could be used not only to check whether the given audio data is a piece of music or non-music, e.g. corrupted music with disordered note sequence, but also to maintain the content-based organization of databases, as well as to search the entire databases for certain types of music according to the user's preference.

Music style classification is one of the key problems in MIR. It is very difficult for computers to classify and analyze these pieces of abstract information because of the complex temporal relationships present in musical features, including pitch, rhythm and dynamics, however, humans can easily identify different styles and genres. The human brain is a highly complex system possessing an extraordinary capability to process musical information efficiently in real time. For the task of computer-based music analysis, bio-inspired techniques such as Artificial Neural Networks (ANNs) would be a good choice.

With increasing computational power, biologically realistic ANNs have attracted more and more attention. ANNs are widely used nowadays and with very high performance in solving problems including pattern recognition, time series forecasting and function approximation. The third generation of ANNs employs neurons that generate action potentials (spikes) and communicate through synapses. Although computational demanding, these Spiking Neural Networks (SNNs) are biologically more relevant and may also be more powerful [1]. The Liquid State Machine (LSM) [2] is a type of SNN inspired by cortical microcircuits; it has great potential to efficiently classify data with strong temporal features due to its inherent time-dependent dynamics. The universal computational power of LSM in real-time computing has been shown in [3]. Therefore, music classification that requires temporal analysis is a suitable application for the LSM.

Music can be seen as a very large class of audio signals that is "extremely variable and hence challenging" [4]. Methods for discriminating music from speech, including analysis based on energy contour statistics, and various classification strategies like Gaussian mixture models have been reviewed [4]. In [5] it is proposed to classify two music styles using statistics of pitches, intervals and rhythms, referred to as a "shallow structure description". Three types of classifiers, nearest neighbours, Bayesian classifier and self-organizing maps are trained on the extracted features and then used to classify jazz and classical music. In [6], thirteen statistical features like average and standard deviation of MIDI key number and duration are extracted and fed into classifiers. Three methods using Bayesian classifier, linear classifier and Neural Networks with Cascade-Correlation architecture are developed. The result shows that the neural network is the most powerful approach. In the work of [7], the music segments are represented by pitch histograms. Many of the proposed music analysis methods are based on offline processing or data buffering to extract features, so that static inputs can be presented to classifiers. The temporal relationships between notes are not greatly emphasized.

The application of LSM in music instruments classification is explored in [8]. A "democratic" LSM model that consists of multiple LSM and uses majority voting was proposed to classify bass guitar and flute. The music samples used are 16-bit mono Wave-files and pre-processed by Fast Fourier Transform (FFT) algorithm. It is reported that the democratic LSM gives high accuracy on this task. The music composers' classification between Haydn/Mozart and Beethoven/Bach using LSM are also discussed.

In this work, two music classification problems are discussed: music and non-music, and music style classification between classical and ragtime. The paper is organized as follows. Section II presents the architecture of the LSM system, the spike-encoding scheme, as well as the design of the experiments for the two problems. Classification results and the comparison between the LSM system and two other classifiers, Recurrent Neural Network

H. Ju and A.M.J. VanDongen are in the Program for Neuroscience and Behavioral Disorders of the Duke-NUS Graduate Medical School, Singapore (65-6015-7075; e-mail: g0901908@nus.edu.sg; antonius.vandongen@duke-nus.edu.sg).

J-X. Xu is in the Department of Electrical and Computer Engineering of the National University of Singapore (e-mail: elexujx@nus.edu.sg).

and K-Nearest Neighbourhood, are presented and discussed in section III, while general conclusions are in section IV.

II. Experiment Setup

The LSM has three parts: i) an input layer, ii) the liquid filter, and iii) a set of one or more readout neurons. Input neurons receive spikes encoding the sensory data (in our case a musical piece) and pass the spikes to the liquid filter through synaptic connections. Connections of input neurons to the central liquid filter are randomly initialized and the connection probability depends on the location of each input neuron with respect to the liquid filter, such that spatial information can contribute to the classification result. The role of the liquid filter is to expand the complex spatio-temporal input signal which encodes the musical data, into a high-dimensional feature space, facilitating linear classification. The liquid filter is able to take into account the temporal structure of the input signal because it possesses a 'fading memory' [3], which results from recurrent connectivity between excitatory neurons, the dependence of synaptic efficacy on recent input history and the exponential-decay of membrane potential changes induced by synaptic inputs.

Readout neurons are trained to perform the actual classification. They receive synaptic inputs from all the neurons in the liquid filter and project the filter state from high dimensional space into a low dimension using algorithms like linear regression or Fisher's linear discriminant. The final decision will be made by processing the output of the readouts.

A. *Architecture of the LSM*

All the experiments discussed in this paper are using the Matlab *CSim package* as described in [9] and [10]. Figure 1 shows the schematic diagram of our music classification system. Piano music data are encoded into 60 spike trains, with each train corresponding to one piano key (pitch); spike trains are fed into the liquid filter through the input layer. Readout neuron processes the state of the liquid filter over time and final decisions are made from its output values.

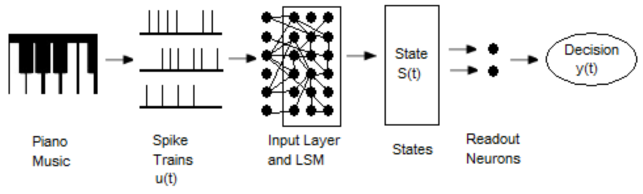

Fig. 1 The Music Classification System

The liquid filter consists of leaky integrate-and-fire (LIF) neurons with recurrent dynamic synaptic connections. LIF neurons are commonly used to model the behaviour of real biological neurons [11]. The standard LIF neuron model is used here, shown in equation 1:

$$\tau_m \frac{dV_m}{dt} = -(V_m - V_{rest}) + R_m(I_{syn} + I_{inject} + I_{noise}) \quad (1)$$

where the membrane time-constant τ_m is set to 30 ms; V_{rest} is 0 mV, the input resistance R_m is 1 MΩ; I_{inject} is set to 13.5nA, and I_{noise} is 1nA. Neurons have an absolute refractory period, which is 3 ms for excitatory neurons, and 2 ms for inhibitory neurons. These parameters follow [12], which are based on biological recordings.

In the LSM 80% of neurons are chosen to be excitatory and the remaining 20% are inhibitory [2]. All the neurons are located at points with integer coordinates in a 3D space. The probability of connection between two neurons is defined as:

$$P(D) = C * \exp\left(\frac{-D^2(a,b)}{\lambda^2}\right) \quad (2)$$

where $D(a,b)$ stands for the Euclidean distance between two neurons a and b. As λ increases, both the connection probability and the average connection length will increase. λ is set to 1.5 here. The value of C is set to 0.3 for excitatory to excitatory (E E) connections, 0.2 for excitatory to inhibitory (E I), 0.4 (I E), and 0.1(I I) according to [13] based on recordings in cortical brain areas.

Input neurons are all excitatory with the outgoing connections configured as static spiking synapse. C is 0.3 for connections to excitatory neurons in the filter and 0 to inhibitory neurons. Consequently, there are no connections from the input layer to inhibitory neurons. λ is set to 5. The input layer is constructed as a grid of 12×5 neurons, corresponding to 5 aligned octaves. The input layer is located 2 units' away from the front layer of the central liquid filter, as shown in Figure 2. Only a few selected neurons' ingoing and outgoing connections are drawn for clarity.

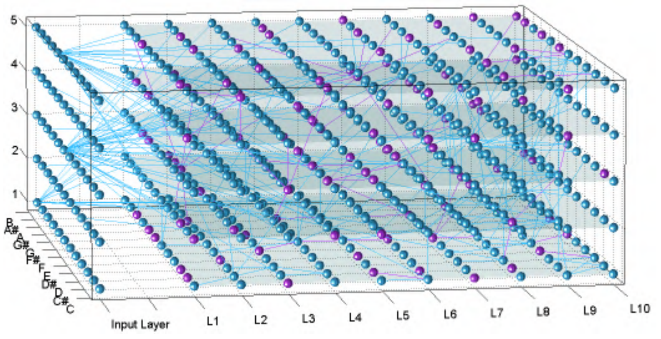

Fig. 2 The architecture of the LSM : the Input Layer (the most left layer) and the Liquid Filter (L1 – L10)

B. *Dataset and Spike Encoding*

The dataset used in this analysis contains classical and ragtime piano music from well-known composers (Table I) in MIDI format, downloaded from Bernd Krueger's website (Classical) [14] and Warren Trachtman's website (Ragtime) [15]. After checking the MIDI files, we selected 150 files in which 80 are classical and 70 are ragtime. The Matlab MIDI toolbox described in [16] is used to load and analyse music files. Three types of information are considered in all the experiments presented in this paper: pitch, which are represented in integers ranging from 0-127 that correspond to piano keys, for example, middle C (C4) is 60; onset time of the notes, which indicates the time when the key is pressed;

and duration that denotes how long the piano key press is sustained.

TABLE I
LIST OF COMPOSERS

Classical	Ragtime
Albéniz, Isaac (1860-1909)	Scott Joplin (1868 - 1917)
Bach, Johann Sebastian (1685-1750)	James Scott (1885 - 1938)
Balakirew, Mili Alexejewitsch (1837-1910)	Joseph Lamb (1887 - 1960)
Beethoven, Ludwig van (1770-1827)	Ferdinand (Jelly-Roll) Morton (1890 - 1941)
Borodin, Alexander (1833-1887)	James Hubert (Eubie) Blake (1883 - 1983)
Brahms, Johannes (1833-1897)	Charles L. Johnson (1876 - 1950)
Burgmueller, Friedrich (1806-1874)	Tom Turpin (1873 - 1922)
Chopin, Frédéric (1810-1849)	James Reese Europe
Debussy, Claude (1862-1918)	Irene Giblin
Granados, Enrique (1867-1916)	Percy Wenrich
Grieg, Edvard (1843-1907)	May Aufderheide
Haydn, Joseph (1732-1809)	Tad Fischer
Liszt, Franz (1811-1886)	H.H. McSkimming
Mendelssohn, Felix (1809-1847)	Zez Confrey
Moszkowski, Moritz (1854-1925)	Cy Seymour
Mozart, Wolfgang Amadeus (1756-1791)	E. Philip Severin
Mussorgsky, Modest (1839-1881)	Arthur Marshall
Ravel, Maurice (1875-1937)	Roy Bargy
Schubert, Franz (1797-1828)	
Schumann, Robert (1810-1856)	
Tchaikovsky, Peter (1840-1893)	

In MIDI files, the keys in each octave are coded by 12 pitches, therefore, the 12x5 neurons in the LSM input layer cover octaves 2-6 with the pitch number range of 36-95, and each neuron represents one piano key. Almost all the pitches are within octaves 2-6. The very few pitches which were outside this range are assigned to the value of either 36 or 95. The reason for using only 60 input neurons instead of 127 is that we want to keep the input dimension to be small relatively to the filter, without losing too much information, such that there is no conflict with the working principle of the LSM, which is to expand input dimensionality. If the number of input neurons is not significantly small compared to the filter, the effect of dimensionality expansion would not be obvious. It should be noted that a large size of the liquid filter is very resource demanding and could dramatically increase computational complexity.

The 12×5 arrangement of input neurons makes adjacent octaves sit beside each other as shown in Figure 2, such that the input spatial information can be one of the factors contributing to the final classification decision. It can be seen in the figure that the number of synaptic connections from each input neuron is decided not only by random initialization but also by its location (coordinates). More connections are created for the neurons in the centre of the input layer as they are relatively closer to the liquid filter in the Euclidean distance metric according to formula 2.

Spike encoding is based on the natural frequency of the piano keys. The standard natural frequency of the piano key corresponding to the MIDI pitch number *m* follows the formula 3:

$$F(m) = 440 * (\sqrt[12]{2})^{m-69} \qquad (3)$$

In our LSM, each input neuron *n* is designed to receive spikes with its own fixed frequency that is calculated as:

$$F_{input}(n) = L + \frac{F(n) - L_{natural}}{H_{natural} - L_{natural}} * (H - L) \qquad (4)$$

where $H_{natural}$ and $L_{natural}$ (Hz) are the pitch number 95's and 36's natural key frequencies respectively. Equation (4) guarantees that any natural frequency value within [$L_{natural}$, $H_{natural}$] is linearly mapped to the range of [*L*, *H*] in Hertz that can be adjusted by us. Spikes with a frequency of $F_{input}(n)$ are generated when the corresponding piano key n is pressed and held.

C. Music and Non-Music Classification

In this experiment we use classical music only. In fact, any style of music is valid in this case and the generality of this experiment holds.

Each MIDI file is divided into segments of 60 seconds, which are used as the data sets for training or testing. The remaining tail segment that has the length of less than 60 seconds is discarded. Non-music segments are generated from music segments by randomly swapping notes. In detail, for a pair of notes (*i*, *j*) that is randomly selected within a music segment, the pitch number and duration of note i are swapped with those of note j, while the onset time remains untouched. For each segment, such swaps are done for 100×*N* times, where *N* is equal to the number of notes in that segment. This operation completely destroys the temporal relationship between notes, while still maintaining pitch and duration statistics. Figure 3 illustrates a non-music stimulus generated from a segment of music. Each spike is represented by a dot; a burst of spikes appears as a bar.

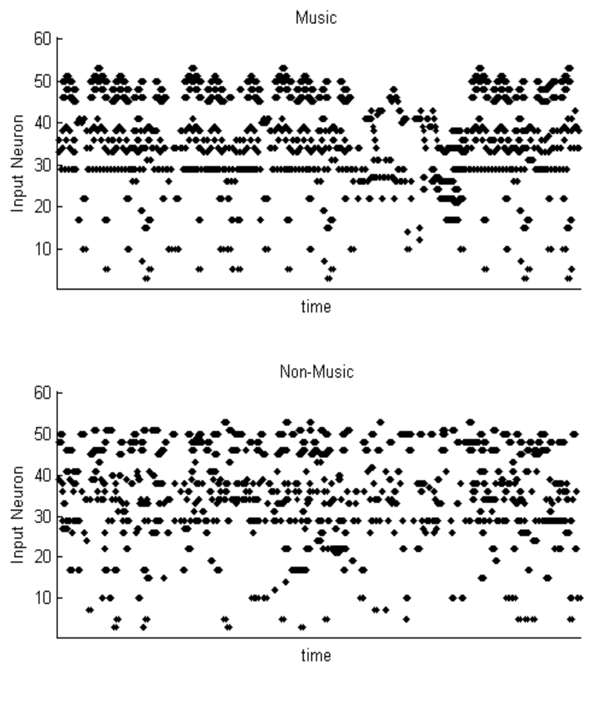

Fig. 3 An example of stimulus: Music and Non-Music

To effectively utilize the fading memory in the LSM, music segments are compressed from 60 seconds to several seconds by decreasing inter-spike intervals, because the LSM memory fades within seconds. A faster playback speed may facilitate the LSM to grasp temporal features in the music. Adjusting parameters to stretch the LSM's memory period and playing music in normal speed to get real-time classification is an alternative solution, but essentially there is no difference from our fast-playing method; therefore, the classification results should not be affected.

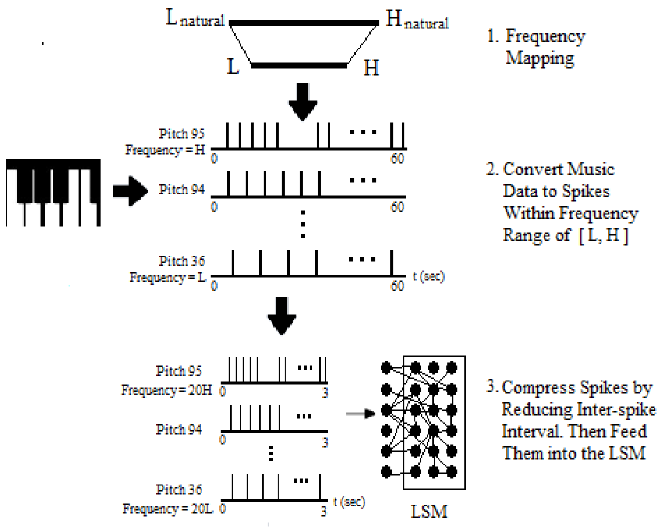

Fig. 4 The spike encoding scheme

Figure 4 shows the schematic diagram of spike encoding as discussed above. This figure gives an example of compressing a 60 seconds music segment to 3 seconds. In this case, compression in step 3 increases the spikes frequency by 20 times, i.e. from [L, H] in step 2 to [20L, 20H] in step 3. So if in the first step 'Frequency Mapping', we set L to be $L_{natural}/20$ and H to be $H_{natural}/20$ as in formula 2, the compressed spikes that are fed into LSM in the step 3 will have the natural frequency range [$L_{natural}$, $H_{natural}$], but fast-played in 3 seconds.

In this music and non-music classification, we set the encoding spike frequency range [L, H] to [$L_{natural}/20$, $H_{natural}/20$] which is [3.27 Hz, 98.78 Hz]. We will investigate the performance of LSM by using different compression time.

The classical music files are divided into two sets: the training set with 50 MIDI files and testing set with 30. After segmentation and generation of 'non-music' by repeated swapping of notes, the number of training instances increases to 234×2 for the two classes and testing increases to 177×2. It should be noted that there is no sharing of MIDI files in the testing and training sets.

D. *Music Style Classification*

In this section, the LSM is trained to classify two styles of music: Classical vs. Ragtime. Even though this is a case of binary classification, more readout neurons can be easily added to achieve multi-class classification.

Ragtime and Classical music have different rules for composition, for example, in contrast to classical music, syncopated rhythms are used in ragtime and there are many melody accidentals. Figure 5 gives an example of a classical and a ragtime stimulus.

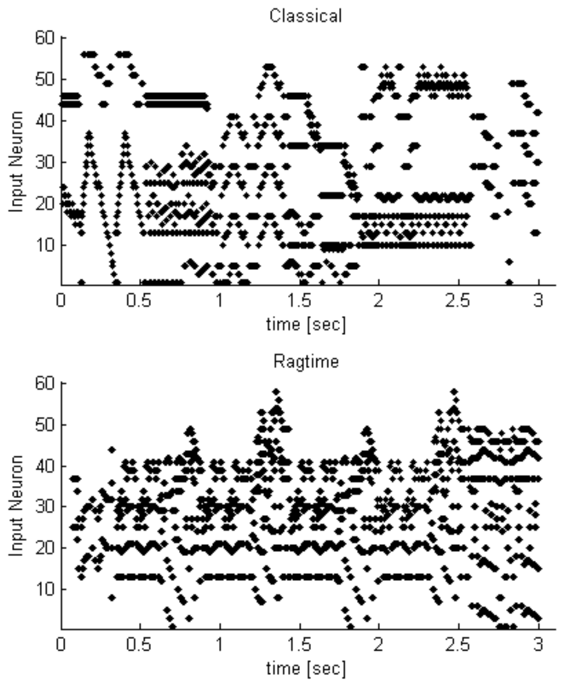

Fig. 5 An example of classical and ragtime stimulus

50 MIDI files from each style are used for training and 20 for testing, with no sharing of files. Each MIDI file is then segmented in the same way as described in the subsection above with a length of 60 seconds. The range of [L, H] is the same as the previous subsection, [$L_{natural}/20$, $H_{natural}/20$]. The performance of the LSM with different number of layers and different compression time will be studied in this experiment. As the length of the music may depend on the style, segmentation of music files may result in unequal number of instances for the two styles in the training and testing data sets. To prevent bias, we used an equal number of segments for each style: 123×2 for training and 51×2 for testing.

To demonstrate the real-time classification capability of the LSM, we designed another experiment, in which we concatenate two randomly selected music segments from each style into one long segment. Each segment is compressed to 0.3 seconds long because we find this compression time gives the best performance in the classification of single segment discussed above. Thus, the trained LSM is tested on the concatenated segments having 0.6 seconds long. The concatenated segment has the sequence of either classical-ragtime or ragtime-classical. 50 such segments are generated from each sequence for testing.

In this experiment, there are two challenges for LSM: First, it has never seen any of the testing segments before and during training, and there is no sharing of files between testing and training data sets. This is a common criterion for all the experiments discussed in this paper. Second, LSM has to report immediately when the music style changes.

III. Results and Discussion

Single readout neuron is used for all the experiments, and it is responsible to output a decision value every time the state of the filter is sampled. Linear regression is used for training the readout neuron because of its low computational complexity. The readout trained by linear regression can output floating value, instead of binary, which could be thought as a measure of similarity between the input stimulus and the class that the readout represents.

A. Music and Non-Music Classification

The readout neuron is trained to output 1 if the input stimulus is generated from a music segment; otherwise it is trained to output 0. The decision is made by averaging the readout's outputs from sampling of the liquid filter's states. When the input segment reaches the end, the decision is music if the average value is greater than 0.5, or non-music if smaller than 0.5.

Figure 6 illustrates the output of the readout for a given music stimulus. The red dots are the target values during training. It can be seen that the majority of the outputs of the readout neuron lies above 0.5, thus, the average value of all the data points is greater than 0.5 in this case and the correct classification is made.

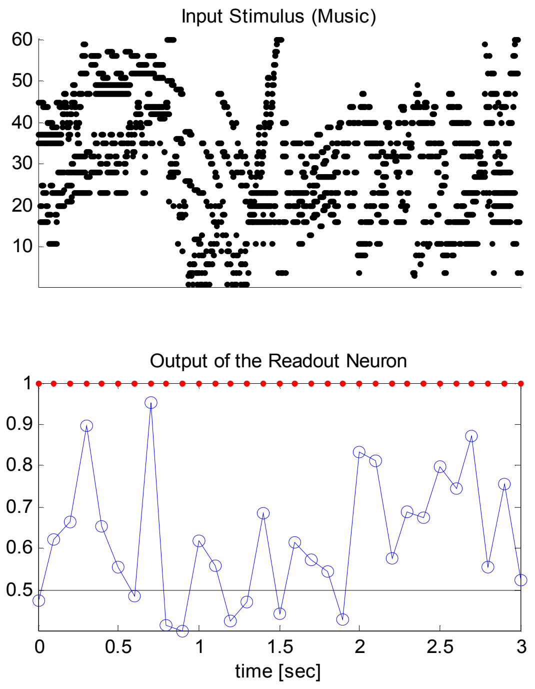

Fig. 6 Outputs of the readout in response to a music stimulus

The results of experiments employing different compression time, while keeping all the other parameters unchanged, are shown in Figure 7. The number of liquid filter state samples is fixed to 30 for every stimulus in training and testing data sets, because high sampling rate of the LSM states or large training data set will cause out of memory error in Matlab. Each data point plotted in this figure, i.e. the performance vs. compression time, is the average value of 50 runs. The music vs. non-music classification seems to be independent of the randomly-initialized connections in the LSM, as there is no significant outlier in the results for all the runs. Compression of 0.3 and 0.9 seconds result in an admirable high percent correct (82.8% and 82.1% respectively) during testing. This figure indicates that long inter-spike interval will generally lead to poor performance, which may be due to the low frequency spikes that lack the ability to excite the network at a minimal level, and hence, weaken the LSM fading memory effect. In addition, shorter compression time means shorter input stimulus and faster playing-speed, which can effectively utilize the fading memory in the LSM. However, if the frequency of spikes is too high and the compression time is too short, the LSM will not be able to grasp the temporal feature very well, as shown in the case of 0.1 second in the figure.

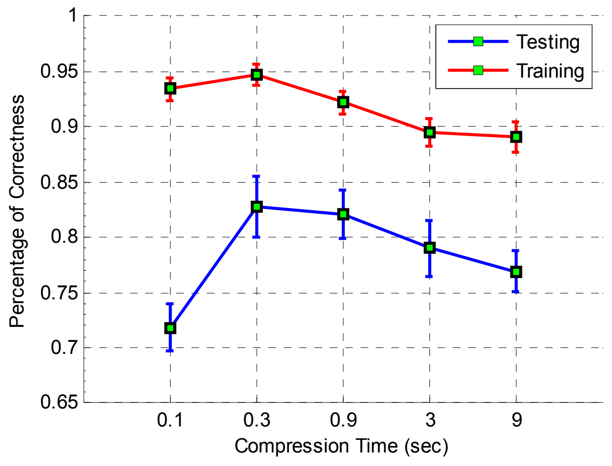

Fig. 7 Percentage of correctness in training and testing vs. compression time (The error bar at each data point represents the standard deviation.) for music and non-music classification

TABLE II
Number of Wrongly Classified Segments

Compression Time	0.1	0.3	0.9	3	9
Avg. No. Of Wrongly Classified Non-Music	62.22	43.98	48.38	44.28	40.12
Avg. No. Of Wrongly Classified Music	37.42	17	14.98	30.02	41.52
Mean Difference	24.8	26.98	33.4	14.26	-1.4

Table II lists the average number of wrongly classified instances on testing set for different compression time. We observed that the number of non-music segments that were wrongly classified is larger than music during testing, especially for the trials with a high percentage of correctness. This is because music has a highly organized form, while the generated non-music pieces involve randomness. A random pattern, after being projected by the LSM into high dimensional feature space, would maintain its random character and tend to be distributed evenly in feature space. This makes the decision boundary between music and non-music ambiguous in the feature space. Therefore, using linear regression in readout neuron will give relatively poor performance for non-music segments. If the trained classifier is weak, there would be smaller difference between the numbers of wrongly classified instances of the two styles due to the ambiguous boundary.

B. *Music Style Classification*

The readouts are trained in the same way as music vs. non-music classification, reporting one when the input is classical music, zero otherwise. Classification performance is investigated for different compression time as well as for the liquid filter with different number of layers, while keeping all the other parameters unchanged. The performance is again evaluated by averaging 50 runs for each data point as plotted in Figure 8. It can be seen that the percentage of correctness drops as the compression time increases. This agrees with the results obtained from music and non-music classification. With 10 layers and 0.3 second compression time, LSM achieves best performance for the testing set, 94.08% average success rate with the standard deviation of 2%. For the training set, the LSM learns almost perfectly, nearly 100% correct, by using only 5 layers for 0.3 and 0.9 seconds compression. The performance on the testing set starts to drop when less than 6 layers are used. When using single layer, the result on testing set goes down to 77.8% for 0.3 second compression. For the case of 3 seconds, the performance is not as good as the others, only getting 59.57% correct by using single layer.

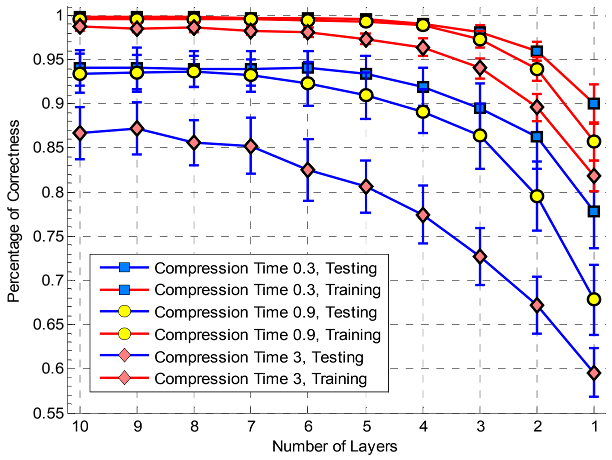

Fig. 8 Percentage of correctness in training and testing vs. number of layers by using different compression time (mean +/- standard deviation) for music style classification

Figure 9 (A) plots an example of the readout's output in the experiment of concatenating two segments. It can be clearly seen that the output values increases above 0.5 after 0.3 seconds in the ragtime-classical case. By integrating the first and second half of the readout's output values, the LSM is able to figure out the style sequence correctly at the percentage of 81.31% on average of 100 runs with the standard deviation of 5.18%.

The memory trace in the LSM resulting from the first half of the stimulus does affect the classification of the second half. To illustrate this, we firstly generate 50 classical-ragtime instances by randomly concatenating segments from each style, and then create another 50 instances by reversing the order of the two segments, as the example shown in Figure 9, in which (B) ragtime-classical is created by reversing the order of the two segments in (A). In this example, in the first 0.3 second of (A), the readout can classify correctly, but failed in the last 0.3 second of (B). Such two sets of 50 long segments are put together to form a testing set, and we tested the LSM for 500 runs. The result shows that, the LSM can correctly classify the first 0.3 second at the percentage of 91.91% ± 3.11%, while 85.91% ± 3.43% for the last 0.3 second. This strongly implies that the classification of the second half of the stimulus is affected by the memory from the first half.

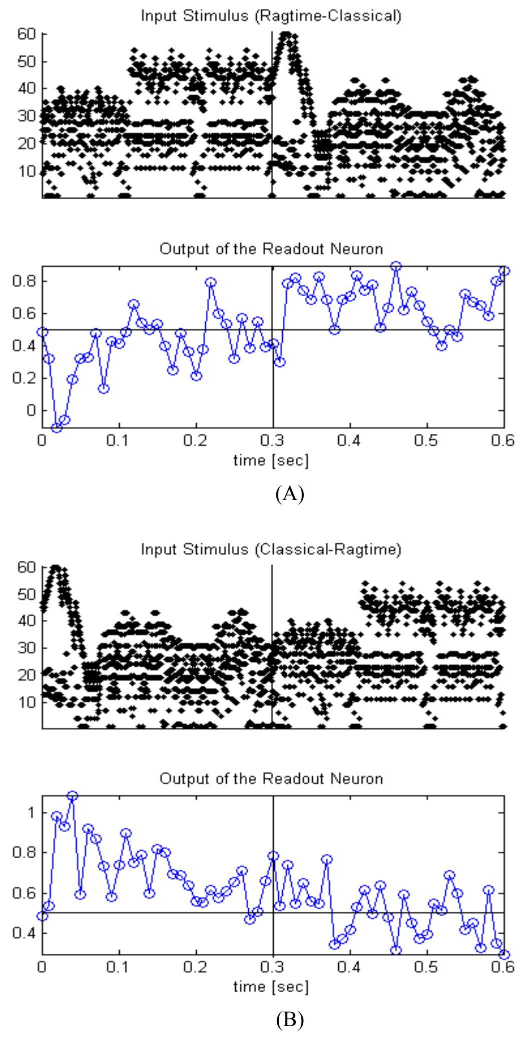

Fig. 9 An example of classifying a concatenated music segment: (A) Ragtime-Classical (B) Classical- Ragtime, generated by swapping the order of the two segments in A

C. *Comparison with other methods*

For comparison, Recurrent Neural Network (Elman Network) and K-Nearest Neighbourhood (KNN) classifiers are implemented to perform the music style classification task.

The training and testing data sets are the same as the one used in LSM. Here, each segment of music is "re-sampled" every 0.15 second. In other words, at every sampling point, the piano keys pressed at that time are represented by a vector that contains the corresponding pitch number. Note that multiple keys may be pressed at the same time. Thus, each music segment with 60 seconds' length is represented by 400 vectors, each containing the keys that are pressed at that sampling time.

Recurrent neural network is capable to classify data sequences as its recurrent connections allow it to have memory and maintain certain "states". The Elman network is a two-layer recurrent network that can be trained to recognize temporal pattern. Here we use the standard Elman network implementation provided by the Matlab neural network toolbox. The network is trained for 200 iterations to output 1 for classical music and 0 for ragtime. After adjusting the number of hidden neurons and trying various types of the output neuron, the best performance is obtained by using 120 hidden-layer tangent sigmoid neurons and one linear output neuron. The average accuracy over 100 runs is 92.62% (standard deviation 3.93%) on the testing data set, which is about 1.5% less than the proposed LSM, and 94.09% (standard deviation 3.08%) on training, 5.5% less than the LSM. The standard deviation of the Elman network is also larger. Besides this, compared with the LSM with 0.3 second compression time and 10 layers, the training of the Elman network takes much longer time.

KNN is a classifier that classifies a testing data based on its neighbours. It searches the whole training data set to locate the testing data's k nearest neighbours, and the label of that testing data is assigned to the class that occurs most frequently among these k neighbours. Here, the k neighbours are selected based on the Euclidian distance. The highest accuracy on the testing data by using KNN is 73.53% when K=2, which is not as good as the LSM and the Elman network.

IV. CONCLUSIONS

In this work, we have developed a biologically realistic LSM system to classify two styles of music (Classical vs. Ragtime), and music versus 'non-music' with disordered note sequence. The Elman network and KNN classifiers are also implemented for comparison purpose. The result shows that the proposed LSM system with 10 layers and 0.3 compression time has the best performance. The classifiers are trained and tested in only hundreds of samples. Higher success rate will be expected if more data are available.

LSM's excellence in music classification has been demonstrated in all the experiments discussed above, particularly in the segments concatenation experiment. Even though in this paper we increased the playback speed of the music by reducing the inter-spike intervals, classification in normal playing speed with the same performance can be easily achieved by adjusting parameters in LSM as it is just scaling operation.

From the experimental results, it can be seen that the performance of music style classification will not be largely affected by using only 5 layers (300 neurons). With 2 layers the performance is still satisfactory. As the LSM is a universal computer, simply adding more readout neurons and adjusting the memory-less readouts can generalize our binary classification to a multi-class problem. This great flexibility is an impressive merit of the LSM.

ACKNOWLEDGMENT

We would like to thank Bernd Krueger's website and Warren Trachtman's website for providing the data.

REFERENCES

[1] S. Ghosh-Dastidar and H. Adeli, "Spiking Neural Networks", *International Journal of Neural Systems*, vol. 19, No. 4 (2009) 295-308.

[2] W. Maass, T. Natschlager, H. Markram, "Real-time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations", *Neural Computations,* vol. 14, 2531-2560, 2002.

[3] W. Maass, H. Markram, "On the computational power of circuits of spiking neurons", *Journal of Computer and System Sciences*, 69, pp. 593-616, 2004.

[4] J. Foote, "An Overview of Audio Information Retrieval", *ACM-Springer Multimedia Systems*, 1998.

[5] Pedro J. Ponce de Leon and Jose M. Inesta, "Pattern Recognition Approach for Music Style Identification Using Shallow Statistical Descriptors", *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 37, No. 2, 2007.

[6] Dannenberg, Thom, and Watson, ``A Machine Learning Approach to Musical Style Recognition" in *1997 International Computer Music Conference*, International Computer Music Association (September 1997), pp. 344-347.

[7] B. Thom, "Unsupervised Learning and Interactive Jazz/Blues Improvisation", in Proc. of AAAI 2000, pp. 652-657.

[8] Leo Pape, Jornt R. de Gruijl, and Marco Wiering, "Democratic Liquid State Machines for Music Recognition", Speech, Audio, Image and Biomedical Signal Processing using Neural Networks, Volume 83b, Berlin, 2008.

[9] T. Natschläger, H. Markram, W. Maass, Computer models and analysis tools for neural microcircuits, in: R. Kötter (Ed.), A Practical Guide to Neuroscience Databases and Associated Tools, Kluwer Academic Publishers, Boston, 2002, Ch. 9.

[10] T. Natschläger, "CSIM: A Neural Circuit Simulator", available at: http://www.lsm.tugraz.at/csim/index.html , 2008.

[11] Eliasmith, C. and Anderson, C. *Neural engineering: computation, representation and dynamics in neurobiological systems*. Cambridge, MA: MIT Press. pp. 81-89. 2004.

[12] Joshi, P., "From memory-based decisions to decision-based movements: A model of interval discrimination followed by action selection". *Neural Networks, 20, pp.* 298-311. 2007.

[13] Gupta, A., Wang, Y., and Markram, H., "Organizing Principles for a diversity of GABAergic interneurons and synapses in the neocortex." Science, 287, 273-278, 2000.

[14] Bernd Krueger, Classical Piano MIDI Files, 1996. Available at: http://piano-midi.de/midi_files.htm

[15] Warren Trachtman, Ragtime Piano MIDI Files, 1995. Available at: http://www.trachtman.org/ragtime/

[16] Eerola, T. & Toiviainen, P. (2004). *MIDI Toolbox: MATLAB Tools for Music Research*. University of Jyväskylä: Kopijyvä, Jyväskylä, Finland. Available at: https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/miditoolbox/