COMP47650 Deep Learning Project

Student Information
Name: Yingjie Niu
Student Number: 18209791

Abstract

Sound Recognition is one of the most widely studied area of Deep Learning. It has many branches of downstream tasks and applications. The sound recognition technique can be applied in different scenarios, like sound type classification and speech recognition. This project implements two popular neural network architectures: Recurrent Neural Network(RNN) and Convolutional Neural Network(CNN) on a sound type classification task. The dataset used is from Kaggle, which has 5.67GB training videos containing 41 different type of sounds.

1 Introduction

Deep neural network models have been proved to be effective in many prediction tasks. Sound Recognition is one of the most widely studied area of Deep Learning. It has many branches of downstream tasks and applications. The sound recognition technique can be applied in different scenarios, like sound type classification and speech recognition. Some sounds are distinct and instantly recognizable, like a baby's laugh or the strum of a guitar. Other sounds are not clear and are difficult to pinpoint. Moreover, we often experience a mix of sounds that create an ambience – like the clamoring of construction, a hum of traffic from outside the door, blended with loud laughter from the room, and the ticking of the clock on your wall. Partly because of the vastness of sounds we experience, no reliable automatic general-purpose audio tagging systems exist. To tackle this problem, Freesound (an initiative by MTG-UPF that maintains a collaborative database with over 370,000 Creative Commons Licensed sounds) and Google Research's Machine Perception Team have teamed up to develop a dataset: Freesound General-Purpose Audio Tagging Challenge dataset.

This project implements two popular neural network architectures: Recurrent Neural Network(RNN) and Convolutional Neural Network(CNN) on the dataset. Convolutional Neural Network(CNN) have been theoretically and empirically proved to be efficient in image classification tasks, while Recurrent Neural Network(RNN) is powerful in series classification and regression tasks. Since neural network architectures prefer different senarios, the main challenge of this project is to design the data format and decide the neural network architecture to be used. As a result, this project has shown the benefit of different audio data pre-processing methods. The best performance on the test dataset is : F1 score , accuracy .

2 Related Work

Audio classification is a widely researched task, many people have proposed and applied different model architectures, feature extraction methods to get a better accuracy. McKinney and Breebaart compared four feature sets on audio classification task, including low-level signal properties, the MFCC, and two new feature sets. Hershey et al. examined various CNN models on large-scale audio classification task, including DNN, AlexNEt, VGG, Inception, and ResNet. Karim et al. proposed a augmentation of fully convolutional neural network on long-short term memory (LSTM-FCN) on time series classification tasks. More specific to this project, there are many discussions on the project website on Kaggle and the Leaderboard also shows the SOTA results.

3 Experimental Setup

3.1 Dataset Overview

Since the vastness of sounds we experience, sound classification heavily relied on human efforts and there was no reliable automatic sound annotating methods. In 2018, Freesound (an initiative by MTG-UPF that maintains a collaborative database with over 370,000 Creative Commons Licensed sounds) and Google Research's Machine Perception Team developed this Freesound Audio Tagging dataset.

3.2 Data Preprocessing

This dataset contains 9437 training audios files(5.69GB) and 9400 test audio files(4.7GB)., and there are totally 41 different type of sounds. Since the number of samples of different classes varied, a data augmentation process has been applied before feature extraction. Figure below shows the training dataset distribution. There are two challenges deserve to be consider: 1. The dataset is imbalanced. 2. Audio samples length varies.

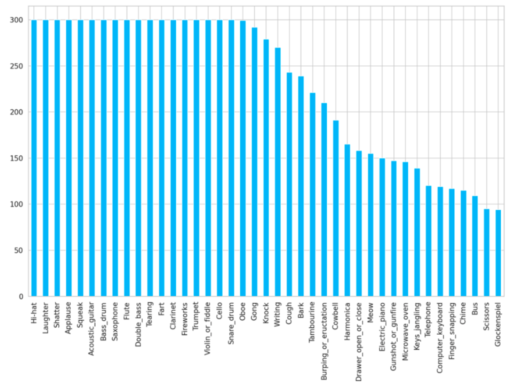

Figure 1: label distributions

3.2.1 Data Augmentation

A Time Shifting data augmentation method is implemented to increase the number of samples of those classes with less than 300 samples. We first read in an audio file in the format of a time series, then we shift the time series to a certain direction by a random proportion of the total length of the original series. After assign the shifted time series the same label as the original one, we got a new sample with a certain label. Figure below is an example of this process.

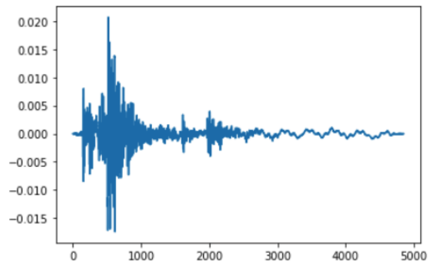

Figure 2: original time series

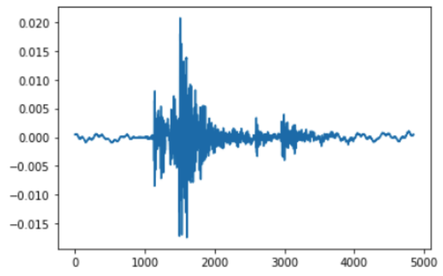

Figure 3: shifted time series

3.2.2 Feature Extraction

Two feature extraction methods on time series data are explored in this project: MFCC and STFT.

Mel Frequency Cepstral Coefficients (MFCCs) In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. MFCC extract a small set of features (24 in our experiment) of an audio signal which are usually used to describe timbre. Figure below shows the MFCC spectral graph of the sample time series above.

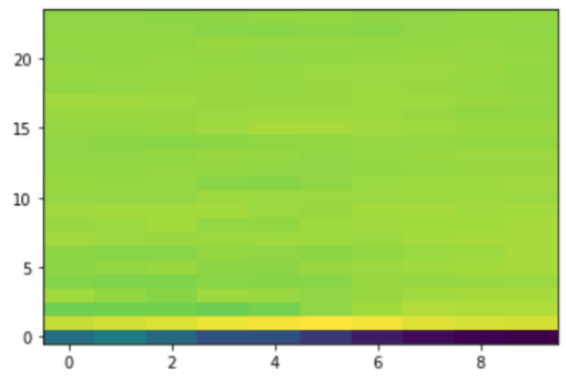

Figure 4: MFCC power spectrum

Short-time Fourier transform (STFT) The classical and popular feature extraction method on time series data is Fourier transform which transform signals depending on time into signals depending on temporal frequency. The reason why we try STFT is because we found that comparing with MFCC spectral graphs, the STFT spectrogram can distinguish different types of sound better. Figure below shows the STFT spectrogram of the sample time series above.

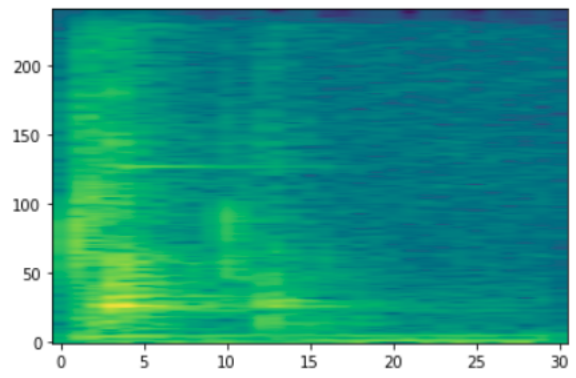

Figure 5: STFT spectrum

3.3 Model Architecture

This project implements two classical neural network architectures: Long-Short Term Memory(LSTM) and Convolutional Neural Network(CNN), and a Transfer learning-based method is also explored.

3.3.1 Long-Short Term Memory

Long short-term memory (LSTM) is a recurrent neural network (RNN) architecture which can process not only single data points, but also entire sequences of data (such as speech or video). It is especially good at processing sequence data which is the reason why we try it at first in this project. Another benefit of using LSTM on this task is, as a RNN architecture, LSTM is able to handle sequence data with different length.

Intuitively, we implement a single layer LSTM on the original dataset(no data augmentation, no feature extraction) as the baseline experiment. Then we try to improve the performance(we use F1 score and accuracy as the performance indicator) by data augmentation, feature extraction, adding more layers in the LSTM model, using Bi-directional LSTM model, etc. Detailed Experimental setup for each experiment is shown in Table 1.

3.3.2 Convolutional Neural Network

Convolutional Neural Network(CNN) is a special neural network architecture which is powerful in image classification tasks. Since we convert the audio to spectrogram, the audio classification task is converted to be a image classification task and we can leverage power of CNN. The difficulty of implementing CNN is to unitize the size of spectrogram. We tried two methods to do this:

Padding We use Height(**H**) and Weight(**W**) to describe the size of one graph where, in our experiment scenario, H is related to the original audio length, and W corresponds to the number of features which is a hyper parameter. After feature extraction, the graphs are with different **H** but the same **W**. So, a straight forward way to unitize graph sizes is to pad the graphs with smaller **H** to $\mathbf{H_{max}}$. Since the audio lengths vary in a large range, $\mathbf{H_{max}}$ might be 10 times larger than a given **H**. To avoid involving too much noise into the training data, we pad the graph by repeating itself. Figure below shows an example of padding one spectrogram.

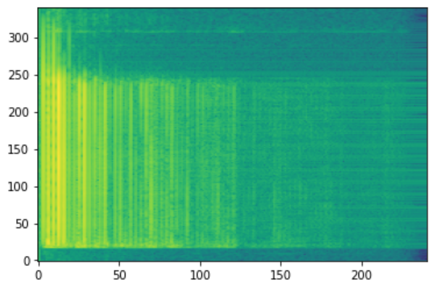

Figure 6: Fourier transform spectrogram

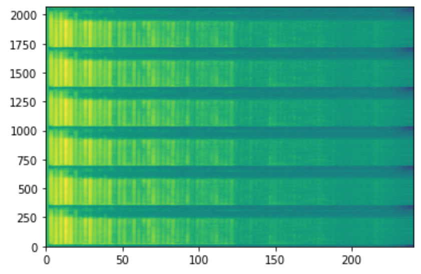

Figure 7: Padded spectrogram

Resize Resizing is a more efficient way of unitizing graph sizes. Different from padding, resizing does not involve noise to the training data. Through resampling pixels or interpolating new pixels, we can reduce or increase the size on an image. In our experiment, we resize all images to the size of $\mathbf{H_{mean}} * \mathbf{W}$. Figure below shows an example of resizing on spectrogram.

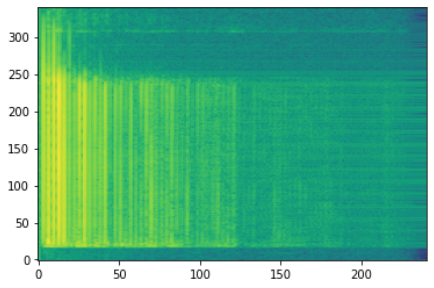

Figure 8: Fourier transform spectrogram

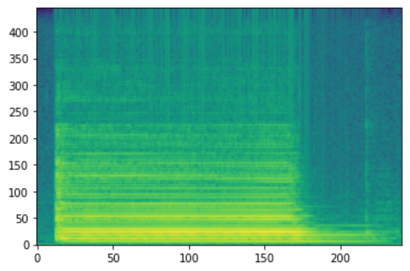

Figure 9: Resized spectrogram

We start with a simple CNN with one convolutional layer and three dense layers. But this network is too naive and it does not converge in the training process. We tried to 1.add more convolutional layers, 2. decrease batch size, 3.decrease learning rate, 4. add batch normalization layers, but none of these works. This is probably because the designed CNN is too shallow to capture the characteristics of each image. So, we tried some transfer learning-based methods. Results of all experiments are shown in the Table 1 in the Results section.

3.3.3 Transfer Learning

Transfer Learning is a machine learning method where, instead of training a model from scratch we use pre-trained models as the starting point, and fine-tune the model for our task. VGG16 is a very deep convolutional neural network for large scale image recognition. It has totally 16 layers(13 Conv layers and 3 Linear layers) and output a 1000 dimension vector. We stack some Linear layers on top of the VGG16 network and convert the output to 41 dimension corresponded to the number of classes in our dataset.

We seperate the whole network into two parts: 1.the pretrained VGG16 network(we call it "VGG16 Net"), 2.the stacked linear layers(we call these linear layers 'the Head'). In terms of how to fine-tune the pre-trained network, we have compared two methods:

Directly fine-tune whole network In this method, we randomly initialize the Head and initialize the VGG16 Net with pretrained parameters, then we fine-tune the whole model parameters.

Train the Head first then fine-tune whole network In this method, we randomly initialize the Head and initialize the VGG16 Net with pretrained parameters. Then we freeze the parameters of VGG16 Net and only allow the optimizer to change parameters of the Head. After several epochs, the Head already learned some knowledge of our dataset. We use these parameters as the initialization of the Head. Then we unfreeze the VGG16 Net parameters and fine-tune the whole model parameter on the basis of good Head initialization. This method is able to increase the converging speed and also increase the model performance.

4 Results & Conclusion

All experiment results are shown in Table 1 below.

Experiment	Data Processing	Model Architecture	Accuracy	F1
Baseline	NO	1 layer LSTM	0.02	0
1	NO	2 layer LSTM	0.06	0.02
2	MFCC	Bi 2-layer LSTM	0.28	0.24
3	MFCC	Bi 3-layer LSTM	0.44	0.40
4	augment+MFCC	Bi 3-layer LSTM	0.48	**0.44**
5	augment+MFCC	Bi 4-layer LSTM	0.43	0.40
6	augment+resize MFCC	3-layer CNN	0.25	0.15
7	augment+resize STFT	3-layer CNN	0.23	0.13
8	augment+padding STFT	3-layer CNN	0.11	0.02
9	augment+padding STFT	4-layer CNN	0.15	0.07
10	augment+padding STFT	VGG16+Head	0.45	0.41
11	augment+resize STFT	VGG16+Head	0.44	0.33
12	augment+resize STFT	VGG16+ Good Init Head	**0.50**	0.34

Table 1: Model performance comparison

Experiment Result Analysis:

1. Through comparing experiment 3 and 4, we can see that the Data Augmentation method is able to improve the model performance.
2. Through comparing the experiment 6 and 7, under the same model architecture and setting up, the MFCC preprocessing method is slightly better than STFT on this task.
3. The results of experiment 7,8,10 and 11 show that resizing works better on shallow CNN but padding favors deep neural networks. This is reasonable because the image size after padding is much larger than it after resizing which means padded images have much more characteristics that need deep neural network to catch.
4. Pre-trained models dosen't work surprisingly well as expected. This might have few reasons: 1) not enough training data, 2) not enough training epochs during fine-tuning

Since we use F1 score as the key performance indicator in this project, experiment 4 is the one with best performance.

5 Future Work

This is a very interesting project and deserve more experiments on it. Unfortunately, we don't have enough time to try all possible combinations of proposed data processing methods, model architectures. and fine-tuning methods. We are not able to fine-tune the pretrained model with more epoches as well since it is very time consuming. The SOTA performance of this task is 0.95 F1 score, so we still have a long way to go!

References

[1] Karim, F. et al. (2017) LSTM Fully Convolutional Networks for Time Series Classification. *IEEE Access* (6):1662 - 1669.

[2] Hershey, S., et al. (2017) CNN architectures for large-scale audio classification. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).*

[3] McKinney, M.F. & Breebaart, J. (2003) Features for Audio and Music Classification.

[4] kaggle.com. (n.d.). Freesound General-Purpose Audio Tagging Challenge. [online] Available at: https://www.kaggle.com/competitions/freesound-audio-tagging [Accessed 21 Apr. 2022].