Deep neural networks for audio scene recognition

Yohan Petetin, Cyrille Laroche, Aurelien Mayoue

DEEP NEURAL NETWORKS FOR AUDIO SCENE RECOGNITION

Yohan Petetin, Cyrille Laroche, Aurélien Mayoue

CEA, LIST, Gif-sur-Yvette, F-91191, France

ABSTRACT

These last years, artificial neural networks (ANN) have known a renewed interest since efficient training procedures have emerged to learn the so called deep neural networks (DNN), i.e. ANN with at least two hidden layers. In the same time, the computational auditory scene recognition (CASR) problem which consists in estimating the environment around a device from the received audio signal has been investigated. Most of works which deal with the CASR problem have tried to find well-adapted features for this problem. However, these features are generally combined with a classical classifier. In this paper, we introduce DNN in the CASR field and we show that such networks can provide promising results and perform better than standard classifiers when the same features are used.

***Index Terms*—** Deep neural networks; deep beliefs networks; audio scene recognition.

1. INTRODUCTION

1.1. Generalities

The CASR problem consists in determining automatically the context or environment around a device [1]. A variety of features have been proposed for CASR, but the majority of the past work uses features that are well-known for structured data, such as speech and music. In this way, time-domain (zero-crossing rate), frequency-domain (band-energy ration, spectral centroid, spectral flatness) and cepstral (Mel-frequency cepstral coefficients) features are naturally used in the literature [1] [2] [3] [4]. Only few recent articles have proposed new sets of features which try to encode some relevant information for unstructured environmental sound classification. The choice of these new features is often inspired by other research fields than audio one such as image processing (spectrogram pattern [5], histogram of gradient (HOG) features [6]), chaos theory (Recurrence Quantification Analysis descriptors [7]) or compressed sensing (Matching Pursuit-based features [8]). In this paper, we do not discuss on the relevance of audio features for CASR but we investigate classification approaches. Indeed, whatever the complexity of the features proposed in the literature, the classification step is always based on standard machine learning approaches such as K-nearest neighbors [1] [5] [8], Gaussian Mixture Models (GMM) [1] [8], hidden Markov models [2] [3], Support Vector Machines (SVM) [4] [6] [7]. Or, while DNN have led to significant advances in automatic speech recognition [9], this approach has never been used in the field of CASR to the best of our knowledge. In this paper, we study how to deploy DNN for audio context recognition and we show that DNN can produce promising results even when we use standard audio features which are not necessarily optimized for the CASR problem.

1.2. Feed forward artificial neural networks

Feed forward artifical neural networks (ANN) are popular computer architectures which can be used for classification. More precisely, when the objective is to classify a feature of interest $\mathbf{x}$ among C classes, an ANN estimates the probabilities p_j, $j \in \{1, \cdots, C\}$, of each class given the input feature $\mathbf{x}$. In our audio classification problem, the input $\mathbf{x}$ represents the concatenation of audio features [10] [11], such as cepstral (Mel-frequency cepstral coefficients (MFCC)) and frequency features (spectral centroid, spectral flatness,...); the class represents the audio context (car, bus, office, street, restaurant, ...). A graphical representation of this architecture is given in Figure 1.

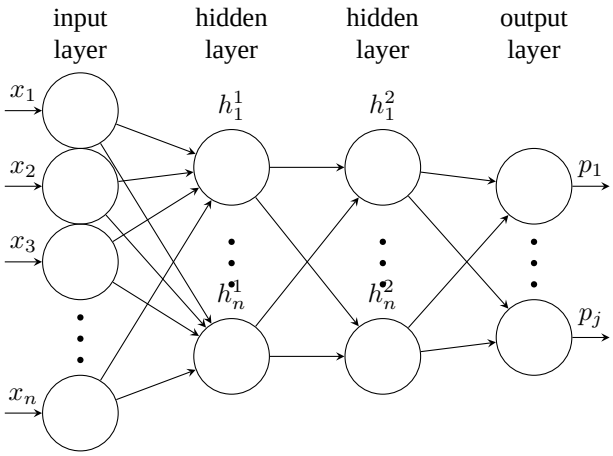

Fig. 1. An ANN is described by an input (a feature vector), a given number of hidden layers, a given number of neurons per layer and an output which describes the class probabilities.

In order to compute the outputs p_j of the ANN, we first

need to compute the ouput of each hidden unit. In an ANN, the connection between the $k-1$-th hidden layer and the k-th one is described by a matrix of weights $\mathbf{W}^k$, and a bias vector $\mathbf{b}^k$; the output h_j^k of the j-th neuron of the k-th layer is then computed from

$$h_j^k = f\left(\sum_i w_{ij}^k h_i^{k-1} + b_j^k\right), \tag{1}$$

where $f(.)$ is the sigmoid function:

$$f(x) = \text{sigmoid}(x) = \frac{1}{1+e^{-x}}. \tag{2}$$

Finally, the output is computed via the softmax nonlinearity,

$$p_j = \frac{e^{\sum_{i=1} w_{ij}^p h_i^{p-1} + b_j^p}}{\sum_k e^{\sum_{i=1} w_{ik}^p h_i^{p-1} + b_k^p}}, \tag{3}$$

where p is the number of layers (without counting up the input layer).

The training of ANN (i.e. the estimation of parameters $\mathbf{W}^k$ and $\mathbf{b}^k$) relies on supervised methods such as the Back-Propagation (BP) algorithm [12] whose the principle will be reminded in section 3. However, when the number of hidden layers and neurons increases, supervised methods are not reliable and ANN are difficult to tune. Particularly, these methods can be stuck in a poor local optima when we look for estimating the parameters. Recently, new procedures for training DNN (i.e. ANN with at least two hidden layers) have been proposed to overcome the limitations of classical training algorithms [13] and rely on an unsupervised pre-training which aims at initializing properly the parameters of the DNN. The rest of this paper is organized as follows. In Section 2, we describe the pre-training step of DNN which relies on Restricted Boltzman Machines (RBM) and Deep Belief Networks (DBN) and which are both probabilistic graphical models. In Section 3, the principle of the supervised training via the BP algorithm is recalled. Finally, in section 4, we focus on the tuning of DNN for CASR problem by performing experimentations on an audio context dataset.

2. PRE-TRAINING OF DNN VIA DBN

We now focus on the initialization of the parameters of DNN by considering DBN which are generative graphical model. Thus, the initialization of the parameter of a DNN relies on those of the associated DBN. However, maximizing the likelihood of a DBN is impossible. Consequently variational methods based on RBM models have been developed and consists in training separately each layer of the DBN as an RBM. These models are described in our next paragraph.

2.1. RBM

An RBM is a probabilistic graphical model which connects a set of m visible random variables (r.v.), $\mathbf{v} = (v_1, \cdots, v_m)$, with a set of q hidden r.v., $\mathbf{h} = (h_1, \cdots, h_q)$ [14]. In this model, the joint probability density function (pdf) of the visible and hidden units depends on an energy function and reads

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})}, \tag{4}$$

where

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} w_{ij} v_i h_j, \tag{5}$$

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{6}$$

for a Bernoulli-Bernoulli RBM (BBRBM) (i.e. an RBM in which v_i and h_j take their values in $\{0, 1\}$). Equations related to a Gaussian-Bernoulli RBM (GBRBM) and which are more adapted for real values can be found in [9].

From an unlabeled visible dataset $(\mathbf{x}^1, \cdots, \mathbf{x}^N)$, our objective is to learn the parameters of the RBM. More precisely, starting from (4)-(6), we intend to maximize the likelihood $p(\mathbf{x}^1, \cdots, \mathbf{x}^N) = \prod_{i=1}^N p(\mathbf{x}^i)$, where

$$p(\mathbf{x}^i) = \sum_{\mathbf{h}} p(\mathbf{x}^i, \mathbf{h}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{x}^i,\mathbf{h})}, \tag{7}$$

w.r.t. a_i, b_j and w_{ij}.

However, the gradient of the log-likelihood,

$$\frac{1}{N}\sum_{k=1}^N \frac{\partial \log p(\mathbf{x}^k)}{\partial w_{ij}} = \frac{1}{N}\sum_{k=1}^N \sum_{h_j} p(h_j|\mathbf{x}^k) x_i^k h_j - \sum_{v_i, h_j} v_i h_j p(v_i, h_j), \tag{8}$$

is not computable but can be interpreted as the sum of two expectations. Consequently, (8) can be approximated by a Monte Carlo method. More precisely, in an RBM (4), one can show that

$$p(h_j = 1|\mathbf{v}) = \text{sigmoid}(b_j + \sum_i w_{ij} v_i), \tag{9}$$

where the sigmoid function $\text{sigmoid}(.)$ is defined in (2) and that

$$p(v_i = 1|\mathbf{h}) = \text{sigmoid}(a_i + \sum_j w_{ij} h_j). \tag{10}$$

Finally, the first expectation in (8) is easy to approximate by sampling according to $p(v_i|\mathbf{h})$; sampling according to $p(v_i, h_j)$ is more difficult but can be achieved via a Gibbs sampler in which we sample alternatively from $p(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^m p(v_i|\mathbf{h})$ and $p(\mathbf{h}|\mathbf{v}) = \prod_{i=j}^q p(h_j|\mathbf{v})$. In practice, the expectations are approximated with only 1 sample and the Gibbs sampler relies on 1 iteration. This procedure is called the Contrastive Divergence (CD-1) algorithm and leads to an approximate maximization of (8) via a gradient descent method.

2.2. Deep Belief Networks

Let us now consider the DBN probabilistic model defined by a layer of visible unit $\mathbf{x} = \mathbf{h}^0$ (our input data) and $p-1$ hidden layers, denoted $\mathbf{h}^1, \cdots, \mathbf{h}^{p-1}$. The pdf of $(\mathbf{h}^0, \mathbf{h}^1, \cdots, \mathbf{h}^{p-1})$ in a DBN reads

$$p(\mathbf{h}^0, \mathbf{h}^1, \cdots, \mathbf{h}^p) = \prod_{i=1}^{p-2} p(\mathbf{h}^{i-1}|\mathbf{h}^i) p(\mathbf{h}^{p-2}, \mathbf{h}^{p-1}), \quad (11)$$

where $p(\mathbf{h}^{p-2}, \mathbf{h}^{p-1})$ is an RBM and coincides with (4), and $p(\mathbf{h}^{i-1}|\mathbf{h}^i)$ is deduced from (10).

Again, the maximization of the likelihood $p(\mathbf{x})$ in model (11) w.r.t parameters $\mathbf{W}^k$ and $\mathbf{b}^k$ associated to each layer $\mathbf{h}^k$ is not possible. A greedy layer wise procedure has been proposed in the literature [13] [15] [16] and consists in approximating the DBN (11) as a stacking of RBM (4).

Strictly speaking, $p(\mathbf{h}^{k-1}, \mathbf{h}^k)$ in (11) does not satisfy (4) and so is not an RBM, except for $k = p-1$. However, justifications of the following procedure can be found in [16] [13] and relies on Kullback Leibler Divergence arguments.

In summary, the unsupervised training of a DBN (i.e. the pre-training of our DNN) consists of the following steps; starting from a training dataset $\{\mathbf{x}^1, \cdots, \mathbf{x}^i, \cdots, \mathbf{x}^N\}$:

1. train the first RBM $(\mathbf{h}^0, \mathbf{h}^1)$ (i.e. compute $\mathbf{W}^1$ and $\mathbf{b}^1$ associated to the first layer) via the procedure described in section 2.1;
2. compute the output associated to the data set $\{\mathbf{x}^1, \cdots, \mathbf{x}^i, \cdots, \mathbf{x}^N\}$ via (1)-(2) using the parameters $\mathbf{W}^1$ and $\mathbf{b}^1$ obtained after the pre-training;
3. train the next RBM $(\mathbf{h}^1, \mathbf{h}^2)$, $\cdots$, $(\mathbf{h}^{p-2}, \mathbf{h}^{p-1})$ by repeating steps 1. and 2.

3. FINE-TRAINING OF DNN

We now consider that the parameters estimated by the pre-training algorithm are used for the initialization of the supervised training algorithm of the DNN. So now we assume that we have a set of labeled data

$$E = \{(\mathbf{x}^1, \mathbf{d}^1), \cdots, (\mathbf{x}^i, \mathbf{d}^i), \cdots, (\mathbf{x}^N, \mathbf{d}^N)\} \quad (12)$$

where $\mathbf{d}^i = [d_1^i, \cdots, d_K^i]^T$ is the known class vector associated to $\mathbf{x}^i$ and K the number of classes: $d_{j=C}^i = 1$ if $\mathbf{x}^i$ belongs to the C-th class and $d_{j\neq C}^i = 0$ otherwise. Note that this set could be different from the one used in the previous section for the learning of the associated DBN.

Supervised training consists in tuning matrices $\mathbf{W}^k$ and biases $\mathbf{b}^k$ from the set E in (12). Here, our objective is to minimize the cross entropy $C = -\sum_{k=1}^K d_k \log(p_k)$ between the output of the DNN $\mathbf{p} = [p_1, \cdots, p_K]^T$ and the target probabilities $\mathbf{d} = [d_1, \cdots, d_K]^T$. The BP method is a popular algorithm to compute recursively the gradient of C w.r.t. the weights w_{ij}^k and the biases b_j^k of the DNN [12]. Finally, the algorithm includes a gradient descent method in order to approximate the parameters which minimize C.

In summary, from a given labeled data $(\mathbf{x}, \mathbf{d})$ and for a given iteration l:

1. Compute the output $\mathbf{p}$ associated to the input $\mathbf{x}$ by using weights $w_{ij}^k(l-1)$ and biases $b_j^k(l-1)$ of the previous iteration $l-1$. Remember that $w_{ij}^k(0)$ and $b_j^k(0)$ coincide with the parameters estimated by the pre-training step;
2. Backpropagate the gradient of the error C in the DNN, i.e. compute the gradient $\Delta w_{ij}^k(l)$ and $\Delta b_j^k(l)$ of C w.r.t the weights and the biases of the DNN.
3. Update the weights $w_{ij}^k(l) = w_{ij}^k(l-1) - \epsilon \Delta w_{ij}^k(l)$, and the biases $b_j^k(l) = b_j^k(l-1) - \epsilon \Delta b_j^k(l)$, where ϵ is the learning rate;

Many refinements have been proposed in order to improve the computation of the weights and the biases from the training set E in (12). These refinements rely on a random mini-batch to compute the gradient of the error and the momentum method to improve the speed of learning [17].

4. SIMULATIONS

4.1. Dataset

We present the results for audio context classification that we have obtained with DNN. The dataset that we use in this section is the publicly available audio scene dataset acquired by the LITIS Rouen [18]. The dataset is composed of 3026 recordings whose duration is 30 seconds in such a way that about 1500 minutes of audio scene have been recorded. The data are scattered in 19 classes : plane, bus, busy street, cafe, car, student hall, train station hall, kid game hall, market, metro-paris, metro-rouen, billard pool hall, quiet street, restaurant, pedestrian street, shop, train, high-speed train and tubestation. More details on the dataset can be found in [6].

Our experimentations are based on a standard feature set which consists in computing 12 MFCC, its first order derivatives (ΔMFCC) and 6 subband spectral flatness coefficients [19] for every 15ms-spaced frames of length 30ms.

To evaluate our system, we have followed the protocol proposed in [6] i.e. 80% of the examples were used for training while the remaining recordings were kept for testing (and results were averaged over 20 different splits of the dataset). As an evaluation criterion, we have considered the recognition rate. Since the classes are not represented by the same number of recordings, the recognition rate reads

$$\text{Rec\_Rate} = \frac{1}{C} \sum_{i=1}^{C} \frac{\text{TP}(i)}{\text{Card}(C(i))},$$

where $C = 19$, $\text{TP}(i)$ is the number of examples of class i correctly classified and $\text{Card}(C(i))$ the number of examples in class i. The final decision for a recording is taken by first

averaging the output of the DNN for each input frame which forms the recording and next choosing the class with the best result.

On one hand, our simulations aim at studying the performances of DNN for CASR in function of the number of hidden layers and the number of neurons for a given layer. On the other hand, we also study the effect of the number of the concatened frames of 30ms (1, 5, 10...) at the input of the DNN.

Finally, to be sure that our experiments are reproducible by others, we mention the specific parameters we have used for the learning procedures. The number of epochs for the pre-training and the fine-training is 100 and 300 respectively; the learning set is set to 1 for the pre-training and 0.1 for the supervised training; a batch size of 100 is used for both training procedure; the momentum is set to 0.5 for the first five epochs of both training and next to 0.9. Finally, a weight cost is set to 0.2×10^{-5}, for the pre-training. An interpretation of these parameters can be found in [17].

4.1.1. Influence of the size of the DNN

In this paragraph, we consider a fixed number of 15 input frames (so the size of the input layer of our DNN is $15 \times 30 = 450$) and we compare the performances in function of the number of hidden layers and the number of neurons for each hidden layer. For simplicity, we have considered the same number of neurons for each hidden layer. In figure 2 we have displayed the recognition rate in function of these parameters. Overall, DNN perform better when the number of hidden layers and neurons is greater but it can be seen that when the number of neurons is weak, increasing the number of hidden layers does not necessarily improve the performances. The worse recognition rate (80%) is obtained for 2 hidden layers of 50 neurons and the best performances (91.6%) are obtained with 5 hidden layers of 500 neurons. For reasons of space, we have not reproduced the confusion matrix associated to this configuration. Roughly speaking, all classes have a recognition rate greater than 80%, except the quiet street and the pedestrian street which have a recognition rate of 66.66% and 75%, respectively. Theses classes are mainly confused with the shop and market classes. For larger DNN, we have not observed a major improvement. Indeed, for a DNN with 7 hidden layers and 1000 neurons, the recognition rate is 92.2%.

4.1.2. Influence of the input of the DNN

We now set the number of hidden layers to 3 and the number of neurons to 50 and we perform a simulation in function of the number of input frames. Increasing the number of input frames has the advantage to reduce the computational cost when we need to do many classifications. From a computational cost point of view, it is clear that it is preferable to use $30 \times 15 = 450$ coefficients at the input of the DNN rather than doing 15 classifications with 30 coefficients. However, when the final decision is taken after several classifications, the mean error rate per recording is optimal for 10-15 frames as we see in Fig. 3.

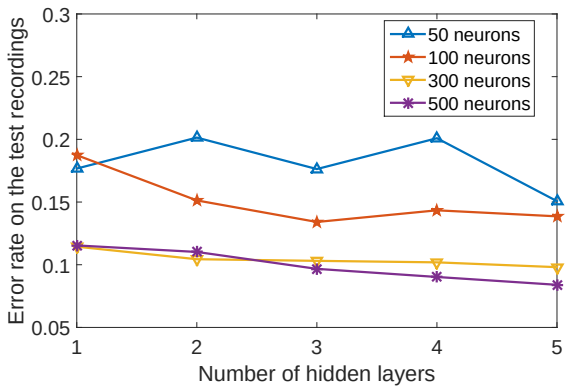

Fig. 2. Performances of DNNs for audio classification scene in function of the size of the DNN. Here, the number of input frames is 15.

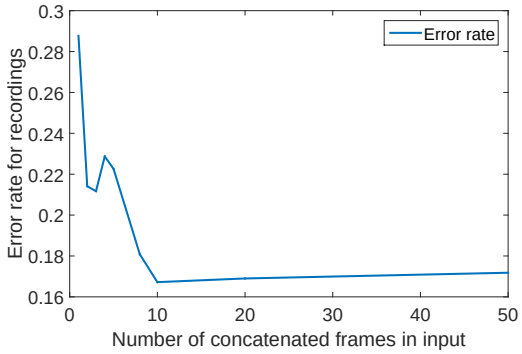

Fig. 3. Error rate in terms of recordings for the validation set in function of the number of input frames. The DNN has 3 hidden layers of 50 neurons.

We have also computed a classifier based on Gaussian Mixture models (GMM). We have trained 4 mixtures for each class via the Expectation Maximization (EM) algorithm. The best recognition rate for this classification method is 80.91% and is obtained by considering one input frame. It seems that the concatenation of frames is only interesting for architectures like DNN because they permit to encode the temporal connections between the frames. This observation is also confirmed by the fact that DNN present similar results if we do not consider the ΔMFCC which describe such connections. Finally, we have also computed a SVM classifier with a Gaussian Kernel and 15 concatenated input frames. The recognition rate is 86.5 %.

5. CONCLUSION

We have proposed a DNN-based approach for the CASR problem. The rationale of the training algorithms associated to DNN has been recalled and the performances of these architectures have been studied in function of their size and of the number of concatenated input frames, which define the input layer of the DNN. DNN have been compared with more classical classifiers such as GMM and SVM, with the same features: the optimal recognition rates obtained are 92%, 81% and 86.5%, respectively. The relevance of the features used in our simulations have not been discussed, but we underline that DNN applied to standard features give similar results as well-defined features (HOG) classified by an SVM approach [6] following the same protocol on the same dataset (best performance is 92% in both cases). In this way, several solutions could be exploited in order to improve the performances of DNN in this context. First, more adapted features (HOG features for example) could be used at the input of the DNN. Alternatively, we could also let the DNN extract automatically the relevant features by using directly the spectrum as input.

REFERENCES

[1] V. T. Peltonen, J. T. Tuomi, A. Klapuri, J. Huopaniemi, and T. Sorsa, "Computational auditory scene recognition," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002, pp. 1941–1944.

[2] A. Eronen, V. Peltonen, V. Tuomi, A. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi, "Audio-based context recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, pp. 321–329, 2006.

[3] L. Ma, B. Milner, and D. Smith, "Acoustic environment classification," *ACM Transactions on Speech and Language Processing*, pp. 1–22, 2006.

[4] M. Perttunen, M. Van Kleek, O. Lassila, and Riekki J., "Auditory context recognition using SVMs," in *Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM08),Valencia, Spain*, 2008, pp. 102–108.

[5] P. Khunarsal, C. Lursinsap, and T. Raicharoen, "Very short time environmental sound classification based on spectrogram pattern matching," *Information Sciences*, vol. 243, pp. 57–74, 2013.

[6] A. Rakotomamonjy and G. Gasso, "Histogram of gradients of time-frequency representations for audio scene classification," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 23, no. 1, pp. 142–153, 2015.

[7] G. Roma, W. Nogueira, and P. Herrera, "Recurrence quantification analysis features for environmental sound recognition," in *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013, pp. 1–4.

[8] S. Chu, S. Narayanan, and C.-C. J. Kuo, "Environmental sound recognition with time-frequency audio features," *IEEE Transactions on Audio, Speech and Langage Processing*, vol. 17, no. 6, pp. 1142–1158, 2009.

[9] G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, Mohamed A-R., N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Procesing. Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[10] A. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech and Langage Processing*, vol. 20, no. 1, pp. 14–22, Jan. 2012.

[11] H. A. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*, Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Neurocomputing: Foundations of research," chapter Learning Representations by Back-propagating Errors, pp. 696–699. MIT Press, Cambridge, MA, USA, 1988.

[13] G. E. Hinton, S. Osindero, and Y-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, July 2006.

[14] H. Larochelle and Y. Bengio, "Classification using discriminative restricted boltzmann machines," in *Proceedings of the 25th International Conference on Machine Learning*, 2008, pp. 536–543.

[15] C. Poultney, S. Chopra, and Y. Lecun, "Efficient learning of sparse representations with an energy-based model," in *Advances in Neural Information Processing Systems (NIPS) 2006*. 2006, MIT Press.

[16] B. Yoshua, *Learning Deep Architectures for AI*, Now Publishers Inc., 2009.

[17] G. E. Hinton, "A practical guide to training restricted Boltzmann machines.," in *Neural Networks: Tricks of the Trade (2nd ed.)*, Grgoire Montavon, Genevieve B. Orr, and Klaus-Robert Mller, Eds., vol. 7700 of *Lecture Notes in Computer Science*, pp. 599–619. 2012.

[18] "LITIS Rouen audio dataset," https://sites.google.com /site/alainrakotomamonjy/home/audio-scene.

[19] A. Ramalingam and S. Krishnan, "Gaussian mixture modeling of short-time Fourier transform features for audio fingerprinting," *IEEE Trans. Info. For. Sec.*, vol. 1, no. 4, pp. 457–463, 2006.