ACOUSTIC SCENE CLASSIFICATION FROM BINAURAL SIGNALS USING CONVOLUTIONAL NEURAL NETWORKS

Rohith Mars, Pranay Pratik, Srikanth Nagisetty, Chong Soon Lim

Panasonic R&D Center Singapore
{rohith.mars, pranay.pratik, srikanth.nagisetty, chongsoon.lim}@sg.panasonic.com

ABSTRACT

In this paper, we present the details of our proposed framework and solution for the DCASE 2019 Task 1A - Acoustic Scene Classification challenge. We describe the audio pre-processing, feature extraction steps and the time-frequency (TF) representations employed for acoustic scene classification using binaural recordings. We propose two distinct and light-weight architectures of convolutional neural networks (CNNs) for processing the extracted audio features and classification. The performance of both these architectures are compared in terms of classification accuracy as well as model complexity. Using an ensemble of the predictions from the subset of models based on the above CNNs, we achieved an average classification accuracy of **79.35%** on the test split of the development dataset for this task. In the Kaggle's private leaderboard, our solution was ranked 4^{th} with a system score of **83.16%** — an improvement of $\approx 20\%$ over the baseline system.

Index Terms— DCASE 2019, acoustic scene classification, convolutional neural networks, binaural signals, mixup.

1. INTRODUCTION

Humans perceive their surroundings primarily through the visual and audio cues presented to their eyes and ears, respectively. Though visual stimuli provide a substantial amount of information regarding the scene, it is inarguable that audio cues also play a vital role in determining the type of the environment we are immersed in. For example, an immersive experience through virtual reality (VR) is deemed satisfactory only when the associated audio aligns with the visual scene. In a simpler scenario, a person standing near a beach with eyes closed can easily infer that they are near the shore from the repetitive sound pattern of the waves crashing on the rocks or from the sound of the seagulls. It is easy to conclude that acoustic characteristics of certain environments have their own unique signature, which aids humans in distinguishing an audio scene from another.

The objective of acoustic scene classification is to empower a machine to automatically recognize the audio scene from the audio signals they are provided with. Such "machine listening" tasks fall under the broader umbrella of computational auditory scene analysis (CASA) [1, 2]. Over the past few years, advancement of deep learning algorithms along with availability of large datasets and increase in computational power has helped to further push the performance of such machine listening systems.

The Detection and Classification of Acoustic Scenes and Events (DCASE) challenge, has played a major role in providing common datasets for development, setting algorithmic benchmarks and furthering the research in deep learning for audio signals, especially for tasks such as scene classification, event detection and audio tagging. For the scene classification task introduced in DCASE 2013 challenge, the best performing algorithm used a machine learning approach, more specifically, a treebagger classifier using hand-crafted features extracted from the audio recordings [3]. In the 2016 edition, most of the solutions involved deep learning approach, with the top performance achieved by a fusion of convolutional neural network (CNN) and binaural I-vectors [4]. Continuing a similar trend from the previous year, the top performing algorithms for the DCASE 2017 scene classification task employed CNNs for the audio spectrogram representations [5] and used generative adversarial network (GAN) for data augmentation [6]. In the 2018 edition of the DCASE challenge, the best performance was achieved by use of CNNs with adaptive division of multiple spectrogram respresentations [7].

Similar to the previous editions, the DCASE 2019 challenge [8] consists of separate challenge tasks, with Acoustic Scene Classification being one among them. This task is further divided into three subtasks, wherein we participate in the DCASE 2019 Task 1A. In this subtask, the development data and evaluation data are obtained from the same recording device. We built our proposed solution framework inspired by the success of utilizing 2-D time-frequency (TF) representations of binaural recordings with CNNs for classification. However, instead of using computationally expensive audio feature extraction steps and CNN models with large number of parameters, we utilize audio feature extraction with minimal computation and light-weight CNN models. In addition, we also explore the effect of using rectangular kernels and non-uniform pooling operations in CNN architecture as opposed to conventional square kernels and uniform pooling for achieving the same task and compare these distinct architectures in terms of accuracy as well as complexity. We obtain the final prediction by ensembling the outputs from the best-performing models identified using the test split from the development set.

We begin this paper by describing our audio pre-processing, feature extraction and data augmentation steps in Section 2. In Section 3, we provide details on the two separate CNN architectures used in our solution. The details of the database provided for the challenge, the accuracy achieved by our solution on the test split of the development dataset as well as the Kaggle's private leaderboard are provided in Section 4. Finally, the conclusions are presented in Section 5.

2. FEATURE EXTRACTION & DATA AUGMENTATION

In this section, we describe the audio pre-processing steps as well as the binaural audio feature extraction process. The extracted features are then provided as input to the CNN for predicting the acoustic scene class. In addition, we also discuss the data augmentation step

https://doi.org/10.33682/6c9z-gd15

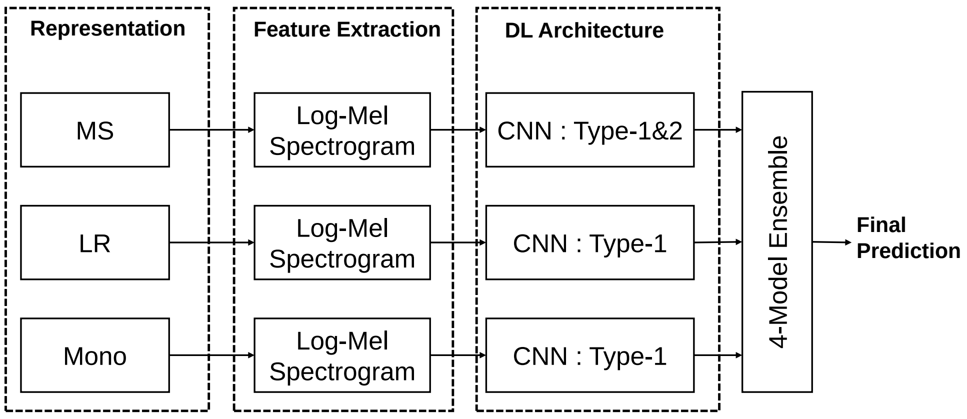

Figure 1: Our overall solution framework consisting of binaural audio representations, feature extraction steps, multiple CNNs & model ensembling for final prediction of the acoustic scene.

used to improve the model's generalization to unseen data.

2.1. Binaural audio feature extraction

In our proposed system, we use the originally provided audio recordings sampled at 48 kHz without down sampling. The time domain audio signals are then normalized by amplitude and then converted to the TF representation to extract the temporal and spectral characteristics. As such, we first compute the short-time Fourier transform (STFT) of the normalized time-domain audio signal. The frame size for the STFT operation is fixed at 2048, with a frame overlap of 50% and hanning window. Due to the large dimension of the linear STFT representation, we further compute the corresponding Mel-spectrogram representation using 128 Mel-bands. The use of Mel-scale is more close to the human auditory system and provides additional advantage of having smaller dimensionality than conventional linear STFT. As the final step, we compute the $\text{Log}(\cdot)$ of the Mel-spectrogram to reduce the dynamic range of the values and make the feature space more Gaussian in distribution as reported in [9]. On the computed Log Mel-spectrograms, we performed feature normalization to achieve zero mean with unit variance. This mean and standard deviation was computed using the training data and the same were used on the validation/test split.

Since the recorded data in this task is binaural in nature, the Log Mel-spectrograms are computed separately for the Left (L), Right (R), Mid (M), Side (S) representation. In addition, we also use the conventional mono representation of the binaural signal for computing the Log Mel-spectrogram. The MS representation is obtained from the LR representation as follows

$$\begin{aligned} M &= (L+R)/2 \\ S &= (L-R)/2. \end{aligned} \tag{1}$$

The use of LR, MS & mono representations for audio classification have been explored in earlier editions of DCASE audio scene classification task and has been reported to achieve superior performance [5]. However, our framework differs from [5] in few aspects. Firstly, we do not split the 10 second audio clips to smaller audio chunks. In other words, the entire 2-D Log Mel-spectrogram of size 128×469 per channel is provided as input to the CNN. Secondly, instead of using each channel as a separate input to the CNN and concatenating the corresponding CNN layers at a later stage in the network, we combine the individual channels at the first layer of convolution itself. Finally, we do not perform the background subtraction (BS) method as well as the harmonic percussive source separation (HPSS) on the mono representation used in [5] as they involve further processing after the downmix operation and are thus computationally more expensive. The entire framework of our solution depicting the audio representations, feature extraction, multiple CNNs and ensembling step is shown in Figure 1.

2.2. Data Augmentation

It is well-known that deep learning algorithms perform well when they are trained using large amounts of data. However, depending on the task, the amount of labelled data for training maybe limited or constrained. As a result, deep learning algorithms may not fully capture the intra-class and inter-class variations in the data. In such situations, data augmentation plays a crucial role by increasing the amount and variance in the training data. For acoustic signals, conventional augmentation techniques include pitch shifting, time stretching, adding background noise and dynamic range modulation [10]. Another approach for augmentation is to mix the clips of same acoustic class by splitting and shuffling [11]. Recently, the use of GANs for data augmentation has also been explored in [6].

In our proposed method, to ensure a better generalization capability for the neural network, we perform the augmentation method proposed in [12], termed as *mixup*. The use of *mixup* for improving the performance of acoustic scene classification task has been explored in [13, 14]. In *mixup*, two random training examples (x_i, x_j) are weighted and mixed together along with their class labels (y_i, y_j) to form virtual training examples $(\tilde{x}, \tilde{y})$ as

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1-\lambda)x_j \\ \tilde{y} &= \lambda y_i + (1-\lambda)y_j, \end{aligned} \tag{2}$$

with $\lambda \in [0,1]$ acquired by sampling from the β distribution $\beta(\alpha, \alpha)$, with α being a hyper parameter.

Input: $128 \times 469 \times 2$ or $128 \times 469 \times 1$
Conv2D (64, $\{3 \times 3\}$), BatchNormalization, ReLU MaxPooling2D $\{2 \times 2\}$
Conv2D (128, $\{3 \times 3\}$), BatchNormalization, ReLU MaxPooling2D $\{2 \times 2\}$
Conv2D (256, $\{3 \times 3\}$), BatchNormalization, ReLU MaxPooling2D $\{2 \times 2\}$
Conv2D (512, $\{3 \times 3\}$), BatchNormalization, ReLU
Conv2D (512, $\{3 \times 3\}$), BatchNormalization, ReLU
GlobalMaxPool2D
Dense(256, ReLU)
Dense(10, Softmax)

Table 1: CNN: Type-1 Architecture. Conv2D($n, \{p \times q\}$) represents 2D convolution operation with n filters of kernel size $p \times q$.

Input: $128 \times 469 \times 2$ or $128 \times 469 \times 1$
Conv2D (64, $\{3 \times 7\}$), BatchNormalization, ReLU MaxPooling2D $\{3 \times 1\}$
Conv2D (128, $\{3 \times 1\}$), BatchNormalization, ReLU MaxPooling2D $\{4 \times 1\}$
Conv2D (256, $\{11 \times 1\}$, padding= "valid") BatchNormalization, ReLU
Conv2D (512, $\{1 \times 7\}$), BatchNormalization, ReLU
GlobalMaxPool2D
Dense(256, ReLU)
Dense(10, Softmax)

Table 2: CNN: Type-2 Architecture. Note that we use rectangular kernels and non-uniform pooling operation throughout the network.

3. CNN ARCHITECTURE

The audio features extracted by the pre-processing and data augmentation steps explained in Section 2 are provided as input to a CNN. In this work, we experimented with two distinct architectures of CNN. The first CNN architecture is similar to the VGG-style architecture, which uses a constant $\{3 \times 3\}$ square-shaped kernels. However, we use significantly less number of convolution and dense layers as compared to the original VGG-16 architecture. In the second CNN architecture, we employ rectangular kernels for convolution and non-uniform pooling operation for the frequency and temporal dimension of the audio spectrogram. CNNs with rectangular kernels have been previously used for a variety of tasks such as scene classification [15, 16], keyword spotting [17] and music genre classification [18, 19]. The use of such rectangular kernels help to treat the spectral and temporal components of the audio with different context sizes as compared to square-shaped kernels. In the following subsections, we further elaborate on the above two CNN architectures.

3.1. CNN: Type-1

This CNN architecture is similar to the VGG-style architecture. It consists of 5 convolutions with increasing number of filters, i.e., $(64, 128, 256, 512, 512)$. The kernel size is chosen as $\{3 \times 3\}$ and is kept constant for all the convolution layers. We also apply batch normalization [20] and ReLU non-linear activation for each convolution layers. Max pooling $\{2 \times 2\}$ operation is performed at the first three convolution layers to reduce the dimensionality. Finally we perform global max pool operation to gather all the components, which is then connected to a dense layer of 256 units with ReLU activation. The output layer consists of 10 units corresponding to the number of scene classes and undergoes softmax operation to obtain the prediction probabilities. The CNN: Type-1 has ≈ 4 million parameters.

3.2. CNN: Type-2

In this CNN architecture, we employ rectangular kernels instead of square kernels. It consists of 4 convolutions with increasing number of filters, i.e., $(64, 128, 256, 512)$. For the convolution layer-1, we apply a kernel of size $\{3 \times 7\}$ for low-level feature extraction, followed by a max pooling with size $\{3 \times 1\}$. After reducing the dimension in the frequency axis, convolutions with kernel size $\{3 \times 1\}$ is applied in convolution layer-2 to extract frequency patterns for each time-frame. We further reduce the dimension in the frequency axis by using $\{4 \times 1\}$ max pooling. In the convolution layer-3, we use a kernel size of $\{11 \times 1\}$ and perform "valid" convolutions. This step ensures that spectral patterns are learnt with entire frequency dimension being compressed. We do not perform pooling across time dimension and the last convolution layer uses filter size of $\{1 \times 7\}$ to learn only the temporal characteristics. Similar to CNN-1, we apply batch normalization and ReLU non-linear activation for each convolution layers. After the convolution layers, all the components are collected using the global max pooling operation, which is further input to a fully connected layer of 256 units with ReLU activation. The output layer consists of 10 units corresponding to the number of scene classes and undergoes softmax operation to obtain the prediction probabilities. The CNN: Type-2 uses ≈ 1.4 million parameters. This is 3 times lower number of parameters as compared to CNN: Type-1 and therefore, a comparatively less-complex architecture.

By employing the above distinct architectures of CNNs, we expect each of them to learn different low-level and high-level features of the audio spectrogram. While CNN: Type-1 treats the frequency and temporal dimension equally using square kernels, CNN:Type-2 treats these dimensions with different context sizes using rectangular kernels. Note that for LR & MS representations, the input size is $128 \times 469 \times 2$, with each channel arranged back to back and we combine the individual channels at the first layer of convolution itself. For the case of mono representation, the input size is $128 \times 469 \times 1$.

Methods	Mean accuracy (%)
Baseline	62.5
CNN:Type-1 - Mono	73.04
CNN:Type-1 - LR	74.20
CNN:Type-1 - MS	75.19
CNN:Type-2 - Mono	69.86
CNN:Type-2 - LR	69.79
CNN:Type-2 - MS	72.90
Ensemble	**79.35**

Table 3: Mean accuracy on the test split from the development set using the baseline system, the proposed CNN architectures for each representation and after ensembling. For ensembling, the top-4 best performing models are selected.

4. DATABASE & RESULTS

The TAU Urban Acoustic Scenes 2019 dataset [21] for this task is the extension of the 2018 TUT Urban Acoustic Dataset, consisting of binaural audio recordings from various acoustic scenes in different cities. The recordings were made using the Soundman OKM II Klassik/studio A3, electret binaural microphone and a Zoom F8 audio recorder using 48 kHz sampling rate and 24 bit resolution. For each acoustic scene class, such recordings were collected from different locations in the city. Each original recordings were split into segments with a length of 10 seconds as development and evaluation set. The audio scenes are namely {"Airport", "Indoor shopping mall", "Metro station", "Pedestrian street", "Public square", "Street with medium level of traffic", "Travelling by a tram", "Travelling by a bus", "Travelling by an underground metro" & "Urban park"}.

From the training split of the development set, we use a random split of 15% as the hold-out validation set for hyperparameter tuning. We do not utilize any external data or pre-trained models for training our system. The optimization is performed using the Adam optimizer [22], with an initial learning rate of 0.001 and a maximum epoch of 200 with a batch size of 32 samples. We reduce the learning rate by a factor of 0.1 if the validation loss does not decrease after 5 epochs. We use early stopping method to stop the training if the validation loss does not decrease after 10 epochs. The categorical cross-entropy is chosen as the loss function. For the data augmentation step using *mixup*, we kept $\alpha = 0.3$ for all the models. The baseline system [23] also used a CNN based approach on Log Mel spectrogram of 40 bands, consisting of two CNN layers and one fully connected layer. We chose Keras [24] as our deep learning framework for all experiments.

For the CNN: Type-1, using the mono representation, we get an average accuracy of 73.04%. In comparison, the MS and LR representation, we achieve a mean accuracy of 75.19% and 74.2%, respectively. For CNN: Type-2, the mean accuracy are 69.86%, 72.9% & 69.79% using the mono, MS and LR representation, respectively. From both these results, we conclude that the MS representation is best suited for this task. The performance drop in CNN: Type-2 can be attributed to the low-complex architecture used. We also note that better tuning of the parameters may be required to enable this CNN to better capture the spectral and temporal patterns which can lead to performance improvement as well.

Scene Label	Baseline system Accuracy (%)	Proposed system Accuracy (%)
Airport	48.4	90.2
Bus	62.3	92.0
Metro	65.1	74.7
Metro station	54.5	80.2
Park	83.1	65.1
Public square	40.7	80.5
Shopping mall	59.4	75.5
Street pedestrian	60.9	87.5
Street traffic	86.7	80.2
Tram	64.0	79.4
Average	**62.5**	**79.3**

Table 4: Comparison of class-wise accuracy on the test split from the development set using the baseline system and the proposed system after ensembling.

Based on performance on the test split, we select the top-4 best performing models (CNN: Type-1 : MS, LR, Mono, CNN: Type-2: MS) for the final ensembling. We ensemble the output predictions from each of the 4 models by computing the geometric mean of the predictions. The final prediction is done by selecting the class with maximum probability on the ensembled prediction. After this ensembing step, we obtain a mean accuracy of **79.35%** on the test split of the development set.

The classification results for all the proposed models and ensembled solution compared with the baseline system are shown in Table 3. The class-wise accuracy of the proposed system after ensembling for the test split is compared with the baseline system is shown in Table 4. It can be seen that the proposed system achieves better accuracy for all classes except for "Park" and "Street traffic". For the evaluation on Kaggle leaderboard set, we used the entire development set for training the proposed system. In the Kaggle's private leaderboard [25], the baseline system achieved a system score of 63.00%. In comparison, our solution was ranked 4^{th} with system score of **83.16%**, thereby achieving an improvement of $\approx 20\%$ over the baseline system.

5. CONCLUSIONS

In this paper, we provided the details of our solution to the DCASE2019 Task1A - Acoustic Scene Classification. We described the audio pre-processing, feature extraction steps and the various binaural representations used as input the neural network. The architecture of two distinct and light-weight CNNs used for the classification are described. We compared the performance of these CNNs on each binaural representations in terms of classification accuracy as well as their complexity. After ensembling multiple models, our system achieves an average accuracy of **79.35%** on the test split from the development set. The solution was ranked 4^{th} with system score of **83.16%** in the Kaggle's private leaderboard.

6. REFERENCES

[1] D. Wang and G. J. Brown, *Computational auditory scene analysis: Principles, algorithms, and applications*. Wiley-IEEE press, 2006.

[2] T. Virtanen, M. D. Plumbley, and D. Ellis, *Computational analysis of sound scenes and events*. Springer, 2018.

[3] D. Li, J. Tam, and D. Toub, "Auditory scene classification using machine learning techniques," in *AASP Challenge on Detection and Classification of Acoustic Scenes and Events*. IEEE, 2013.

[4] 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," DCASE2016 Challenge, Tech. Rep., September 2016.

[5] Y. Han, J. Park, and K. Lee, "Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification," *the Detection and Classification of Acoustic Scenes and Events (DCASE)*, pp. 1–5, 2017.

[6] S. Mun, S. Park, D. K. Han, and H. Ko, "Generative adversarial network based acoustic scene training set augmentation and selection using svm hyper-plane," *Proc. DCASE*, pp. 93–97, 2017.

[7] Y. Sakashita and M. Aono, "Acoustic scene classification by ensemble of spectrograms based on adaptive temporal divisions," *IEEE AASP Challenge on DCASE 2018 technical reports*, 2018.

[8] http://dcase.community/challenge2019/.

[9] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "A comparison of audio signal preprocessing methods for deep neural networks on music tagging," in *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 1870–1874.

[10] 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.

[11] T. Inoue, P. Vinayavekhin, S. Wang, D. Wood, N. Greco, and R. Tachibana, "Domestic activities classification based on cnn using shuffling and mixing data augmentation," *DCASE 2018 Challenge*, 2018.

[12] H. Zhang and et.al, "mixup: Beyond empirical risk minimization," *International Conference on Learning Representations*, 2018.

[13] K. Xu, D. Feng, H. Mi, B. Zhu, D. Wang, L. Zhang, H. Cai, and S. Liu, "Mixup-based acoustic scene classification using multi-channel convolutional neural network," in *Pacific Rim Conference on Multimedia*. Springer, 2018, pp. 14–23.

[14] J. J. Huang and J. J. A. Leanos, "Aclnet: efficient end-to-end audio classification cnn," *arXiv preprint arXiv:1811.06669*, 2018.

[15] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2015, pp. 1–6.

[16] D. Battaglino, L. Lepauloux, N. Evans, F. Mougins, and F. Biot, "Acoustic scene classification using convolutional neural networks," *IEEE AASP Challenge on Detec*, 2016.

[17] T. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," *Proc. Interspeech*, 2015.

[18] A. Schindler, T. Lidy, and A. Rauber, "Comparing shallow versus deep neural network architectures for automatic music genre classification," in *9th Forum Media Technology (FMT2016)*, vol. 1734, 2016, pp. 17–21.

[19] J. Pons, T. Lidy, and X. Serra, "Experimenting with musically motivated convolutional neural networks," in *2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI)*. IEEE, 2016, pp. 1–6.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[21] T. Heittola, A. Mesaros, and T. Virtanen, "TAU Urban Acoustic Scenes 2019, Development dataset," Mar. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2589280

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] http://dcase.community/challenge2019/task-acoustic-scene-classification#baseline-system.

[24] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[25] https://www.kaggle.com/c/dcase2019-task1a-leaderboard/leaderboard.