Audio Event Classification Using Deep Neural Networks

Zvi Kons, Orith Toledo-Ronen

IBM Research – Haifa, Haifa University Mount Carmel, Haifa 31905, Israel
{zvi,oritht}@il.ibm.com

Abstract

We present in this paper our work on audio event classification for outdoor events. As the main classification method we employ a deep neural network (DNN) and compare this to other classification methods. We propose a novel improvement to the pre-training process of the network which is useful when training with Gaussian data. Our experimental results are based on an audio corpus extracted from the FreeSound.org website repository. We show that the DNN has some advantage over other classification methods and that fusion of two methods can produce the best results.

Index Terms: audio classification, deep neural network, support vector machines, feature extraction

1. Introduction

Extracting various types of information from speech is a topic for considerable amount of research. On the other hand, there is much less work on audio events analysis. In the SMART EU project[1] we work on a framework for searching and analyzing of new types of information from multimedia data of the physical world. As part of this project, we would like to know what we could learn about the world around us using audio data.

The scope of this work is to extract some basic information from the audio sources, such as an indication of the presence of some audio events. This could be for example the information that an audio segment contains traffic or crowd noises. This audio event detection information can later be fused with additional data such as input from other audio and video sensors to produce higher reasoning.

There are several different approaches for audio event classification. The basic flow in most of them is to extract spectral based features and then apply some classification tools to distinguish between the classes. Simple approaches follow speech processing techniques with MFCC features and GMM based classification [1]. Other use features such as spectral features, LPC, perceptual features and mixtures of features. Different classifiers are also used where the common ones are: nearest neighbors, SVM, RBFNN and random forest [2],[3],[4],[5]. More complex techniques use audio pattern matching as in [6],[7].

Since each article tests its work with different audio data sets, different audio classes and different protocols, it is very hard to compare the different techniques and to determine which one is the best. Publicly available audio databases which can be used to evaluate various algorithms are common for the speech processing world but none is available for this type of audio event classification task. Our first task was therefore to create one.

Recently a significant improvement in speech recognition was achieved by using Deep Neural Networks (DNN) for phoneme classification [8]. In this article we turned to see if this approach is useful for audio classification too. We show how to use a DNN with the audio data and suggest improvements to the training process. We compare the DNN to other two classifiers. The first one is a Support Vector Machine (SVM) classifier. The second on is based on Gaussian Mixture Models (GMM).

Previous attempts to use DNN with audio data are described in [10] and [11]. In [10] the authors describe music genre classification using convolutional DNN. They report classification rate of about 70% for this task (no comparison to other classifiers is given). In [11] DNN are used to classify audio event during sport events. They achieve 72% classification rate which is somewhat worse than the SVM with 74%.

This paper describes our attempts to use DNN for audio event classification. The remainder of this paper is organized as follows. Section 2 describes the data set that we have created and used for our experiments. In Section 3 we present the DNN classification approach and our improvements to the training process. We also describe other classifiers that we have tested in comparison to the DNN. Section 4 then describes our experimental setup and presents our experimental results. Finally, in Section 5 we conclude with some discussion of the results.

2. Audio data

The audio data for our experiments was obtained from the FreeSound.org site[2], which is a repository of audio samples uploaded by users and covers a wide range of acoustic events. The samples selection was based on the tags that had been provided by the users. Most of the samples are of outdoor recordings with sounds from ordinary streets and different kinds of live events. A total of 84 files were obtained containing 228 minutes of audio. All samples were uncompressed, converted to one channel and downsampled to 22 KHz if needed.

The samples were then annotated manually. The annotation included locating and labeling segments with the relevant audio classes. Overlapping segments (with two or more audio classes at the same time) were allowed. Table 1 lists the four selected audio classes and the total amount of audio data contained in each class.

The list of the audio files along with their annotation can be made available by request from the authors.

[1]http://www.smartfp7.eu/

[2]http://www.freesound.org/

Class	Description	Length (minutes)
Crowd	Crowd of people	63
Traffic	Cars and road noises	37
Applause	Applause, yelling and cheering	27
Music	Various kinds of music recorded outdoor	29

Table 1: The content of the FreeSound data set including the description of the four audio classes and the corresponding amount of audio.

3. Audio classification

3.1. Audio features

The speech files are divided into 5-second segments with 50% overlap. For each segment we extract a vector of 192 features as follow. The segment itself is divided into frames. Each frame has length of 46 milliseconds and there is 50% overlap between the frames. The first 64 features are the mean of the MFCC over the entire segment. The second set of 64 features is their standard deviation (STD). The last set of 64 features is calculated by first taking the STD of the log spectrum over the frame and then applying the Mel-spaced filter banks. This type of feature helps differentiate between smooth spectrum and spectrum with peaks (e.g. musical tones).

This whole feature set was found to be the best after experimenting with several others, more conventional features.

3.2. Classification using DNN

The DNN classifier consists of a multilayer feed-forward perceptron network. Since it is very difficult to train the entire network using discriminative training, a generative pre-training stage is applied first to train one layer at a time. After the pre-training stage, the whole network is fine-tuned using a back propagation process.

The generative training is based on the restricted Boltzmann machine (RBM) training model [9]. The RBM is made of one layer network where the i'th input node v_i is connected with the j'th binary output node $h_j \in \{0,1\}$ using weights matrix W_{ij} and bias vectors b_j and a_i.

The training of the weights is an iterative process. We performed 200 iterations in our experiments. During each iteration, we calculate the hidden layer from the inputs using

$$P(h_j=1,v)=\sigma\left(\sum_i W_{ij} v_i + b_j\right), \quad (1)$$

where

$$\sigma(x)=\frac{1}{1+e^{-x}}. \quad (2)$$

A backward iteration then estimates the inputs back from the hidden layer using

$$P(\tilde{v}_i=1,h)=\sigma\left(\sum_j W_{ij} h_j + a_i\right) \quad (3)$$

for binary inputs and

$$\tilde{v}_i=\sum_j W_{ij} h_j + a_i \quad (4)$$

for Gaussian inputs. In our case, the inputs for the first layer are Gaussian and binary for the next layers. Finally, the hidden layer is calculated again using the reconstructed inputs from (3) or (4):

$$\tilde{h}_j=\sigma\left(\sum_i W_{ij} \tilde{v}_i + b_j\right). \quad (5)$$

The weights and bias terms are updated using gradient descent rules. For example, the gradient rule for the weights is based on the correlation between the inputs and the outputs:

$$\Delta W_{ij}=\langle v_i h_j\rangle - \langle \tilde{v}_i \tilde{h}_j\rangle, \quad (6)$$

where $\langle\cdot\rangle$ represents averaging over the available training data. For a detailed description of this training procedure see [9].

If we put aside the original derivation of this process from the RBM formulation, we can consider this as a generative model that tries to minimize the error between v and $\tilde{v}$. Under this perspective we notice that there is a large difference between the binary and the Gaussian cases. Because of (4) the scale of the weights is limited by the scale of the inputs. An attempt to increase the weight to signify a stronger connection would also increase the error between v and $\tilde{v}$. This doesn't happen in the binary case since the estimated values in (3) are always mapped to the region [0,1] using the σ function. Previous attempts tried to solve this by adding the variance into the RBM formulation [12].

In this work we present a novel approach by deviating from the RBM formulation and considering an asymmetric connection between the inputs and outputs. We modify (4) and add scaling factors s_j to the hidden layer nodes. Equation (4) is now converted into:

$$\tilde{v}_i=\sum_j W_{ij} s_j h_j + a_i. \quad (7)$$

The tuning of the new weights is done by taking the gradient direction that minimizes the difference between v and $\tilde{v}$:

$$\Delta s_j=\langle h_j \sum_i (v_i-\tilde{v}_i) W_{ij}\rangle. \quad (8)$$

The advantage of this method is that there is no need to change the calculation of the hidden layer in (1) so the same weights are used for the initialization of the neural network.

By applying this technique the pre-training error between v and $\tilde{v}$ for the Gaussian data was reduced by about 19%. We discuss the affect of those weights on the classification results in Section 4.2.

After the pre-training stage is complete we use the weights and biases ($W_{ij}^{(n)}$ and $b_j^{(n)}$) for each layer n to initialize a feed-forward neural network. We add a final classification layer with one output for each audio class. The network output is calculated as in eq. (1) except that the outputs are not quantized:

$$h_j^{(n)}=\sigma\left(\sum_i W_{ij}^{(n)} h_i^{(n-1)} + b_j\right), \quad (9)$$

where $h^{(0)}$ is the input vector and the output of the last layer $h_k^{(N)}$ represents the score for audio class k.

The weights are trained using the neural networks back propagation rule. We are interested in the EER point but the training data is unbalanced (e.g. 27 minutes of applause data vs. 201 minutes without applause). To fix that we give different weights to the error terms that are used in the calculation of the back propagation gradient. Those weights are inversely proportional to the size of each class.

3.3. Classification using SVM

We compared the DNN classifier to an SVM classifier [14]. We used an RBF kernel function (with $\gamma = 0.2$). For the other parameters we used the default values as configured in [15]. We trained a one-versus-all binary SVM classifier for each class by taking all the training data of the class as positive examples and all the rest of the data as negative examples. We normalized the input features to the range [0,1] with the scaling estimated from the training data and applied on both the training and the test data. We used the libSVM software (version 3.12) [15] for training and scoring the SVM with probability estimates prediction.

3.4. Classification using GMM

Another classifier we used in comparison to the DNN is a simple GMM classifier. We trained a GMM per class using the class data. The classification scores are the log-likelihood score of each class normalized by a score of a Universal Background Model (UBM). The UBM is a GMM that was trained using all the training data. This approach was successfully used for speaker recognition as described in [16]. However, in our system the UBM and the class models are trained independently without adaptation. We allow each class to have a different number of Gaussians, and optimize the number of Gaussians per class with cross validation. For each class we then train a separate UBM with the same number of Gaussians as in the class model. Because the GMM has diagonal covariance models, we first apply a Principal Component Analysis (PCA) transform to reduce the correlation between the features. We perform the PCA on the input features and select the first 50 components of the transformed feature vector as input for training the GMM. The PCA transformation was trained on the entire training set. We used the Speech Signal Processing Toolkit (SPTK) tool [17] to train and score the GMMs.

4. Experiments

We conducted the experiments using the data described in Section 2. Each classification method was tested 50 times in cross validation using the holdout evaluation method. In each evaluation the speech samples were randomly split into two subsets: 80% of the audio files for training and the rest 20% for testing. Since the holdout evaluation performance depends on the actual split between training and test, we perform random subsampling by repeating each evaluation 50 times and averaging the results. The same data splitting into training and testing subsets were used for the evaluation of all the classification methods.

The classification results were evaluated using the error rate at the equal error rate point (EER). The EER was calculated for each audio class independently and the results were averaged over the 50 cross validation evaluations. We also report the average error over all the audio classes. In addition to the average EER results, we provide the Standard Error (SE), which is the standard deviation divided by the square-root of the number of evaluations for both the per-class and the overall average results. A large part of the data is labeled with more then one class. Therefore, it is impossible to produce a confusion matrix.

4.1. Network structure

We first explored the DNN structure by modifying the number of nodes in the hidden layers. Our DNN has n=3 hidden layers, with the same number of nodes in each layer. The classification results as a function of the number of nodes in the hidden layers is shown in Table 2. As we can see, the class with the lowest EER is the Applause class, and the most difficult class to classify is the Music class. For easier comparison between the results we consider a simple averaged EER of all the classes.

We can now see that the optimal performance is achieved with 200 nodes. Therefore, in all our following experiments, the DNN will have 3 hidden layers with 200 nodes in each layer.

Audio Class	**50**	**100**	**200**	**300**
Crowd	19.67 ± 0.98	18.06 ± 1.11	17.79 ± 0.92	19.81 ± 1.18
Traffic	15.73 ± 0.58	15.53 ± 0.67	15.71 ± 0.67	15.37 ± 0.60
Applause	9.10 ± 1.01	8.8 ± 1.00	8.04 ± 0.85	9.05 ± 1.06
Music	23.78 ± 1.21	23.04 ± 1.23	21.61 ± 1.16	22.15 ± 1.08
Average	**17.07 ± 0.62**	**16.36 ± 0.62**	**15.79 ± 0.57**	**16.60 ± 0.61**

Table 2: Average EER ± SE in percent as a function of the number of nodes in the hidden layers of the DNN.

4.2. Scaling factors

We have tested what is the impact of the new scaling factors (eq. (7)) by doing the test once with all $s_j = 1$ and then repeating the test and allowing adjustments of the scales by (8). The main impact of the scaling is in reducing the pre-training error between v and $\tilde{v}$ of eq. (8). For the Gaussian data the error was reduced by 19%. This helps to reduce the number of iterations needed by the back-propagation until convergence. This number was reduced by 7%. These results are shown in Table 3.

In terms of the classification performance, the results are displayed in Table 4. As can be seen from this table, the effect of the new technique on the classification results is small (compared to the standard error of the results). By introducing the new scaling factors into the pre-training process a minor degradation in the classification performance is observed. It seems that the back-propagation algorithm is able to achieve similar goals even if the starting point is less optimal.

We expect that this technique will be more useful for reducing the computation time when larger amounts of data are used for the training and larger networks are used for classification [13].

Parameter	No Scaling	Scaling
RBM error	0.489	0.397
BP iterations	43	40

Table 3: RBM training error after 200 iterations and Back Propagation training iterations without and with the scaling factors in the RBM training of the DNN.

Audio Class	No Scaling	Scaling
Crowd	17.79 ± 0.92	18.84 ± 1.03
Traffic	15.71 ± 0.67	15.97 ± 0.75
Applause	8.04 ± 0.85	8.76 ± 1.14
Music	21.61 ± 1.16	20.51 ± 1.14
Average	**15.79 ± 0.57**	**16.02 ± 0.60**

Table 4: Average EER ± SE in percent without and with the scaling factors in the RBM training of the DNN.

4.3. Comparing with SVM and GMM

In our next experiment, we compare the performance of the DNN to the other two classifiers described in Section 3. Table 5 shows the classification results of the DNN classifier in comparison with the GMM and SVM classifiers. As we can see, the GMM has worse performance than the other two classifiers, while the SVM and DNN classifiers have comparable performance. The weakness of the GMM classifier for the audio event detection task that we see in our experiment is in line with the findings in [18]. The DNN classifier achieves the best overall performance in most of the per-class results except for the Music class for which the SVM performs better.

Audio Class	GMM	SVM	DNN
Crowd	24.48 ± 0.98	19.82 ± 1.15	17.79 ± 0.92
Traffic	23.85 ± 0.96	16.38 ± 0.77	15.71 ± 0.67
Applause	15.48 ± 1.33	10.18 ± 0.81	8.04 ± 0.85
Music	23.74 ± 1.35	17.77 ± 1.15	21.61 ± 1.16
Average	**21.89 ± 0.64**	**16.04 ± 0.55**	**15.79 ± 0.57**

Table 5: Average EER ± SE performance comparison between three classifiers: GMM, SVM, and DNN.

4.4. Score Fusion

We perform a simple fusion at the score level. A new score is generated by adding the final scores obtained from the DNN and the final score of the SVM. The EER is then calculated using this new fused score.

The results of the fusion are shown in Table 6. From these results we learn that the score fusion between the DNN and SVM classifiers provides 6.5% improvement relative to the DNN alone. Most of the gain in the fusion is achieved by improvement of the Music classification. Similarly, if we look at the relative improvement of the fused score compared to the SVM, most of the improvement is achieved in the Applause class with overall relative improvement of 8.0%.

Audio Class	DNN+SVM
Crowd	18.04 ± 1.07
Traffic	14.88 ± 0.63
Applause	7.89 ± 0.79
Music	18.21 ± 1.08
Average	**14.76 ± 0.54**

Table 6: Average EER ± SE in percent for score fusion of the DNN and SVM classifiers.

5. Conclusions

In this paper we have experimented with audio event classification using different types of classifiers. We have found that a DNN classifier is very useful for this task and performs a little better than SVM. A small additional improvement can be gained by combining the DNN and the SVM scores.

We proposed a new method for improving the pre-training process of the network by introducing new scaling factors in the reconstruction step of the RBM training. This method seems to be useful for reducing the pre-training error rate and helps the back-propagation to converge faster, with only minor degradation in classification performance.

The current error rates are a bit high for practical use. The high correlation between the different classifiers suggests that the main problem is not the classification technique. Additional improvement would require better features. We would probably need some features that can better describe the time dependencies in the signals. Another direction to improve the results would be to integrate the output from several segments.

6. Acknowledgments

This work is part of the SMART EU project (FP7-287583, http://www.smartfp7.eu/) partially funded by the European commission in the scope of the 7th ICT framework.

7. References

[1] Aronowitz, H., "Segmental Modeling for Audio Segmentation," *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 4, pp.393-396, 15-20 April 2007.

[2] P. Dhanalakshmi, S. Palanivel, and V. Ramalingam. 2009. "Classification of audio signals using SVM and RBFNN". Expert Syst. Appl. 36, 3 (April 2009), 6069-6075.

[3] McKinney, M. F., Breebaart, J. (2003). "Features for audio and music classification." Proc. 4th Int. Conf. on Music Information Retrieval.

[4] Zixing Zhang; Schuller, B.; , "Semi-supervised learning helps in sound event classification," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp.333-336, 25-30 March 2012.

[5] Bach, J.; Meyer, A.; McElfresh, D.; Anemuller, J.; , "Automatic classification of audio data using nonlinear neural response models," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp.357-360, 25-30 March 2012.

[6] Burred, J.J.; , "Genetic motif discovery applied to audio analysis," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp.361-364, 25-30 March 2012.

[7] Kumar, A.; Dighe, P.; Singh, R.; Chaudhuri, S.; Raj, B.; , "Audio event detection from acoustic unit occurrence patterns," *IEEE International Conference on Acoustics, Speech and Signal Processing* , pp.489-492, 25-30 March 2012.

[8] Hinton, G. *et al*, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups" IEEE Signal Processing Magazine, 2012.

[9] Hinton, G. E., Osindero, S. and Teh, Y., "A fast learning algorithm for deep belief nets.", Neural Computation, 2006.

[10] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng., "Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks." Communications of the ACM, vol. 54, no. 10, pp. 95-103, 2011.

[11] Ballan, Lamberto, et al. "Deep networks for audio event classification in soccer videos." Multimedia and Expo, ICME 2009.

[12] Cho, KyungHyun, Alexander Ilin, and Tapani Raiko. "Improved learning of Gaussian-Bernoulli restricted Boltzmann machines." Artificial Neural Networks and Machine Learning–ICANN 2011 : 10-17.

[13] T. N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak and A. Mohamed, "Making Deep Belief Networks Effective for Large Vocabulary Continuous Speech Recognition," in Proc. ASRU, December 2011.

[14] Burges, C. J. C., "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, 2, 121–167, 1998.

[15] Chang, C.-C. and C.-J. Lin. *"LIBSVM: a library for support vector machines"*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[16] D. A. Reynolds, T. F. Quatieri ,R. B. Dunn, "Speaker verification using adapted Gaussian mixture models", *Digital Signal Processing*, Vol. 10, No.1-3, 2000.

[17] "Speech Signal Processing Toolkit (SPTK) version 3.6," Software available at http://sp-tk.sourceforge.net/.

[18] Li Lu,Fengpei Ge,Qingwei Zhao,Yonghong Yan, "A SVM-based Audio Event Detection System", International Conference on Electrical and Control Engineering, 2010.