Urban Sound Source Classification and Comparison

Yihua Yang *yiy011@ucsd.edu*
Ling Hong *lihong@ucsd.edu*
Ke Liu *keliu@ucsd.edu*
Chenwei Dai *chdai@ucsd.edu*

***Abstract*—Automatic urban sound recognition is less investigated compared to image processing but with highly research value in multi-media and urban informatics. In this paper, we explore several deep learning methods trying to compare the performance of each feature and model based on the accuracy of sound classification.The dataset we use is UrbanSound8K which contains 8732 labeled sound excerpts ($\leq 4s$) of urban sounds from 10 classes. By comparing the result, we found that the MFCC feature of sound combined with Resnet18 model works excellently on sound recognition.**

***Index Terms*—sound classification, deep learning, MFCC**

I. Introduction

Sonic analysis of urban environments has aroused more and more interests recently. Unlike visual images, the urban sound is less constructed and full of noises due to complex acoustic scenes in real life. Thus, it is far more challenging to identify and classify these sound sources from different environments.

However, many applications will benefit from the recognition of urban sounds. For example, intelligent wearable devices will sense the acoustic scenes by identifying specific urban sounds and provide extra hearing-aids to the elderly persons [1]. Mobile robots can also take advantages of the audio information to recognize unstructured environments so as to automatically adjust their functionality in the navigation systems [2]. What is more, recognizing urban sound will provide a good way of pre-processing in other audio-based research area such as the speech recognition. Researchers can take advantages of the background sounds classification and then separate the speech signals from different background noises for the advanced research [1]. Therefore, it is of great value for researchers to investigate in this field.

In this project, we have developed several deep learning algorithms such as CNN, DNN, LSTM and other modern neural networks like VGG and Resnet to the sound classification task. The input to our models are short urban sound audios. By extracting two important features, namely the Mel-spectrogram (Mel) and Mel-frequency Cepstral Coefficients (MFCC), we trained the models with the above feature vectors and output the predicted category label that a specific sound audio belonged to. We compared the performance of multiple algorithms with different features and explored why several models achieved better performance.

The paper is organized as follows: Section II describes the related work and background acknowledge about urban sound classification problem and Section III introduces the dataset and features we are using in this project. The methods and models are presented in section IV, followed by the section V presenting the results and discussion details about each model. Finally we draw some perspectives about the future work in section VI.

II. Related work

An early method to handle sound recognition problem was proposed by Sawhney and Maes [3], which modeled the temporal evolution of audio features and employ recurrent neural networks and a k-nearest neighbour criterion to model the mapping between features and categories. However, such temporal feature could hardly obtain an accuracy larger than 70%. Then Eronen et al. [4] discovered that the MFCC feature that based on human recognition of soundscape and how human ears discern different frequencies. Some soundscape research [5] extract MFCC features to describe the local spectral envelop of audio signals and classify with SVM (radial basis function kernel) and random forest. The result of [5] shows accuracy of approximately 70% which is much lower than deep learning method. Other strategy using HMM [6] to translate an audio into events performs poorly in the presence of interfering acoustic noise.

Since noise is unavoidable in practical situation, we must find features and models more robust. Therefore, researchers tend to find powerful machine learning methods like CNN [7], RNN with high dimensional input features such as spectrograms. These improve robustness because of the discriminative capabilities of the back end classifiers.

Our work is distinctive in the following ways. First, we combine different deep learning strategies with two different features: Mel and MFCC, to compare the performance of different features and models. Second, we try to apply Resnet18 and Vgg11, such state-of-art networks used for image classification, to our sound recognition.

III. Dataset and Features

A. The Urban Sound Dataset

In this project, we used the UrbanSound8K dataset [8] containing 8732 labeled sound audios($\leq 4s$). These audios come from 10 urban sound classes including air_conditioner, car_horn, children_playing, dog_bark, drilling, engine_idling, gun_shot, jackhammer, siren, and street_music. The data is split into three sets - the train set containing 6112 audios, validation set with 873 audios and another test set with 1746 audios.

B. Feature Extraction

Usually we transform the audio data from time field into the frequency field, then take advantages of the spectrogram features as the input of machine learning models. Here we extracted two features that are most frequently used in audio processing area, namely Melspectrogram and MFCC by using the Python library librosa [9].

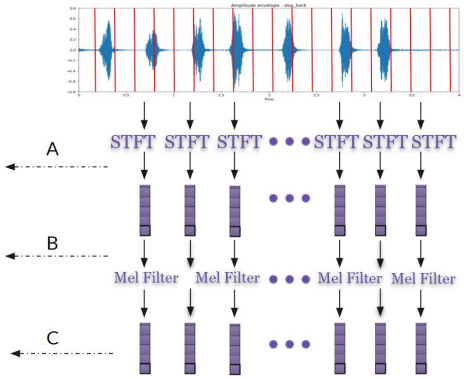

Fig. 1: Melspectrogram extraction process

1) Melspectrogram: Figure 1 outlines the basic process of obtaining the Mel-scaled spectrogram(Melspectrogram). The original audio data was split into multiple shorter frames in the time field. Usually, these frames may overlap with each other by specifying proper size of sliding windows. For each audio clip, we applied Short-Time Fourier Transform (STFT) [10] and got the magnitude spectrogram. The STFT formula is shown as follows.

$$STFT\{x(t)\}(\tau,\omega) = X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)\mathrm{e}^{-j\omega t}dt$$

where $x(t)$ is the input signal to be transformed and $w(\tau)$ is the window function.

Afterwards, the mel-scaled filter-bank, shown in Figure 2, was applied to the spectrogram and transformed the STFT bins into Mel-frequency bins. Obviously, filters in this filter bank are not equally distributed because the filters are denser in the low frequency area than the high frequency area. The mel-scaled bank is designed to mimic the way that human auditory system non-linearly responds to sound in different frequency band [1].

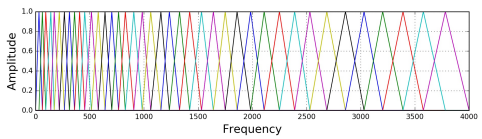

Fig. 2: Mel-scaled filter bank [11]

2) Mel-frequency cepstral coefficients (MFCC): Mel-spectrogram features may be highly correlated, which will degrade the performance of some machine learning models. Thus, the cepstral coefficient is designed based on Melspectrogram to decorrelate the filter bank coefficients. In this case, we applied the Discrete Cosine Transform (DCT) [12] to the logarithm of Melspectrogram features and generated a compressed representation of mel frequencies. The DCT formula is described as below.

$$X_k = \sum_{n=0}^{N-1} x_n \cos\frac{\pi}{N}(n+\frac{1}{2})k$$

where $k = 0, 1, ..., N-1$. Then the final MFCCs are the amplitudes of the resulting spectrum we got from DCT.

IV. Methods

In the first three models, we use 20% data for test and 80% for K-fold cross-validation. We built our models based on the Keras framework [13].

A. K-Fold Cross-Validation

We get pretty good accuracy in the first stage of the experiments, while the models are comparatively complicated. The number of total parameters are 3049994, 4976874, 16794298 in DNN, CNN, LSTM respectively. To avoid overfitting, we use K-fold cross-validation [14] in later experiments. The method is concise. The only parameter K refers to the number of groups that our data is to be split into. In each training, we pick a different group as validation set and the remaining belong to the train set. Using the average accuracy in all K times training, we evaluate the models. We assign K with 8.

B. Models

1) Deep Neural Networks: Deep Neural Network (DNN) shows great advantages in speech recognition and image recognition. It uses perceptrons to imitate the learning process of our brain. The learning process is composed of two stages: feedforward and backpropagation. Through backpropagation, we update the weights and bias of each layer. Each layer learns some features of the input. If there are enough training data, the network could exhibit rather great result. The structure of DNN is shown in Figure 4.

Compared with image recognition, speech recognition needs much less layers, typically 3-4 layers is enough. After trying different layers and other tools to avoid overfitting, our generated DNN model has 3 hidden layers with 512 perceptrons in each layer, followed by RELU activation function and 10% dropout. The output layer has 10 perceptrons corresponding to 10 sound classes followed by softmax activation function.

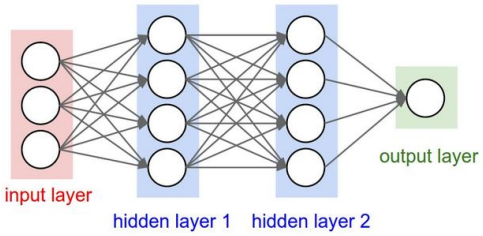

Fig. 4: Structure of DNN

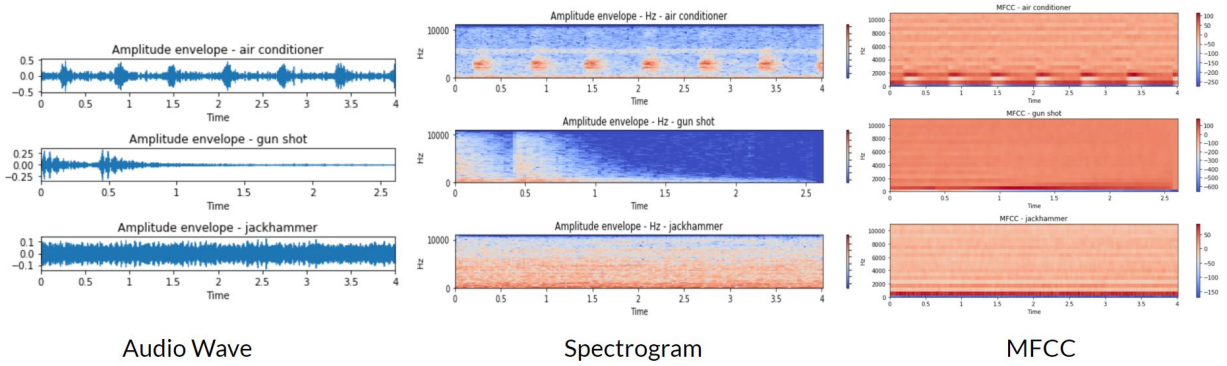

Fig. 3: Examples of the MFCC features

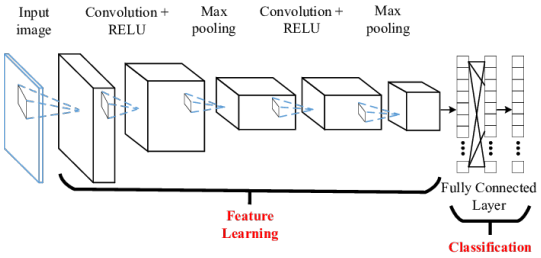

Fig. 5: Structure of CNN [15]

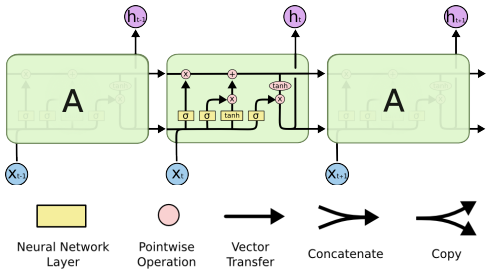

Fig. 6: The repeating module in an LSTM [16]

2) Convolutional Neural Networks: A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers with each layer followed by an activation function. The structure of CNN is shown in Figure 5. Here we use 3 convolutional layers using 3*3 filter banks to generate feature map followed by RELU activation function and then go through max pooling layers with 2*2 size. The generated features will then be flattened to connect with a fully connected layer with 256 neurons, after activated by RELU activation function, fully connect with the output layer which followed by softmax activation function.

3) Long Short Term Memory (LSTM): The problem of DNN and CNN is that they cannot model the variance of the time series characteristic. We investigate some research on speech recognition nowadays. Many use Long Short-Term Memory to preserve the influence of the previous status. And LSTM [17] can to some extent avoid gradient exploding and gradient vanishing problem. Remembering information for long period of time is practically its default behavior [16].

The core idea behind LSTM is the cell state. It runs straight down the entire chain, with only some minor linear interactions. The information flow along it just unchanged easily [16]. As shown in Figure 6. In a single module, the first step is to decide what information to throw away from the cell state. Next, it decides what new information we're going to store in the cell state. This includes the input gate layer (a sigmoid layer) deciding which values we'll update and a tanh layer creating a vector of new candidate values. Then, it is time to update the old cell state into the new one. Finally, decide the output based on our cell state. After running a sigmoid layer, put the cell state through tanh and multiply it by the output of the sigmoid gate.

After experiments, we decide to use 2 fully-connected layers after the LSTM layer and choose *RELU* and *softmax* as activation functions. To avoid overfitting, we use a dropout rate 20%.

4) Other modern neural networks: VGG is a Convolutional Neural Network architecture proposed in the year 2014. The model achieves 92.7% top-5 test accuracy in ImageNet. ImageNet is the one on the largest data-set available, which has 14 million hand-annotated images.

ResNet is an artificial neural network (ANN) of a kind that builds on constructing known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections, or short-cuts to jump over some layers. Typical ResNet models are implemented with double- or triple- layer skips that contain nonlinearities (RELU) and batch normalization in between [19].

V. Results and Discussion

A. DNN results

According to the result shown in Figure 9, we can see that there are some discrepancies between results of training set and validation set which might because we have too many perceptrons to handle relative small-size dataset and thus incur

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig. 7: VGG structure [18]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 8: ResNet structure [20]

overfitting. We can also observe that the accuracy obtained by using MFCC feature is better.

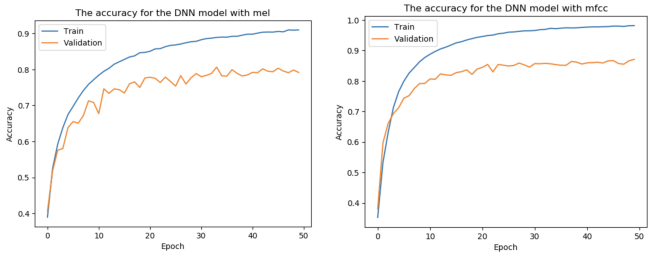

Fig. 9: Accuracy of DNN with Mel and MFCC

B. CNN results

By 8-fold cross-validation, we got the average accuracy in each epoch during the training process as Figure 10 shows. For both the MelSpectrogram and MFCC features, the accuracy in the validation set is higher compared to the training set within several starting epochs. This may result from the fact that we dropped out some data in the training process in case of overfitting. Therefore, at first not all the data was learned so it is possible that we got a higher accuracy in the validation set at the very beginning. As the training process went on, the model got higher accuracy in the training set.

On the other hand, CNN has achieved a slightly better performance using MFCC than using MelSpectrogram according to Table I.

TABLE I: Results of 8-fold cross-validation

model	**MFCC**			**Mel**		
	train	***val***	***test***	***train***	***val***	***test***
DNN	0.982	0.793	0.872	0.910	0.722	0.876
CNN	0.941	0.887	0.896	0.985	0.886	0.878
LSTM	0.998	0.755	0.909	0.998	0.932	0.916

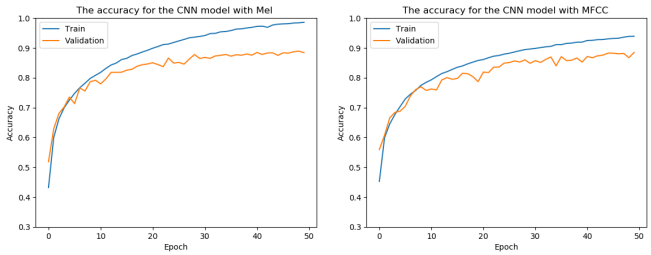

Fig. 10: Accuracy of CNN with Mel and MFCC

C. LSTM model structure and results

Here are the average accuracy of LSTM model using Mel and MFCC respectively. For each epoch, we calculate the average accuracy of all folds. Compared to DNN and CNN models, LSTM model has a good start. In the first epoch, the average training accuracy is higher than 60% no matter which feature we use. In the first 20 to 25 epochs, the accuracy improves and becomes stable.

The problem confusing us is that the validation accuracy is much lower than the training accuracy and test accuracy. We checked the code for training and plotting but have not found the potential bugs. We split data randomly and it seems to work well on DNN and CNN. But it is likely that our splitting is not random enough. Using Mel, the max accuracy on validation set in each training is 0.620, 0.855, 0.841, 0.866, 0.652, 0.745, 0.828 and 0.672. Obviously, our LSTM model does not have a stable performance on all data sets. Unluckily, we have not solved it before the due. Hope we can find a better way to split data in the near future.

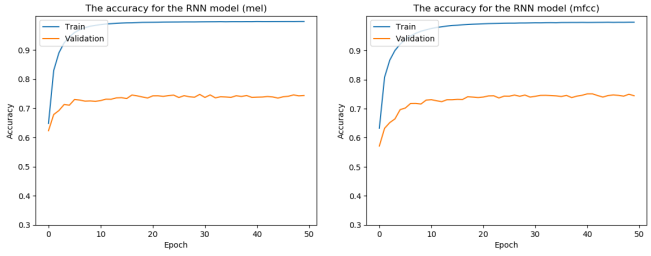

Fig. 11: Accuracy of LSTM with Mel and MFCC

D. VGG11 and ResNet18 results

The two models mentioned below, VGG and ResNet, both have good performance on image classification task. To see whether they are also suitable for sound classification, we

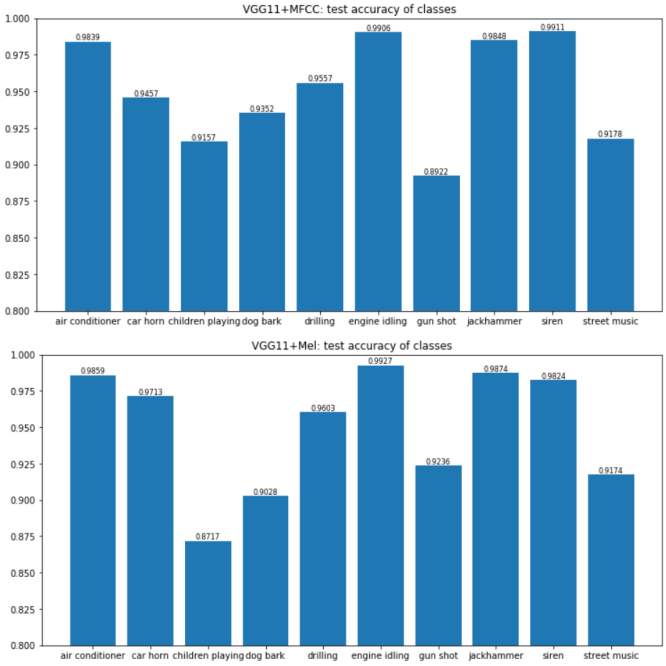

Fig. 12: VGG11+MFCC/Mel test accuracy for classes

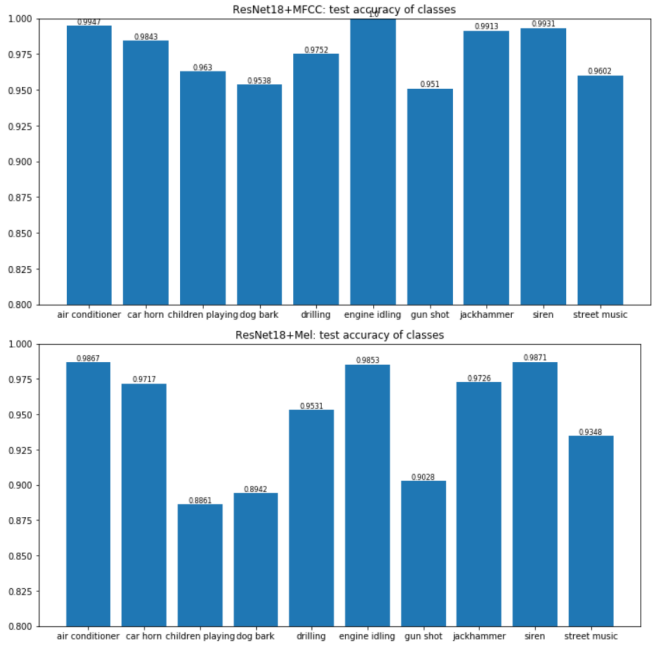

Fig. 13: ResNet18+MFCC/Mel test accuracy for classes

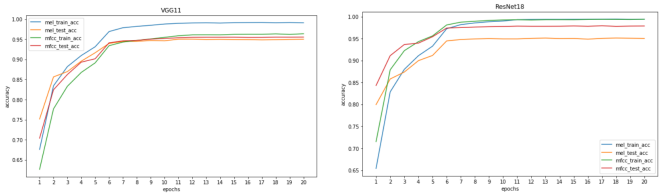

Fig. 14: VGG11 and ResNet18 accuracy

reconstruct these two models (not transfer learning which reserves pre-trained parameters) and explore the performance on our sound feature data.

Considering the dataset is rather small and to mitigate overfitting, we choose the model structure with less layers. The results of VGG11 and ResNet18 are shown in figure 14.

From the results, we can see that both the networks have satisfying performance on sound classification, and the training finishes within 20 epochs with high accuracy.

On average, MFCC features achieve higher accuracy on test dataset, which is over 96%. And the accuracy of both models and both features beat the custom model of DNN, CNN and LSTM. However, the deficiency is also obvious. These two comparatively complex models is of larger size, and the training time for every epoch is much longer.

There is a trade-off between the accuracy and model size. For this project, our main target is the accuracy, so definitely VGG and ResNet beat the custom models. But if we are to train models or implement them on resource constrained platforms with tolerance for accuracy lost, we may have different choices on models.

First, considering the dataset which is pretty small, we tried to build custom models which have fewer layer to mitigate overfitting. Then, for higher accuracy, we decided to try some well-known models proven to have good performance to see whether they can perform well for our sound classification task after some modifications.

VI. Conclusion and Future Work

In this project, we have built several deep learning models such as CNN, DNN, LSTM to explore how they performed in classifying the urban sound sources with the extracted Mel and MFCC features. Based on our experiments, LSTM achieved much better performances in test data than CNN and DNN, which met our expectation. The audio signals are highly time-related and meanwhile LSTM can preserve the influence of the previous status, thus it has higher accuracy than the other two. Furthermore, different features may also have different performance in the classification task. For instance, the Mel-specrogram behaves better in LSTM and DNN while MFCC behaves better in CNN, VGGNet and ResNet. Meanwhile, experiments show that VGGNet11 and ResNet18 have very excellent performance on sound classification tasks.

We explore a lot in feature extractions, training different models and comparing them. Due to limited time, there are still some ideas in the air. First of all, our approach to splitting

TABLE II: VGG11 and ResNet18 accuracies

model	**MFCC**		**Mel**	
	train	***test***	***train***	***test***
VGG11	0.964	0.956	0.991	0.950
ResNet18	0.994	0.979	0.994	0.951

data seems not random enough. We did pay much attention on it before. And we should figure out a better way to solve problems in LSTM model. As we stated before, carefully labeled sound data is scarce. The dataset in our experiments is a perfect one except that the size is pretty small. Data augmentation helps augment the training set with synthetic data that are created by modifying some parameters of the real data. There are already frameworks proposed for sound data augmentation [21]. It can figure out the appropriate augmentations for the appropriate training set. We could also add de-noising stage before we train the data to see if it could obtain better result. Furthermore, our result is only based on UrbanSound8K dataset, we could find other dataset to verify the generalization of our models.

VII. Contributions

During the project, Yihua Yang proposed many ideas about which models we could implement and built models such as Resnet18 and Vgg11. Ling Hong did the feature extraction part and implemented CNN model. Chenwei Dai and Ke Liu built LSTM model and DNN model respectively and then visualized results obtained. During the process, group members discussed a lot about how to improve our model further and wrote report together.

Here is our Github link: https://github.com/Elsa1024/SP19_ECE228_proj

References

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

[2] S. Chu, S. Narayanan, C.-C. J. Kuo, and M. J. Mataric, "Where am i? scene recognition for mobile robots using audio features," in *2006 IEEE International conference on multimedia and expo*. IEEE, 2006, pp. 885–888.

[3] N. Sawhney and P. Maes, "Situational awareness from environmental sounds," 1997.

[4] A. Eronen, J. Tuomi, A. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi, "Audio-based context awareness - acoustic modeling and perceptual evaluation," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, vol. 5, April 2003, pp. V–529.

[5] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: ACM, 2014, pp. 1041–1044. [Online]. Available: http://doi.acm.org/10.1145/2647868.2655045

[6] S. Chaudhuri, M. Harvilla, and B. Raj, "Unsupervised learning of acoustic unit descriptors for audio content representation and classification." 01 2011, pp. 2265–2268.

[7] H. Zhang, I. McLoughlin, and Y. Song, "Robust sound event recognition using convolutional neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 559–563.

[8] 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.

[9] LibROSA, "Librosa api document," https://librosa.github.io/librosa/, 2019 (accessed June 14, 2019).

[10] Wikipedia, "Short-time fourier transform," https://en.wikipedia.org/wiki/Short-time_Fourier_transform, 2019 (accessed June 14, 2019).

[11] H. Fayek, "Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what's in-between," https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html, 2019 (accessed June 14, 2019).

[12] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.

[13] Keras, "Keras: The python deep learning library," https://keras.io/, 2019 (accessed June 14, 2019).

[14] https://machinelearningmastery.com/k-fold-cross-validation/.

[15] P. Kamencay, M. Benčo, T. Miždoš, and R. Radil, "A new method for face recognition using convolutional neural network," 2017.

[16] https://colah.github.io/posts/2015-08-Understanding-LSTMs/fn1.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Microsoft Research*, 2015.

[19] Wikipedia: Residual neural network. [Online]. Available: https://en.wikipedia.org/wiki/Residual_neural_network#cite_note-1

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Visual Geometry Group, Department of Engineering Science, University of Oxford*, 2014.

[21] R. Lu, Z. Duan, and C. Zhang, "Metric learning based data augmentation for environmental sound classification," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2017, pp. 1–5.