Visual Sound Assistant

Submitted by:
Ke Xia(POC)
Xin Yang
Song Yang
Tong Wu

Team Project Number:
S18-39

Advisor:
Prof. Yingying Chen

April 28, 2018

Electrical and Computer Engineering Department
Rutgers University, Piscataway, NJ 08854

Abstract

As the smartphone becomes popular, we want to apply mobile technology on the hearing problems support area and reach a goal to help them identify what's happening around by analyzing sounds and coming up with information in messages. These text information can remind people to take corresponding actions if necessary. Considering professional teachers and time are needed for the sign language and assistant devices are expensive, we'd like to develop an App based on physical devices with a microphone.

Our work focuses on the deep learning algorithm. Steps include feature extracting and classification. We use the neural network to train the classification model. The final result is represented as an Android application with functions of real-time sampling, test and provide a friendly interface. The accuracy of the classification model is around 96%.

Keywords: sounds classification, neural network, MFCC, mobile App development, Android, web server

Table of Contents

1. Introduction

1.1. Background

Hearing loss is common in the human world. The World Health Organization estimates that 15 percent of adults, or roughly 766 million people, suffer from hearing loss. That number is rising as the population expands and the proportion of older adults becomes larger. Some people were born with hearing issues. Some people suffer hearing loss when they become older. They are usually difficult to take interactions with the environment including detecting environment and taking actions according to environmental changes. For children, it can affect the ability to learn spoken language. For older people, it makes them being lonely and difficult to keep themselves away from danger.

The most common ways that help people with the hearing disorder are using hearing-aid and cochlear implants. However, taking a hearing-aid is uncomfortable and very fewer people keep hearing-aids during their sleepings. Cochlear implants are expensive and not mature enough. Therefore, we want to use deep learning algorithms to design a technology which can help them improve interactions with the environment.

1.2. Related Work

Due to the huge market, many scientists and developers are trying to reinvent the hearing aid by artificial intelligence technology.

Albert Bregman, a psychologist at McGill University in Montreal, Canada, who proposed in 1990 that the human auditory system organizes sounds into distinct streams. A stream essentially corresponds to sound emitted from a single source, such as a nearby friend. Each Soundstream is unique in its pitch, volume, and the direction from which it comes.

In the early 2000s, traditional approaches such as Hidden Markov Models combined with feedforward artificial neural networks are commonly used in speech recognition.

1.3. Feature Extraction

Feature extraction is detecting the sound from the environment and extracting useful features from it. That needs knowledge of Signal Processing. There is a number of interesting features that could potentially exhibit different behavior for speech, music, and noise and thus may help the system classify the sound signal.One of the most typical features used for information extraction in the audio analysis is the Mel

Frequency Cepstral Coefficients (MFCCs), that have already been used for sound environment classification in hearing aids.

1.3.1. 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. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum").

1.3.2. Main process of MFCCs

- Pre_emphasize: Filtering low-frequency noise.
- Frame: Separating sound signal into 20 ~ 50ms frames.
- Hamming window: Isolating each frame and shifting it in case of overlap.
- FFT (Fast Fourier Transformation): Getting the power spectrum of the signal.
- Mel banks: Through 40 trigonometric filters, getting the Mel power array.
- DCT (Discrete Cosine Transformation): Getting the MFCC.

1.4. Classification Algorithm

1.4.1. Logistic Regression

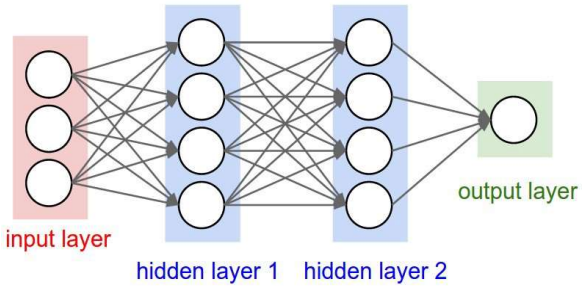
Statistical classification is to identify which given category the target belongs to. It is considered as a supervised learning where a training set of correctly identified observations is available. In machine learning, we use the logistic regression model to do the classification. Consider the predict function as the likelihood of y = 1 or y = 0:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Then we build a Cost function and use methods to minimize the cost function such as Gradient Descent. However, the accuracy of logistic regression is not high enough and it needs to be transformed when the features are not linearly separable.

1.4.2. Neural Network

Neural Network could be considered as a multi-layer of logistic regressions. Each neural model is a logistic unit. The network is consistent with layers. Different layers may perform different kinds of transformations on their inputs. The first layer is input and the last layer is the result of classification. The more layers in a network, the more complicated it is.



Neural Network has the high accuracy of classification and can be used in many complex situations with non-linear relationships.

2. Methods and Approach

2.1. Methods

2.1.1 System Design

The server is designed with a CS framework. CS framework can be used for many experiments. CS is a multi-threaded, event-driven, object-oriented and distributed framework. CS is supported on MS-Windows, Linux, and the real-time OS Pharlap. The whole system is combined with the client (Android application) and the server (build on AWS).

2.1.2 Neural Network on Server

The neural network is built on the Google TensorFlow framework with GPU support. This framework is based on Python 2.7 on Ubuntu 17.10. To accomplish this part, several third-party libraries like numpy, matplotlib, and librosa are imported for the convenience of computation and adjustment.

The server is deployed the TensorFlow Serving for the reuse of pre-trained neural network model. The Android client runs TensorFlow Lite to generate prediction data using real-time processed features as the input. The initial graph is saved on standard TensorFlow, after implementing on Android and collecting enough new cases, TensorFlow Serving will re-train the neural network and synchronize the latest model between clients.

2.2. Use of Standards

We use Ubuntu OS on Linux Server since it has many user-friendly features and has a better server performance. We use Linux based system to do basic feature exacting, neural network design, and learning algorithm test. A Linux server has the lowest cost and convenient enough to develop.

TCP/IP and HTTP protocols are used in server to build a connection to client or transfer files. We should also appreciate LibROSA which is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. The LibROSA with its MFCC functions helps us analyze the sound features and extract them. The server provides REpresentational State Transfer (REST) style API. A JSON string that contains the classification result that being requested would return.

Our application is developed on Android Studio 3.0 with Java Development Kit 25.

2.3. Experiment

2.3.1 MFCC

The MFCC feature extraction should be implemented in Java for Android compatibility. But first we need to build the model in Python, given the suggestion of using the third-party librosa library by our advisor, we successfully implemented the MFCC from librosa. In this case, the Java version should be the same as the one in librosa to ensure the accuracy of our model. With reference to the source code of librosa, we developed our own version of MFCC in Java, which can take an audio buffer as input and generate a sequence of numbers in the given length as output to represent the audio input.

2.3.2 Neural Network

For the classification model, we chose the fully connected neural network for its performance and concision. We have tried lots of configurations of the structure of the neural network, the final model we made consists of one input layer, three hidden layers, and one fully connected output layer. The neurons of each hidden layer are around 300. Considering the performance of Android devices, we choose to generate a 40-digit-long MFCC sequence as the input of the neural network, which decides the input layer has 40 neurons. The optimizer we select is the gradient descent optimizer, using a learning rate 0.01 for 5000 batches in total.

The classification categories are:

- Air Conditioner Running
- Car Horn
- Kids Playing Around
- Dog Barking
- Drilling Work Nearby
- Car Engine Running
- Gunshot Alert
- Jackhammer Nearby
- Siren Ringing
- Street Music Playing

The dataset we utilize is the UrbanSound8k, which contains 10 categories, so the output layer contains 10 neurons. By running our models on the UrbanSound8k dataset, we can have the best result to be around 96% with a 0.852 F-score.

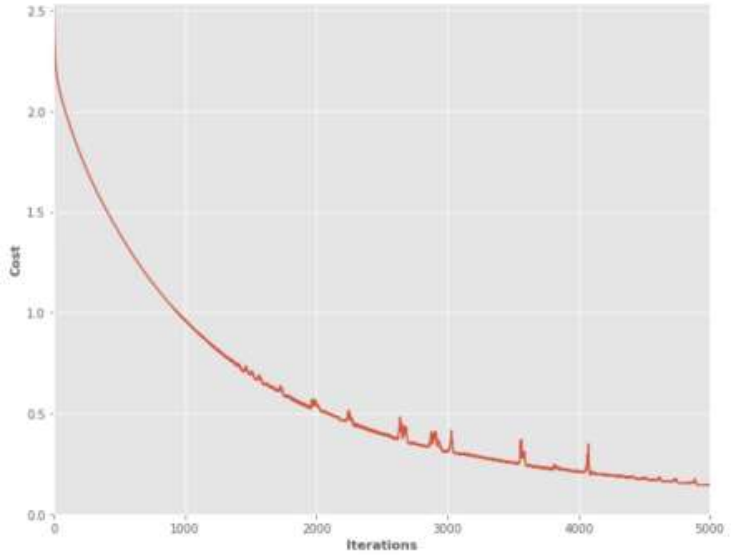


Figure 1-1 F-Score of Neural Network

Thanks to the properties of TensorFlow, we stored the pre-trained model as a .pb file and uploaded it to both our server and Android clients. Based on the pre-trained model, we can achieve the prediction result by doing simple computation instead of the time-consuming training.

2.3.3 Android Client

The Android client is mainly to implement an audio recorder that can keep recording audio samples and analyze the feature and classify. For the recorder part, our implementation is based on the example given by Google. We designed an Android service to ensure the background running, and keep launching a recording and a recognizing thread to make sure the samples taken are contiguous. A concurrent lock is made to guarantee the atomicity of audio buffer processing.
The MFCC module implemented in Android will extract the features of the audio sample and do exactly what the MFCC in librosa does. The MFCC processed features are saved in a vector having 40 floating numbers, which will later be used as the input of the neural network.

The main activity can show the detected events along with the time and confidence. Also, all detected events will be shown as system notification to ensure the user can see it when this program is running in the background. To prevent low priority events disturbing the user, we gave each event a priority value ranging from 1 to 3. The higher value indicates the more urgent situation. The user can switch between the three priorities to filter unnecessary notifications.

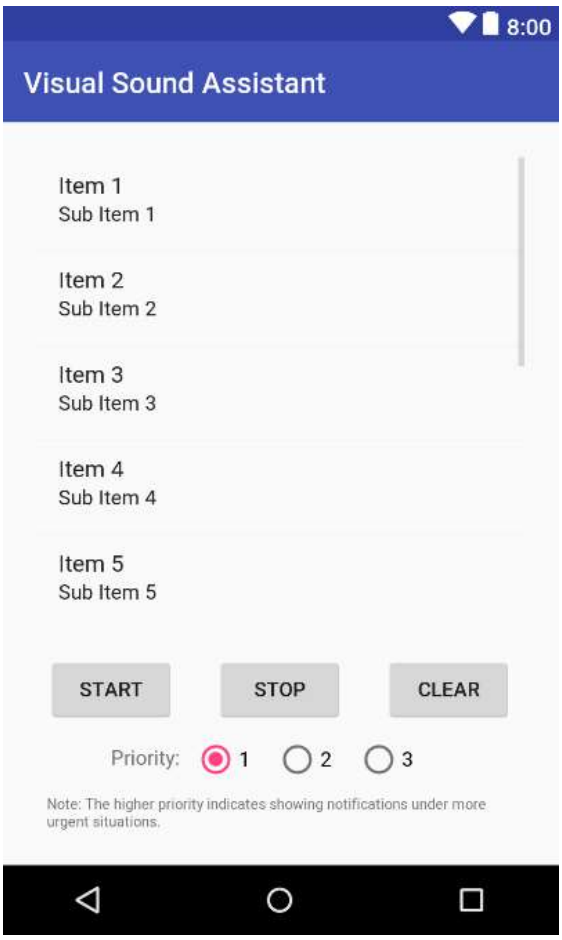


Figure 2-2 Android Application Interface

To help make the more accurate result, our application will upload the detected event's MFCC feature to our server for further model training. To keep the client up-to-date, the application will check if the server released a new model version when it starts, and download the model to upgrade the neural network model.

2.3.4 Server

The server setup for the Android application built on Amazon Web Service with the Ubuntu 64 bit Deep Learning AMI Version 6.0. It installed deep learning environments like tensorflow. tensorflow serving, and other common data analysis library like numpy, scipy in order to support some online features for the application. The REST-style web service is developed with python framework Flask which is a microframework for Python based on Werkzeug, Jinja2, and good intentions.

The workflow of the server is described in the structure graph below:

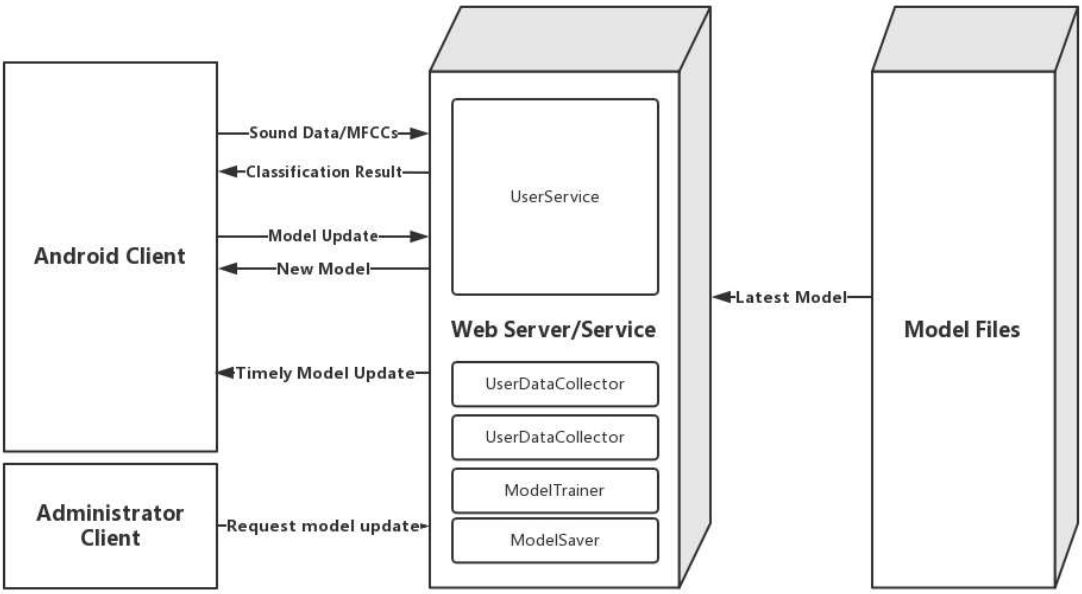


Figure 2-3 workflow of server

What the server does is not only providing an interface for Android Client calling for classification result but also running a learning algorithm that use the data both from data sets and users to train a better model. The model updating occurs only when an Administrator requests or when a user uses the application long enough (every 2 months in our prototype).

3. Cost and Sustainability Analysis

3.1 Economical Cost

As mentioned above, our main goal is to develop an Android application which helps people with hearing issues to collect sound information in common life environment. The training model of our product prototype is finished on our own PC and the cost is zero. We also use our own Android devices (mostly on our phones) to develop and test the application, and the fee to maintain our product is zero except the labor cost now.

The web server is needed to maintain our application and train our classify model with valid cases. The free version of Amazon Web Service (AWS) is enough to support us finish the development procedure. AWS comes with latest binaries of deep learning frameworks pre-installed in many separate virtual environments, including TensorFlow. The platform of the server is Ubuntu and the image size is 75GB. The web service will generate cost if we plan to maintain the training part more than a year.

An Android device is necessary to run our application. Our application supports the mainstream types of Android phones now, but the spent of Android device should not be considered into the device cost because Phone is widely used nowadays and it’s unnecessary to purchase the other phone to use our application.

We have tried to reduce the energy cost of our project by transplanting the model training process to the web server. We designed to train our sound recognition model to improve the sound accuracy. This part costs a lot of energy compared with other functions and users usually does not need the new model for the first time. The reduced energy cost is counting to 20-25% approximate.

3.2 Social Impact

Machine learning and mobile technology are now popular both in research and practical use. It brings new changes to everyone’s life. Our motivation is to apply new method into hearing aids area and offers a new way to improve people's lives.

Due to the loss of hearing, many people in our real life suffers from the blackness of sound information. Imagine a situation: a person was in his house alone and suddenly fire alarm ringing, he would miss this alarm because of hearing issues. The loss of sound information can be a fatal problem. The application we

developed solves this problem with text messages, and impact people lives positively, especially to the people who with hearing issues.

Nowadays, Hearing amplifiers are one of the widely used hearing aid devices. However, professional hearing aids can be up to 300 dollars. That is a huge expense to poor people and families. Our Application is free now or costs 1 dollar in the future that everyone can afford it.

Taking into account user privacy issues, we will protect the privacy of every user. Even though we use valid cases to train our network collected by user's devices, the data the network received is feature value, we cannot get to know the information behind the feature value, such as the location. The recognition results will be sent to users directly so developers and administrators have no authority to access user's data.

4. Conclusions

4.1 Results

For our capstone design, an environmental sound recognition system for Android mobile devices to help people with hearing issues was developed. The application uses an online sound extracting service which helps reduce power consumption. Besides, the MFCC feature extraction algorithms were applied to accurately identify sounds. We made an application successfully reaches our goal, it can identify the sound of up to 10 classifications categories.

We can have the best result to be around 96%. We tested the application and the result shows that it works pretty well in the noise environment, handgun shooting, siren, dog barking with the normal sound volume in real life. We tested the application with 40db noise environment, and it works fine as long as sounds like siren and shooting is as loud as it should be.

4.3 Comparing with existing solutions

There is some real-time environmental sound recognition system already finished, some of the works great in power consumption when processing recognizing, some use many kinds of sound extraction algorithms or strategies. Other than MFCC another widely used feature is spectral centroid, which measures the brightness of a sound. Brightness is especially relevant for continuous instrumental sounds and is calculated as the amplitude-weighted average of all partials for the whole duration of the event. The spectral centroid is also used in many research demonstrations. To learn more deep in the algorithm is what out project should work on continuously.

We believe that online service is necessary not only for feature exacting and model training but the server can also offer many up-to-date news and alarms since our original idea is to help people with hearing issues. More information can fix the decision tree of notifications.

4.3 Future Suggestions

Looking into the future we propose to filter out sounds. Since our project has a lack of denoising algorithms in sound recognizing, we would like to have more layers when training, try some other sound extracting algorithms. In order to make the application a product, a complete backend with strong information source is also required.

Our application may get inaccurate classification result in the noisy environment. More sound process operations needed to add, such as noise reduction.

We also plan to transplant our application to other operating systems, such as Linux and iOS, so we can gain more users.

5. Acknowledgments

We would like to thank Prof. Yingying Chen, a respectable scholar who provided us valuable guidance, great encouragement and support.

Also, we would like to thank Prof. Hana Godrich for her kindness and passion. Without her instruction throughout the whole program, we cannot finish this project and achieve our goal .

We shall extend our gratefulness to the Department of Electronic & Computer Engineering and all the faculties for their generous support.

6. REFERENCES

[1] Piczak, Karol J. "Environmental sound classification with convolutional neural networks." Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on. IEEE, 2015.

[2] UrbanSound8k: https://serv.cusp.nyu.edu/projects/urbansounddataset/urbansound8k.html

[3] Librosa: https://github.com/librosa/librosa

[4] Google Recorder: https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/lite/java/demo