Urban Sound Classification
with Neural Networks

Junqian Zhang

University of Milan
Data Science and Economics

Abstract. The aim of this project is to explore whether neural network can work on urban sound classification. It experiments on three neural network architectures: Convolution Neural Network, Convolutional Recurrent Neural Network and Parallel Recurrent Convolutional Neural Network, with three audio feature extraction methods: spectrogram, mel-frequency cepstral coefficients, and chromagram, to see how different combinations of architecture of neural network and feature influence the classification performances.

Keywords: Sound Classification · Image Classification · Convolution Neural Network · Convolutional Recurrent Neural Network · Parallel Recurrent Convolutional Neural Network.

1 Introduction

Automatic classification of environmental sound is an interesting and hot topic in research field. Great many researchers have proven that neural network has good performance on sound classification. This project trains neural networks for multi-classification of urban sound events based on audio files by Tensorflow 2. It wants to explore how different combinations of architectures of neural network and feature input influence urban sound classification, and look for the best configuration for this problem.

Three questions are come up for achieving the final target. Firstly, although spectrogram has proven powerful in audio classification, if more features are provided, whether the classification performance can be improved? Secondly, audio has not only frequency features, but also time features, whether classification accuracy increases when concerning time features? Thirdly, neural network has shown strong capability in image recognition, whether audio classification can have better results by representing the audio as color images?

To solve these question, popular sound feature extraction methods: spectrogram, mel-frequency cepstral coefficients, and chromagram are employed. The audio files are converted to numerical features and color images respectively to serve as inputs. The chosen architectures for experiments are VGG based Convolution Neural Network, Convolutional Recurrent Neural Network and Parallel Recurrent Convolutional Neural Network.

2 Data Description

The dataset used for this experiment project is **UrbanSound8K** dataset [1], containing 8,732 urban sounds labeled with 10 classes: air conditioner, car horn, children playing, dog bark, drilling, enginge idling, gun shot, jackhammer, siren, and street music. The dataset is predefined as 10 folds. This project will take folds 1, 2, 3, 4, 6 as *training* set, fold 5 as *validation* set, and folds 7, 8, 9, 10 as *test* set.

All the audio files are processed by Python *librosa* package [2]. Each of them is loaded as a floating point time series with sample rate **22,050** Hz. Since original audio files are of different lengths, each is split each sound clip into **50%** overlapping segments of **41** frames to fix the input (approximately 0.93s), borrowing from Karol J. Piczak [3]. So, as illustrated in **Fig.1**, the training size is **27,809**, validation size is **5,689** and test size is **20,555**. **Fig.2** shows how the labels of training set distribute. Obviously, data of *car horn* and *gun short* are far less than data of the other labels.

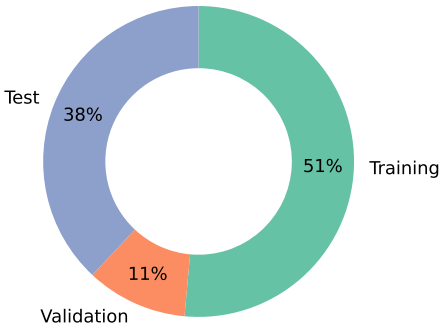


Fig. 1: Data Distribution

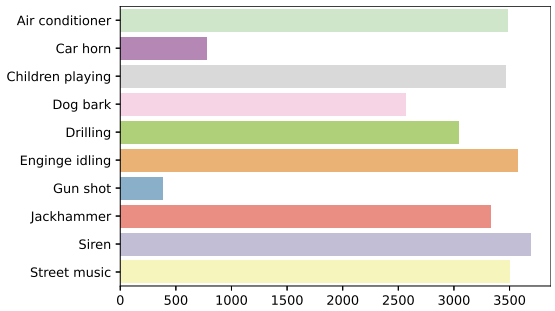


Fig. 2: Label Distribution of Training set

2.1 Feature Extraction

In this project, raw time series are transferred to different forms to to convert the signal into a time-frequency representation as input. **Log-scaled mel-spectrogram**, **Mel-frequency cepstral coefficients** (MFCCs) and **Chromagram** are the three features extracted.

To get Log-scaled mel-spectrograms, spectrogram is computed by librosa.feature.melspectrogram and then converted to decibel (dB) units by librosa.power_to_db with **128** mel-bands. So each of the time series is transferred to an array of size 128 ×41.

50 MFCCs are extracted by librosa.feature.mfcc and are converted to decibel (dB) units too. Then mfccs feature shape of any raw signal is 50 ×41.

Chromagrams are computed from raw time series with **12** chroma bins to produce. So each of the raw signal is transferred to an array of size 12 ×41.

For more information from audio and also for expanding the channels, the corresponding delta of all the features are computed.

Limited to the resources, two different kinds of features are adopted as input. The first kind is only spectrograms, and the second is combination of spectrograms, MFCCs, and chromagrams, which is called as "All Features" in the following.

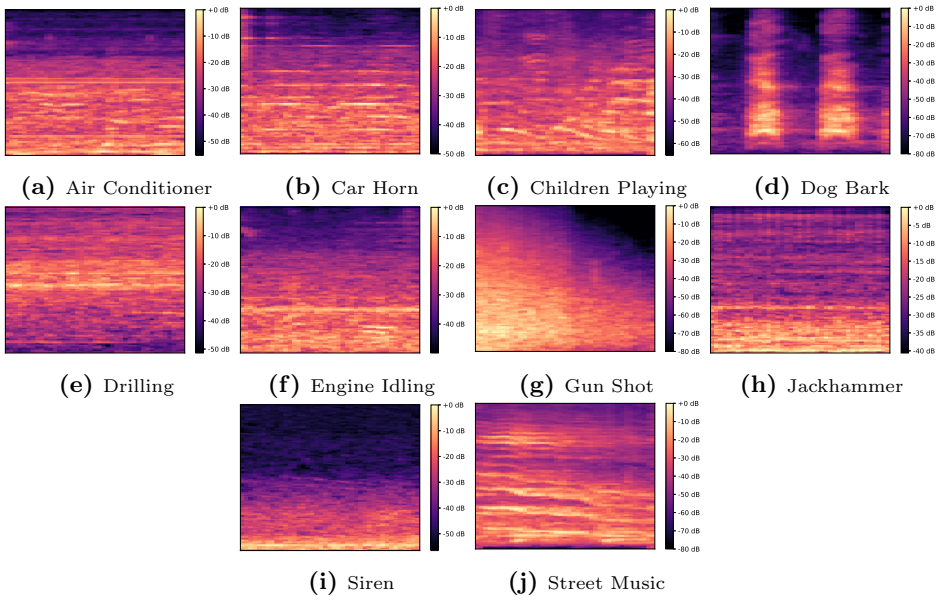
2.2 Image Generation

Spectrogram, MFCCs and chromagram can all be displayed in plots. In **Fig.3**, spectrograms of each class of sound are displayed. Manifestly, different classes of audio have significant characteristics, so it is possible to classify sound represented as color images. In this project, only images of spectrogram and combined images are used as inputs. The idea of combined images comes from Boddapati et al. [4]. They are formed by combining spectrogam plots, MFCCs plots and chromagram plots into single-color images, where Spectrograms are *red*, MFCCs are *green* and chromagrams are *blue*. All the signals are firstly transferred to log-scaled mel-spectrograms, mfccs and chromagrams respectively, and then these features are displayed in plots. All the plots as images are finally converted to arrays of same shape (32, 32, 3) and of type float 32,and then rescaled in a 0-1 range (each value is divided by 255).

Taking one piece of *dog bark* as example, clearly, in **Fig.4**, we can see that mel-spectrogram presents something like objects in **Fig.(a)**, and also, MFCCs plot and chromagram shows some shape. **Fig.5** displays the new plots after conversion to shape (32, 32, 3). Fortunately, although some details got lost, the broad outlines remain, making it possible to do classification task with combined images. **Fig.5** shows how the audio sample combine into a combined image.

3 Methodology

For all the three architectures experimented in this work, the loss function used for training is **Categorical Cross-Entropy**, and optimizer is **Adam** with learning rate **0.001**. Besides the last

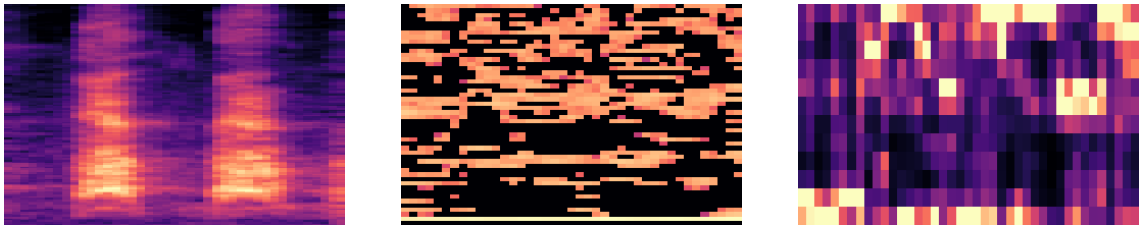


(a) Air Conditioner **(b)** Car Horn **(c)** Children Playing **(d)** Dog Bark

(e) Drilling **(f)** Engine Idling **(g)** Gun Shot **(h)** Jackhammer

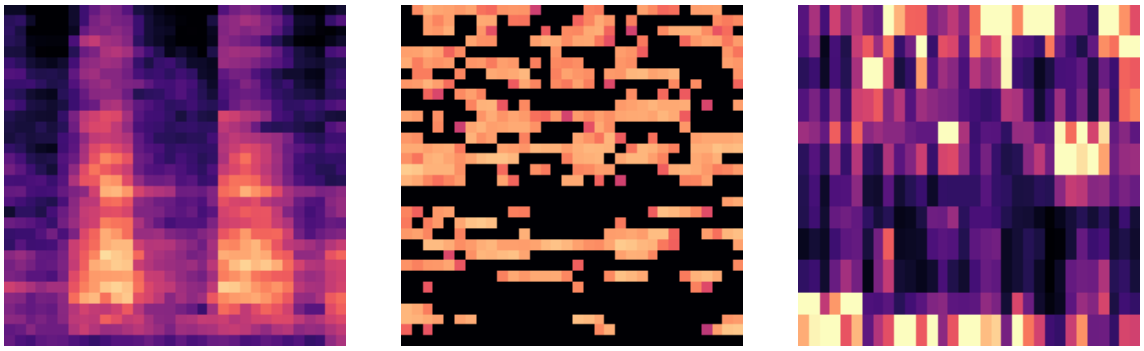
(i) Siren **(j)** Street Music

Fig. 3: Mel-spectrogram of each Label



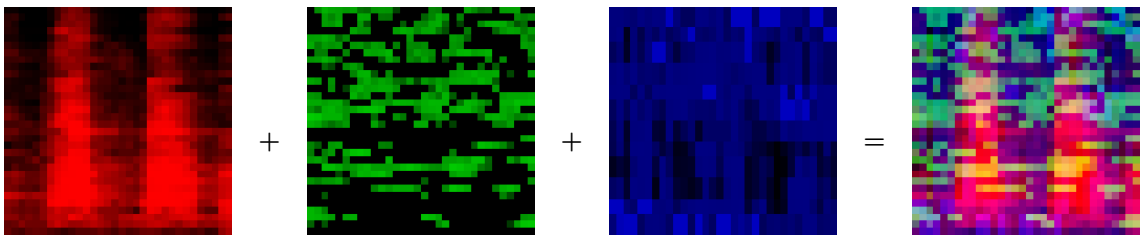
(a) Mel-Spectrogram **(b)** MFCCs plot **(c)** Chromagram

Fig. 4: Original Audio Representation Plots



(a) Mel-Spectrogram **(b)** MFCCs plot **(c)** Chromagram

Fig. 5: Audio representation plots (32, 32, 3)



(a) Red Spectrogram **(b)** Green MFCCs **(c)** Blue Chromagram **(d)** Combined

Fig. 6: Formation of Combined Image (32, 32, 3)

fully connected layer uses **Softmax** as activation function, the other layers all employs **Rectified Linear Unit** (ReLU) as activation function. The assessment of the performance on training set and validation set is **Accuracy** which is the percentage of samples predicted correctly. Batch sizes of all the models are set to **16**, and epochs are all set to **50**. To reduce overfitting and halt the training at the right time, **Early Stopping** is introduced. Early stopping is triggered when validation loss no longer decreases and simultaneously **5** epochs are set as a delay to trigger early stopping. Test performance across the test folds is evaluated by both **Accuracy** which is the percentage of labels predicted correctly, and **Standard Deviation**.

3.1 Convolutional Neural Network

This work adopts **VGG based Convolutional Neural Network** (CNN), which developed by K. Simonyan and A. Zisserman for the ImageNet challange of 2014. VGG architecture is based on blocks, and thus it is easier to organize the neural networks. Shown in **Fig.7**, each block contains 2 Conv2D layers and 1 max pooling layer. After running all the blocks, outputs will get flattened and then fed to two fully connected layers for final classification. Both 1-block and 3-block architectures are used for training classification with images and features as input respectively.

The hyperparameters is set as a simplified form of original work. 2 Conv2D layers in the same block are set same. 32, 64, 128 are the number of the filters for first, second, and third block respectively. For all the blocks, kernel size is (3,3), stride is 1, pool size is (2,2) and to prevent overfitting, **Dropout** is used after each block. The dropout rate is set as 0.2. For the two fully connected layers in each model, 128 and 10 neurons are set in sequence.

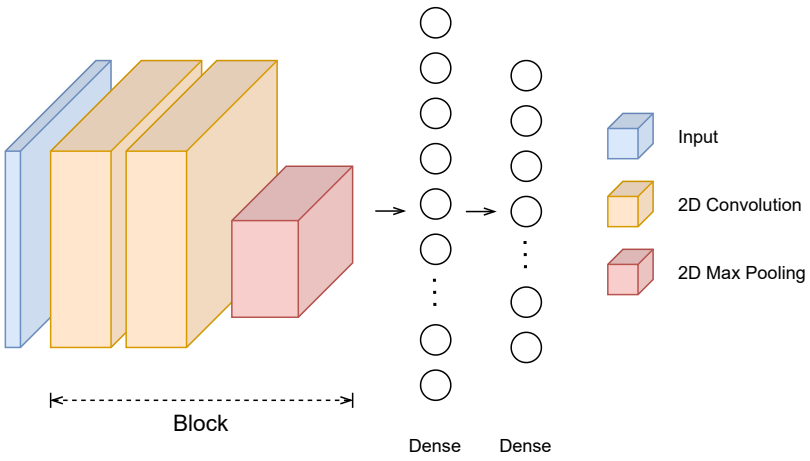


Fig. 7: Architecture of VGG based Convolutional Neural Network

3.2 Convolutional Recurrent Neural Network

In the case of CNN, filters move along y axis same as x axis to detect the object or pattern in the matrix. However, different from the "common" image recognition or pattern recognition, such as fruit classification, where both axes represent size, y axis and x axis now are of different scales and different meanings. One axis means frequency, and the other axis means time, so different operation should be taken to learn audio classification. What's more, samples along time series may highly correlated to several previous samples. If we only use CNN, we miss the time features along x axis, which may bring serious bias. To capture the time domain, the idea of Sang et al. [5] and Nasrullah et al. [6] is implemented here. They introduced **Recurrent Neural Network** (RNN) and then combined it with CNN, so to train the **Convolutional Recurrent Neural Network** (CRNN).

Inputs of CRNN are of one-channel features. The architecture of CRNN is described in **Fig.8**, which can be divided into three stages. The first stage contains three 2D convolution layers and three max pooling layers. The output of final max pooling is assured to have single vector on frequency axis. The second stage is single **Gated Recurrent Unit** (GRU) layer. Finally, the output is flattened to feed two fully connected layers for classification.

The hyperparameter setting of the first stage is listed in **Table 1**. In Conv2D, the values in the cell represent (filter, kernel size, stride). Dropout is used after each max pooling layer to reduce overfitting. The output of first stage is reshaped to (41, 64) and then feed GRU layer which has 32 units. After another Dropout with rate 0.2, the data finally arrive in fully connected layers where the neurons are set 128 and 10 respectively.

Table 1: Hyperparameter Setting of CRNN

	Spectrogram	**All Features**
Conv2D	(16, (3,3), 1)	(16, (3,3), 1)
Pool Size	(1, 4)	(1, 5)
Dropout	0.1	0.1
Conv2D	(32, (3,3), 1)	(32, (3,3), 1)
Pool Size	(1, 4)	(1, 4)
Dropout	0.1	0.1
Conv2D	(64, (3,3), 1)	(64, (3,3), 1)
Pool Size	(1, 5)	(1, 5)
Dropout	0.1	0.1

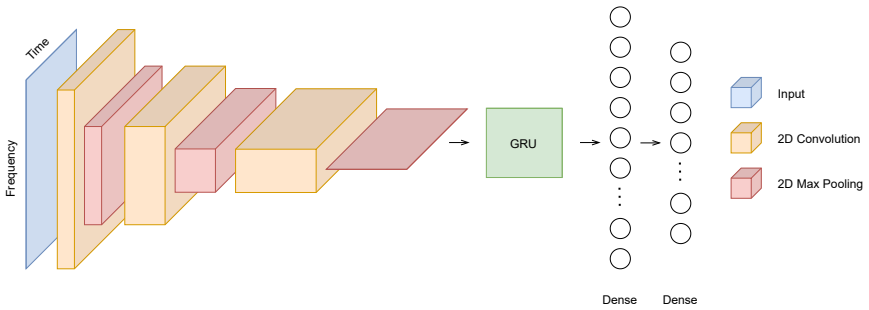


Fig. 8: Architecture of Convolutional Recurrent Neural Network

3.3 Parallel Recurrent Convultional Neural Network

Besides CRNN, another hybrid architecture used in this project containing RNN is what proposed by Lin et. al [7], **Parallel Recurrent Convolutional Neural Network** (PRCNN). This neural network is composed of parallel CNN and Bi-RNN blocks as shown in **Fig.9**. In the case of CRNN, although the design of architecture considers time domain, it does not get temporal relationships of original raw signal but the output of convolution layers. In contrast, PRCNN takes one-channel features as input and extracts spatial features and temporal features in parallel manner, and then two outputs from the two blocks are fused into a one representation. The concatenated result is feed to a fully connected layer which has 10 neurons to obtain the final classification.

In the CNN block, similar to the work of Lin et. al [7], there are **5** repetitive operation chains, which are 2D convolution layers followed by 2D Max Pooling layer. The kernel size of the 5 operation chains are all set as (3, 1), stride as 1 and pooling size as (2, 2) which is smaller than the one set in

the original paper (they set (4, 4)). The number of filters is 16 for first chain, 32 for second chain, and 64 for the last three chains.

In the RNN block, input data are firstly processed by 2D max pooling layer to reduce the dimension. The chosen pooling size is (4, 2). The output is then embedded to feed a bidirectional GRU with **32** units. Better than a standard RNN, bidirectional GRU considers not only the previous context but also the following context so is more suitable for the urban sound characteristics.

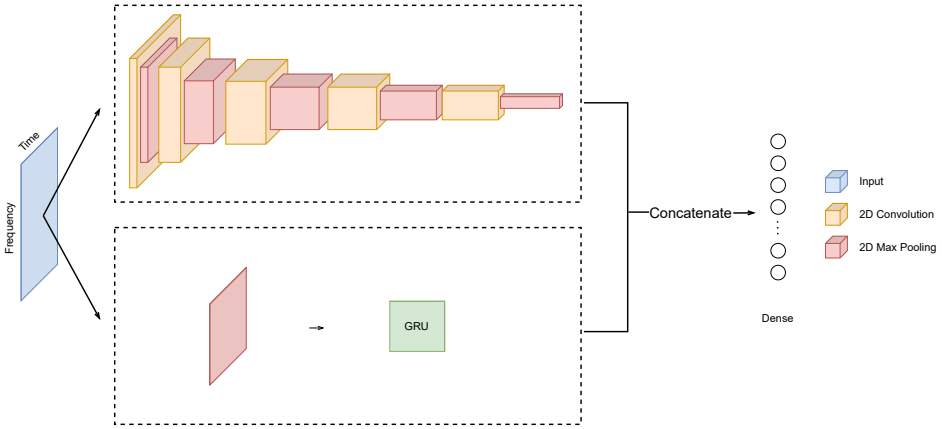


Fig. 9: Achitecture of Parallel Recurrent Convolutional Neural Network

4 Experiment Results

In total, 12 different models are experimented as listed in **Table 2**, with different architectures, and different types of inputs and thus they can be divided into 4 categories which are Models 1-4, Models 5-8, Models 9-10 and Models 11-12. Models 1-4 take images as inputs, whereas the rest of models take features as inputs. All the images are of shape (32, 32, 3). Features inputs for CNN in Models 5-8 are all of shape (time, number of features, 2), where 2 represents the number of channels and the second channel of data is the corresponding delta of the features in the first channel. Models 9-12 takes inputs of shape (time, number of features, 1). Due to the data preprocessing, time of all the inputs is 41.

Table 3 reports the performances of models on each test fold and also average accuracy and standard deviation across the test folds. **Fig.10-13** depict how accuracy and loss changes as epochs increase during training process.

Model	Architecture	Input	Input Shape	Training Parameters
Model 1	VGG - 1 block	Image of Spectrogram	(32, 32, 3)	1,060,138
Model 2	VGG - 3 blocks	Image of Spectrogram	(32, 32, 3)	550,570
Model 3	VGG - 1 block	Combined image	(32, 32, 3)	1,060,138
Model 4	VGG - 3 blocks	Combined image	(32, 32, 3)	550,570
Model 5	VGG - 1 block	Spectrogram features	(41, 128, 2)	5,254,154
Model 6	VGG - 3 blocks	Spectrogram features	(41, 128, 2)	1,598,858
Model 7	VGG - 1 block	All features	(41, 190, 2)	7,793,674
Model 8	VGG - 3 blocks	All features	(41, 190, 2)	2,172,298
Model 9	CRNN	Spectrogram features	(41, 128, 1)	38,218
Model 10	CRNN	All features	(41, 190, 1)	38,218
Model 11	PRCNN	Spectrogram features	(41, 128, 1)	54,570
Model 12	PRCNN	All features	(41, 190, 1)	61,162

Table 2: Summary of all the models

Model	Fold 7	Fold 8	Fold 9	Fold 10	Average	Standard Deviation
Model 1	0.50	0.52	0.44	0.51	0.4925	0.0359
Model 2	0.49	0.54	0.53	0.51	0.5175	0.0222
Model 3	0.47	0.47	0.48	0.53	0.4875	0.0287
Model 4	0.47	0.55	0.44	0.52	0.4950	0.0493
Model 5	0.51	0.57	0.56	0.59	0.5575	0.0340
Model 6	0.54	0.55	0.56	0.61	0.5650	0.0311
Model 7	0.48	0.57	0.53	0.55	0.5325	0.0386
Model 8	0.53	0.59	0.56	0.62	0.5750	0.0387
Model 9	0.55	0.57	0.59	0.62	0.5825	0.0299
Model 10	0.55	0.55	0.61	0.60	0.5775	0.0320
Model 11	0.53	0.56	0.52	0.62	0.5575	0.0450
Model 12	0.57	0.61	0.53	0.61	0.5800	0.0383

Table 3: Summary of Performance

Category 1, the first 4 models which takes images as input shows lower accuracy although they have far more training parameters. With the same architecture of neural network, different kinds of images share similar classification accuracy. And with the same image inputs, increase of complexity of CNN does not make a difference. In all the subplots of **Fig.10**, the 4 models suffer from severe overfitting, although Dropout and Early Stopping are both adopted. And validation loss increases as training loss decreases, especially in simplest case Model 1. The training accuracy can reach around 90% by the end of training, but the validation accuracy is hard to reach 50%, implying that model of VGG based CNN with images as input has trouble with generalization.

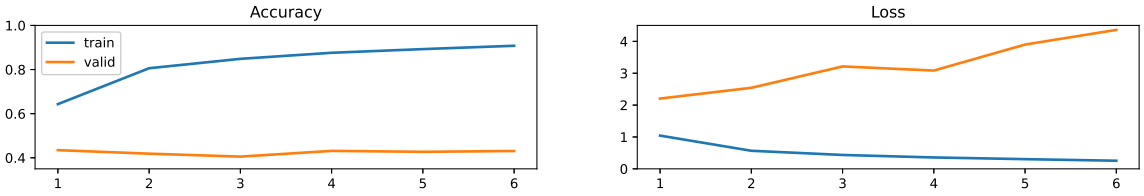
Since **Category 2, Models 5-8** takes inputs of two channels for CNN, they have larger size of input and thus need more parameters, time and computer resources for training. Compared to Models 1-4 which take images as input, they have same architectures but better performances. Obviously, models with more blocks can have higher accuracy. From **Fig.11**, overfitting problem still exists, but better than the case in Models 1-4. What's more, validation loss has lower magnitude.

Category 3, Models 9-10 start to consider time features. As what I expect before the experiment, performance improves. After introducing RNN, not only average accuracy increases but also overfitting is controlled. Model 9 has achieved highest accuracy and lower standard deviation (only a bit higher than Model 2). From **Fig.12 (a)**, clearly, it has smoother curves of accuracy and loss for both training and validation, proving that this model has lower variance. However, more features do not bring more satisfactory results. Model 10 with more features has slight lower average accuracy than Model 9 and higher standard deviation. In **Fig.12 (b)**, both accuracy and loss fluctuate throughout the training process, suggesting the high variance of Model 10.

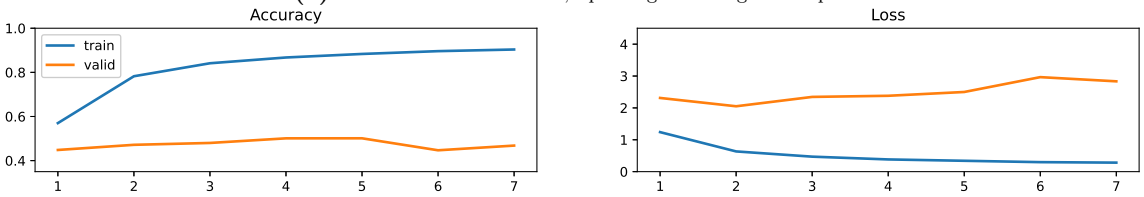
Different from Models 9-10, **Category 4, Models 11-12** deal with spatial features and temporal features at the same time for better learning along time axis. However, PRCNN obtains lower average accuracy and higher standard deviation. The good point is that, from **Fig.13**, the validation curves of accuracy and loss get closer to training curves. Meanwhile, the evolution of the validation accuracy and loss as functions of epochs are both more stable, indicating that PRCNN has lower variance and higher robustness.

5 Error Analysis

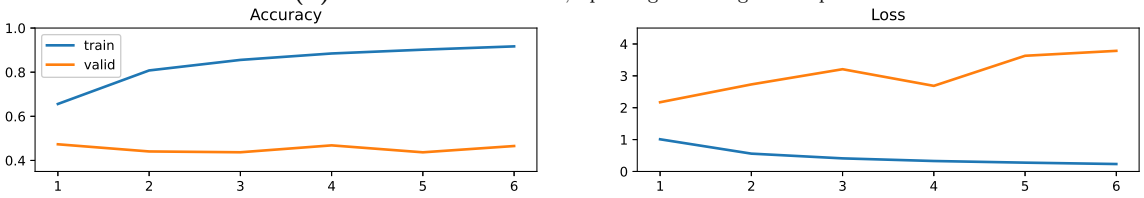
For more details of different models, *Recall* is introduced. Recall means that, for one target label, how many predicted labels are correct out of all the labels predicted as target label, suggesting that how efficiently the classifier can capture the features of the target class. The heatmap in **Fig.14** illustrates recall of each class in different models. The most significant differences among models come from *air conditioner* and *jackhammer*. We can find that the first category Models 1-4 have fairly bad performance on classifying *air conditioner* and *jackhammer*. Models 5-8 which give more information as inputs, compared to first category models, have higher recall in detecting *jackhammer*, especially Model 8. But second category models still have low recall in *air conditioner*. By



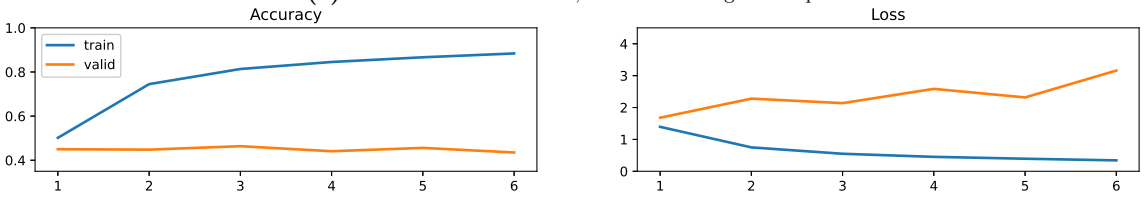
(a) Model 1: VGG - 1 block, Spectrogram Images as Input



(b) Model 2: VGG - 3 block, Spectrogram Images as Input

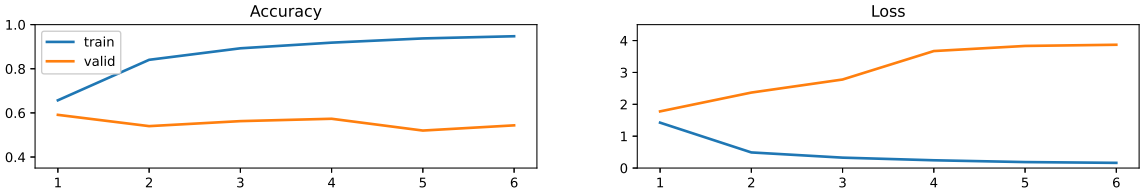


(c) Model 3: VGG - 1 block, Combined Images as Input

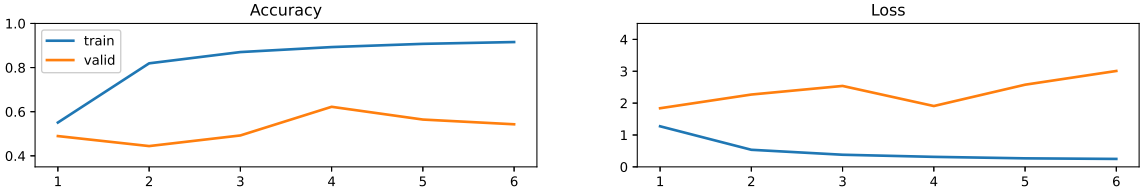


(d) Model 4: VGG - 3 block, Combined Images as Input

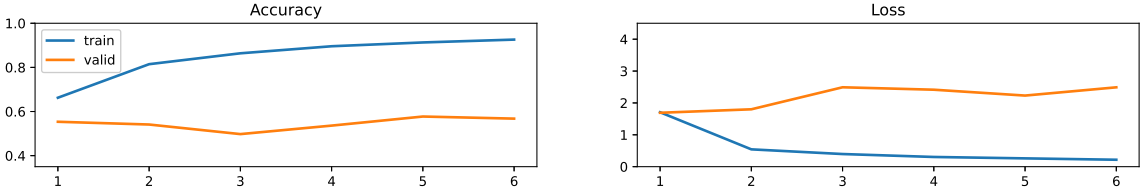
Fig. 10: Accuracy and Loss of Category 1 (CNN, Images as Input)



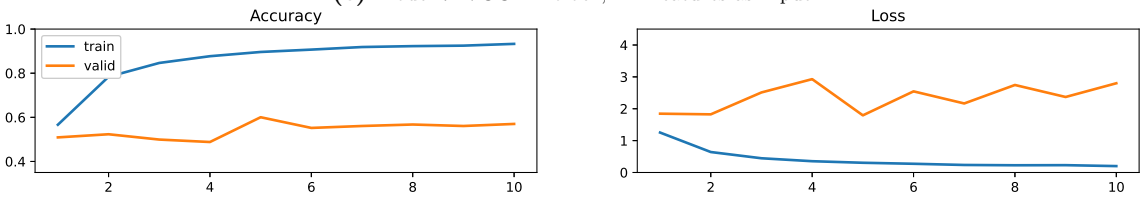
(a) Model 5: VGG - 1 block, Spectrograms as Input



(b) Model 6: VGG - 3 block, Spectrograms as Input

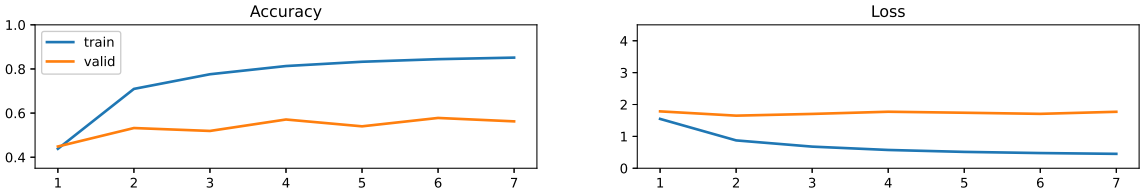


(c) Model 7: VGG - 1 block, All Features as Input

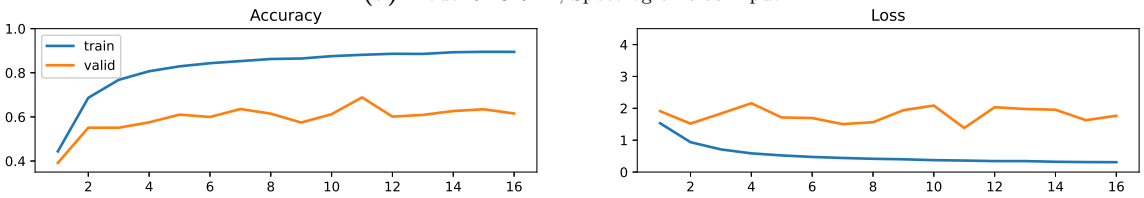


(d) Model 8: VGG - 3 block, All Features as Input

Fig. 11: Accuracy and Loss of Category 2 (CNN, Features as Input)

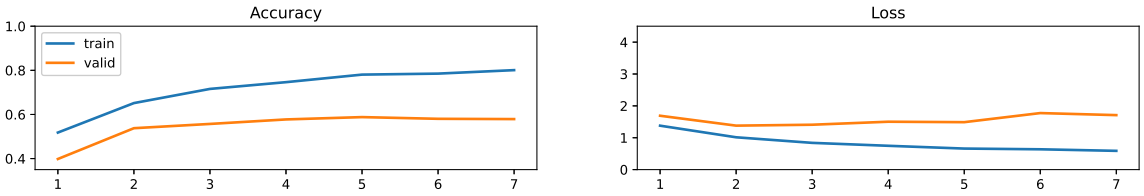


(a) Model 9: CRNN, Spectrograms as Input

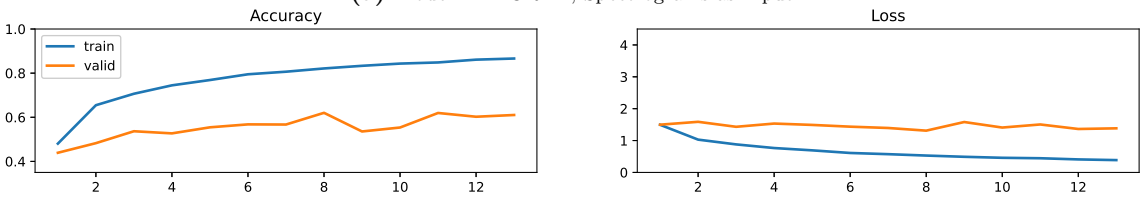


(b) Model 10: CRNN, All Features as Input

Fig. 12: Accuracy and Loss of Category 3 (CRNN, Features as Input)



(a) Model 11: PCRNN, Spectrograms as Input



(b) Model 12: PCRNN, All Features as Input

Fig. 13: Accuracy and Loss of Category 4 (PRCNN, Features as Input)

introducing RNN, Models 9-10 successsfully raise the recall of *air conditioner*. In addition, category 4 models demonstrate more efficiency in classifying *air conditioner* and *jackhammer*.

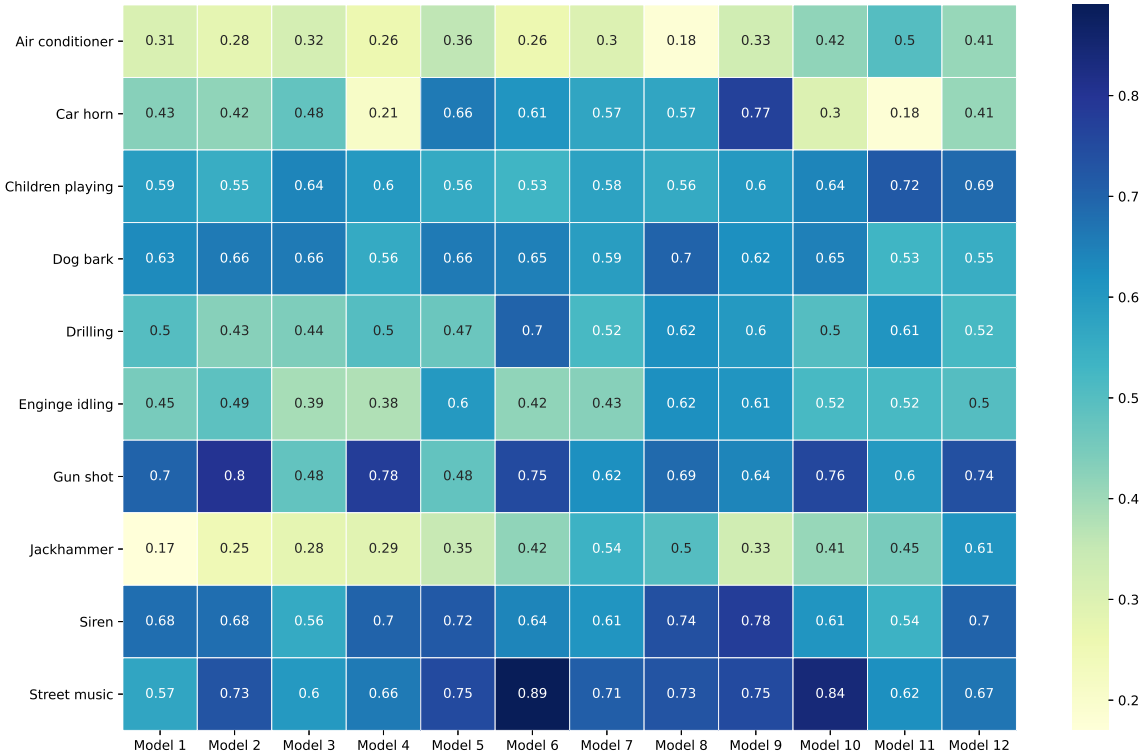


Fig. 14: Recall of Models

Fig.15 displays the confusion matrices of 4 representative models from each category, who have the best performance in their categories. In the confusion matrix, the rows represent ground truth, while the columns represent the prediction.

Model 2 has the best performance in Category 1. In Model 2, *air conditioner* is largely confused by *children playing*, *engine idling* and*street music*. It makes sense since the image of spectrogram of these 4 classes look similar, bringing more difficulties to image recognition. Model 2 also classifies 848 of 2,075 samples of *jackhammer* as *drilling*, implying that recognition mechanism fail to detect the features of *jackhammer*. And Model 2 is hard to figure out the difference between *children playing* and *street music*.

Model 8 takes all features and corresponding delta as input, and 3-block CNN as architecture. It often classifies *jackhammer* as *drilling*, similar to the Model 2. *Air conditioner* is mainly confused by *dog bark*, *engine idling* and *jackhammer*, worse than the case in Model 2. *jackhammer*'s performance improves by less error on other labels.

Model 9 takes all features as input and uses CRNN, less feature information while more time features than Model 2. The good thing is that the recall of *air conditioner* increases, and main error results from *street music*. But the problem of *jackhammer* is much worse than Model 8.

More advanced than Model 9, Model 12 employs PRCNN and thus is able to learn time features more wisely. Obviously, performance on *air conditioner* and *jackhammer* both get promoted. However, performance on *street music* declines.

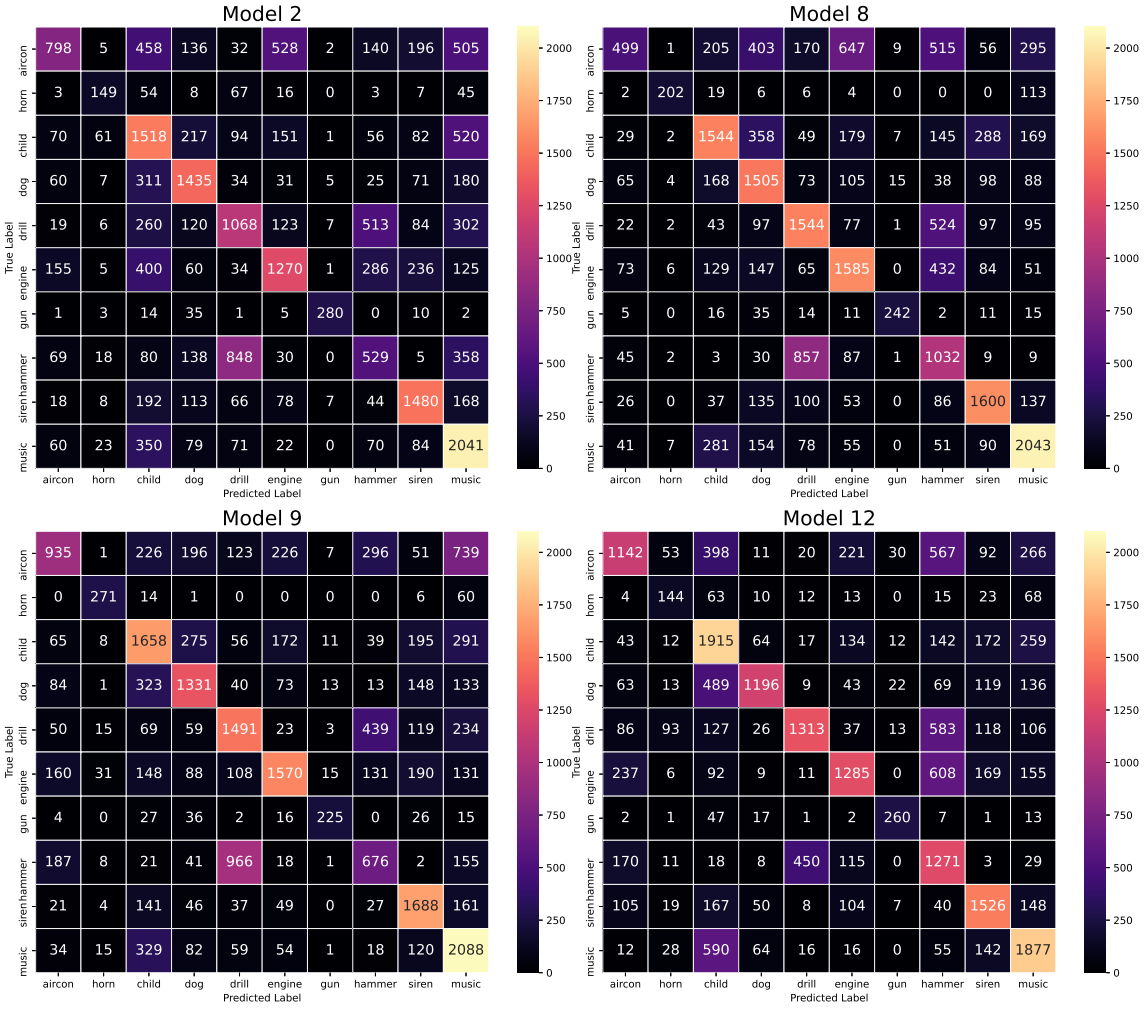


Fig. 15: Confusion Matrix

All the models have trouble in distinguishing *jackhammer* and *drilling*, and as complexity of model increases, this problem never disappear. But recall of *drilling* is not too low compared to that of other labels and final accuracy, suggesting that *jackhammer* may share large number of features with *drilling*. Although the problem remains, from **Fig.14**, we can find that concerning more time features as what PRCNN does or adding more features as what Model 5-8 do, can help relief the situation.

6 Conclusions

In this work, through the experiment results and error analysis, we can answer the three questions proposed in the beginning. Firstly, audio represented by images does not work well in this

task. Results of Model 1-4 proves that image recognition mechanism failed in my experiment. Image recognition is not an optimal way for urban sound classification. Secondly, more features like MFCCs and chromagram, cannot bring big difference on performance compared to the only spetcrogram case. And in some cases, more features increase the instability of the model, resulting in higher variance and lower robustness, especially in CRNN and PRCNN. Thirdly, after concerning time domain, accuracy makes significant progress. However, the way of learning time domain influences the final result. Overall, the best model for the classification so far is Model 9, with spectrogram features as input and CRNN as architecture.

The highest accuracy obtained so far is 58%, and simultaneously some of the folds can reach more than 60%. The performances of neural networks chosen in this work are not high. But they are underestimated because the hyperparameters are not tuned limited to the resources, but follow the simiplified design of original paper. In addition, all the audio has been processed to less than 1 second, but in fact CRNN and PRCNN are proposed for longer music files in the previous research work. Probably, splitting audio files into longer pieces may lead to a more excellent result, and classification can benefit more from models which concerning time features, such as CRNN and PRCNN.

7 Appendix

This project is performed through Python language, and code book is shared on Github.

8 Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

References

1. J. Salamon, C. Jacoby and J. P. Bello, "A Dataset and Taxonomy for Urban Sound Research", *22nd ACM International Conference on Multimedia*, Orlando USA, Nov. 2014
2. McFee, Brian, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. "librosa: Audio and music signal analysis in python." In Proceedings of the 14th python in science conference, pp. 18-25. 2015
3. Karol J. Piczak, "Environmental sound classification with convolutional neural networks", *IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING*, SEPT. 17–20, 2015
4. Venkatesh Boddapati, Andrej Petef, Jim Rasmusson, Lars Lundberg, "Classifying environmental sounds using image recognition networks", *International Conference on Knowledge Based and Intelligent Information and Engineering Systems*, 2017
5. Jonghee Sang, Soomyung Park, Junwoo Lee, "Convolutional Recurrent Neural Networks for Urban Sound Classification using Raw Waveforms", *26th European Signal Processing Conference (EUSIPCO)*, 2018
6. Zain Nasrullah, Yue Zhao, "Music Artist Classification with Convolutional Recurrent Neural Networks", Mar. 2019
7. Lin Feng, Shenlan Liu, Jianing Yao, "Music Genre Classification with Paralleling Recurrent Convolutional Neural Network", Dec. 2017