ACOUSTIC SCENE CLASSIFICATION BY ENSEMBLING GRADIENT BOOSTING MACHINE AND CONVOLUTIONAL NEURAL NETWORKS

Eduardo Fonseca, Rong Gong, Dmitry Bogdanov, Olga Slizovskaia, Emilia Gomez, Xavier Serra

Music Technology Group, Universitat Pompeu Fabra, Barcelona

name.surname@upf.edu

ABSTRACT

This work describes our contribution to the acoustic scene classification task of the DCASE 2017 challenge. We propose a system that consists of the ensemble of two methods of different nature: a feature engineering approach, where a collection of hand-crafted features is input to a Gradient Boosting Machine, and another approach based on learning representations from data, where log-scaled mel-spectrograms are input to a Convolutional Neural Network. This CNN is designed with multiple filter shapes in the first layer. We use a simple late fusion strategy to combine both methods. We report classification accuracy of each method alone and the ensemble system on the provided cross-validation setup of TUT Acoustic Scenes 2017 dataset. The proposed system outperforms each of its component methods and improves the provided baseline system by 8.2%.

Index Terms— acoustic scene classification, gradient boosting machine, convolutional neural networks, ensembling

1. INTRODUCTION

Humans have the ability to identify the environment where they are (e.g., park or beach) and recognize acoustic events that occur around them (e.g., a baby crying or a car passing by) from the incoming sounds they perceive. However, these tasks are not trivial for machine listening systems that attempt to accomplish them. The computational analysis of environmental sounds to automate tasks like the ones mentioned has recently received growing attention from the research community. The consecutive editions of the IEEE AASP Challenges Detection and Classification of Acoustic Scenes and Events“ (DCASE) provide the scenario where to evaluate and benchmark different approaches for acoustic scene classification and acoustic event detection [1]. In particular, DCASE 2017 challenge comprises four tasks: acoustic scene classification (task 1), detection of rare sound events (task 2), sound event detection in real life audio (task 3), and large-scale weakly supervised sound event detection for smart cars (task 4) [2]. This work concerns task 1, i.e., Acoustic Scene Classification (ASC), that can be defined as the task of associating a label to an audio stream thereby identifying the particular context or environment where the audio stream was generated [1]. The acoustic scene hence consists of all the acoustic information that is typically present in a given context, including background noises and specific acoustic events. ASC can trigger applications that range from audio collections management [3] and intelligent wearable interfaces [4] to the development of context-aware applications [5].

Traditionally, ASC systems have been based on a two-stage approach where *i)* pre-designed features or descriptors are extracted from the audio signal and *ii)* they are utilized as input to a classifier. This *feature engineering* based approach relies heavily on the capacity of the features to capture relevant information from the audio signal for the task under consideration, which may require substantial expertise and effort. One of the most popular hand-crafted features in ASC are cepstral features, e.g., MFCCs, which have been taken from the speech recognition field and have been widely utilized for ASC [6, 7, 8]. Also, a number of low-level features computed either from the time or frequency domain (e.g., zero-crossing rate or spectral centroid) have been utilized [7]. Some typical examples of classifiers used for this task are GMM [6] and SVM [8], the latter being used in the winning system for DCASE 2013 challenge.

As opposed to the previous approach that relies on handcrafting features, other techniques are based on *learning representations* from data. In particular, deep learning has recently become a widespread approach among the audio research community. In this case, the system is able to learn an internal representation from a simpler one at the input (typically, a time-frequency representation, e.g., spectrogram), and hence the two stages described before (feature engineering and classifier) are optimized jointly. Among the various deep learning approaches available, Convolutional Neural Networks (CNNs) have proved to be effective for several audio related tasks, e.g., speech recognition [9], automatic music tagging [10] or environmental sound classification [11]. In the specific case of ASC, several well-ranked submissions in the DCASE 2016 challenge were CNN-based, e.g., [12, 13]. Also in the context of DCASE 2016 challenge, a number of highly ranked submissions were based on the ensemble of different models, including the winning system [14], where the scores of a feature engineering based method (MFCC & i-vectors) were fused with those of a feature learning based method (CNN).

In this paper, we present a system for ASC that leverages both of the approaches presented above. On the one hand, a number of low-level time- and frequency-based audio features are extracted and input to a classifier. We decided to use Gradient Boosting Machine (GBM) due to its high performance as the winning solution in Kaggle challenges.[1] On the other hand, a CNN learns features from a log-scaled mel-spectrogram representation of the audio signal. By combining two methods of different nature, our intention is to obtain a system that takes advantage of the complementary information that they provide. The remainder of this work is organized as follows. Section 2 describes the methods (GBM and CNN) that compose our proposed system, as well as the late fusion strategy utilized. In Section 3 we present the dataset used and the evaluation setup that we follow. Results for the different methods (GBM, CNN and esemble) are presented in Section 4 and we end this work with the conclusions in Section 5.

[1] https://www.kaggle.com/

2. SYSTEM DESCRIPTION

2.1. Gradient Boosting Machine

Gradient boosting machine [15] is a powerful technique for building predictive models. It selects a loss as the objective function, and uses the additive model of many weak learners—typically regression trees—to minimize the loss. The parameters of added trees are tuned by a gradient descent algorithm. There are two GBM frameworks which are used widely in the data science community: XGBoost [16] and LightGBM.[2] The former is very popular among Kaggle community where it has been used for many competitions. The latter is a newcomer, which includes several improved features:

- It uses histogram based algorithms, which aggregate continuous features into discrete bins, to speed up training and reduce memory usage.
- It grows the tree by leaf-wise, which can reduce more loss than the level-wise algorithm.

In our experiments, we also found that LightGBM is faster than XGboost on training and achieves a slightly better overall classification accuracy. In consequence, we choose LightGBM as the GBM framework for the experiment.

2.1.1. Feature Extraction and Pre-processing

To consider the temporal characteristics, we segment each recording of 10s into 10 equal length non-overlapped sequences. We then extract features on each sequence using *FreesoundExtractor*,[3] a feature extractor from Essentia open-source library for audio analysis [17]. This extractor is originally used by Freesound[4] in order to provide sound analysis API and search by similar sounds functionality. It allows calculating hundreds of sound and music features. However, we discard some music-related features in rhythm, key, chords and tonal categories since we do not observe much musical trait in the development dataset. We further discard some feature statistics such as histogram and covariance matrix due to their high dimensionality and sparsity. The selected features and their dimensionality are listed in Table 1. The features are calculated on frame-level by using a 4096 samples frame size and a 2048 samples hop size. All other parameters are set to FreesoundExtractor default values. We then perform four statistical aggregations—mean, variance, mean of the derivative and variance of the derivative—to the frame-level feature vector of each sequence. Finally, a $\mathbb{R}^{820\times1}$ (205×4) feature vector is output for each sequence. In the cross-validation experiment (Section 3), we fit a mean and variance standardization scaler for each fold by using the features of the training dataset, which is then used for scaling the training and test set. In the final prediction step, we fit a standardization scaler for the whole development dataset and then apply it to the evaluation dataset.

2.1.2. LightGBM Parameters

Since ASC is a multiclass classification problem, we use logarithmic loss as the objective function, which yields a $\mathbb{R}^{15\times1}$ prediction probability for each sequence (considering 15 acoustic scenes). The three most important parameters are set as *i)* Learning rate: 0.05, *ii)* Number of trees: 500, and *iii)* Number of leaves: 255. All other parameters are default values. All parameters are held unchanged through the 4-fold cross-validation experiment and the model for the prediction of the evaluation dataset.

We keep all 820 dimensions features because according to our pilot experiment, removing irrelevant features only affected the training speed rather than improving the prediction accuracy. The parameter tuning process has also been simplified because several techniques in LightGBM such as *weak learner*, *Taylor approximation of the loss function* and *bagging*, make the system robust to over-fitting [18].

Table 1: Selected features extracted by *FreesoundExtractor*. Dim: dimensionality.

Feature name	Dim	Feature name	Dim
Bark bands energy	32	Tonal features	3
ERB bands energy	23	Pitch features	3
Mel bands energy	45	Silence rate	3
MFCC	13	Spectral features	32
HPCP	38	GFCC	13

2.2. Convolutional Neural Networks

CNNs appear to be a reasonable choice for this task for various reasons. First, if they are presented with a time-frequency representation of audio, they are able, in theory, to capture spectro-temporal modulation patterns that can be relevant to identify the different acoustic scenes. Furthermore, when the input to the CNN is a time-frequency representation, the width and height dimensions of the convolutional filters can be related to the time and frequency axes, respectively.

2.2.1. Input Representation and Pre-processing

We use log-scaled mel-spectrogram as the input representation to the CNN. To compute it, first, the 2-channel wav files are down-mixed to mono, and short-time Fourier transform (STFT) is applied using Hamming windows of 40 ms with 50% overlap. After calculating its power, a mel filter bank is applied consisting of 128 bands ranging from 0 to 22050 Hz (the sampling rate being 44.1 kHz) according to Slaney's formula [19]. Following results reported in [20], we use a filter bank with triangular filters in the frequency domain presenting a peak value of one. Finally, the resulting mel energy values are logarithmically scaled. The whole procedure was carried out using the Librosa library (v0.5.1) [21].

Resulting log-scaled mel-spectrograms are normalized to zero mean and unit standard deviation for the training set of every fold (see Section 3). Later on, the corresponding test set for every fold is standardized with the values from the training set normalization. Then, the spectrogram corresponding to every full 10s recording (consisting of 501 frames) is split into non-overlapping time-frequency patches (T-F patches) or *sequences* of 1.5s (i.e., 75 frames[5]). In this way, for every recording we obtain a total of 7 sequences (the last one being padded with the last original frame until reaching the desired duration). Every sequence, i.e., a T-F patch of $\mathbb{R}^{75\times128}$, is the input to the CNN and the minimum classification unit that will be aggregated to make decisions at the recording level.

[2] https://github.com/Microsoft/LightGBM

[3] http://essentia.upf.edu/documentation/extractors_out_of_box.html

[4] https://freesound.org/

[5] This duration is selected as the result of preliminary experiments, among durations ranging from 1 to 3s in steps of 0.5s.

2.2.2. Network Architecture

The proposed CNN architecture is depicted in Table 2.

Table 2: Proposed CNN architecture.

Input: 1 x (75,128)
Conv1: 48x (3,8)|32x (3,32)|16x (3,64)|16x (3,90) + BN + ReLU
Max-Pooling: (5,5)
Conv2: 224x (5,5) + BN + ReLU
Max-Pooling: (11,4)
Dense: 15 units + softmax

The architecture is comprised of two convolutional layers alternated with max-pooling operations and a final fully connected dense layer. For the design of the convolutional filters, we hypothesize that for the acoustic material of many of the scenes under consideration, the spectro-temporal patterns are more relevant along the frequency domain (e.g., spectral envelope shapes and background noises) rather than in the time domain (e.g., onsets/offsets and attack-decay patterns of specific acoustic events). Most CNN architectures proposed in the literature use squared filters and only one filter shape in the first convolutional layer [11, 12, 14]. In contrast, some recent works suggest to employ, in the first layer, *i)* filter shapes that are not squared but rectangular, and *ii)* different co-existing filter shapes. This has shown to be effective for learning timbre representations in music audio classification tasks [22], and for learning features at multiple time resolutions in acoustic event recognition [23]. Motivated by those works and based on our assumption above, we decided to experiment with several configurations of filters with multiple *vertical shapes*[6] in the first layer (results are reported in Section 4.2). By doing this, we intend to aid the learning process towards what we intuitively assume as more important for our task. To implement this, the first convolutional layer is, in turn, an ensemble of several convolutional layers, each one with filters of one shape, that are eventually merged. In order to obtain feature maps of the same size, zero-padding is applied to the network's input.

Filter shapes are specified in Table 2 as *number of filters* x *(time, frequency)*. The number of filters are 112 and 224 for the first and second convolutional layers, respectively. The proposed final architecture presents, in *Conv1*, four different sets of filters, each of them presenting one different shape. For simplicity, and based on initial experiments, it was decided to use a fixed time dimension of 3 for all filters in this layer. In *Conv2* filters are squared. All filters have unitary stride in both dimensions. In both convolutional layers L2 regularization is applied with a parameter of 10^{-5}.

After every convolutional layer, batch normalization (BN) is applied [24] and the activation function is Rectified Linear Unit (ReLU) [25]. We use max-pooling after the two convolutional layers, which provides downsampling of the feature maps while adding some invariance along the time-frequency dimensions. More specifically, after *Conv1*, max-pooling is applied over squares of dimension 5. After *Conv2*, the pooling operation is designed to be global in the time domain so as to select only the most prominent feature, and with a dimension of 4 in the frequency domain, following previous work [12]. After the last max-pooling operation, resulting feature maps are flattened. Finally, the output layer is a dense layer, followed by a softmax activation function with 15 output units corresponding to the 15 acoustic scenes.

Network weights are initialized with a uniform distribution. The loss function is categorical cross-entropy and the optimizer is Adam with a learning rate of 0.001. The training is stopped through early stopping if the validation accuracy is not improved during 15 epochs, up to a maximum of 200 epochs. Training samples are shuffled between epochs, so that the batches (of size 64) are formed differently in order to increase data variability. The system is implemented using the Keras library (v2.0.2) [26].

2.3. Late Fusion

We use *arithmetic mean* as the late fusion method, which combines prediction probabilities from GBM and CNN systems by taking the arithmetic mean of the probabilities for each recording:

$$pred^i = \text{argmax}(\frac{proba^i_{\text{GBM}} + proba^i_{\text{CNN}}}{2}) \quad (1)$$

where $proba^i_{\text{GBM}}$ and $proba^i_{\text{CNN}}$ are respectively the prediction probabilities for the recording i from GBM and CNN systems; $pred^i$ is the predicted label. This strategy led to better results than geometric mean.

3. EXPERIMENTS

3.1. Dataset

We use the *TUT Acoustic Scenes 2017* dataset, which is split into a development dataset and an evaluation dataset, of 4680 and 1620 audio recordings respectively. The development dataset is provided at the beginning of the challenge, together with ground truth. It includes 15 acoustic scenes[7] each of them containing 312 recordings of 10s. A four-fold cross-validation setup is provided so as to make results reported strictly comparable. Along with the dataset, the challenge provides a two-layers multilayer perceptron (MLP) baseline system. Its prediction accuracy is reported in Table 3.

3.2. Evaluation Setup

We use the development dataset for training and testing both GBM and CNN models, according to the suggested four-fold cross-validation setup. Since no parameter tuning is performed for GBM, we use the entire training set in each fold to train the model. For the CNN, a 15% validation set is randomly split from the training data of every class in each fold to early-stop the training process. As explained in Sections 2.1.1 and 2.2.1, the output of the GBM and CNN models for every input sequence is a $\mathbb{R}^{15\times 1}$ vector with the probabilities of the sequence belonging to every label. The class prediction at the recording level is computed by averaging class-wise scores across sequences and finding the class with the maximum average score. For the final proposed ensemble system, we carry out the late fusion as explained in Section 2.3. Figure 1 shows the evaluation process. The metric used is classification accuracy, i.e., the number of correctly classified audio recordings divided by the total amount of recordings and we report the average accuracy across the four folds. The evaluation dataset is used to predict acoustic scenes with our final proposed system for the challenge submission.

[6] We denote vertical filters as those whose frequency dimension is much larger than its time dimension.

[7] A list of the scenes together with more details about the dataset can be found in `http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/task-acoustic-scene-classification`.

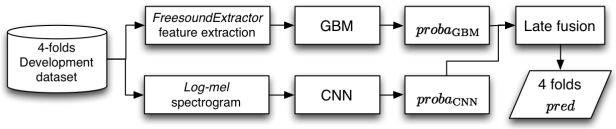

Figure 1: Evaluation diagram.

4. RESULTS AND DISCUSSION

4.1. Gradient Boosting Machine

The four-fold mean prediction accuracy of our LightGBM system is 80.8%, which improves the MLP baseline by 6%. The simplicity of this method—using an out-of-box feature extractor and no extensive parameter tuning— shows the suitableness of the Essentia extracted features for this task and the robustness of the LightGBM system.

4.2. Convolutional Neural Networks

We experimented with different filter configurations in the first convolutional layer. Accuracy results for the architectures using those configurations are listed in Table 3, together with the accuracy of the MLP-based baseline.

Table 3: ASC performance using the proposed CNN with various filter configurations in the first layer.

System	Filter configuration *#filters* x *(t, f)*	#params	acc (%)
MLP	-	-	74.8
CNN Q=1	112x (5,5)	648k	77.8
CNN Q=1	112x (3,40)	659k	78.1
CNN Q=2	64x (3,20) | 48x (3,70)	660k	78.7
CNN Q=3	48x (3,10) | 32x (3,30) 32x (3,60)	656k	79.6
CNN Q=4	48x (3,8) | 32x (3,32) 16x (3,64) | 16x (3,90)	657k	**79.9**

We design filter configurations with Q sets of filters in the first layer, every set presenting one filter shape. Therefore, Q refers to the number of different filter shapes. Every set of filters can have a different number of filters, as seen in the second column of Table 3, but the total amount of filters is always 112. For every configuration, the filters' frequency dimensions were defined of different lengths in order to cover spectral signatures of different nature, i.e., ranging from narrow patterns to patterns that are more spread in frequency. By adjusting the number of filters for each set and the filters' dimensions, it was intended to keep the amount of network parameters approximately constant (except for the case of squared filters), so that results are comparable. Accuracies reported are the outcome of averaging results of three runs of every experiment. Although a more thorough study would be required to draw strong conclusions on the specific effect of varying *i)* the number of filter shapes, *ii)* the number of filters per shape, and *iii)* the dimensions of the filters, it seems that the combination of vertical filters and different filter shapes in the first layer is beneficial for the task. Since the case Q=4 provides the best results (a 5.1% improvement with respect to the MLP baseline), we use it in the proposed architecture.

4.3. Late Fusion

As explained in Section 2.3, the scores from GBM and CNN for every recording are averaged to produce the final system scores, from which the acoustic scene label is predicted. After this late fusion, the system provides a classification accuracy of 83.0% on the development set, which means an improvement of 8.2% with respect to the MLP baseline. Further, it implies an improvement of 2.2% and 3.1% when compared to the GBM and CNN approaches, respectively.[8] This demonstrates that both models provide complementary information and their fusion is able to increase performance substantially, even with a very simple fusion method. We believe the performance can be further improved by employing more sophisticated fusion strategies, e.g., logistic regression. Figure 2 shows the confusion matrix for the proposed ensemble system, where it can be seen which acoustic scenes are misclassified the most. The worst case occurs clearly between 'residential area' and 'park', which are perceptually very similar. Also, the system often confuses 'tram' and 'train', 'grocery store' and 'cafe/restaurant', and 'library' and 'home'. Confusion matrixes for the GBM and CNN along with additional discussion and materials can be found in [9].

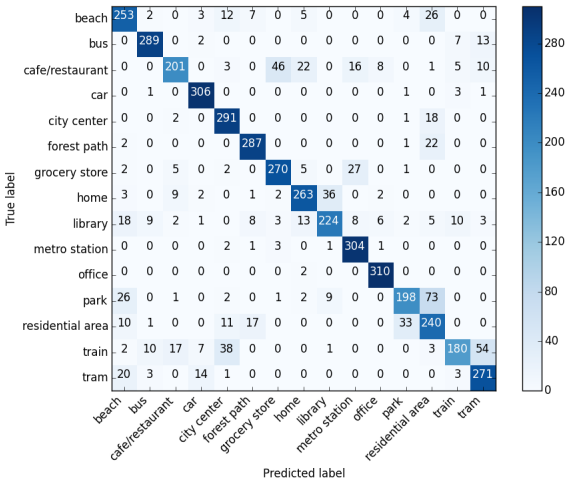

Figure 2: Confusion matrix for the proposed ensemble system evaluated on the development dataset.

5. CONCLUSION

This work proposes a system for ASC that consists of the ensemble of two methods of different nature: one that inputs a collection of hand-crafted features to a GBM, and a CNN that learns representations from log-scaled mel-spectrograms. We have shown how a simple late fusion of them already brings substantial performance improvement, which demonstrates that they provide complementary information beneficial for ASC. The proposed system achieves a classification accuracy of 83.0% on the TUT Acoustic Scenes 2017 development dataset. We believe that the proposed approach of combining two methods of different nature can be generalizable to other audio processing tasks, and we intend to test its effectiveness beyond ASC.

[8] Although the accuracy obtained by the GBM is higher than that of the CNN, their comparison is not totally fair. The latter uses 15% of the training data as validation set while the former uses the entire training set for training.

[9] https://edufonseca.github.io/DCASE2017-Task1-ASC/

6. ACKNOWLEDGMENT

This work is partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 688382 "AudioCommons", and the European Research Council under the European Union's Seventh Framework Program, as part of the CompMusic project (ERC grant agreement 267583), and the Spanish Ministry of Economy and Competitiveness under the Maria de Maeztu Units of Excellence Programme (MDM-2015-0502). We are grateful for the GPUs donated by NVidia.

7. REFERENCES

[1] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 16–34, 2015.

[2] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, submitted.

[3] C. Landone, J. Harrop, and J. Reiss, "Enabling access to sound archives through integration, enrichment and retrieval: The EASAIER project." in *ISMIR*, 2007, pp. 159–160.

[4] Y. Xu, W. J. Li, and K. K. Lee, *Intelligent wearable interfaces*. John Wiley & Sons, 2008.

[5] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Mobile Computing Systems and Applications*. IEEE, 1994, pp. 85–90.

[6] J.-J. Aucouturier, B. Defreville, and F. Pachet, "The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music," *The Journal of the Acoustical Society of America*, vol. 122, no. 2, pp. 881–891, 2007.

[7] A. J. Eronen, V. T. Peltonen, J. T. Tuomi, A. P. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi, "Audio-based context recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 321–329, 2006.

[8] G. Roma, W. Nogueira, and P. Herrera, "Recurrence quantification analysis features for auditory scene classification," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.

[9] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in neural information processing systems*, 2009, pp. 1096–1104.

[10] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 6964–6968.

[11] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[12] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, and T. Virtanen, "DCASE 2016 acoustic scene classification using convolutional neural networks," in *Proc. Workshop Detection Classif. Acoust. Scenes Events*, 2016, pp. 95–99.

[13] Y. Han and K. Lee, "Convolutional neural network with multiple-width frequency-delta data augmentation for acoustic scene classification," DCASE2016 Challenge, Tech. Rep., 2016.

[14] H. Eghbal-Zadeh, B. Lehner, M. Dorfer, and G. Widmer, "CP-JKU submissions for DCASE-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[15] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[16] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016.

[17] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, X. Serra, *et al.*, "Essentia: An audio analysis library for music information retrieval." in *ISMIR*, 2013, pp. 493–498.

[18] H. Zhang, S. Si, and C.-J. Hsieh, "GPU-acceleration for Large-scale Tree Boosting," *ArXiv e-prints*, June 2017.

[19] M. Slaney, "Auditory toolbox," *Interval Research Corporation, Tech. Rep*, vol. 10, p. 1998, 1998.

[20] Q. Kong, I. Sobieraj, W. Wang, and M. D. Plumbley, "Deep neural network baseline for DCASE challenge 2016," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*.

[21] B. McFee, M. McVicar, O. Nieto, S. Balke, C. Thome, D. Liang, E. Battenberg, J. Moore, R. Bittner, R. Yamamoto, D. Ellis, F.-R. Stoter, D. Repetto, S. Waloschek, C. Carr, S. Kranzler, K. Choi, P. Viktorin, J. F. Santos, A. Holovaty, W. Pimenta, and H. Lee, "librosa 0.5.0," Feb. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.293021

[22] J. Pons, O. Slizovskaia, R. Gong, E. Gómez, and X. Serra, "Timbre analysis of music audio signals with convolutional neural networks," *arXiv preprint arXiv:1703.06697*, 2017.

[23] H. Phan, L. Hertel, M. Maass, and A. Mertins, "Robust audio event recognition with 1-max pooling convolutional neural networks," *arXiv preprint arXiv:1604.06338*, 2016.

[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[25] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[26] F. Chollet, "Keras," https://github.com/fchollet/keras, 2017.