What's That Sounds?
Machine Learning for Urban Sound Classification

Li, Jingyang jil050@ucsd.edu
Wang, Yinuo yiw010@ucsd.edu
Zhu, Hao h8zhu@ucsd.edu
Zhang, Yuhan yuz053@ucsd.edu

***Abstract*— In this project, we used four machine learning models, Neural Network, Convolutional Neural Network, Recurrent Neural Network, Boosting and Ensemble models to classify the urban sound 8k dataset. Our results show that the models have good performance according to the accuracy of the classification.**

I. Introduction

A. Background and Motivation

The classification of environmental sound has lots of applications in large scale and content-based multimedia indexing and retrieval. Nowadays, because of multimedia sensor networks and large quantities of online multimedia contents, people pay more attention to sound classification in urban environments. There are various scenarios and fields such as health services and communication where an accurate urban sound recognition system can assist people. For example, in the area of health services, it can help people who have disability to recognize sound of their surroundings. Also, with the help of sound classification, while giving recommendations or providing services, applications in mobile devices can not only base on location information but also base on surrounding sound information.

There are mainly two tasks in our project:

1) Extract features from audios
2) Use different machine learning models to classify urban sounds and compare their performance

B. Input and Output

In this project, by using UrbanSound8K dataset [1][2], we trained models including NN (Neural Network), CNN (Convolutional Neural Network), RNN (Recurrent Neural Network) and boosting to classify sounds in urban environments.

- The input to NN model is a matrix that each row combined three features - mel-frequency spectrum, Contrast and Chroma. We then use a two-hidden-layer Neural Network to output a class ID indicating what's the sound.
- The input to CNN model are images generated by calculating and displaying the mel-frequency spectrum. We then use a pre-trained Convolutional Neural Network to output a class ID indicating what's the sound.
- The input to RNN model is a matrix with each row as the Mel-frequency cepstrum coefficient of one sample. We then use a Recurrent Neural Network to output a class ID indicating what's the sound.
- The input to Boosting model is the same with the features used in the first model. We then use a gradient boosting to output a class ID indicating what's the sound.

The output of the models is class ID. The TABLE I gives a summary of the input and the input format of these models.

Model	Input	Input format
NN	MFC Contrast Chroma	A matrix that each row combined three features
CNN	Mel-frequency spectrum image	Images of Mel-frequency spectrum
RNN	Mel-frequency Cepstrum Coefficient	A matrix of Mel-frequency cepstrum coefficient
Boosting	Same as NN	Same as NN

TABLE I

Inputs of different models

II. Related Work

A. Works on sound classification

Conventionally, solving the audio classification problems involving two steps, extracting features and classification. The features extracted are sophisticated hand crafted features. For example, MFCC[3] is widely used over decades. The extracted features are then fed in to classifier like SVM[2], HMM[4], GMM[5] and NMF[6] to perform the classification task.

After Deep Neural Networks (DNN) and Convolutional Neural Networks (CNN) becoming popular in recent years, more and more research tend to exploit the inherent feature learning that comes with these architectures. They input less processed data including raw waveform or spectrum into the architecture and let the model extract the feature.

In [7], audio signals are split into segments using overlapped windows and converted to log-Mel spectrum, then input into a BLSTM RNN. The proposed RNN model outperforms their baseline, a fully connected Neural Network.

In [8], different kinds of deep CNNs have been tested on subsets of YouTube-100M dataset. The audios were windowed and convert to log-Mel spectrum to input into the model. Multiple architectures including fully connected DNN as baseline, AlexNet, VGG, Inception and ResNet are used. The ResNet and Inception has the best performance being able to capture the common features in different areas in the audio representation.

In [9], features extracted by CNN is found better than MFCC when used in same classifier. But that does not mean traditional features are out of date or use. In [10], a model of i-vectors based on MFCC is evaluated along with CNN on sound classification task, showing that they capture complementary information, and the performance was even better when combining these two models.

B. Works on UrbanSound8K

There are many previous studies on the dataset we study on, UrbanSound8K. Among them, there are both studies on traditional methods that involve feature engineering, and studies that let the model to extract feature.

In [2], baseline for the dataset UrbanSound8K is presented, MFCC is extracted as feature, on which 5 classification algorithms, SVM, decision tree, k-NN, random forest, ZeroR are tested. SVM and decision tree have the best classification result with an average of 70% in accuracy. The method was week on distinguishing between timbre similarity, especially between wide-band noise-like continuous sounds, like air conditioners and idling engines, jackhammers and drills.

In [11], CNN was evaluated on UrbanSound8K. The architecture has two convolutional layers and two fully connected layers and was trained with dropout. The input was also log-scaled mel-spectrograms on windowed sound clips. The result of the experiment shows that longer window and probability voting among windowed segments at output has better performance, reaching 73% in accuracy.

In [12], raw waveform down-sampled to 8kHz and standardized to 0 mean and variance 1 are used to train a very deep CNN from 3 layers to 34 layers. The 18-layer model has the best accuracy of 71.78%, which is close to the result of CNN model with spectrum input above, but used much more layers to achieve.

III. Dataset and Features

A. Dataset

The dataset we use is UrbanSound8K Dataset [2], which is available at the following webpage: `https://urbansounddataset.weebly.com/urbansound8k.html`. In this dataset, there are 8732 urban sound excerpts which are labeled and divided into 10 folders. Each audio file has a name formatted as [fsID]-[classID]-[occurrenceID]-[sliceID].wav, where

- fsID is the Freesound ID of the recording from where this excerpt (slice) is taken.
- classID is a numeric identifier of the sound class.
- occurrenceID is a numeric identifier to distinguish different. occurrences of the sound within the original recording.
- sliceID is a numeric identifier to distinguish different slices taken from the same occurrence.

We use data from 80% audios to train our model, and use the other 20% to test the performance of the models.

Numeric Identifier	Sound Class
0	air_conditioner
1	car_horn
2	children_playing
3	dog_bark
4	drilling
5	engine_idling
6	gun_shot
7	jackhammer
8	siren
9	street_music

TABLE II
SOUND CLASS ID

Fig.1 is an example of sound wave from raw data. [2]

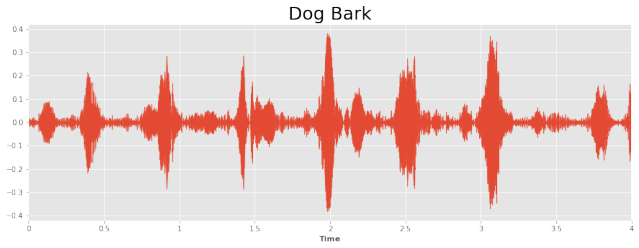

Fig. 1. Sample of sound wave of dog bark

B. Features

We extract audio features using LibROSA[13]. Features of 4 different categories are extracted:

- Spectrogram: log-Mel spectrogram
- Mel-frequency cepstral coefficients (MFCC)
- Pitch feature: chroma
- Spectral statistic representations: spectral contrast[14]

1) Mel-frequency spectrum: The Mel-frequency spectrum is a nonlinear Mel-scaled representation of the linear cosine transformation of short-term power spectrum of a sound. [15] Below is the formula to convert frequency to Mel-scale, it provides a model of the frequency perception of human. [16]

$$M(f) = 1125 \ln(1 + f/700)$$

2) Mel-frequency cepstrum coefficients (MFCC): Mel-frequency cepstral coefficients (MFCCs) are coefficients of MFC, which are widely used features in sound recognition, [17] because this feature mimics the way the human hearing system works. [18] There are a few steps to extract the MFC and MFCC features. First, window a given piece of signal to divide it into n small samples. After that, apply the Fast Fourier Transform to all the samples and get their spectrum. Then, scale these spectrum using Mel-frequency Filter to obtain the Mel-frequency spectrum. Take log of the spectrum and apply IFFT to get the Mel-frequency cepstrum and the Mel-frequency cepstrum coefficients.

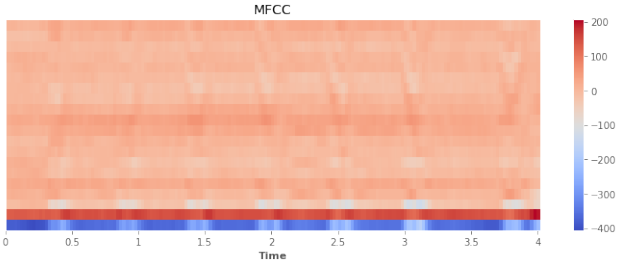

Fig. 2. Example of MFCC feature of Dog Barking

3) Chroma: Chroma-based features refer to the color of a pitch, which is an important tool for analyzing music. [18][19] Their main property is that they capture important characteristics of music such as harmonic and melodic. [17] Chroma of a signal can be extracted by the following way. Window a signal into short-time pieces, then use a short time fourier transform(STFT) to get the spectrum. Then, apply Constant-Q transform(CQT), Chroma Energy Normalized (CENS), and so on to get this feature. [17]

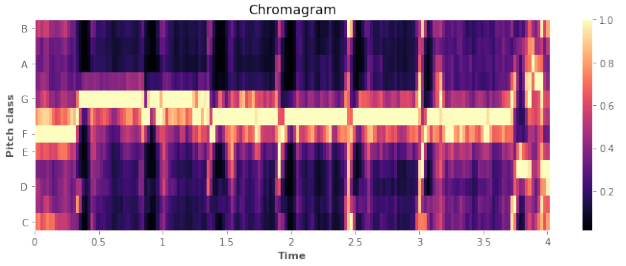

Fig. 3. Example of chroma feature of Dog Barking

4) Contrast: Spectrum contrast represents the relative spectral distribution instead of average spectral envelope.[14] It takes in to consideration of the spectral peak, the spectral valley, and their difference in the subband of each frequency. [20]

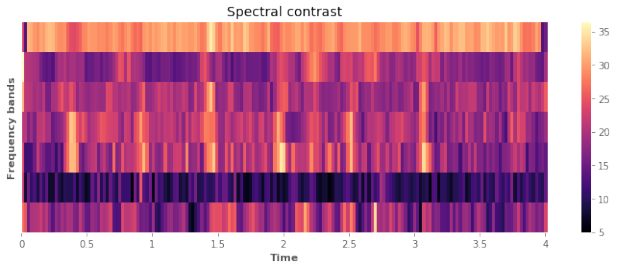

Fig. 4. Example of contrast feature of Dog Barking

IV. Methods

A. Neural Network (NN)

We use a two-hidden-layer Neural Network to classify audios, which contains 512 cells in each layer.

1) Mathematical expressions: Neural Network consists of basic computational unit "neurons". For each neuron, assume that it takes inputs as $x_1, x_2, ..., x_k$ and outputs. [21][22]

$$f_{out} = f_{sigmod}(W^T x) = f_{sigmod}(\sum_{i=1}^{k}(W_i x_i + b)) \tag{1}$$

where W is weights, b is bias. f_{sigmod} is the sigmod function which is one of activation function. [21] $f_{sigmod}(x) = \frac{1}{1+exp(-x)}$. In a neural network, there can be multiple layers. We use a 4-layer NN model with 2 hidden layers. Suppose that L_{in} is the input layer, L_{out} is the output layer, $L_{hidden1}$ and $L_{hidden2}$ are the hidden layers. Suppose that in layer l, there are n neurons. Then suppose the sum of the weighted inputs of the ith neurons in the layer l is $o_i^{(l)}$. Then we can get the $o_i^{(l+1)}$ by the following equation. [21] For $l = 1$

$$o_i^{(2)} = \sum_{j=1}^{n}(W_{ij}^{(1)} x_j + b_i^{(1)}) \tag{2}$$

For $l > 2$

$$o_i^{(l+1)} = \sum_{j=1}^{n}(W_{ij}^{(l)} f_{sigmod}(o_j^{(l)}) + b_i^{(l)}) \tag{3}$$

2) Backpropagation algorithm: The backpropagation algorithm is used to calculate the gradient of the loss function. Basically, we calculate the above steps repeated to train the neural network. By doing this, we can reduce the cost function. Suppose that the loss function is $L(W, b)$, then the parameters can be updated by the following equations. [21]

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} L(W, b) \tag{4}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} L(W, b) \tag{5}$$

B. Convolutional Neural Network (CNN)

We extract mel-frequency cepstrum feature from audio files and then display the power of the mel-frequency cepstrum. Saving all these features as images. Then, we use a 34 layer ResNet to train the model. The convoutional neural network is mainly used to do image classification. In CNN, we input an image of $height \times width \times dimension$. There are a few layer of CNN going through to implement a classification. [23]

1) convolution layer: input image, choose parameters, use filters with strides, padding, and apply convolution on the image and use ReLU activation to deal with the matrix.
2) Pooling layer: reduce dimensionality size using pooling.
3) Add 33 more convolutional layers.
4) Fully connected layer: feed the flattened output into an FC Layer.
5) Output the class and classify images.

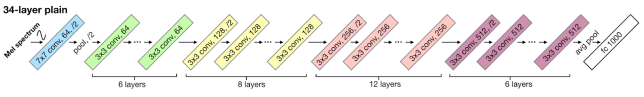

Fig. 5. A model of 34-layer CNN

The residual module introduces skip connections. In this way, it is easier for network layers to represent the identity mapping. [24]. The building block of the ReLU model can be expressed by the following equation.[24]

$$y = F(x, \{W_i\}) + W_s x \tag{6}$$

x is the input vector of the layer and y is the output vector of the layer. $F(x, \{W_i\})$, which

is the residual mapping to be learned. [24] The following shows a block.

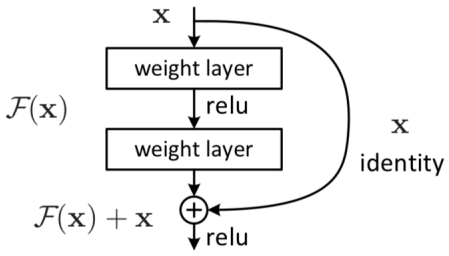

Fig. 6. A building block of ReLU model

C. Recurrent Neural Network (RNN)

RNN can be considered as a particular Convolutional Neural Network with memory and it can save information from previous training. A RNN is just like an assembly line formed by multiple copies of the same network and each network pass message to the next network. In each network, the output of the hidden layer will be saved in the memory. Beside input data, memory can be considered as another input, which may make an influence on training in the next network.

However, this simple RNN has two apparent problems. Firstly, Memory is very short-term. For example, when information from the first hidden layer is passed to the second hidden layer, the information in second will be affected and changed. Therefore, the message passed to the third hidden layer actually is the combination of information of the first and second hidden layer. In the implementation of RNN, information from closer layer will normally have bigger weight. Thus, as the increment of the number of layers, the weight of more previous layer will become less and less and they will make less influence on the current layer. Secondly, simple RNN can not control what will be saved into memory. Not all information needs to be saved and unrelated information needs to be ignored.

In order to avoid above two problems, we use another model of RNN - the Long short term memory(LSTM) [25][26][27].

Memory cell (C): It is the memory space in LSTM model and controlled by gates.

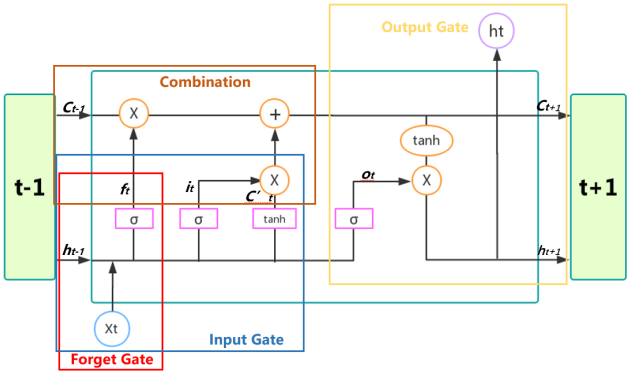

Fig. 7. Structure of LSTM [27]

Forget gate: It is used to decide what information should be thrown away from the memory cell.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{7}$$

Input gate: It controls what input information will be saved. Input information includes current input data and information from previous memory and hidden layer.

$$i_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_i) \tag{8}$$

$$C'_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{9}$$

Combination: Combine previous memory and current memory together.

$$C_t = f_t * C_{t-1} + i_t * C'_t \tag{10}$$

Output gate: It controls what parts of the memory cell state should output to next hidden layer.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{11}$$

$$h_t = o_t * tanh(C_t) \tag{12}$$

Based on the above LSTM structure, the model can only save related and important information in the memory cell and forget information when the model doesn't need it. Thus, problems of simple RNN will be solved and RNN model will have better performance.

D. Boosting

Boosting is an algorithm that mainly aims at reducing bias and variance in supervised learning problems. The main idea of Boosting is using a set of weak learners and combining them together to get a strong learner. Each time, the boosting model will change the weight of samples based on the predictions of the last round training. In our project, we chose to use XGBoost. It is a boosting method based on GBDT aims at solving problems in a fast and accurate way. The basic algorithm of Gradient Boosting is shown as the following Input: training set $(x_i, y_i)_i^n = 1$, loss function $L(y, F(x))$, number of iterations M. Algorithm:
1. Initialize model with a constant value:

$$F_0(x) = argmin \sum_{i=1}^{n} L(y_i, \gamma) \tag{13}$$

2.For m = 1 to M: First, compute the pseudo-residuals:

$$r_i m = -[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}]_{F(x_i)=F_{m-1}(x)} \tag{14}$$

Then we Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals. Next step is to Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = argmin \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)) \tag{15}$$

At last, update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{16}$$

The output will be $F_M(x)$

V. Experience, Results and Discussion

A. Experience

For all the four models, we use grid-search method to find the best model and also tried out many different number of layers of each model. In neural network model, we tuned our hyper-parameter of learning rate, optimizer, batch size. And on the other hand, we also adjust our layers' parameters - such as the number of the layers, the cells and the activation function of each layer. After searching and comparing in different parameters, we found that the best model should be two-hidden-layer neural network model with 512 cells in each hidden-layer and the activation function is sigmoid. For the hyper-parameter, we found that the best ones are with learning rate of 0.01, using Gradient Descent Optimizer and each batch has a size of 1000.

In Convolution Neural Network, we use transfer-learning network layer - ResNet to do the training. So there is not so much to choose, we only change different layer of ResNet. In our project, 34-layer ResNet is the suitable one.

In Recurrent Neural, we study open source code[Ref] about LSTM model and implement it by RNN package in tensorflow. We define the weights, bias, input and output place holder based on data format. Then, we define the prediction with single layer LSTM and loss function and use Adam as training optimizer. From grid-search method, we get optimized hyper-parameters. Learning_rate is 0.01, batch_size is 3000, and the number of iterations is 1500.

In Boosting model, we choose to use XGBoost toolkit. In this model, we tuned the hyper-parameter of max_depth, learning_rate, num_rounds and regularization methods. Using grid-search, we find out that the best parameter is max_depth of 3, learning_rate of 0.3, num_rounds of 100 and using *l2* regularization method.

After we get all those four model, we try to combine all the models and resulting in a better one. Ensemble is the proper method and it can reduce the model's bias and variance simultaneously. [28] The common methods for ensembling are:

- **Bagging**: Use a different, random subset of the training set to train each base model. All the base models have equal weights and they will vote to get the final results. The idea behind this method is just like Random Forest.
- **Boosting**: Train base model iteratively and change samples' weights based on the error rate of the last iteration.
- **Blending**: Train different base models with disjoint data and average (weighted) their outputs. Simple implementation, but lack use of training data.
- **Stacking**: In general, it is to train a multi-layer (generally two layers, the default two layers in this article) learner structure, the first layer (also called the learning layer) with n

different classifiers (or models with different parameters) will The predicted results are combined into a new feature set and used as input to the next level classifier. It can use the whole training set more effectively. This method will be discussed in detail in the next paragraph.

Theoretically, to make ensembling work, there are two key features:

- The correlation between base models need to be as lower as possible. The larger the diversity of the ensemble model, the lower the bias of the final model.
- The gap of performances of the base models should not be too large.

In our project, I think stacking is the proper method to use. The process of stacking was similar to cross-validation. Stacking also splits data into several segments and the number of segments is determined by how many models one use to do stacking. Since we had 4 base models, the training data was first divided into 4 parts. Then our ensemble model started to train models in turn. Each time, the procedure picked one model out of four along with 3 segments of training trials. These 3 segments were trained as the training set for this base model and then the remaining *hold-out* segment was predicted. At the same time, the predictions on the test data were also saved. In this way, each base model predicted one non-overlapping segment of the training set at each iteration and predicted all of the test data. After all 4 iterations were completed, we obtained a matrix which was a number of training data rows by the number of base models matrix, and then this matrix was used as the training data for the second layer of the final model. The final model we also choose to use boosting model. As you can see in Table 3., the results of ensembling model are also the same compare with boosting. We think that may because the result of base models are great enough to combine them together will get almost the same result with the best one.

B. Results

The results can be shown as TABLE III.

Rate	Training accuracy	Testing accuracy	F-Score	Precision	Recall
NN	0.902	0.822	0.82	0.84	0.813
CNN	0.963	0.878	0.877	0.891	0.869
RNN	0.919	0.83	0.827	0.832	0.832
Boosting	0.999	0.916	0.92	0.925	0.916
Ensemble	0.999	0.912	0.916	0.921	0.912

TABLE III

RESULTS OF MODELS

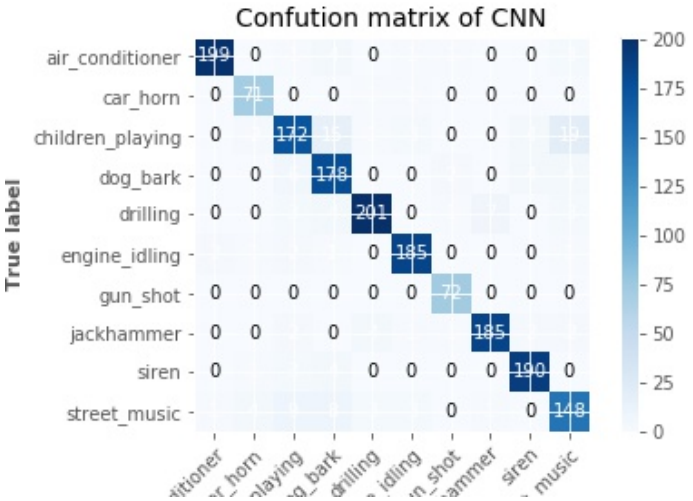

Fig. 8. Confusion matrix of CNN

C. Discussion

1) Meaning of urban sound classification: With the help of sound classification, while giving recommendations or providing services, applications in mobile devices can not only base on location information but also base on surrounding sound information. Thus, applications can provide more accurate service based on surrounding environment.

Also, it can improve the life quality of the disabled who suffers from hearing problems. If a device can provide a description of surrounding environment based on the sound classification to the disabled, it will definitely help them avoid some dangers and make their life more convenient.

2) Meaning of the results: From the above result table, we can tell that our results are quite great. Its because we did many analysis and experiments for extracting proper features and finding suitable methods. For each feature, we try to

understand their physical meanings and choose a suitable one for different using purposes. For each neural network, we explored it thoroughly and grid-searched for best hyper-parameters.

3) Long-term audio recognition: For audio longer than 4 seconds, we use sliding windows to divide them into 4 second pieces and recognizing those pieces to classify the entire audio. Also, we transfer to predicting results into .srt files, which means we can recognize live sound in videos and display them as captions.

4) 10-Fold cross-validation: In order to avoid both training and testing set having related samples, we can take data from 9 of the 10 predefined folds in UrbanSound8K Dataset as training set and take data from the remaining fold as testing set and repeat this process 10 times[1]. We implement these 10-fold cross-validation in our CNN model and average test accuracy is about 82%. The reason why we don't choose to use this kind of cross-validation is that in each fold, UrbanSound8k website has put some audios that from the same long audio. We think that if not shuffling it and directly using it will make our model leak of some information, this will lead an incomplete model. And also, we think that recognition of the fraction of audio is also meaningful. So instead of doing cross-validation on 10 fold, we randomly shuffle data into 8:2 of training and testing and do it many times to get the average result.

VI. Conclusion and Future Work

In this project, we implement different machine learning methods to realize urban sound classification of ten classes. Firstly, we extract features from audio files in urban sound 8K dataset and features include Mel-frequency spectrum, Mel-frequency cepstrum coefficients(MFCC), Chroma and contrast. Then, based on different features, we use four machine learning methods, including Neural Network, Convolutional Neural Network, Recurrent Neural Network and Boosting to classify urban sound to ten classes. By using grid-search method, we find optimized hyper-parameters of these machine learning method to improve the accuracy. Also, urban sound classification is a meaningful project and it can make people life more convenient.

Our project has done quite a lot of work but also there are something more we can do if we have more time. First of all, we want to distinguish continuous sounds in broadband area and also want to recognize the distinction of foreground and background sound. This can be achieved via using a band-pass filter or other kind of filters and then the output waves are what we want to use for classification. There are also larger audio datasets available. Audio set[29] for example has 632 classes and more than 2,000,000 sound clip. It is commonly known that the amount of data is the larger the better.[30]

VII. Contributions

A. Yuhan Zhang

Implement CNN and DNN with 10 fold cross-validation and search for good hyper-parameters.

B. Jingyang Li

Implement RNN-LSTM model to classify urban sound and finish sections related to RNN in report. Cooperate with teammates, design poster, extract features from dataset and finish section Experience, Results, Discussion and Conclusion in report.

C. Hao Zhu

Implement Neural Network, boosting method and also ensemble method. Participated in the design of the poster and the writing of report.

D. Yinuo Wang

Analyze the dataset, obtained and plot images of features, worked on poster and writing the report including abstract, introduction, dataset and features, NN and CNN methods.

References

[1] "Urbansound8k dataset," https://urbansounddataset.weebly.com/urbansound8k.html.

[2] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *22nd ACM International Conference on Multimedia (ACM-MM'14)*, Orlando, FL, USA, Nov. 2014, pp. 1041–1044.

[3] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 4, pp. 357–366, August 1980.

[4] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *2010 18th European Signal Processing Conference*, Aug 2010, pp. 1267–1271.

[5] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real-life recordings," in *18th European Signal Processing Conference*, 07 2014.

[6] J. F. Gemmeke, L. Vuegen, P. Karsmakers, B. Vanrumste, and H. Van hamme, "An exemplar-based nmf approach to audio event detection," in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Oct 2013, pp. 1–4.

[7] G. Parascandolo, H. Huttunen, and T. Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings," *CoRR*, vol. abs/1604.00861, 2016. [Online]. Available: http://arxiv.org/abs/1604.00861

[8] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. W. Wilson, "CNN architectures for large-scale audio classification," *CoRR*, vol. abs/1609.09430, 2016. [Online]. Available: http://arxiv.org/abs/1609.09430

[9] J. Pons and X. Serra, "Randomly weighted cnns for (music) audio classification," *CoRR*, vol. abs/1805.00237, 2018. [Online]. Available: http://arxiv.org/abs/1805.00237

[10] H. Eghbal-zadeh, B. Lehner, M. Dorfer, and G. Widmer, "A hybrid approach with multi-channel i-vectors and convolutional neural networks for acoustic scene classification," *CoRR*, vol. abs/1706.06525, 2017. [Online]. Available: http://arxiv.org/abs/1706.06525

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

[12] W. Dai, C. Dai, S. Qu, J. Li, and S. Das, "Very deep convolutional neural networks for raw waveforms," *CoRR*, vol. abs/1610.00087, 2016. [Online]. Available: http://arxiv.org/abs/1610.00087

[13] Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. 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*, Kathryn Huff and James Bergstra, Eds., 2015, pp. 18 – 24.

[14] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai, "Music type classification by spectral contrast feature," in *Proceedings. IEEE International Conference on Multimedia and Expo*, vol. 1, Aug 2002, pp. 113–116 vol.1.

[15] En.wikipedia.org., "Mel-frequency cepstrum," 2019. [Online]. Available: https://en.wikipedia.org/wiki/Mel-frequency_cepstrum

[16] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937. [Online]. Available: https://doi.org/10.1121/1.1915893

[17] M. Kattel, A. Nepal, A. Shah, and D. Shrestha, "Chroma feature extraction," 01 2019.

[18] T. Khdour, P. Muaidi, A. Al-Ahmad, S. Alqrainy, and M. Alkoffash, "Arabic audio news retrieval system using dependent speaker mode, mel frequency cepstral coefficient and dynamic time warping techniques," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 7, pp. 5082–5097, 06 2014.

[19] En.wikipedia.org., "Chroma feature," 2019.

[20] "spectral features," https://musicinformationretrieval.com/spectral_features.html, accessed: 2019-05-25.

[21] "Multi-layer neural network," http://deeplearning.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/, accessed: 2019-06-02.

[22] "Artificial neural network," https://en.wikipedia.org/wiki/Artificial_neural_network, accessed: 2019-06-02.

[23] Prabhu, "Understanding of convolutional neural network (cnn)ŁŁdeep learningk."

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition."

[25] "Understanding rnn," https://www.jianshu.com/p/30b253561337.

[26] colah, "Understanding lstm networks," http://colah.github.io/posts/2015-08-Understanding-LSTMs/, August 2018.

[27] youzipi, "Deep learning: Lstm," https://blog.csdn.net/pipisorry/article/details/78361778, October 2017.

[28] A. Chandra, H. Chen, and X. Yao, *Trade-Off Between Diversity and Accuracy in Ensemble Generation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 429–464. [Online]. Available: https://doi.org/10.1007/3-540-33019-4_19

[29] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 776–780.

[30] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, pp. 78–87, Oct. 2012. [Online]. Available: http://doi.acm.org/10.1145/2347736.2347755