Author(s) Mäkinen, Toni; Kiranyaz, Serkan; Gabbouj, Moncef

Title Content-based audio classification using collective network of binary classifiers

Citation Mäkinen, Toni; Kiranyaz, Serkan; Gabbouj, Moncef 2011. Content-based Audio Classification using Collective Network of Binary Classifiers. Evolving and Adpative Intelligent Systems EAIS, 2011 IEEE Workshop, 11 - 15 April 2011, Paris, France. IEEE Workshop on Evolving and Adpative Intelligent Systems EAIS Piscataway, NJ, IEEE . 116-123.

Year 2011

DOI http://dx.doi.org/10.1109/EAIS.2011.5945911

Version Post-print

URN http://URN.fi/ URN:NBN:fi:tty-201306271275

Content-based Audio Classification using Collective Network of Binary Classifiers

Toni Mäkinen

Serkan Kiranyaz

Moncef Gabbouj

Tampere University of Technology
Department of Signal Processing
P.O. Box 553, Tampere, Finland

***Abstract*—In this paper, a novel collective network of binary classifiers (CNBC) framework is presented for content-based audio classification. The topic has been studied in several publications before, but in many cases the number of different classification categories is quite limited and needed to be fixed *a priori*. We focus our efforts to increase both the classification accuracy and the number of classes, as well as to create a *scalable network* design, which allows introducing new audio classes incrementally. The approach is based on dividing a major classification problem into several *networks* of *binary classifiers (NBCs)*, where each NBC adapts its internal topology according to the classification problem at hand, by using *evolutionary Artificial Neural Networks (ANNs)*. In the current work, feed-forward ANNs, or the so-called Multilayer Perceptrons (MLPs), are evolved within an architecture space, where a *stochastic optimization* is applied to seek for the optimal classifier configuration and parameters. The performance evaluations of the proposed framework over an 8-class benchmark audio database demonstrate its scalability and notable potential, as classification error rates of less than 9% are achieved.**

Keywords - audio content - based classification; evolutionary neural networks; particle swarm optimization; multilayer perceptron

I. Introduction

The rapid growth of the database sizes both in the Internet and home computers has created new and challenging tasks in maintaining the flexibility in handling such large amounts of data. Audio content-based retrieval offers several advantages and possibilities over traditional text-based queries, as manual annotation of audio information in large databases is not convenient or perhaps feasible at all. Moreover, in many practical situations it would be ideal to retrieve certain kind of audio content from a large database using a *reference* audio clip, for example when searching for a certain type of music or environmental sounds. For this, audio content-based classification is needed, which is studied in this work using a novel approach of collective (evolutionary) classifier networks.

The idea of content-based audio indexing and retrieval was first presented by Wold et al. in [1], where pitch, loudness, brightness and bandwidth features were used in classifying the used audio database (*Muscle Fish*). Since then, the research has been rather active, and many classification schemes have been proposed to improve the accuracy of the classification performance. In this paper, the focus is put purely on the audio *classification* problem, meaning that audio *segmentation* and *change-point detection* approaches are out of the scope of this study. The pure audio classification approaches found from the literature can be divided into two main categories, namely the *model-based* and the *rule-based* methods. The latter ones are convenient when no complete training data is available, as the classification is performed in an unsupervised manner. This is accomplished by using *thresholds* for different audio features, as performed in [2]-[4]. There are, however, several problems with the unsupervised learning techniques, such as the need for high amount of heuristics, and the limited number of classes that need to be fixed *a priori*. Such drawbacks limit their practical use for dynamic, ever-growing multimedia repositories, which are common in many environments and applications today.

In model-based methods, based on supervised learning, Chen et al. in [5] used a *Support Vector Machine (SVM)* to classify audio from two movies into 5 classes, namely music, speech, environmental sound, speech with music, and music with environmental sound. The results (~78% classification accuracy) showed improvements in classification error rates when compared to *k-Nearest Neighbour (kNN), Artificial Neural Networks (ANNs),* and *Naive Bayes (NB)* classifiers. Rather high training dataset (70% of the entire database) was used in achieving the results, whereas Zhu et al. in [6] used the same kind of SVM approach with a smaller training set and additional validation set, achieving more or less similar outcome with [5]. Chu and Champagne [7] used a slightly different approach for SVM-based classification by introducing their FFT-based noise-robust spectrum. Improved classification results were achieved in noisy test cases, but only speech, music, and noise were classified in their work. SVM was used also in [8], where it was applied to transform domain indexing by using a non-standard audio codec in a music genre-classification application. As a popular classifier, SVM was also used, along with the *Hidden Markov Models (HMMs),* in [9] to classify audio content into five non-silent classes. In [9], a unique HMM-model is trained for each non-silent class using MPEG-7 features. Training set encapsulated 50% of the entire dataset in achieving the reported accuracy rates, which are, however, highly dependent on the selection of the SVM parameters, which is a well-known fact in the field. Peeters in [10] used *Gaussian Mixture Models (GMMs)* together with the

This work was supported by the Academy of Finland, project No. 213462 (Finnish Centre of Excellence Program (2006 - 2011)

HMMs to model individual classes in the context of music genre recognition (with six categories). *Principal Component Analysis (PCA)* and *Linear Discriminant Analysis (LDA*) were needed to lower the feature space dimensionality, whereas the classification itself was done based on the generated statistical models. The classification results obtained in music genre recognition are close to the state-of-the-art, but selection of the best classifier configuration remains an unsolved problem. Another supervised classification approach was presented by Harb and Chen in [11], where modeling based on human perception was applied. An average classification accuracy of 63.5% was achieved for six music genres.

In general, the aforementioned audio classification efforts, and many alike, put lots of effort in adjusting the classifier parameters, so as to "fit" to the specific classification problem at hand as close as possible. However, it is obvious that this is not too applicable for a general classifier that should achieve robust and efficient performance levels over generic audio databases. Therefore, for any audio repository, the setting of the classifier parameters, as well as the choice of the classifier configuration, should be optimal, so as to maximize the classification accuracy. Also, the number of different classes is usually quite limited and specific to a certain audio domain, whereas in order to have a reliable retrieval performance on a versatile database, more classes should be supported. Furthermore, support for dynamic updates in the databases is rare at the moment, as in most cases the training dataset and the number of classes need to be fixed beforehand.

In order to address these problems, in this paper, we shall focus on a global and data-adaptive framework design that embodies a collective network of evolutionary binary classifiers (CNBC). The main idea of the framework was first introduced in a previous work [12], for another application area, being now specifically designed for audio classification purpose. Earlier, fundamentally similar type of approaches of constructing an ensemble of neural networks (a.k.a. *neuro-ensemble*) have been introduced (see e.g. [13]), but, to our knowledge, the framework structure presented in this paper has not been used in audio classification scheme before. The issues specifically targeted in our approach are:

- *Evolutionary Search*: Seeking for the optimum classifier network architecture among a certain collection of different configurations (the so-called *Architecture Space, AS*).
- *Evolutionary Update in the AS*: Keeping only "the best" individual configuration in the AS among indefinite number of evolution runs.
- *Class / feature Scalability*: Support for varying number of classes and audio features. A new class / feature can be dynamically inserted into the framework without requiring a full-scale re-configuration or re-training.
- *High efficiency* for the evolution (training) process: Using as compact and simple classifiers as possible.
- *Maximizing the classification accuracy:* Using several audio features to take advantage of the discrimination power of each one of them.

In this work, *Multilayer Perceptrons (MLPs)* are evolved in the proposed CNBC framework. The recently proposed *Multi-Dimensional Particle Swarm Optimization (MD-PSO)* [14] is used as the primary evolutionary search technique.

The rest of the paper is organized as follows. Section II briefly presents the applied evolutionary ANNs and the MD-PSO technique, whereas Section III describes the feature extraction process and introduces the audio features used. The proposed CNBC framework and the evolutionary update mechanisms are explained in detail in Section IV, and the classification results and performance evaluation over an 8-class database are given in Section V. Finally, Section VI concludes the paper and discusses future research directions.

II. Evolutionary Neural Networks

In this section, we will first briefly discuss the applied evolutionary technique, MD-PSO, which is used in an architecture space to search for the optimal classifier configuration. Second, the concept of evolutionary feed-forward artificial neural networks is introduced. Finally, an overview of the well-known *Back Propagation (BP)* method will be given, which can be used also exhaustively to perform a sequential search for the optimal classifier in an AS.

A. Multi-Dimensional Particle Swarm Optimization

The Particle Swarm Optimization (PSO) was introduced by Kennedy and Eberhart [15] in 1995 as a population-based stochastic search and optimization process. In a PSO process, a swarm of particles, each of which represents a potential solution to the optimization problem at hand, navigates through a search space. The particles are randomly distributed over the search space, and the goal is to *converge* to the global optimum of a function or a system. Each particle keeps track of both its current position, and the best position achieved so far in the search space. The latter is called the *personal best* value (*pbest*), while the PSO process keeps also track of the *global best* solution achieved so far by the whole swarm (*gbest*). During their journey in the search space with discrete time iterations, the *velocity* of each particle in the next iteration is computed by the best position of the swarm (position of the particle *gbest*, the *social* component), the best personal position of the particle (*pbest*, the *cognitive* component), and the current velocity of the particle (the *memory* term). Both *social* and *cognitive* components contribute randomly to the position of the particle in the next iteration.

In this research we will use the multi-dimensional (MD) extension of the basic PSO (*bPSO*) method, the MD-PSO. Instead of operating with a fixed number of dimensions, *D*, the MD-PSO algorithm is designed to seek both *positional* and *dimensional* optima within a certain dimension range $\{D_{\min}, D_{\max}\}$. For this, each particle has *two sets* of components, each of which has been subjected to two independent and consecutive processes. The first set is a regular positional PSO, taking care of the traditional velocity updates and positional shifts in the *D*-dimensional search (solution) space, whereas the second set is the dimensional PSO, allowing the particles to navigate through dimensions. Accordingly, now each particle keeps track of its latest position, velocity and personal best position in a *particular dimension,* so that when the particle revisits the same dimension later, it can perform its regular "positional" update. The dimensional PSO process of each particle may then move the particle to another dimension, where it will remember its positional status and shall be updated with the positional PSO process at that dimension. The swarm, on the other hand, keeps now track of the *gbest* particle in *each dimension*, and the dimensional PSO process of each particle uses its personal best dimension (in which the personal best fitness score has been achieved so far). Finally, the swarm keeps track of the global best dimension, *dbest*, among all the personal best dimensions. Thus, the *gbest* particle in the *dbest* dimension represents the optimum solution found.

In a MD-PSO process at time (iteration) t, each particle a in the swarm with S particles, $\xi = \{x_1,..,x_a,..,x_S\}$, is represented by the following symbols:

$x_{a,j}^{d_a(t)}(t)$: j^{th} component of the *position* of particle a in dimension $d_a(t)$.

$\tilde{x}_{a,j}^{d_a(t)}(t)$: j^{th} component of the *personal best position* of particle a in dimension $d_a(t)$.

$v_{a,j}^{d_a(t)}(t)$: j^{th} component of the *velocity* of particle a in dimension $d_a(t)$.

$d_a(t)$: dimension of particle a.

$\tilde{d}_a(t)$: *personal best dimension* of particle a.

$vd_a(t)$: dimensional velocity of particle a.

$gx_j^d(t)$: j^{th} component of the *global best position* of swarm in dimension d.

Let f denote a *fitness function* that is to be optimized within a certain dimension range, $\{D_{\min}, D_{\max}\}$. Without loss of generality, assume that the objective is to find the *minimum* of f at the optimum dimension within a multi-dimensional search space. Assume also, that the particle a visits (back) the same dimension after T iterations (i.e. $d_a(t) = d_a(t+T)$). Then, the personal best position can be updated at iteration t+T as,

$$\tilde{x}_{a,j}^{d_a(t+T)}(t+T) = \\ = \begin{cases} \tilde{x}_{a,j}^{d_a(t)}(t) \quad if\ f\left(x_{a,j}^{d_a(t+T)}(t+T)\right) > f\left(\tilde{x}_{a,j}^{d_a(t)}(t)\right) \\ x_{a,j}^{d_a(t+T)}(t+T) \quad else, \end{cases} \qquad (1)$$

$$j = 1,2,\ldots,d_a(t+T).$$

Furthermore, the personal best dimension of particle a can be updated in iteration $t+1$ as,

$$\tilde{d}_a(t+1) = \\ = \begin{cases} \tilde{d}_a(t) \quad if\ f\left(x_{a,j}^{d_a(t+1)}(t+1)\right) > f\left(\tilde{x}_{a,j}^{\tilde{d}_a(t)}(t)\right) \\ d_a(t+1) \qquad else. \end{cases} \qquad (2)$$

Fig. 1 shows an example MD-PSO and *bPSO* particles. Particle a in *bPSO* is at (fixed) dimension, D = 5, and contains only positional components, whereas in MD-PSO, particle a contains both the positional and dimensional components. The dimension range for MD-PSO is given by $\{D_{\min}, D_{\max}\}$ = {2, 10}, so that 9 sets of positional components are included in a.

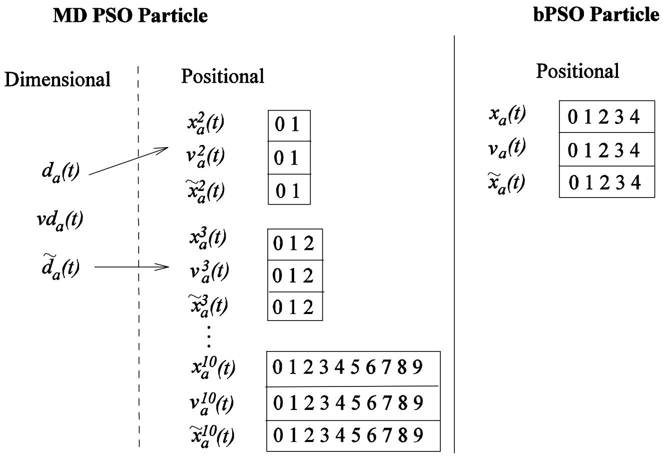


Figure 1. MD-PSO (left) vs. bPSO (right) particle structures for dimensions $\{D_{min}=2, D_{max}=10\}$. At time t, $\boldsymbol{d_a(t) = 2, \tilde{d}_a(t) = 3}$.

In this example the particle a currently resides at dimension 2 ($d_a(t) = 2$), while its personal best dimension is 3 ($\tilde{d}_a(t) = 3$). Hence, at time t a positional PSO update is first performed over the positional components of $x_a^2(t)$, after which the particle may move to another dimension with respect to the dimensional PSO. Recall that each positional component $x_a^2(t)$ represents a potential solution in the data space to the problem. The algorithmic flowchart and further details about MD-PSO can be obtained from [16].

B. MD – PSO for Evolving MLPs

The MD-PSO seeks (near-) optimal networks in an AS, which can be defined over any type of ANNs with any properties. All network configurations in the AS are enumerated into a *hash table* with a proper hash function, which ranks the networks with respect to their complexity, i.e. associates higher hash indices to networks with higher complexity. The MD-PSO can then treat each index as a unique dimension in the search space. The dimension thus corresponds to the optimal classifier architecture, while the position (solution) encapsulates the optimum network parameters (connections, weights and biases). Suppose, for the sake of simplicity, that a certain *range* is defined for the minimum and maximum number of MLP layers, $\{L_{\min}, L_{\max}\}$, as well as for the number of neurons in the hidden layer l, $\{N_{\min}^l, N_{\max}^l\}$. The sizes of both input and output layers, $\{N_i, N_o\}$, are determined by the problem, and hence fixed. The AS can then be defined by only two range arrays:

$$R_{\min} = \{N_i, N_{\min}^1, \ldots, N_{\min}^{L_{\max}-1}, N_o\},$$
$$R_{\max} = \{N_i, N_{\max}^1, \ldots, N_{\max}^{L_{\max}-1}, N_o\},$$

where the first one is for the minimum-, and the second one is for the maximum number of neurons allowed for each layer of a MLP. The size of the both arrays is naturally $L_{\max}$ + 1, where the corresponding entries define the range of the l^{th} hidden layer for all those MLPs that can have the l^{th} hidden layer. The terms $L_{\min} \geq 1$ and $L_{\max}$ can be set to any value meaningful for the problem at stake. The hash function then enumerates all potential MLP configurations into hash indices, starting from the simplest MLP with $L_{\min}$ – 1 hidden layers (each of which has the minimum number of neurons given by $R_{\min}$), to the most complex one with $L_{\max} - 1$ hidden layers (each of which has the maximum number of neurons given by $R_{\max}$).

Let N_l be the number of hidden neurons in layer l of a MLP with the input and output layer of sizes N_i and N_o, respectively. The input neurons are merely fan-out units, as no processing is done in them. Let g be the *activation function* (e.g. sigmoid) applied over the weighted inputs and a bias. Thus we can write,

$$y_n^l(p) = g\big(Y_n^{p,l}\big), \qquad Y_n^{p,l} = \sum_m w_{mn}^{l-1} y_m^{l-1}(p) + \theta_n^l, \tag{3}$$

where $y_n^l(p)$ is the output of the n^{th} neuron of the l^{th} hidden / output layer when a pattern p is fed into it, w_{mn}^{l-1} is the weight from the m^{th} neuron in layer $l-1$ to the n^{th} neuron in layer l, and θ_n^l is the bias value of the n^{th} neuron in the l^{th} layer. The training mean square error, MSE, is formulated as:

$$MSE = \frac{1}{2PN_o} \sum_{p \in A} \sum_{n=1}^{N_o} \big(t_n(p) - y_n^o(p)\big)^2, \tag{4}$$

where $t_n(p)$ is the target (desired) output, and $y_n^o(p)$ is the actual output from the n^{th} neuron in the output layer, l=o, for pattern p in the training dataset A with size P, respectively. At time t, the particle a has the positional component formed as,

$$\begin{aligned} &x_{a,j}^{d_a(t)}(t) = \\ &= \Psi^{\mathrm{d_a(t)}}\{\{w_{mn}^0\},\{w_{mn}^1\},\{\theta_\mathrm{n}^1\},\ldots,\{w_{mn}^{o-1}\},\{\theta_\mathrm{n}^{o-1}\},\{\theta_\mathrm{n}^o\}\}, \end{aligned} \tag{5}$$

where $\{w_{mn}^l\}$ and $\{\theta_\mathrm{n}^l\}$ represent the sets of weights and biases of the layer l of the MLP-configuration $\Psi^{\mathrm{d_a(t)}}$. Note that the input layer (l=0) contains only weights, whereas the output layer (l=o) contains only biases. By the means of such a direct encoding scheme, the particle a thus represents all potential network parameters of the MLP architecture at the dimension (hash index) $d_a(t)$. As mentioned earlier, the dimension range $\{D_{\min}, D_{\max}\}$ where the MD-PSO particles can make inter-dimensional jumps, is determined by the AS defined. Apart from the regular limits, such as (positional) velocity range, $\{V_{\min}, V_{\max}\}$, and dimensional velocity range, $\{VD_{\min}, VD_{\max}\}$, the data space can be also limited by some practical range, i.e. $X_{\min} < x_{a,j}^{d_a(t)}(t) < X_{\max}$. Setting the MSE in (4) as the fitness function, to be used in the MD-PSO, enables then performing evolutions of both the network parameters and the network architectures. Further details and an extensive set of network evolution experiments can be found in [14].

C. The Back-Propagation Algorithm

Back Propagation (BP) is the most commonly used training technique for feed-forward ANNs. It is a supervised training technique that has been used in pattern recognition and classification problems in many application areas. Essentially, BP is just a *gradient descent* algorithm in the *error* space, which may be complex and contain many deceiving local minima (multi-modal). Therefore, BP gets easily trapped into a local minimum, making it entirely dependent on the initial (weight) settings. However, due to its simplicity and relatively lower computational cost, BP can be applied exhaustively over the network architectures with *random initializations*, to find out the optimal architecture. Since the AS is composed of only compact networks, with such an exhaustive search the probability of finding (converging) to a (near-) optimum solution in the error space is significantly increased.

The used BP algorithm can be summarized as follows:

1. Initialize the weights w_{mn}^l and biases θ_n^l randomly.
2. Feed a pattern p to the network and compute the output $y_n^l(p)$ of each neuron n in each hidden layer l.
3. Calculate the error between the final output $y_n^o(p)$ of each output neuron and the desired output $t_n(p)$ as $e_n^o(p) = t_n(p) - y_n^o(p)$.
4. For each neuron n, calculate the partial derivatives $\frac{\partial E(p)}{\partial h_n^l}$, where $E(p)$is the total *error energy* defined as $E(p) = \frac{1}{2}\sum_{n \in o}\big(e_n^o(p)\big)^2$, and $h_n^l$is a uniform symbol for the parameters w_{mn}^l and θ_n^l.
5. Update the parameters as follows:

   $$h_n^l(t+1) = h_n^l(t) - \eta \frac{\partial E(p)}{\partial h_n^l}, \tag{6}$$

   where η is a *learning rate* parameter.
6. Repeat steps 2-5 until some stopping criterion is reached.

One complete run over the training dataset is called an *epoch*. Usually many *epochs* are required to obtain the best training results, but, on the other hand, too many training *epochs* can lead to over-fitting. In the above realization of the BP algorithm, the network parameters are updated after every training sample (pattern p). This is called an *online* or *sequential* training mode. Another possibility is the *batch* mode, where all the training samples are first presented to the network, and then the parameters are adjusted so to minimize the total training error. The *sequential* mode is often favored over the *batch* mode, as less storage space is required. Moreover, the *sequential* mode is less likely to get trapped into a local minimum, as updates at every training sample make the search stochastic in nature. Hence, *sequential* BP mode is used for MLP training in this study.

III. Audio Feature Extraction

As a common approach in audio signal processing, the audio signal to be analyzed is first divided into short time windows / frames (20-40 ms), from which the audio features are extracted. This is to prevent averaging the signal over long segments, in which case the discrimination of audio features may decrease significantly. In this study, *three sets* of features are extracted from each audio clip to be classified, divided as:

- *General Audio Features:* These consist of sub-band power (4 bands), band energy ratio (BER), sub-band centroid (SC), zero-crossing rate (ZCR), short average energy (SAE), brightness, bandwidth, spectral roll-off, spectral flux, and fundamental frequency (FF).
- *MFCCs:* The first 24 coefficients of the extracted Mel-frequency cepstral coefficients.

- *Linear Prediction Coefficients (LPC):* The 8^{th} order LP coefficients (the order is based on the *16 kHz sampling frequency* used in the classified audio samples).

The *feature vector (FV)* dimensions for each feature set are thus 13, 24, and 8, respectively.

Extracting the features from short time frames leads into a rather big number of FVs even from a short audio clip. Therefore, in our work, a certain amount of *key frames* (*KFs,* see [4]), are first selected among the frames, being sort of "prototypes", which are chosen so to represent the others as accurately as possible. The ultimate goal is to find concise and representative feature sets from each audio clip, to speed up the classification process without losing any vital information of the original signals. The reasoning behind the idea of exploiting only a small fraction of the audio frames is based on an assumption, that the elementary sounds within an audio clip are immensely repetitive, and often entirely alike. For an efficient KF selection, audio frames with similar acoustical features within an audio clip are *clustered*, and only one or few frames from each cluster are considered as KFs to represent the others in that cluster. Here the number of KFs is empirically set between ~1-2% of the total amount of frames, being relatively lower for longer clips. Thus, the actual number of key frames selected from each clip varies approximately between 40 and 200 frames, depending on the length and variation of the signal. Once the number of KFs is determined, a *Minimum Spanning Tree (MST)* clustering technique is applied. Every node in the tree represents the extracted features of a unique audio frame. The clustering scheme is illustrated in Fig. 2, where the word "Speech Lab" is divided into seven separate clusters, according to the similarity of the extracted frames. More technical details about the audio clustering scheme can be found in [4].

For each of the three extracted features sets, the proposed classification framework evolves a separate binary classifier (BC) per each pre-determined class, so that a unique *network* of BCs (NBC) is created for each class. Note that, as such, the proposed framework performs classification over the KFs, and not the actual audio clips. Hence, a *majority rule* is applied to the classified KFs to decide the final class of the corresponding clip. For this, a specific table is created, where the KF indices corresponding to each audio clip in the database are stored.

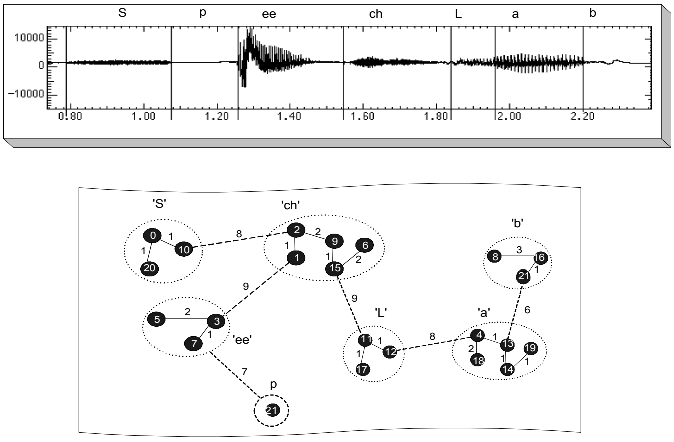


Figure 2. An illustrative MST clustering scheme.

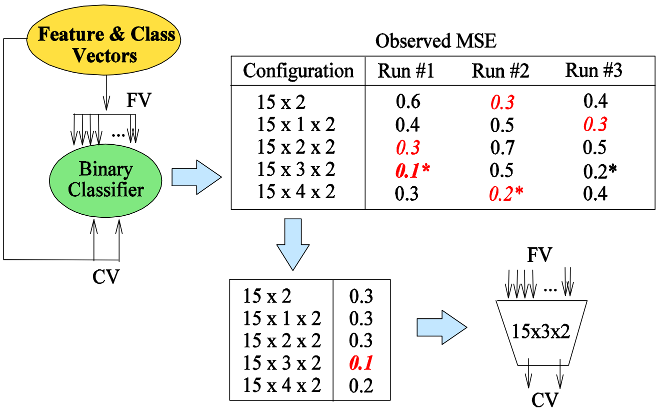
IV. The Classification Framework

This section describes in detail the proposed classification framework: the Collective Network of (Evolutionary) Binary Classifiers (CNBC). The framework takes as an input the extracted feature vectors from the training dataset KFs, after which the internal network topology is configured, and all the corresponding binary classifiers are evolved individually. Before going into full details of the CNBC, the used AS evolutionary update mechanism will be introduced.

A. Evolutionary Update in the Architecture Space

Since the primary evolution technique used, MD-PSO, is a stochastic optimization method, it is not guaranteed that it will always find the optimal solution. Thus, in order to improve the probability of convergence to the global optimum, several evolutionary *runs* can be performed. Let Q_R be the number of runs and Q_C be the number of configurations in the AS. For each run, the objective is to find the optimal classifier within the AS, with respect to some pre-defined criterion. Note that, along with the *best* classifier, all the other configurations in the AS are also evolved simultaneously, so that the configurations are continuously (re-)trained within each run. Thus, during the process, any network configuration may *replace* the current best one in the AS, if it is surpassed in terms of the classification performance criterion. This is also true in the *exhaustive search*, where each network configuration in the AS is evolved using Q_R Back-Propagation (BP) runs.

Fig. 3 demonstrates the evolutionary update operation over a sample AS containing 5 MLP configurations. The bigger table in Fig. 3 shows the training *Mean Square Error (MSE)*, which is the criterion used to select the optimal configuration at each run. The best runs for each configuration are highlighted, and the best configuration in each run is tagged with '*'. In this case, at the end of three runs, the overall best network with MSE = 0.1 has a configuration 15x3x2, and thus it is used as the classifier for all the forthcoming classification tasks, until a new configuration may surpass it in a future run. As can be seen from this example, each BC configuration in the AS can only evolve into a *better* state from the previous one (in terms of the training MSE), which is the main motivation for the proposed evolutionary update mechanism.



Observed MSE

Configuration	Run #1	Run #2	Run #3
15 x 2	0.6	*0.3*	0.4
15 x 1 x 2	0.4	0.5	*0.3*
15 x 2 x 2	*0.3*	0.7	0.5
15 x 3 x 2	***0.1****	0.5	0.2*
15 x 4 x 2	0.3	*0.2**	0.4

15 x 2	0.3
15 x 1 x 2	0.3
15 x 2 x 2	0.3
15 x 3 x 2	***0.1***
15 x 4 x 2	0.2

Figure 3. Evolutionary update in a sample AS for MLP configuration arrays $\boldsymbol{R}_{\boldsymbol{min}} = \{\mathbf{15}, \mathbf{0}, \mathbf{2}\}$ and $\boldsymbol{R}_{\boldsymbol{max}} = \{\mathbf{15}, \mathbf{4}, \mathbf{2}\}$, where Q_R = 3 and Q_C = 5. The best runs for each configurations are highlighted, and the best configuration in each run is tagged with '*'.

B. Collective Network of Binary Classifiers

1) The Topology:

In the CNBC framework, the individual networks of BCs (NBCs) *evolve* continuously with the ongoing evolution sessions by using the *ground truth training data* (*GTD*) given by the user. Each BC in a particular NBC performs binary classification using one of the three extracted FVs. Each NBC has also a “fuser” BC in its output layer, which collects and fuses the binary outputs of all the BCs in the input layer. A single binary output is then generated from each NBC, indicating the *relevancy* of the current input KF to the NBC’s corresponding class. Due to its structure, the CNBC can be dynamically *scaled* into any number of classes, as whenever a new class is defined, a new corresponding NBC will be created and evolved on top of the existing structure. The procedure does not require changing or updating the other NBCs, as long as they pass the so-called *verification test*, which is performed by selecting a specific accuracy threshold, and by seeing whether the existing NBCs classify the training samples of the new class(es) accurately enough (and not confuse with them). In this work, an accuracy threshold of 95% was applied.

As is shown in Fig. 4, in CNBC, a learning problem with many classes and features can be *divided* into as many NBCs (and BCs within) as necessary, so as to negate the need for complex classifiers. This is a notable advantage, since the performance of the training and evolution processes degrades significantly as the classifier complexity increases (due to the well-known *curse of dimensionality* – phenomenon). Another major benefit of the approach, with respect to efficiency, is that the configurations in the AS can be kept very compact, so that unfeasibly large storages and heavy computations can be avoided. This is especially important for the BP method, since the amount of deceiving local minima is significantly lower in the error space for simple and compact ANNs.

In order to maximize the final classification accuracy, a dedicated *class selection technique* is applied. In all the BCs, a *1-of-M* encoding scheme, with *M=2,* is used. Let us denote $CV_{f,1}$ and $CV_{f,2}$ as the first and second output of the f^{th} BC’s *class vector (CV)*, respectively. The class selection in the *1-of-2* encoding scheme is then performed by comparing the two individual outputs, and the encoded output is determined as *positive* if $CV_{f,2} > CV_{f,1}$, and *negative* otherwise. The same encoding scheme applies for the fuser BC output, which determines the output of the whole NBC. A *class selection* block, illustrated at the bottom of Fig. 4, then collects the CVs of each NBC, and selects the “most positive” output among all the NBCs as the final classification outcome. Here a so-called *winner-takes-all* strategy is utilized, where the positive class index, c^*, (“the winner”) is defined as,

$$c^* = \arg \max_{c\,\in[0,C-1]} (CV_{c,2} - CV_{c,1}), \tag{7}$$

where C is the number of classes (NBCs). This way the erroneous cases (false positives), where there exist more than one NBC with positive outcome, can be handled properly.

2) Evolution of the CNBC:

The evolution of the CNBC, or a subset of NBCs, is performed for each NBC individually with a *two-phase* operation, as is illustrated in the upper part of Fig. 4. In Phase 1, the BCs of each NBC are first evolved by giving an input set of FVs and the target CVs (the GTD). Recall, that each CV is associated with a unique NBC, and that the fuser BCs are not yet used at this phase. Once the evolution session is over, the AS of each BC is *saved*, so as to be used for potential (incremental) evolution sessions in the future (Section IV.B.3). The best BC configuration in the AS is used to forward-propagate the respective FVs of the training dataset, in order to compose the BC outputs, which, again, are used as input FV for the corresponding fuser BC. The fuser BCs are then evolved in Phase 2 of the CNBC evolution process, where each fuser BC *learns* the *significance* of its individual BCs (and their feature sets). This can be viewed as a way of applying an efficient *feature selection* scheme, so that the fuser, if properly evolved and trained, can “weight” each BC accordingly. This way the potential of each feature set (and its BC) will be optimally fused according to their discrimination power over each class. Similarly, each BC in the first layer shall in time learn the significance of the individual feature components of the corresponding feature set. That is, the CNBC, if properly evolved, will learn the significance (or the discrimination power) of each feature set, as well as their individual components (the single features).

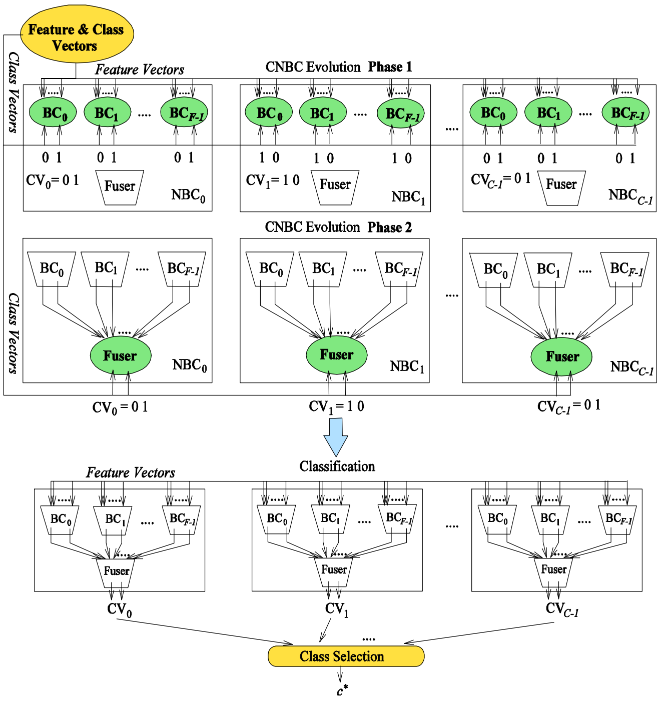


Figure 4. Illustration of the two-phase evolution session over BC architecture spaces in each NBC, and the topology of the CNBC framework with *C* classes and *F* feature sets.

3) Incremental Evolution of the CNBC:

The proposed CNBC framework is designed for continuous “incremental” evolution sessions, where each session may further improve the classification performance of each BC using the advantage of the “evolutionary updates”. The main difference between the initial and the subsequent evolution sessions is in the *initialization* phase of the evolution process: the former uses *random* initialization, whereas the latter starts from the previously saved AS parameters of each classifier in each NBC. Note that the training dataset used for the incremental evolution session may differ from the ones used in the previous sessions, and that each session may contain several runs. The evolutionary update rule hence compares the performance between the previously received, and the *current* (after the update) network over the *current* training dataset. Consequently, for the proposed MD-PSO evolutionary technique, the swarm particles are randomly initialized (as in the initial evolutionary step), with the exception that the *first particle* has its personal best value, *pbest*, set to the optimal solution found in the previous evolutionary session. That is,

$$\begin{aligned} &\tilde{x}_0^d(0) = \\ &= \Psi^d\{\{w_{mn}^0\},\{w_{mn}^1\},\{\theta_n^1\},\dots,\{w_{mn}^{o-1}\},\{\theta_n^{o-1}\},\{\theta_n^o\}\}, \quad (8) \\ &\forall d \in [2, L_{max}+1], \end{aligned}$$

where Ψ^d is the *d*-dimensional MLP-configuration retrieved from the previous AS search.

It is expected that, especially at the early stages of the MD-PSO run, the first particle is likely to be the *gbest* particle in every dimension, guiding the swarm towards the previous solution. However, if the training dataset is considerably different in the incremental evolution sessions, it is quite probable that MD-PSO will converge to a new solution, while taking the past solution (experience) into account. In the case of the BP training technique, the weights w_{mn}^l and biases θ_n^l will be initialized with the parameters retrieved from the last AS search. Starting from this as the initial point, and using the current training dataset with the target CVs, the BP algorithm can then perform its gradient descent in the error space.

V. Experimental Results

The audio database used in the classification experiments consists of 367 clips, divided into 8 classes. The database is gathered mostly from the "FreeSound Project" web page [17], but also RWC Music Database [18] was used to collect the music classes. The abbreviations of the used audio classes are as follows: MS (male speech), FS (female speech), FV (female vocals/singing), MV (male vocals/singing), B (bird chirping), W (water sounds), CM (classical music) and GM (general (pop) music). We left the majority of the database clips (75%) for testing, while a training set containing only 25% of the samples from each class was used to evolve the CNBC. For the AS, we used simple ANN configurations with the following range arrays: $R_{\min} = \{N_i, 8, 2\}$ and $R_{\max} = \{N_i, 16, 2\}$, which indicate that, besides a single-layer perceptron (SLP), all the MLPs contain only one hidden layer, i.e. $L_{\max} = 2$, with no more than 16 hidden neurons. The software implementations were made using Visual C++ 6.0 with FFTW library for FFT processing. Parallel processing was utilized in evolving the CNBC classifiers, yielding an approximate CPU time of 1-1.5 h to obtain the classification results with the exhaustive BP method. However, the needed CPU time is highly dependent on the parameters used, i.e. the number of runs, Q_R, and epochs, Q_E, so that the reported CPU time should be considered as suggestive only (for example, with Q_R=1 and Q_E=100, the CPU time decreases to only 10-15 minutes). The processor used was Intel® Core^{TM}2 Quad Q9400, 2.66 GHz with 8 Gb of RAM.

For the both evolution methods, exhaustive BP and MD-PSO, the number of runs and training epochs (or iterations in the case of MD-PSO) were varied to see their effect on the classification performance, as is shown in Table I. In order to compare the results with a method representing the current state of the art ([5], [6]), also *SVM* classifiers were tested by applying the *libSVM* library. Results with four different kernels (linear, polynomial, radial basis function (RBF), and sigmoid) were evaluated, for which the best classification accuracy of *89.38%* was obtained. Here a *“one against one”* approach (one SVM for each pair of classes) was applied in evaluating the results for the stated multi-class problem.

The performed CNBC evolutions of Table I are much alike to the (batch) training of traditional classifiers (such as ANNs, K-nearest neighbour, Bayesian), where the training data (the features) and the number of classes are all fixed, and the entire GTD is used during the training (evolution). However, as detailed earlier, the CNBC can be also evolved incrementally, i.e. the evolutions can be performed whenever new features / classes are introduced. For evaluating the incremental evolution performance, the training dataset was divided into *three* distinct partitions, containing 4 (MS, FS, CM, and W), 2 (B and MV) and 2 (*FV* and *GM*) classes, respectively. Three *stages* of incremental evolutions were then performed, where at each stage the CNBC was further evolved using only the dataset belonging to the new classes in the corresponding partition. At the end, the resulting CNBC with $4 + 2 + 2 = 8$ NBCs, encapsulating $8 \times (3 + 1) = 32$ BCs within, was created. Verification test between each stage was performed (with the 95% accuracy threshold) to determine whether the existing NBCs needed to be re-trained with the new training samples. A final classification accuracy of 87.38% was achieved, indicating only a moderate loss of performance when compared to the classification accuracies listed in Table I. It is thus evident that the proposed CNBC design can cope up with the incremental evolutions also.

The *confusion matrix (CM)* given in Table II is composed from the classification results of the exhaustive BP evolutions with Q_R = 5, and Q_E = 200. The rows of the CM correspond to the ground truth labels of the classes, whereas the columns indicate the actual classifications results. The average *precision (P)* and *recall (R)* calculated from the CM are:

$P = 0.9151 \qquad R = 0.9005,$

respectively. The overall classification error rate of ~8.8% with 8 classes indicates a substantial level of classification accuracy, considering the quite limited training dataset used (25%), and the somewhat overlapping audio classes with inter-class similarities (e.g. male speech / male vocals). It can be noticed that the music classes are classified perfectly, whereas some

TABLE I. Classification Accuracies of the Both Evolutionary Techniques with Different Number of Epochs and Runs.

		Number of Epochs / Iterations		
	Evol. Method	**Q_E =100**	**Q_E =200**	**Q_E =300**
Q_R =1	**BP**	90.11%	91.07%	91.07%
	MD PSO	87.91%	88.64%	88.64%
Q_R =5	**BP**	90.48%	**91.21%**	**91.21%**
	MD PSO	89.01%	**91.21%**	87.91%

TABLE II. CONFUSION MATRIX OF THE EXHAUSTIVE BP EVOLUTIONS WITH $Q_R = 5$, AND $Q_E = 200$.

Actual Result		MS	FS	FV	MV	B	W	CM	GM
Ground Truth	MS	**27**	4	0	0	0	0	0	3
	FS	0	**22**	2	0	1	0	0	0
	FV	0	0	**39**	0	0	0	1	0
	MV	2	0	1	**32**	0	0	1	0
	B	0	0	0	0	**37**	1	0	0
	W	0	1	0	0	0	**18**	0	7
	CM	0	0	0	0	0	0	**40**	0
	GM	0	0	0	0	0	0	0	**34**

confusion occurs between the speech classes and, rather surprisingly, between the water sound and general pop music classes. Nevertheless, for the tested database, the results are more accurate than those obtained using the "one against one" SVM approach.

VI. CONCLUSIONS

In this paper, a novel CNBC framework was introduced to address the problem of accurate and efficient content-based audio classification within large and dynamic audio databases. The achieved classification results show improvements in accuracy when compared to the tested Support vector machine (SVM) classifiers. Furthermore, two notable advantages can be mentioned on behalf of the proposed approach: First, the dynamic and evolving structure of the framework supports for dynamic variations of audio classes in the considered database, meaning that there is no need to re-train the entire classifier network again whenever new data is added to the database. Second, the optimum classifier for the classification problem at hand can be searched by the underlying evolution technique, which allows creating a dedicated classifier to discriminate a certain class type from the others by using only some specific feature set. This negates the necessity of configuring the classifier parameters and configurations strictly for some specific audio dataset, and thus, hopefully, broadens the usage of the framework for varying type of audio databases.

Future research will be concentrating on supporting larger number of audio classes and developing more descriptive audio features. Due to the structure of the CNBC, it would be ideal to have features with high discrimination power over one (or some particular) class(es). We aim to develop a *perceptual* audio key frame extraction scheme that would take into account the human auditory system. Also, additional classifiers, such as weak classifiers, RBFs and random forests, are planned to be applied within the CNBC network, because of their strong discriminating power reported for certain type of classification tasks in the literature. Finally, content-based indexing and retrieval of audio (and video) clips is a natural way to add more value to the framework, and will be certainly put under study in a near future. Based on the achieved classification accuracy, reliable retrieval results can be expected in the future research.

REFERENCES

[1] E. Wold, T. Blum, D. Keislar, and J. Wheaton, "Content-based classification, search, and retrieval of audio", in IEEE Multimedia Journal, Vol. 3, No. 3, 1996, pp. 27-36.

[2] C. Wu and C. Hsieh, "Multiple change-point audio segmentation and classification using an MDL-based Gaussian model", in IEEE Transactions on Audio, Speech, and Language Processing, Vol. 14, No. 2, March 2006, pp. 647-657.

[3] W. Pan, Y. Yao and Z. Liu, "An unsupervised audio degmentation and classification approach," in Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2007.

[4] S. Kiranyaz, A. F. Qureshi, and M. Gabbouj, "A generic audio classification and segmentation approach for multimedia indexing and retrieval", in IEEE Transactions on Audio, Speech, and Language Processing, Vol. 14, No. 3, May 2006, pp. 1062-1081.

[5] L. Chen, S. Gündüz and M. T. Özsu, "Mixed type audio classification with support vector machine," in IEEE International Conference on Multimedia and Expo (ICME), pp. 781–784, April 2006.

[6] Y. Zhu, Z. Ming and Q. Huang, "SVM-based audio classification for content-based multimedia retrieval", in Proc. of the 2007 International Conference on Multimedia Content Analysis and Mining (MCAM), Weihai, China, 2007, pp.474-482.

[7] W. Chu and B. Champagne, "A noise-robust FFT-based spectrum for audio classification", in Proc. of Acoustics, Speech and Signal Processing (ICASSP), May 2006, pp. V-213-V-216.

[8] E. Ravelli, G. Richard, and L. Daudet, "Audio signal representations for indexing in the transform domain", in IEEE Transactions on Audio, Speech, and Language Processing, Vol. 18, No. 3, March 2010, pp. 434-446.

[9] E. Dogan, M. Sert, A. Yazici, "Content-based classification and segmentation of mized-type audio by using MPEG-7 features", in Proc. of the First International Conference on Advances in Multimedia (MMEDIA), July 2009, pp. 152-157.

[10] G. Peeters, "A generic system for audio indexing: Application to speech/music segmentation and music genre recognition", in Proc. of the 10th Int. Conference on Digital Audio Effects (DAFx-07), Bordeaux, France, September 2007.

[11] H. Harb and L. Chen, "A general classifier based on human perception motivated model", in Multimedia Tools and Applications Journal, Vol. 34, No. 3, 2007, pp. 375-395.

[12] S. Kiranyaz, M. Gabbouj, J. Pulkkinen, T. Ince and K. Meissner, "Network of evolutionary binary classifiers for classification and retrieval in macroinvertebrate databases", in IEEE International Conference on Image Processing, September 2010, pp. 2257 – 2260.

[13] H. Abbass, "Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization", in the IEEE Congress on Evolutionary Computation 2003, pp. 2074 – 2080 Vol. 3.

[14] S. Kiranyaz, T. Ince, A. Yildirim and M. Gabbouj, "Evolutionary artificial neural networks by multi-dimensional particle swarm optimization", Neural Networks, vol. 22, pp. 1448 – 1462, Dec. 2009.

[15] J. Kennedy, R Eberhart., "Particle swarm optimization", in Proc. of IEEE Int. Conference On Neural Networks, vol. 4, pp. 1942–1948, Perth, Australia, 1995.

[16] S. Kiranyaz, T. Ince, A. Yildirim and M. Gabbouj, "Fractional Particle Swarm Optimization in Multi-Dimensional Search Space", *IEEE Trans. on Systems, Man, and Cybernetics – Part B*, pp. 298 – 319, vol. 40, No. 2, 2010.

[17] The Freesound Project web page, http://www.freesound.org/.

[18] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical, and jazz music databases", in Proc. of 3rd International Conference on Music Information Retrieval, Oct. 2002