ACOUSTIC SCENE CLASSIFICATION USING DEEP LEARNING

Rohit Patiyal, Padmanabhan Rajan

School of Computing and Electrical Engineering
Indian Institute of Technology Mandi
Himachal Pradesh, INDIA
rohit21122012@gmail.com, padman@iitmandi.ac.in

ABSTRACT

Acoustic Scene Classification (ASC) is the task of classifying audio samples on the basis of their soundscapes. This is one of the tasks taken up by Detection and Classification of Acoustic Scenes and Events 2016 (DCASE-2016) challenge. A labeled dataset of audio samples from various scenes is provided and solutions are invited. In this paper, use of Deep Neural Networks (DNN) is proposed for the task of ASC.

Here, different methods for extracting features with different classification algorithms are explored. It is observed that DNN works significantly better as compared to other methods trained over the same set of features. It performs at par with the state-of-the-art techniques presented in DCASE-2013.

It is concluded that the use of MFCC features with DNN works the best, giving 97.6 % cross-validation score on the development dataset-2016 data for a particular set of parameters for the DNN. Also training a DNN does not take larger run times compared to others methods.

***Index Terms*—** Audio Scene, Machine Learning, Deep Neural Networks

1. INTRODUCTION

This paper serves as the technical report for the accompanied submission from IIT Mandi, India to the Acoustic Scene Classification task of DCASE Challenge 2016. The task involves predicting the scene label of an unknown audio sample. The system responsible to perform the task is to be trained using the dataset provided in the DCASE challenge. This dataset contains labelled audio samples of different classes which can be used to perform supervised learning to train a classifier model.

In our submission, Deep Neural Networks (DNN) classifier with Mel Frequency Cepstral Coefficients (DNN) features is used. Results of various experiments is reported in this paper where other features are explored. Along with that, DNN parameters are also varied to achieve better accuracies. Rationale for using MFCC and DNN is well supported by the experimental results obtained.

Note that all results are reported as per the cross-validation setup provided in DCASE-2016.

2. FEATURES EXPLORED

The following feature sets have been explored for the task.

2.1. Mel-Frequency Cepstral Coefficients (MFCC)

MFCC collectively represent the short-term power spectrum of a sound sample.

These are widely used as features in applications like speech recognition, speaker recognition, genre classification and similarity measures in music information retrieval. These are sensitive to noise and not speech, so are more appropriate to be applied here in the case of scene classification.

MFCCs are obtained as follows:[1]

1. Take the Short Time Fourier Transform of an audio sample.
2. Map the power spectrum obtained above onto the mel scale by multiplying the spectrum with mel-scaled triangular overlapping filters (mel filter banks)
3. Take the log of mel spectrum obtained above
4. Take the discrete cosine transform (DCT) of the log spectrum to obtain required number of coefficients.
5. MFCCs are the amplitudes output of the DCT.

In the implementation, the librosa library from the baseline is used to extract the features.

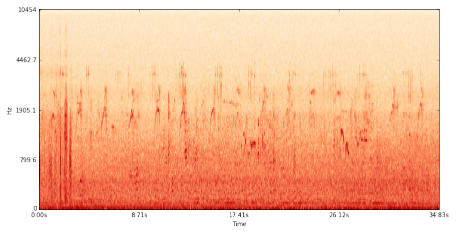

Figure 1: MFCC Features plotted as spectrum

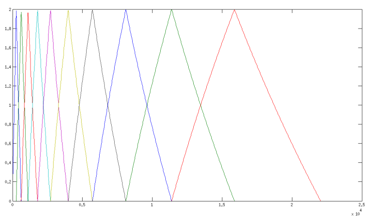

Figure 2: Mel scaled filter banks

In figure 2, each column in the plot represents the value of the MFCC coefficient in a particular frame. The shape of the mel filters is shown in figure 3 which leads to more resolution in lower frequency and less resolution in the higher frequency range.

2.2. Linear Frequency Cepstral Coefficients (LFCC)

LFCC are similar to MFCC but the only difference is that the features are not mel scaled but distributed uniformly. In this all the steps are done similar to MFCC but only the step of multiplying the spectrum with mel filter banks is changed. Here the filter banks are linear in the frequency scale [2] . Features are extracted using voicebox tool [3].

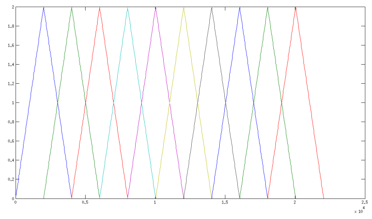

Figure 3: Linear scaled filter banks

2.3. Antimel Frequency Cepstral Coefficients (AntiMFCC)

AntiMFCC are just the opposite of MFCC features. Here, the difference is that the spectrum is multiplied with filter banks which are just the reverse of mel filter banks along the frequency axis i.e. narrower triangles (higher resolution) at the higher frequency and wider (lower resolution) in the low frequency range [2]. Features are extracted using voicebox tool [3].

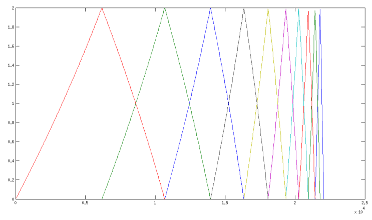

Figure 4: Antimel scaled filter banks

2.4. Chroma Features

Chroma features are powerful representation for music audio. In this, the entire spectrum is projected onto bins representing distinct semitones (or chroma) of the musical octave [4].

This is considered in order to compare with other frequency scaling technique observed above which are mel, linear, and antimel scaling.

Since, in music, notes exactly one octave apart are perceived as similar, knowing the distribution of chroma features even without the absolute frequency (i.e. the original octave) can give useful musical information about the audio and may even reveal perceived musical similarity that is not apparent in the original spectra [4]. Features are extracted using the default librosa library.

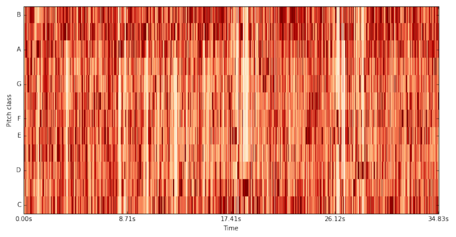

Figure 5: Chroma Features plotted as spectrum

2.5. All-Pole Group Delay Features (APGDF)

In features like MFCC, which are derived from short term magnitude spectrum, the phase spectrum remains unused. A useful representation of the phase is the all-pole group delay function [5].

The group delay function is defined as the negative derivative of the phase spectrum. All pole Group Delay Features are derived as follows:

1. Perform all-pole modeling on the audio frames with a prediction order of say p = 20 and obtain the filter coefficients a(κ).
2. From the a(κ), form the frequency response H(ω).
3. Compute the group delay function by taking the negative derivative of the phase response of H(ω). In practice, the derivative is computed using the sample-wise difference.
4. Take DCT on the group delay function.

Implementation is done using the librosa library.

3. CLASSIFICATION METHODS

3.1. Gaussian Mixture Model (GMM)

A GMM based classifier is a classification technique in which each class is modeled using a mixture of multivariate Gaussian distributions components. The components are identified by their mean vectors and covariances matrices and, the mixture distribution is obtained by adding the components with weights termed as mixing coefficients. [6]

The parameters for each component are estimated from the data using iterative Expectation Maximization (EM) algorithm which maximizes the likelihood or using Maximum A Posteriori (MAP) method from a model with priors. [6] The expression for multivariate Gaussian distribution is :

$$f(x) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(x-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu})\right)$$

Here $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean vector and the covariance matrix of the data feature values respectively. The baseline system uses the GMM from the scikit-learn library [6]

3.2. Deep Neural Network from TensorFlow

Tensorflow is an open source software library for numerical computation using data flow graphs. It provides a simplified interface similar to the popular scikit-learn python library [6], following the fit-predict model, named TF learn / skflow.

Deep learning API of TF learn / skflow allows users to manipulate the number of layers, nodes, iterations, etc. [7]

A Deep Neural Network is a supervised learning technique that learns a function $f(\cdot) : R^m \rightarrow R^o$ with the help of training data set, where m is the number of dimensions for input and o is the the number of dimensions for output. For a data instance having features $X = x_1, x_2, ..., x_m$ and a target y, the neural network learns a non-linear function approximator for the task of classification. This is done for all training i nstances and the weights between nodes of the network are calculated using algorithms like backpropagation, AdaGrad, or other optimizers. Any test sample is passed on through the network to get the output as the posterior probabilities for the targets. [6]

4. SETUP USED

4.1. Hardware Setup

To perform the experiments, a single computer is used and the run time benchmarks are based on it. The specifications of the machine are :

Component	Value
CPU cores	32, Intel Xeon
CPU Clock	2.00 GHz
RAM	48228 MB

Table 1: Hardware used for the experiments

4.2. Methodology

The process followed to obtain the evaluation results as given in the baseline system is shown in fig 6. At the time of training, each audio sample is divided into frames and features are extracted from each frame. Features from the frames are then given to a classifier, DNN in this case, where it learns the weights with the help of the corresponding training label.

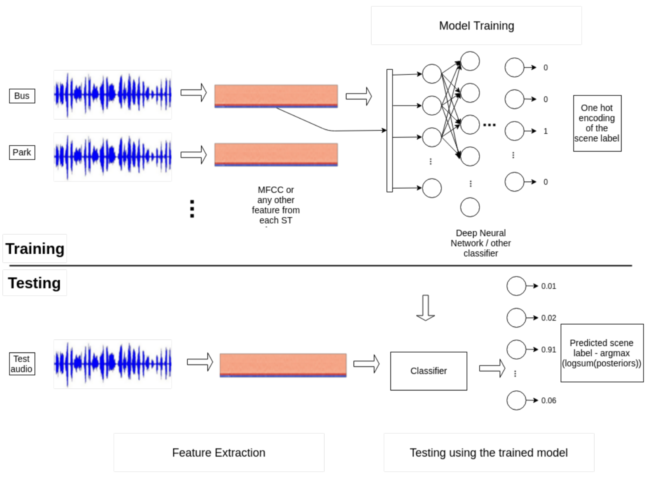
Model Training
Bus
Park
MFCC or any other feature from each ST
0
0
1
0
One hot encoding of the scene label
Deep Neural Network / other classifier
Training
Testing
Test audio
Classifier
0.01
0.02
0.91
0.06
Predicted scene label - argmax (logsum(posteriors))
Feature Extraction
Testing using the trained model

Figure 6: Methodology for system evaluation

When an unknown sample is to be tested, a similar procedure is followed but this time, probability for a frame belonging to each class is calculated using the classifier model and then these probabilities are aggregated for the complete audio sample using logadd operation. The label of the class with the highest score is assigned to the audio sample.

5. EXPERIMENTAL RESULTS

A set of experiments were performed in order to finalize with a system to be proposed for the challenge. DCASE-2016 acoustic scenes dataset is used and the results are obtained using the provided development mode cross-validation scheme.

5.1. Results using Baseline MFCC-GMM Classifier

The given baseline implementation uses MFCC features and Gaussian mixture models to do the classification.

Here, in Table 2, accuracy of this is reported, where we can see that changing the number of mixture models does not affect the accuracy. Also the run time increases significantly for larger number of mixture models.

Mixtures	Accuracy	Run Time (mins)
4	68.7	11.91
8	70.0	45.42
16	70.0	65.71
32	69.5	82.65
64	68.6	326.16
128	68.1	698.59

Table 2: Performance of baseline system with different number of mixture models.

5.2. Results using Different Features with DNN

In order to understand the effect of features for the task, different feature sets have been observed with the same classifier. The classifier considered for this task is a DNN with 3 layers each having 100 nodes and is trained with a learning rate of 0.01 in batches of size 1024 and number of steps for training is kept as 1000. This is done using using the skflow APIs from the tensorflow library

In Table 3, the performances of DNN with different features has been shown. We can clearly see that MFCC performs far better as compared to other features. Also, cepstral coefficients obtained using linear and antimel filter banks, i.e. LFCC and AntiMFCC are better than others but not MFCC.

Features	Accuracy
MFCC	84.2
LFCC	78.4
AntiMFCC	63.7
Chroma Features	40.8
APGD Features	39.0

Table 3: Performance of different feature sets with DNN of size [100,100,100] with learning rate as 0.01, batch size as 1024 and number of steps as 1000.

5.3. Results using MFCC-DNN Classifier

As observed in earlier experiments 3, MFCC outperforms as compared to other features. So, further experiments are performed using MFCC along with different DNN sizes.

As evident from Table 4, using a DNN as the classifier gives a significant improvement from the baseline accuracy. The size and parameters of the network is varied to obtain optimal accuracy in the development mode. The run time observed is also less as compared to the baseline system.

Layers	Nodes/layer	Steps	Accuracy	Run Time (mins)
2	100	2000	86.6	10.92
2*	500	4000	91.1	25.21
3	1000	2000	92.9	71.33
3*	1000	2500	97.6	58.62
3	1000	3000	94.5	64.81
3*	1000	5000	93.3	111.61
3	2000	3500	92.5	234.132

Table 4: Performance of MFCC-DNN system with different parameter values. The star in some of the rows represent use of exponential decay in place of a constant value for the learning rate.

6. SETUP OF SUBMISSION SYSTEM PROSPOSED

The submission system proposed uses MFCC features with the following parameters which are to be passed into DNN of a specific configuration as shown next in Table 6. The MFCC parameters used are shown below :

Parameter	Value
Window	Hamming asymmetric
No. of mfcc coeff.	20
No. of mel bands	40
FFT length	2048
fmin	0
fmax	22050
htk	false
delta & acc. width	9

Table 5: MFCC feature parameter values used in the proposed system.

It is to be noted that these are no different from the baseline system. There are no significant improvements observed when the values are changed. On the other hand, as observed in Table 4, the DNN structure parameters do matter and the following configuration works the best among others. The parameter values for the DNN proposed are summarized in the following Table 6.

Parameter	Value
Layers	3
Nodes per Layer	1000
Steps	2500
Learning Rate	Exp. decay (0.1-0.01)
Batch size	1024

Table 6: DNN parameter values used in the proposed system.

7. RESULTS OF SUBMISSION SYSTEM PROSPOSED

The results obtained from the proposed system using the development mode cross-validation setup is shown below in Table 7

Here column 1 represents the scene label of the audio. The second represents the number of samples of the corresponding scene and third represents the number of samples predicted by the system to be of the scene. The fourth is the accuracy calculated as per the challenge rules, which is, in principle, the **recall** value obtained for each scene.

Scene label	Nref	Nsys	Accuracy
beach	78	78	100.0 %
bus	78	81	100.0 %
cafe/restaurant	78	83	100.0 %
car	78	76	97.5 %
city center	78	77	98.9 %
forest path	78	78	100.0 %
grocery store	78	76	97.4 %
home	78	80	97.2 %
library	78	74	92.6 %
metro station	78	78	100.0 %
office	78	80	100.0 %
park	78	78	98.8 %
residential area	78	77	98.7 %
train	78	67	85.3 %
tram	78	87	97.5 %
Overall accuracy	1170	1170	97.6 %

Table 7: Accuracy of the proposed system for each scene and overall.

Most of the samples are classified correctly except those of the confusing classes, e.g. in train and tram. The accuracies in each fold is show in Table 8

CV Step	Accuracy
Fold 1	98.7 %
Fold 2	96.8 %
Fold 3	95.5 %
Fold 4	99.3 %
Overall	97.5 %

Table 8: Accuracy of the proposed system for each fold and overall.

8. CONCLUSION

In conclusion, Deep Neural Networks with MFCC works significantly better than the baseline system. The accuracies range from 86.6 % to 97.6 % when DNNs of different sizes and parameters are used. The feature extraction parameters does not affect the accuracies much. Also, particular set of values of parameters of DNN, as reported in table 6, performs better as compared to other values. DNN with 3 layers with each layer containing 1000 units trained in batches of 1024 using an exponentially decaying learning rate of 0.1 to 0.01 in 2500 steps performs the best, giving a recall accuracy of 97.6 % on DCASE-2016 dataset in CV development mode.

9. REFERENCES

[1] M. Sahidullah and G. Saha, "Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition," *Speech Communication*, vol. 54, no. 4, pp. 543–565, 2012.

[2] H. Lei and E. L. Gonzalo, "Mel, linear, and antimel frequency cepstral coefficients in broad phonetic regions for telephone speaker recognition." in *INTERSPEECH*, 2009, pp. 2323–2326.

[3] M. Brookes *et al.*, "Voicebox: Speech processing toolbox for matlab," *Software, available [Mar. 2011] from www. ee. ic. ac. uk/hp/staff/dmb/voicebox/voicebox. html*, vol. 47, 1997.

[4] D. Ellis, "Chroma feature analysis and synthesis." [Online]. Available: http://labrosa.ee.columbia.edu/matlab/chroma-ansyn/

[5] A. Diment, R. Padmanabhan, T. Heittola, and T. Virtanen, "Group delay function from all-pole models for musical instrument recognition," *Computer Music Multidisciplinary Research 2013, Revised Selected Papers, Lecture Notes in Computer Science, Springer Verlag*, 2014.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/