THE CAU-ET ACOUSTIC SCENERY CLASSIFICATION SYSTEM FOR DCASE 2020 CHALLENGE

Technical Report

Yerin Lee[1]*, Soyoung Lim*[1]*, Il-Youp Kwak*[1]

[1] Chung-Ang University, Department of Applied Statistics,
Seoul, South Korea, {dldpfls14, isy921,ikwak2}@cau.ac.kr

ABSTRACT

The acoustic scenery classification problem is an interesting topic that has been studied for a long time through the DCASE competition. This technical report presents the CAU-ET's submitted scenery detection system to the DCASE 2020 challenge, Task 1. In our method, we generated Mel-spectrogram from audio. From the log-mel spectrogram, we got Deltas, Delta-deltas, and Harmonic-percussive source separation(HPSS) features as inputs of our deep neural network models. The classification result of the proposed system was 67.14% in subtask A and 95.27% in subtask B for each development set.

Index Terms— deltas and delta-deltas, harmonic-percussive source separation, mixup, convolutional neural network, ResNet, Multi-Input model, ensemble

1. INTRODUCTION

The goal of Task 1 in DCASE 2020 is to classify a test recording into one of the provided predefined classes that characterize the acoustic scenes in which it was recorded[1]. Task 1 is divided into two tasks. Both are concerned with the basic problem of acoustic scene classification. Subtask A targets the classification of audio into ten classes. Subtask A's audio data are recorded and simulated with a variety of devices. The development dataset comprises 40 hours of data from device A, and smaller amounts from the other devices. Audio is provided in single-channel 44.1kHz 24-bit format. Subtask B is concerned with the classification of audio into three classes and targets low complexity. This dataset contains data recorded with a single device (device A). Audio is provided in binaural, 48kHz 24-bit format.

2. ARCHITECTURE

2.1. Audio Preprocessing

In past DCASE challenges, most of the top team approached to forming image like spectrograms as inputs for Convolutional Neural Networks (CNN). We also used log mel spectrogram. The audios in the subtask A are mono and have a common sampling rate of 44.1kHz. But in the subtask B are binaural and have a sampling rate of 48kHz. To generate each spectrogram, we used 2048 FFT points, a hop-length of 1024 samples, 128 frequency bins and HTK formula. And we extracted log mel spectrogram.

2.1.1. Log-mel energies, Deltas, Delta-deltas

For feature extraction, our approach was inspired by McDonnell's past work on DCASE 2019 competition [2], that utilize log-mel energies, deltas, and delta-deltas from the log-mel energies. The deltas and delta-deltas imply the first and second temporal derivatives of the spectrum. For subtask A, with raw data in mono, we, therefore, had three input channels, log-mel, deltas, and delta-deltas, to our deep learning models. For subtask B, since we have binaural sample, we need 6 input channels.

2.1.2. Harmonic-percussive source separation

The Harmonic-percussive source separation(HPSS) decomposes monaural audio into two channels: one contains the harmonic sounds and other contains the percussive sounds. HPSS was inspired by Sakashita[3]. The Mel-spectrogram is also obtained from HPSS applied to mono audio. We had 2 channels for subtask A, because it is mono audio, and we had 4 channels for subtask B .

2.2. Network Architecture

Most of the top teams on the leader board last year used CNNs with audio data. We also applied CNNs on preprocessed features, log-mel spectrogram, deltas, delta-deltas, and HPSS. The process of our architecture expressed in Figure 1.]Subtask B is similar to subtask A, but subtask B does not consider ensemble to keep the light-weighted model.

2.2.1. Subtask A

The figure 2 describe our CNN modeling architecture for Subtask A. We proposed to use four different modeling architecture for the ConvBlock, a building block layers. The first architecture for ConvBlock is CNN module taken from Sakashita (2018) and Han (2017) [3, 4]. This neural network is a convolution model inspired by VGGNet. The second architecture for ConvBlock is ResNet module, the skip-connection network. The Third architecture for ConvBlock is LCNN module that utilize Max Feature Map(MFM) activation inside of the skip-connection network. The Fourth architecture for ConvBlock is Inception module like architecture that concatenating two tensors.

2.2.2. Subtask B

The figure 3 describes our modeling architecture for subtask B. We used reduced ResNet for each feature, HPSS and Deltas-Deltadeltas. Because of the model size, we have 2 convolution lay-

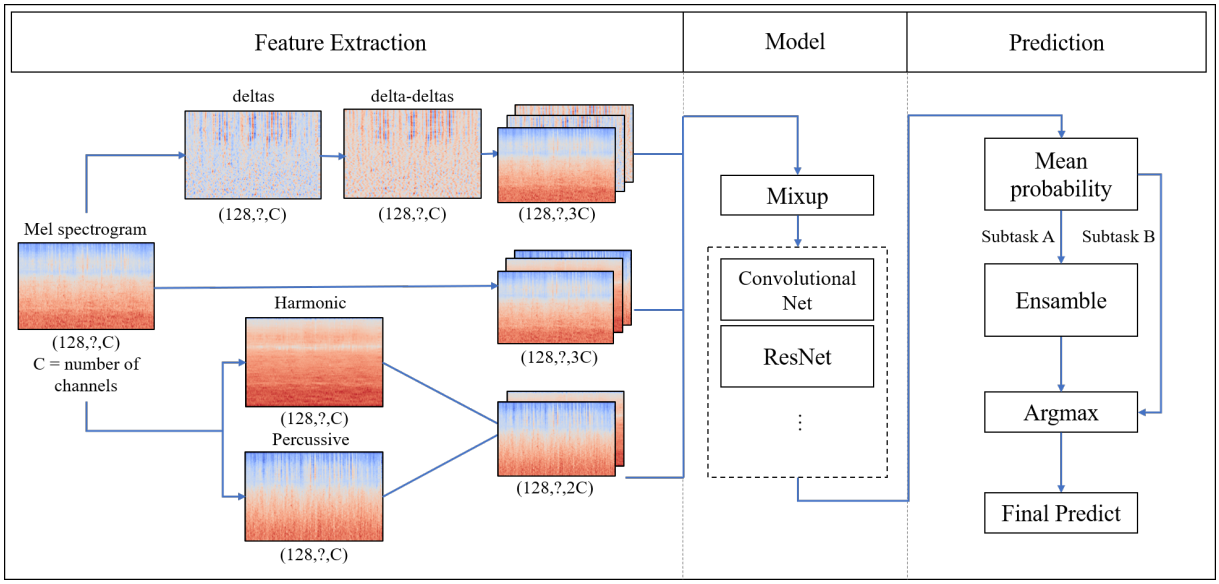

Figure 1: The CAU-ET system architecture

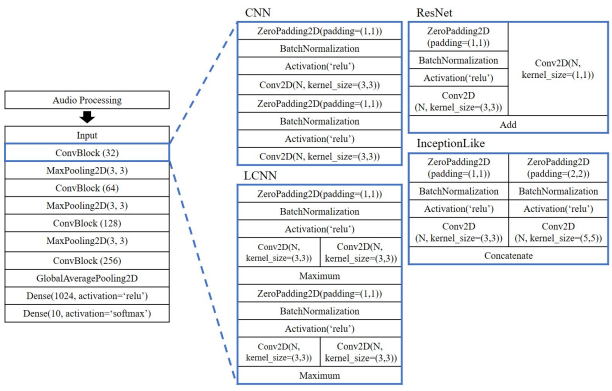

Figure 2: CNN architecture for Subtask A. Four different modeling architectures for ConvBlock are proposed. Those four architectures are inspired by VGGNet, ResNet, LCNN and InceptionNet.

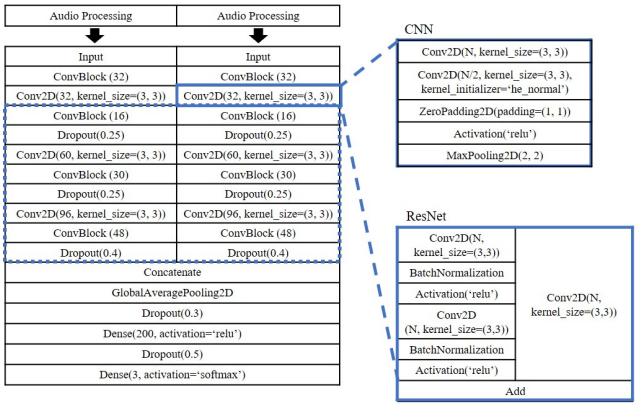

Figure 3: CNN and ResNet architecture for Subtask B. When we use ResNet in ConvBlock, the parts marked with dotted lines have been deleted.

ers and 1 short cut connection in our reduced ResNet [5]. We had two multi inputs, HPSS and Deltas-Deltadeltas, and used CNN and reduced Resnet architecture for modeling ConvBlock. Each input had its convolution layers to extract the features which were then concatenated and feed to the fully connected layers below[6].

3. DATA AUGMENTATION

Mixup is an effective data augmentation method. Mixup had proposed a general augmentation approach: mixing different samples of the training set according to their weights, and mixing labels according to their weights.The method is as follows:

$$X = \lambda X_i + (1 - \lambda) X_j$$

$$y = \lambda y_i + (1 - \lambda) y_j$$

where $\lambda \in [0, 1]$ and it is acquired by sampling from the beta distribution $\beta(\alpha, \alpha)$, $\alpha \in (0, \infty)$. X_i and X_j are different data, y_i and y_j are their corresponding label. In our experiment, we used the mixup to augment the log mel spectrograms and set α at 0.4. We additionally used crop augmentation on the time axis.

4. EXPERIMENTS

4.1. Data sets

The subtask A used TAU Urban Acoustic Scenes 2020 Mobile dataset. The dataset contains recordings from 12 European cities in 10 different acoustic scenes using 4 different devices. Additionally, synthetic data for 11 mobile devices was created based on the original recordings. Of the 12 cities, two are present only in the evaluation set.

The dataset for subtask B is TAU Urban Acoustic Scenes 2020 3Class. The dataset contains recordings from 12 European cities in 10 different acoustic scenes. The 10 acoustic scenes are grouped into three major classes, Indoor, indoor, outdoor, and transportation. All datasets were collected by Tampere University of Technology between 05/2018 - 11/2018[1].

The dataset is provided with a training/test split in which 70% of the data is included for training, 30% for testing. In subtask A, some devices appear only in the test subset. To create a perfectly balanced test set, a number of segments from various devices are not included in this split. We used 30% of train set as validation set.

4.2. Experiment Setting

We trained all our models with 100 epochs. And by 60 epochs, the learning rate was reduced from e^{-3} to e^{-5}. We applied sigmoidal decay using a learning rate scheduler implemented in keras. We used Adam optimizer.

4.3. Network Ensemble

In subtask A, we used ensemble to improve the performance. Different good models trained independently are likely to be good for different reasons. We computed initial predictions from each different model. We used weighted average and argmax for prediction.

5. RESULT

In this section, we report the performance of our proposed models on development set for subtask A and B.

5.1. Subtask A

For the subtask A, we used the following configurations:

- Deltas-DeltaDeltas-CNN: 'Deltas-DeltaDeltas' indicate that we used log-mel energies, deltas, and delta-deltas features described in Section 2.1.1. 'CNN' indicate that we used CNN architecture described in the Figure 2.
- Deltas-DeltaDeltas-ResNet: 'ResNet' indicate that we used ResNet architecture described in the Figure 2.
- Deltas-DeltaDeltas-LCNN: 'LCNN' indicate that we used LCNN architecture described in the Figure 2.
- Deltas-DeltaDeltas-InceptionLike: 'InceptionLike' indicate that we used Inception like architecture described in the Figure 2.
- HPSS-CNN: 'HPSS' indicate that we used HPSS feature described in Section 2.1.2.
- HPSS-ResNet
- HPSS-LCNN
- HPSS-InceptionLike
- Deltas-DeltaDeltas-Ensemble: This model ensembled four 'Deltas-DeltaDeltas' models listed above.
- HPSS-Ensemble: This model ensembled four 'HPSS' models listed above.
- 0.5Deltas-DeltaDeltas+0.5HPSS-Ensemble: Ensembled all 8 models with equal weight.
- 0.8Deltas-DeltaDeltas+0.2HPSS-Ensemble: Used weighted ensemble, multiplied 0.8 on 'Deltas-DeltaDeltas' models and multiplied 0.2 on 'HPSS' models

Table 1 shows the experimental results for subtask A. Each model applied to deltas-deltadeltas in subtask A had an accuracy of over 60%, while the same model applied to HPSS had a lower accuracy. So we used weighted ensemble, and the best result came out when we used weighted average: 0.8 for delta-deltadeltas and 0.2 for HPSS. And the accuracy was 67.14%.

Model	TASK1 A	Dev(%)
1	Deltas-DeltaDeltas-CNN	61.31
2	Deltas-DeltaDeltas-ResNet	62.83
3	Deltas-DeltaDeltas-LCNN	62.15
4	Deltas-DeltaDeltas-InceptionLike	61.89
5	HPSS-CNN	54.61
6	HPSS-ResNet	56.23
7	HPSS-LCNN	56.30
8	HPSS-InceptionLike	54.65
9	Deltas-DeltaDeltas-Ensemble	66.80
10	HPSS-Ensemble	59.53
11	0.5Deltas-DeltaDeltas+0.5HPSS-Ensemble	65.29
12	**0.8Deltas-DeltaDeltas+0.2HPSS-Ensemble**	67.14

Table 1: Subtask A Results

5.1.1. Our submission

We used same model, 12th model in Table 1 for all four submissions. The only difference is the datasets that used for model training. The Table 2 describe our 4 different submissions. The Lee_CAU_task1a_1 and Lee_CAU_task1a_2 are similar submissions with two independent training. We trained these two models (12th model in Table 1) using the training set. On the other hand, for the submission of Lee_CAU_task1a_3 and Lee_CAU_task1a_4, we trained our model using the combined data of both training set and validation set. When we were training our model on train set, we found that our model tend not to overfit as the number of epoch increases. If our model is robust enough, it might be better to train using both training set and validation set to encompass more diverse environmental conditions in our trained model.

Submission ID	Model	Training
Lee_CAU_task1a_1	12	T
Lee_CAU_task1a_2	12	T
Lee_CAU_task1a_3	12	T+D
Lee_CAU_task1a_4	12	T+D

Table 2: Description for Subtask A submissions. Model 12 is described in Table 1. 'Training' refers to the data on which the model was trained. 'T' is only the training set and 'T+D' is combined data of training set and validation set.

5.2. Subtask B

Table 3 shows the experimental results of subtask B. In all results, it exceeds the accuracy of the Baseline system, and all models have a model size of less than 500 KB. For the subtask B, we used the following configurations:

- Deltas-DeltaDeltas-ResNet: 'Deltas-DeltaDeltas' indicate that we used log-mel energies, deltas, and delta-deltas features described in Section 2.1.1. 'ResNet' indicate that we used reduced ResNet architecture described in the Figure 3.
- HPSS-ResNet: 'HPSS' indicate that we used HPSS feature described in Section 2.1.2.
- Multi-Input-ResNet: 'Multi-Input' indicate that we used both Deltas-DeltaDeltas and HPSS input features.
- Multi-Input-CNN: 'CNN' indicate that we used CNN architecture described in the Figure 3.

Model	TASK1 B	Dev(%)	Size(KB)
1	**Deltas-DeltaDeltas-ResNet**	95.27	494.2
2	**HPSS-ResNet**	93.74	489.8
3	**Multi-Input-ResNet**	92.85	495.6
4	Multi-Input-CNN	92.38	484.7

Table 3: Subtask B Results

5.2.1. Our submission

We submitted 1st, 2nd and 3rd model in Table 3 for subtask B. The Table 4 describe detailed information for our submissions. The Lee_CAU_task1b_1 and Lee_CAU_task1b_1 are 1st model with different training sets. Lee_CAU_task1b_1 trained using only training set and Lee_CAU_task1b_2 trained using combined data of training set and validation set. Lee_CAU_task1b_3 and Lee_CAU_task1b_4 are trained using only training set with 2nd and 3rd model respectively.

Submission ID	model	Training
Lee_CAU_task1b_1	1	T
Lee_CAU_task1b_2	1	T+D
Lee_CAU_task1b_3	2	T
Lee_CAU_task1b_4	3	T

Table 4: Description for Subtask B submissions. Models are described in Table 3.'Training' refers to the data on which the model was trained. 'T' is only the training set and 'T+D' is combined data of training set and validation set.

6. CONCLUSION

This paper provides CAU-ET systems submitted to the acoustic scenery classification challenge for subtask A and B. In subtask A, we considered two features, Deltas-Deltadeltas and HPSS, and four models inspired by VGGNet, ResNet, LCNN and InceptionNet. We ensembled those 8 models and submitted. All four submission have the same modeling architecture but the data used for training are different. In subtask B, we suggested reduced ResNet with Deltas-Deltadeltas and HPSS respectively, and a multi-input ResNet model with two different features, Deltas-Deltadeltas and HPSS. We use only 2-4 convolution layers for each model to minimize model complexity. Submitted systems achieved accuracy of 67.14% for subtask A and 95.27% for subtask B on development set.

7. ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by Ministry of Science and ICT (2020R1C1C1A01013020)

8. REFERENCES

[1] http://dcase.community/challenge2020/task-acoustic-scene-classification.

[2] W. Gao and M. McDonnell, "Acoustic scene classification using deep residual networks with late fusion of separated high and low frequency paths," DCASE2019 Challenge, Tech. Rep., June 2019.

[3] Y. Sakashita and M. Aono, "Acoustic scene classification by ensemble of spectrograms based on adaptive temporal divisions," DCASE2018 Challenge, Tech. Rep., September 2018.

[4] Y. Han and J. Park, "Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification," DCASE2017 Challenge, Tech. Rep., September 2017.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] P. Radu Enucă, "dat64x-capstone-project," https://github.com/raduenuca/dat64x-capstone-project.git, 2019.