RADIAL/ELLIPTICAL BASIS FUNCTION NEURAL NETWORKS FOR TIMBRE CLASSIFICATION

Tae Hong Park
Tulane University
Music Department
102 Dixon Hall
New Orleans, LA 70118
USA

Perry Cook
Princeton University
Computer Science Department
and Music Department
Princeton, NJ 08544
USA

ABSTRACT

This paper outlines a RBF/EBF neural network approach for automatic musical instrument classification using salient feature extraction techniques with a combination of supervised and unsupervised learning schemes. 829 monophonic sound examples (86% Siedlaczek Library [2], 14% other sources) from the string, brass, and woodwind families with a variety of performance techniques, dynamics, and pitches were used for the development of feature extraction, network initialization algorithms, and training of the neural networks resulting in approximately 71% individual instrument and 88% instrument family classification. A novel approach for automatically fine-tuning the system using the Nearest Centroid Error Clustering (NCC) method which determines a robust number of centroids is also discussed.

1. INTRODUCTION

Various flavors of artificial neural networks have been used in situations where complex problems are difficult to solve with a von Neumann type approach. Automatic musical timbre recognition is one of those areas where neural networks have been used to some extent. In this paper we have used 12 instruments with various playing techniques, dynamics, articulations, and pitches. The system was trained through Radial Basis Function (RBF) and Elliptical Basis Function (EBF) neural network models in conjunction with a number of new features and a novel approach for automatically finding robust number of centroids through "Nearest Centroid Error Clustering" (NCC). The system uses a standard bottom-up model with a sampling rate of 22.05 kHz and 2 second sample excerpts including the attack and steady-state portions. There were 12 classes and 829 samples used in total as shown in table 1.

Instrument	Examples	Instrument	Examples
Elec. bass	10	Oboe	55
Violin	105	Bassoon	35
Cello	102	French horn	56
Viola	75	Trumpet	78
Bb clarinet	100	Trombone	82
Flute	99	Tuba	32

Table 1. Instruments and number of samples

Also, pizzicato, spiccato, sordino, vibrato/non-vibrato, long/sustained/short, detaché, espressivo, pianissimo, piano, mezzo-forte, forte, and fortissimo samples were present for the majority of the samples with pitches between 1~3 octaves.

2. OVERVIEW OF FEATURE EXTRACTION

2.1. Feature Extraction

Feature extraction techniques both in time and frequency-domain were utilized. Harmonics were automatically picked using a custom algorithm extracting the first 10 harmonics of a sample [20]. The 12 features applied in testing the RBFN/EBFN were spectral shimmer, spectral jitter, spectral spread, spectral centroid, LPC noise, inharmonicity, attack time, harmonic slope, harmonic expansion/contraction, spectral flux shift, temporal centroid, and zero-crossing rate. Some of the new features developed are briefly discussed below (see [20] for details).

2.1.1. Harmonic Expansion/Compression

A single number describing the expansion and contraction of the harmonics structure of a sample as shown in figure 1. This feature can be observed in plucked strings for example (see [20] for details) .

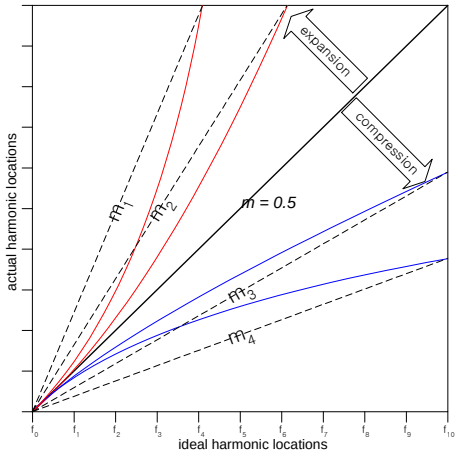

Figure 1. Harmonic expansion/compression

2.1.2. *LPC-Based Noise Content Analysis*

This is a feature utilizing the residual from the LPC (8th order was used) predicted signal and the original signal, and computing the SNR between the residue and original signal.

$$e[n] = s[n] - \hat{s}[n] = s[n] - \sum_{k=1}^{k=p} a_k s[n-k] \quad (1)$$

$$SNR = 10\log(\frac{P\{s\}}{P\{e\}}) \quad (2)$$

$$P\{s[n]\} = \sum_{n=0}^{N-1} s^2[n] \quad (3)$$

2.1.3. *Harmonic Slope*

Slope between the strongest harmonic and the weakest harmonic magnitude values. The idea was to come up with a single number to represent the spectral envelope.

2.1.4. *Harmonic Flux Shift*

This method divides the STFT frames into two groups – the 1st group reflecting the attack part and the 2nd group reflecting the steady-state portion. The difference of the group's 2-norms was computed and used to describe a spectral flux shift.

3. RBF/EBF NEURAL NETWORK

3.1. Why RBFN/EBFN?

RBFNs have found popularity in pattern classification in areas such as speech recognition and prediction [19, 16, 3], phoneme recognition [1], and face recognition [21, 22, 14]. Interestingly, RBFs have especially had noticable results in the vision community where one experiment showed 1.92 error percentage [7] while in another study using a modified RBFN, the detection rate was 99.25% ~ 100% [14]. Considering the impressive results RBFNs have had in face recognition applications, coupled with the fact that there have been no studies made with EBFNs in timbre classification and only one study with RBFNs [6] that we know of it may be meaningful to investigate its possibilities for musical instrument timbre classification.

3.2. RBFN/EBFN Characteristics

The inputs of RBFN/EBFN are directly connected to each basis function and the output of the activation functions are then weighted and summed. This is unlike the common Multi-Layered Perceptrons (MLP) which have linear basis function architectures with inputs weighted before being summed and have sigmoidal or step activation functions. RBFN/EBFNs take non-linear input spaces and output linear activation outputs through a single hidden layer. Using inherent nonlinear approximation properties, RBFNs/EBFNs have the capability to model very complex patterns, which the MLPs can only achieve through multiple intermediary hidden layers [12]. RBFN/EBFNs also have faster learning capacity, are easier to implement, are less complex in structure, and are computationally more efficient than MLPs.

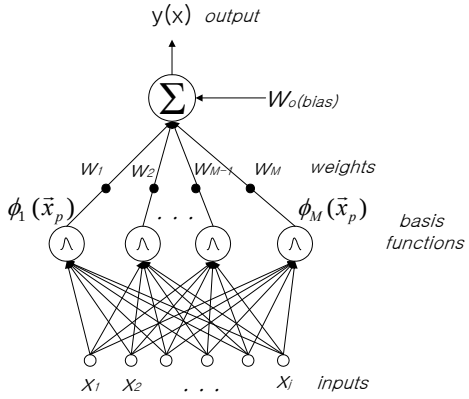

Figure 2. Basic RBFN/EBFN

RBFN/EBFNs are built on unique centroids (means), spreads (standard deviations from means), and activation functions. As with MLPs, weights are adjusted during training but in addition, the spreads and centers of each cluster are also updated. For feature spaces in 2 dimensions a circular cluster is formed for RBFs and elliptic cluster for EBFs; 3-dimensional spaces result in spherical clusters for RBFs and ellipsoids for EBFs; dimensions greater than 3 results in hyperspheres and hyperellipsoids for RBFs and EBFs respectively. RBFs are special cases of EBFs where diagonal covariances matrix are equal.

3.3. RBFN/EBFN Activation Functions

In this paper a Gaussian type activation function was used. As we can observe from equation 5, figure 3, and figure 4 as input samples are further away from the mean (C_1/C_2 in figure 3, μ in figure 4), the activation output decreases exponentially. Hence patterns located at large distances from the mean (cluster centers) will fail to activate a particular basis function while maximum activation is achieved by data samples closest to a cluster's mean. Each cluster has its own Gaussian distribution, mean, and spread.

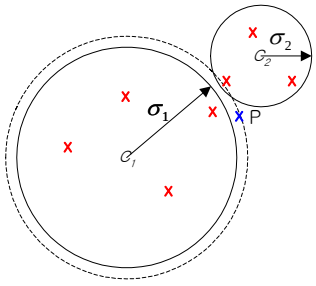

Figure 3. RBF clusters with width σ_1 and σ_2

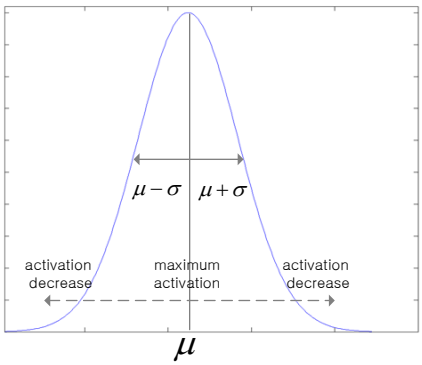

Figure 4. Gaussian normal distribution

The RBF basis function is defined by Euclidian distance *r* and activation function by $\phi(.)$, where x_p is the input sample number *p*, μ_j is the mean for cluster *j*, and *N* is the dimension of input vector x_p.

$$r_{Euclidian} = \|x_p - \mu_j\| = \sqrt{\sum_{n=1}^{N}(x_{np} - \mu_{nj})^2} \quad (4)$$

$$\phi_j(x_p) = e^{-r_{Euclidian}^2 / 2\sigma^2} \quad (5)$$

For EBFs the Mahalanobis distance is used for distance computation. For the EBF activation function we have:

$$r_{Mahalanobis} = \sqrt{(x_p - \mu_j)^T \Sigma_j^{-1} (x_p - \mu_j)} \quad (6)$$

$$\phi_j(x_p) = e^{-r_{Mahalanobis}^2 / 2} \quad (7)$$

Figure 5 illustrates the same pattern space as figure 3 but with the two classes having different spread characteristics. Class 1 uses an elliptical spread, and class 2 the original radial spread. It can be seen that for the EBF-based cluster, class membership of patterns are more flexible and sample *P* is given membership to C_1 with greater elasticity.

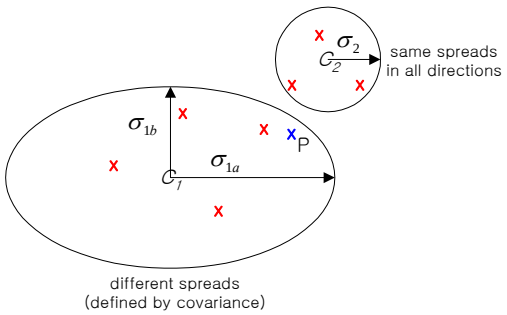

Figure 5. EBF/RBF clustering patterns

3.4. Network Updating

For network training the backpropagation algorithm was applied which uses the delta rule method along with gradient descent. The general gradient type learning formula for each layer is given by:

$$w_{ij}[n] = w_{ij}[n-1] + \Delta w_{ij}[n] \quad (8)$$

The objective is to adjust the weights Δw_{ij} to minimize the error *E* computed as the least-squares-error:

$$E = \frac{1}{2}\sum_{p=1}^{P}\sum_{i=1}^{N}[d_i^{(p)} - y_i^{(p)}(L)]^2 \quad (9)$$

$d_i^{(p)}$ is the target, $y_i^{(p)}$ the actual network output, *L* is the topmost output layer number of a general multi-layer network, *P* the number of training patterns, *p* pattern index, and *N* the dimension of the output space. The weight change Δw_{ij} along with a learning rate scalar η and network error *E* can then be expressed as:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \quad (10)$$

By using gradient descent and Chain Rule approach for the weights, covariance/standard deviations, and centers respectively we get the following update equations for our RBF/EBF networks:

$$e(x^{(p)}) = \frac{1}{2}\sum_{j=1}^{J}\{d_j^{(p)} - y_j(x^{(p)})\}^2 \quad (11)$$

$$w_i(t+1) = w_i(t) + \eta_w \sum_{p=1}^{N} e(x^{(p)})\Phi_i(x^{(p)}) \quad (12)$$

$$w_i(t+1) = w_i(t) + \eta_w \sum_{p=1}^{N} e(x^{(p)}) \quad \text{when } i = 0 \quad (13)$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \eta_\sigma \sum_{p=1}^{N} e(x^{(p)}) w_i \Phi_i(x^{(p)}) \frac{\left(x_j^{(p)} - c_{ij}\right)^2}{\sigma_{ij}^3(t)} \quad (14)$$

$$c_{ij}(t+1) = c_{ij}(t) + \eta_c \sum_{p=1}^{N} e(x^{(p)}) w_i \Phi_i(x^{(p)}) \frac{\left(x_j^{(p)} - c_{ij}\right)}{\sigma_{ij}^2(t)} \quad (15)$$

e is error, *d* target, *y* actual output, *p* pattern index; w_i weights with center index i; η_w, η_σ η_c learning rates for weights, variance, and centroids; Φ_i activation; σ_{ij} the standard deviations; *t* time index; and c_{ij} the mean for centroid i's j^{th} dimension.

3.4.1. Network Training and Initialization

Network training is divided into two stages. The 1st stage consists of guessing initial parameters for the means, weights, covariance, and standard deviations. The second stage iteratively trains the network and updates parameters using equations (11) ~ (15).

In this paper we have used the k-means method to compute the initial parameters. To obtain the initial

weights w, $\frac{\partial E}{\partial w} = 0$ is used to solve for w with respect to the total error squared E yielding (A is the activation output, d is known output):

$$w = (A^T A)^{-1} A^T d \qquad (16)$$

During the computation of the network's initial means, spreads, and weights , problems often arise with inverse matrix computation. That is, issues with singular or near singular matrices arise, especially as the number of centroids increase. Generally speaking, more centroids achieve better results for a particular pattern space but at the same time also results in increased specificity leading to loss of generality. The size of centroids and instability is closely related to the inverse matrix operation $(A^T A)^{-1}$. This is in part due to centroids with very small spreads. Singular value decomposition [11] estimation was applied to improve this issue. EBFNs are particularly prone to singularity problems due to the additional requirement of the inverse covariance matrix in the Mahalanobis distance algorithm. The cause of instability is again some centroids' spreads becoming too small or even singular. After trying out different approaches for improving the singularity issue it turned out that the best way (at least in this study), and consequently the simplest way to get more stable performance was identifying and purging centroids with extreme spreads (small and large) during the initialization phase, generally small in number.

3.5. Further Fine-Tuning: Nearest Centroid Error Clustering (NCC)

At the end of the day, neural network classification is improved through fine-tuning. In this study the final stage for further fine-tuning after k-means was achieved through NCC. The idea basically exploits the observation that errors occur between class boundaries. Figure 6 shows misclassified data (synthesized) in a 3-class system (001, 010, 100) after k-means and network training.

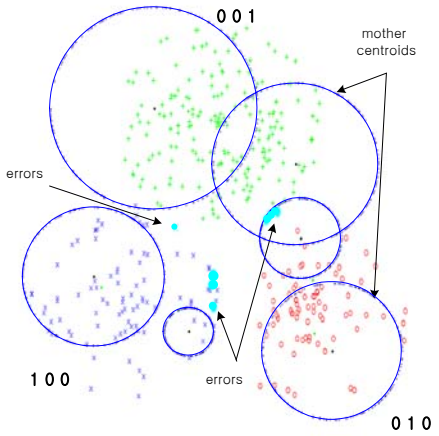

Figure 6. Initial training with a 6 centroid RBFN

For this problem the initial training (k-means + training) does a good job in the classification task (94%) with errors occurring in areas where class boundaries overlap or are close together. By placing additional smaller centroids at those problematic locations automatically, it is possible to improve the performance of the network after further training – an additional fine-tuning and training stage concentrating on localized regions with smaller centroids. This in essence is the basic idea of the NCC method. The algorithm determines the locations of "problematic areas" using information from "mother centroids" (original 6 mother centroids for this example) and spawns new smaller children centroids nearest to a particular mother centroid. The mother centroids are used as guides as they are already "roughly tuned" to a particular pattern space. Although blindly increasing the number of centroids is an option for better performance, there is no consideration of the error feedback reflecting the pattern space. The NCC method on the other hand achieves rapid and more consistent increase in performance utilizing learned information about the pattern space – automatically.

As seen on the left of figure 7 the error pattern and mother centroid which render the minimum distance is initially chosen. Once the error pattern p and mother centroid j is selected, the new child centroid pertaining to pattern p inherits the mother's spread σ_j. The mother's spread σ_j is then used to find any encompassing error pattern neighbors satisfying the general case hyperellipsoid (17). If any error pattern members are found within σ_j, a new spread/center is computed via the arithmetic mean of its members (18). In the above example, the new child centroid has one "sibling." On the other hand if there is only a single member within spread σ_j, the new child's spread is just scaled linearly to decrease its span of influence through (19).

$$\left(\frac{(q_m^{(1)} - q_p^{(1)})}{\sigma^{(1)}}\right)^2 + \left(\frac{(q_m^{(2)} - q_p^{(2)})}{\sigma^{(2)}}\right)^2 + \ldots + \left(\frac{(q_m^{(N)} - q_p^{(N)})}{\sigma^{(N)}}\right)^2 \le 1 \qquad (17)$$

$$\boldsymbol{\mu}_p = \frac{1}{M}\sum_{m=1}^{M} \mathbf{e}_m, m \in \{members\ \ of\ \ new\ \ child\ \ centroid\ \ p\} \qquad (18)$$

$$\sigma_p = \alpha \cdot \sigma_j^{mother}, \text{ where } 0 < \alpha < 1 \qquad (19)$$

This process is repeated and new centroids with reduced spreads and new "error-pattern-influenced" centers are obtained automatically. In order to resume training the network with the additional children centroids has to be re-initialized. The weight re-initialization was achieved via (16). Figure 9 shows the final centroid configuration after NCC and retraining at 100%.

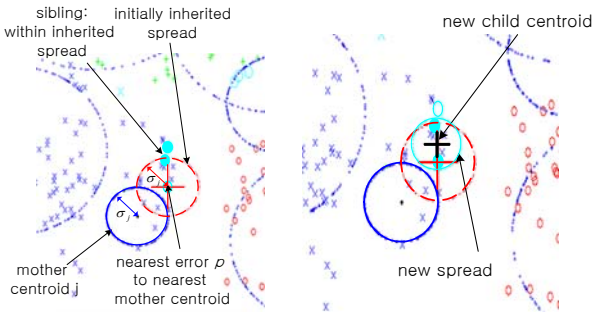

Figure 7. Spawning a new child centroid

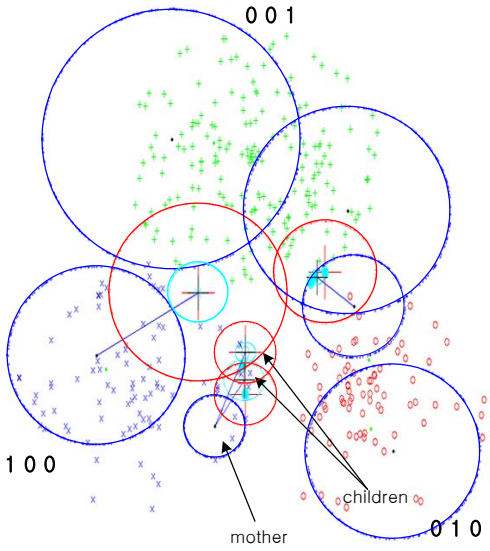

Figure 8. Spawning of new centroids

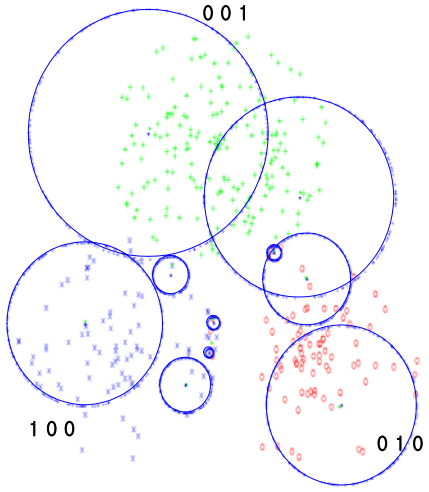

Figure 9. Final centroid configuration: 100%

This additional fine-tuning method could be applied a number of times until a desired performance is achieved. However, due to the mechanics of the algorithm, there will usually be an increase in the number of total centroids, resulting in over-fitting issues. A partial solution to over-fitting was to make it optional whether to include "single-member" children centroids as these tend to address only very localized error patterns. This methodology not only lessened the overall increase of centroids (generally a good idea), but also only included those new children centroids that had at least two members associated with it.

4. CLASSIFICATION RESULTS

The feature extraction algorithms were thoroughly tested with all of the samples and verified manually whenever possible. As we see below (table 2, 5) certain feature combinations worked better than others. To determine the best feature sets for a particular network, a process starting out with all 12 features and eliminating those features that gave poorest results was applied until no further improvements were made. The system's classification performance and training was conducted using 80% of the total 829 samples for network training and the remaining 20% of samples for cross-validation. Each new training/classification session was subjected to a pattern shuffling scheme. Family and individual instrument classification and training was done separately with independent RBFN/EBFNs.

4.1. Instrument Family Classification

For the family recognition task three families were used – strings, woodwinds, and brasses. As we see in table 4 and 5 system performance without cross-validation tends to be about 10% higher. This is likely a case of pattern over-fitting most noticeable with the all inclusive NCC algorithm. The best performance for family recognition was approximately 88% for RBFNs after NCC. Figure 10 shows the confusion matrix run on all the data samples.

Cross-Validation %	1.Shimmer	2.Jitter	3.Spectral Spread	4.Spectral Centroid	5.Inharmonicity	6.Attack Time	7.Harmonic Slope	8.LPC noise content	9.Harmonic Expansion/Compression	10.Spectral Flux	11.Temporal Centroid	12.Zero-Crossing Rate
88	o	o	o	o			o	o	o		o	o
85	o	o	o	o		o		o	o		o	o
84	o	o		o		o	o	o	o	o	o	o
83	o		o	o	o	o	o	o	o	o	o	o
82	o	o	o	o			o	o			o	o
82	o	o	o	o				o	o			
71	o	o		o	o	o	o	o	o	o	o	o
65	o	o	o									
61	o	o	o									

Table 2. Feature set selection for instrument family

	RBFN		EBFN	
	Normal	NCC	Normal	NCC
Correct (%)	78	97	81	92
# of features	9	9	9	9
#of centroids	50	132	50	124
# of epochs	2000	1500	2000	1500

Table 3. Training for families without cross-validation

	RBFN		EBFN	
	Normal	**NCC**	**Normal**	**NCC**
Correct (%)	73	88	85	85
# of features	9	9	9	9
#of centroids	45	54	40	57
# of epochs	2000	1600	9000	7000

Table 4. Family recognition with cross-validation

Figure 11 shows the binary version of the confusion matrix (correct or incorrect). From the graphs we can see that the least confusion for string instrument classification was with brasses and a little confusion existed in the case of the network misclassifying some instruments as woodwinds.

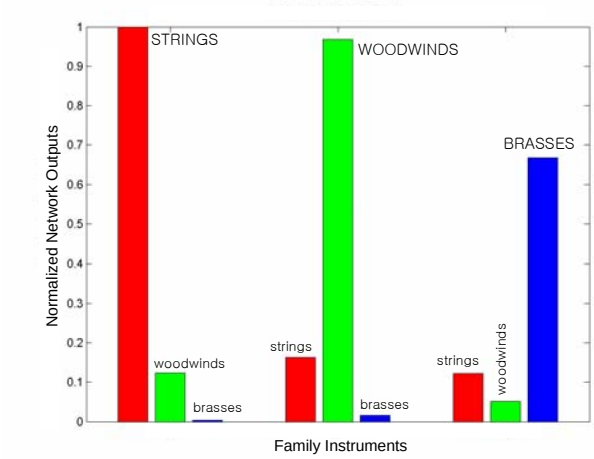

Figure 10. Confusion matrix for instrument family

The last group in figure 10 reflects the confusion of the system when reacting to brass instruments. In the brass section, we can note that the network had noticeable confusion with string instruments but also substantial errors with woodwind instruments. The French horn was the main cause for performance degradation for family classification and also individual instrument recognition as we will see below.

	Strgs.	**Wwinds**	**Brasses**	**%**
Strgs.	**278**	10	4	**96**
Wwinds	24	**261**	4	**90**
Brasses	22	24	**202**	**81**

Figure 11. Confusion matrix for instrument family

4.2. Individual Instrument Classification

Tables 6 and 7 summarize the 12 individual instrument classification results. As in family classification, training without cross-validation resulted in superior system performance. However, with cross-validation the best performance for RBFNs was approximately 71% and 67% for EBFNs.

Cross-Validation %	1.Shimmer	2.Jitter	3.Spectral Spread	4.Spectral Centroid	5.Inharmonicity	6.Attack Time	7.Harmonic Slope	8.LPC noise content	9.Harmonic Expansion/Compression	10.Spectral Flux	11.Temporal Centroid	12.Zero-Crossing Rate
71		o	o	o			o	o		o	o	o
69		o	o	o		o	o	o		o	o	o
69	o	o	o	o		o	o	o		o	o	o
68	o		o	o				o	o	o	o	o
66		o	o	o		o	o	o	o	o	o	o
65	o	o	o	o		o	o	o		o	o	
65	o	o	o	o		o	o	o			o	o
64	o		o	o	o	o	o	o	o	o	o	o
62	o		o	o			o				o	
56	o		o	o			o	o	o	o	o	o
44	o		o	o	o	o	o	o	o			

Table 5. Feature set selection for individual instrument

	RBFN		EBFN	
	Normal	**NCC**	**Normal**	**NCC**
Correct (%)	70	95	78	97
# of features	8	8	8	8
#of centroids	45	214	35	199
# of epochs	3000	2000	3000	2000

Table 6. Training for individual instruments without cross-validation

	RBFN		EBFN	
	Normal	**NCC**	**Normal**	**NCC**
Correct (%)	57	71	63	67
# of features	8	8	10	8
#of centroids	45	59	38	74
# of epochs	10000	8000	10000	8000

Table 7. Individual instrument recognition with cross-validation

Figure 12 shows the binary confusion matrix (soft confusion matrix could not be fitted into this paper, see [20] for details). For brasses such as the trumpet, the network confused it mostly with the clarinet. French horn classification was again the poorest at 32%. The best results were for the tuba (last row) at 91% with most of its errors within its own family and the electric bass. Interestingly, for the brass instruments more than the usual cross-family error observations can be made (excluding the tuba).

5. DISCUSSION

The initial studies show that the system recognizes instrument families with fewer errors than individual instruments, which is congruent with human performance tendencies. This is hardly an unexpected discovery, but nevertheless a confirmation that the system is generally behaving in a predictable manner.

The best performance for family instruments was approximately 88 percent using RBFNs with feature set shimmer, jitter, spectral spread, spectral centroid,

harmonic slope, LPC noise content, harmonic expansion/contraction, temporal centroid, and zero-crossing rates. EBFNs did not do quite as well. However, EBFNs were not exhaustively tested due to time constraints imposed by the additional computational load and instability of the EBFNs developed in Matlab. The majority of the instability for the EBFNs had to do with singularity or near-singularity issues discussed above.

	eb.	vln.	vcl.	vla.	clar.	flute	obo.	bsn.	horn	trp.	trb.	tuba	%
eb.	9				1								90
vln.		83	4	9	4	3	1			1			79
vcl.	1	8	75	10	1		2			1	4		68
vla.		14	16	44				1					59
clar.		4	1		82	11	1			1			82
flute		7	3	2	9	73	4		1				74
obo.		4			9	4	38						69
bsn.			3		1	1		29			1		85
horn	1	3	5		1	6	10	3	18	2	2	5	32
trp.		1	3	4	2	1				67			86
trb.		4	5	1	1	5	1	1	4	7	53		65
tuba											3	29	91

Figure 12. Confusion matrix for individual instruments

New features such as harmonic slope, LPC noise content analysis, and harmonic expansion/contraction have also shown to improve classification of the neural network and as the experimental results show, it was possible to robustly increase the network's performance using NCC starting with a relatively low number of centroids and jumping to a higher centroid configuration. Blindly increasing the number of centroids without NCC generally did not result in acceptable performance and in most cases introduced instability making the higher order networks impossible to use. For individual instruments the best performance was 71% using RBFNs and 67% for EBFNs. The inferiority of the EBF networks was rather a surprise as initial tests, albeit on synthesized two-dimensional pattern spaces resulted in better performance. With careful control of centroid widths perhaps the more flexible EBFNs could be trained to produce better results.

Revisiting the confusion matrix for individual instruments it is apparent that the system has difficulty in classifying French horn timbres. However, classification for the other brass instruments was on the average better than the other families. Also, the brass instruments except for the tuba (all its errors were within its family), especially the trumpet and trombone had their errors outside their family in the strings and woodwinds groups. Although the first reaction may be that this result is a "serious network classification problem," it may not necessarily be an undesirable result. For the French horn for example, most of the confusion occurs with the oboe (10 patterns, especially with long forte notes), flute (6), and cello (5). Although further research needs to be conducted, it is generally the case that improving "cross-family" misclassifications of instruments is relatively easier than "within-family" misclassifications. However, it remains to be seen if additional features or adjusting existing features will indeed improve classification.

Additional observations were made which illustrated that classification is dependent on dynamics and techniques used (details can be found in [20]). One rather surprising result in this study shows that in a number of cases, the system's performance was better for samples that used some variation of "short-note techniques" such as staccato and pizzicato notes. The findings also implicate that the system may work well with percussive instruments especially those percussive instruments that are pitched, this however remains to be seen. Also, sordino and espressivo techniques seem to confuse the system. However, not all instruments adhered to the aforementioned trends.

Comparing the developed neural network system with other artificial systems it is possible to note some similarities and differences in performance as well as testing environment. For k-NN based models, Fujinaga [9], Fujinaga & McMillan [10], Martin & Kim [18], Eronen & Klapuri [8] reported 50.3% (1338/23), 68% (1300/23), 70% (1023/15), and 80% (1498/30) performance (samples/classes) respectively for individual instruments which are similar to the rates obtained with this neural network model (pitch information was provided in [8]). For other neural network systems such as the one used by Kaminskyj cited by Herrera-Boyer [13] a high 90-percentile performance was reported. However, the number of instruments (4) and number of samples (240) employed to evaluate the neural network seem to be less-than-ideal to confidently make an assessment. The same is true for Cemgil [6] who has had seemingly impressive results with 94~100% accuracy using only 40 samples classified into 10 classes. Kostek reported 97% for correctly classifying bass trombone, trombone, English horn, and contra bassoon [15]. However, the pitch information was provided to the system and training patterns and cross-validation patterns came from the same stereo audio file – one channel for training and the other channel for cross-validation.

The preliminary research results in this paper demonstrate that with appropriately trained RBF/EBF networks via salient features it is possible to design an ANN system to automatically recognize musical instrument sounds. The studies also show that machine performance outperforms or are similar to human counterparts in comparable testing environments – 46%~67%[17], 72%[5], 85% [4] for 27, 6, 4 instruments respectively. However, the results should be taken with a

grain of salt, as the number of different examples for each instrument was limited in breath –86% of the samples came from a single library [2], although other samples from personal collections and from the Internet were used in training and evaluating the system (clarinets, flutes, and electric bass – 14%).

6. ACKNOLEDGEMENTS

Tae Hong Park would like to thank Paul Lansky for his insights, valuable help, and support with this research.

7. REFERENCES

[1] Berthold, M. R. "A Time Delay radial Basis Function Network for Phoneme Recognition", *Proceedings of the IEEE International Conference on Neural Networks, vol. 7*, 1994.

[2] Best Service – Sounds & More. Hanauer Straße 91a, 80993 München, Germany.

[3] Birgmeier, M. "Nonlinear Prediction of Speech Signals Using Radial basis Function Networks", *EUSIPCO*, vol. 1, 1996.

[4] Brown, J.C., Houix, O., McAdams, S. "Feature Dependence in the Automatic Identification of Musical Woodwind Instruments", *Journal of the Acoustical Society of America*, 2001.

[5] Campbell, W. C., Heller, J. J. ''The contribution of the legato transient to instrument identification,'' *Proceedings of the Research Symposium on the Psychology and Acoustics of Music*, 1978.

[6] Cemgil, A. T. and Gürgen, F. "Classification of Musical Instrument Sounds using Neural Networks", *Proc. of SIU97*, 1997.

[7] Er, M., Wu S., Lu J., Toh H. "Face Recognition with Radial Basis Neural Networks", *IEEE Transactions on Neural Networks*, Vol. 13, No. 3, 2002.

[8] Eronen, A., Klapuri, A. "Musical instrument recognition using cepstral coefficients and temporal features", *Proceedings of the ICASSP*, 2000.

[9] Fujinaga, I. "Machine recognition of timbre using steady-state tone of acoustical musical instruments", *Proceedings of the ICMC*, 1998.

[10] Fujinaga, I., MacMillan, K. "Realtime recognition of orchestral instruments", *Proceedings of the ICMC*, 2000.

[11] Golub, G. H., Van Loan, C. "The Singular Value Decomposition and Unitary Matrixes", §2.5.3 and 2.5.6 in Matrix Computations, 3rd ed, Baltimore, MD: Johns Hopkins University Press, 1996.

[12] Haykin, S. *Neural Networks: A Comprehensive Foundation*, Macmillan, 1994.

[13] Herrera-Boyer, P., Amatriain X., Batlle E., Serra X. "Towards Instrument Segmentation for Music Content Description: a Critical Review of Instrument Classification Techniques", *International Symposium on Music Information Retrieva*, 2000.

[14] Huang, L., Shimizu A., Kobatake H. "Face Detection Using a Modified Radial Basis Function Neural Network", *Proceedings of the 16th International Conference on Pattern Recognition, vol. 2*, 2002.

[15] Kostek, B. "Soft Computing in Acoustics: Applications of Neural Networks, Fuzzy Logic and Rough Sets to Musical Acoustics", Physica-Verlag, 1999.

[16] Mak, M.W., Allen W. G., Sexton G. Speaker "Identification using Radial Basis Functions", *The 3rd IEEE Int. Conf. on Artificial Neural Networks*, 1993.

[17] Martin, K. D. ''Sound-source recognition: A theory and computational model',' Massachusetts Institute of Technology, Ph.D. thesis, 1999.

[18] Martin, K. D., Kim, Y. E. "Musical instrument identification: A pattern-recognition approach", *Proceedings of the 136th meeting of the Acoustical Society of America,* 1998.

[19] Niranjan, M., Fallside F. "Neural Networks and Radial Basis Functions in Classifying Static Speech Patterns", *Computer Speech and Language*, 1990.

[20] Park, T. H. *"Towards Automatic Musical Instrument Recognition",* Princeton University, Music Department, Ph.D. Dissertation, 2004.

[21] Sato K., Shah S., Aggarwal J. K. "Partial Face Recognition using Radial Basis Function Networks", *Proceedings of the 3rd International Conference on Automatic Face and Gesture Recognition*, 1998.

[22] Thomaz, C.E., Feitosa, R.Q., Veiga, A. "Design of Radial Basis Function Network as Classifier in Face Recognition Using Eigenfaces", *Proceedings of 5th Brazilian Symposium on Neural Networks*, 1998.