Project ID: MA3

Project

Zamplify

Hearing the future

This page is intentionally left blank.

```
var page = {};
```

Final Year Project Final Report

Project Zamplify - A Sound Recognition System for Context Awareness

By

Cheng Kok Hang, Sam 20193070

Cheung Tsz Him, James 20199455

Kwok Hin Kwan, Billy 20188664

Shin Wai Ching, Martin 20191104

Advised by

Prof. Brian Mak Kan Wing

Submitted in partial fulfillment of the requirements for COMP 4981/4988 in the Department of Computer Science, The Hong Kong University of Science and Technology in the academic year 2017 - 2018

Date of submission: April 19, 2018

Abstract

Natural sound can be an important indicator for context recognition. This report presents the design, implementation, testing and evaluation of Zamplify, a real-time natural sound recognition system. The Zamplify Android app and its complementary IoT device continuously recognize context from sound in the surroundings, and provide a customizable trigger-action mechanism that performs actions when certain context is detected. The core technologies used in Zamplify include a convolutional neural network for extracting sound features and a recurrent neural network (with long short-term memory cells) for modelling sequential information of audio. We evaluated the system in a set of 9 daily-life ambient sounds and discussed its performance. The result shows that it is indeed feasible and practical to use sound for context awareness, and demonstrates the potential of a sound-based context recognition system.

Table of Contents

1 Introduction

1.1 Overview

1.1.1 Background

In the recent years, the advancement in machine learning has brought us amazing technologies like voice assistant, photo auto-tagging and face recognition. While computers now know how to read our faces, photos, text and other relatively tangible form of data, they are still unable to infer contextual information about our surroundings. Such context awareness capability has the potential to significantly improve machine intelligence and user experience. In fact, it has already shaped the idea of Context Computing[1]. Since the term was coined, computer scientists and companies have been dreaming about the applications of context awareness in Human-computer Interaction[2], Artificial Intelligence and the Internet of Things[3].

Category	Examples
Physical	Physical event, surrounding object, scene, geolocation
Personal	Routine, activity, emotion, health
Social	Setting, atmosphere

Table 1. Examples of context

1.1.2 Problem

Currently, to a limited extent, a basic level of Context Computing is used by software giants, like Google and Facebook, to improve their product experience and offer contextual recommendations. For example, if a user's GPS location suddenly changes from US-based to Tokyo after disappearing for a few hours, Google may suggest him the points of interest, and Facebook may invite him to post a "check-in" in News Feed.

However, the contextual inference of these services is usually static and based on single-purpose signal-based data such as GPS coordinates, Wifi connection, device proximity, and gyroscope output. Not only do they lack the ability to infer abstract and dynamic context, but also fail to capture fine-grained environmental data such as ambience, social interaction and spontaneous events. For instance, GPS-based systems stop working inside buildings; Wifi-based systems fail when network configuration changes; proximity-based systems require complementary detectors or inter-device communication; gyroscope-based systems only infer gesture and individual movement.

[1] Context computing, also called context-aware computing, is the use of software and hardware to automatically collect and analyze data about a device's surroundings in order to present relevant, actionable information to the end user.
[2] Human-Computer Interaction researches the design and use of computer technology, focused on the interfaces between people and computers.
[3] The Internet of Things is the network of physical devices, vehicles, home appliances and other items embedded with electronics, software, sensors, actuators, and connectivity which enables these objects to connect and exchange data.

1.1.3 Proposed solution

To tackle the problem of existing context recognition methods, we need a more adaptive and multi-purpose type of data for inferring more abstraxt, complex and dynamic context. Image, speech, sound and video are the most promising and widely available data types that satisfy the above requirement. However, image and video data are affected by line of sight - users need to point their devices towards a specific direction to recognize the context. And since speech recognition includes only verbal content, it does not carry enough information about the environment.

a) Image and video b) Sound and audio

Figure 1. Detection range and coverage

The remaining candidate is natural sound, and this made us start to think about the possibilities of using sound to infer context. After thinking about this carefully, we found that the power of sound is often overlooked. In fact, an enormous amount of context-related information can be inferred from sound - whether a user is eating in or just passing by a restaurant, whether a shopper has entered the outdoor area of a shopping mall, whether a pedestrian is crossing the road despite a red pedestrian light being shown, etc. The “noise” around us is indeed a good indicator of context.

1.1.4 Plan

Despite the potential of natural sound, there has not been a sound-based context recognition service in the market. To explore the possibilities of sound-based context awareness system, we decided to start Project Zamplify[4] to develop a sound recognition API that empowers applications with context-aware capabilities and a more engaging user experience. To better demonstrate the use cases, we also developed a device and a mobile app around the API. Both serve as an input channel for recording natural sound, while the app also lets users define tasks to be performed when a certain type of sound is detected. Examples include triggering silent mode when entering indoor area, taking photos when the users click their fingers, waking up sleeping parents when their baby is crying, notifying house owners away from home when a window breaking sound is detected, and so on. The API, together with the peripheral device and mobile app, has marked the first successful attempt to build a decent sound context-awareness system for mobile and home devices.

[4] Zamplify is a term we thought of to represent the process of amplifying the power of natural sound and turning it into useful information.

1.2 Objectives

1.2.1 Goals

Our group achieved the following goals by completing this Final Year Project.

1. Design a cutting-edge machine learning model for context recognition using sound, either based on existing research and self-invented ideas
2. Transform trained machine learning model from a research demonstration to a production-ready system that is scalable and computationally maintainable
3. Explore and evaluate the feasibility and practicality of inferring context from non-speech audio content
4. Apply machine learning to software and/or hardware that is/are used in practical daily life
5. Gain solid experience in the agile development cycle and project management while using software that involves machine learning

1.2.2 Outcomes

Based on the above goals, we produced the following deliverables.

1. A trained and tested audio-in-labels-out machine learning engine, which is able to classify 7 to 9 types of sound with over 80% accuracy
2. A real-time Sound Recognition Model that processes a given sound recording and returns a set of corresponding object and scene labels with corresponding confidence levels
3. An API that manages access to the previously-mentioned machine learning engine from various applications
4. A mobile app that samples sound around the device regularly and allows users to define which actions will be triggered by specific types of sounds
5. A device that actively listens to sound and communicates with the Sound Recognition API over the Internet
6. All relevant documentation and reports required

1.3 Literature Review

To better understand the previous work on sound-based context recognition, we conducted literature review about two important concepts - sound recognition and its application, and trigger-action mechanism.

1.3.1 Prior works in building sound recognition model

Due to the availability of large datasets and increasing computing power, extensive research on audio recognition was conducted in recent years, especially in the field of speech recognition. However, natural sound recognition has not been given the same amount of attention. Prior work in natural sound recognition predominantly uses handcrafted filters with general classifier like SVM[5], GMM[6], etc [4]. Even with conventional deep learning models, like deep CNN[7], the performance is still largely restricted by limited labeled data.

Recently, two research papers introduced breakthroughs in applying deep learning to sound recognition with a satisfying prediction result. In the first paper, SoundNet was proposed for learning sound embedding from unlabeled videos. The model extracts visual information from unlabeled videos and applies transfer learning from the old visual model to a new audio model [12]. To train SoundNet, audio and visual data from unlabeled videos are fed into the model separately. The visual elements pass through two state-of-the-art CNN models (a popular ImageNet[8] CNN and a popular Places[9] CNN) and generate two distributions of labels, namely object and scene. The detached audios pass through a series of convolution and pooling layers, in which the parameters will then be trained to minimize the loss between the last layer and the two distributions from visuals.

A similar task was also discussed in the paper, “Look, Listen and Learn”, which presents a binary classification approach to examine the correspondence of an unlabeled video and an audio track [10]. A one-second audio and a video frame is passed to two CNNs separately to extract audio and visual features. A fusion network, which is composed of a ReLU and a softmax layer, combines the features and makes the final decision on whether the visual and audio data are related. Such an approach gives better results on sound classification tasks than SoundNet.

These two papers have provided a basis for us to apply deep audio understanding to real-life data, which was not thoroughly discussed in previous literature.

---

[5] Support vector machines (SVM) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.
[6] Gaussian mixture model is a probabilistic model for representing the presence of subpopulations within an overall population, without requiring that an observed data set should identify the sub-population to which an individual observation belongs.
[7] Convolutional neural network (CNN) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery.
[8] ImageNet is a large visual database designed for use in visual object recognition software research with over 14 million labeled images.
[9] Places is a scene-centric database by MIT, with 205 scene categories and 2.5 million single-labeled images.

Despite the outstanding performance achieved by SoundNet, it fails to capture sequential information (time-dimension pattern) in the data, reducing its robustness. Therefore, we researched on the possibility of modeling sequential dependency between timestamps of an audio clip. Yet, we found that this could be challenging due to the high dimension and complicated features embedded in sound. Lucky, there is evidence that RNN[10], especially with LSTM[11] [17], may be capable of modelling audio sequence and has shown its edge in speech recognition and natural language understanding. Besides, it is popular to use LSTM together with CNN to capture features and model sequential information at the same time, for example, image captioning [18]. Therefore, we believe it is a good experiment to extend such idea to natural sound recognition.

1.3.2 Prior works in the application of sound recognition on mobile devices

Interestingly, some preliminary research on the application of sound recognition has been conducted by scholars around the world. We have specifically selected and analyzed two papers that include successful end-to-end implementations of mobile sound recognition systems.

The first paper proposed UbiEar, the closest implementation of sound recognition to our project. It is a smartphone-based acoustic event-sensing and notification system for hard-of-hearing young students [11]. The system attains a high degree of accuracy in the sound recognition task and provided inspiration for our project in the aspect of practical use. UbiEar proposed a lightweight CNN model in smartphones for scene recognition. The research also optimizes the responsiveness, energy efficiency and user-friendliness of UbiEar. The use of a CNN model in smartphones promotes more responsive recognition results than a client-server model and gives better performance than a shallow learning model, like decision tree, which was previously used in similar applications [5]. The paper also suggests a lightweight version of a traditional CNN and an adaptive duty-cycling sensing algorithm that reduces recording time to reduce energy consumption. While UbiEar detects nine useful events for hard-to-hear people, for example, knocking on the door, and provides simple notifications, like smartphone vibration, Zamplify focuses on more general sounds and extends the notification to more channels.

Another project called AmbientSense also implemented a similar end-to-end system that recognizes the context in which the mobile phone is located [7]. In the application part, this paper suggested two techniques to reduce the runtime and CPU usage of a real-time sound recognition system - offloading computation to a server and optimizing background threads. It also measured the performance of AmbientSense under different Wifi and cellular conditions, as well as mobile device environments. With these statistics, we are able to gain insight into the speed and efficiency of a practical mobile

[10] A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence.

[11] Long short-term memory (LSTM) is a building unit for layers of a recurrent neural network (RNN). LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell is responsible for "remembering" values over arbitrary time intervals; hence the word "memory" in LSTM.

sound recognition application. However, in the machine learning part, it uses Fast Fourier Transform[12] to pre-process the data and SVM with 23 labels to predict context with an average accuracy. In this project, we replaced this model with our own one to cover more daily use cases.

1.3.3 Prior works in trigger-action mechanisms

The trigger-action (if-then) Model has been broadly studied over the last decade and is widely used for controlling devices in smart homes [1,8,9,13]. One of the most popular realizations of such paradigm is IFTTT[13], which provides over 18 million triggers and actions on its platform [13]. However, only a few triggers on IFTTT takes sound as an input, and they all can only recognize voice commands. Therefore, our application aims at using natural sound as triggers.

1.3.4 Prior context-awareness API Services

Context awareness is a emerging trend in the community. There is an increasing number of APIs aiming at detecting the context from mobile devices. A well-developed one is the Awareness API by Google [19]. The Awareness API combines seven location and context signals, including local time, position, user activity, place, nearby beacons, headphone plug state and local weather conditions. It is not fine-grained enough as most of the signals depend on locations, which is not a good indicator of the current activity of users. Therefore, we see an opportunity to complement Zamplify with Awareness API to provide a new level of context awareness experience.

[12] Fast Fourier Transform (FFT) is an algorithm that computes data transformations from time domain to frequency domain by factoring the DFT matrix into a product of sparse factors in $O(n \log n)$.
[13] IFTTT (If This Then That) is a free web-based service that people use to create chains of simple conditional statements, called applets.

2 Methodology

2.1 Overall Architecture

Zamplify is an integrated system consisting of five major components. The core of our project is a API, powered by a sound recognition model, that provides context awareness capability to connected applications. Other parts, including the Android app and IoT device, are peripherals built around the core API. They serve as the channels of audio input and demonstrate the use cases and potential of our API.

Component	Description
Server-side Sound Recognition Model	A machine learning model hosted on the cloud, taking audio file as input and outputting a predicted probability distribution of object and scene labels
Client-side Sound Recognition Model (Experimental)	An experimental on-device version of the sound recognition model. It is faster and lighter than the server-side version in order to save storage and battery life of mobile devices
API	An interface between the sound recognition model and different devices and applications
Mobile App	An Android app that records surrounding sound on users' device, communicates with the server to provide scene recognition services, and triggers actions upon the detection of predefined scene
IoT Device	Bring sound recognition service to home and businesses with integration with third-party services such as smart speakers (e.g. Google Home, Amazon Echo)

Table 2. Summary of components

The components will be connected in the following way.

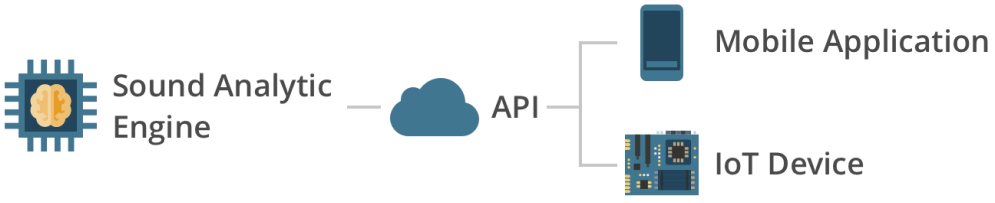


Figure 2. Overall architecture

2.2 Sound Recognition Model

To predict context with sound, we came up with a machine learning model shown in Figure 3. It takes processed audio as input and output the predicted context. The model processes data in two steps - a pretrained model first embeds the input audio into a vector representation of audio features; the classifier then transforms the representation into predicted context. The pretrained model only accepts raw audio (also known as bytecode)[14], compressed audio like MP3 needs to be first converted into raw format before being fed into the model (such conversion is discussed in Section 2.3 API).

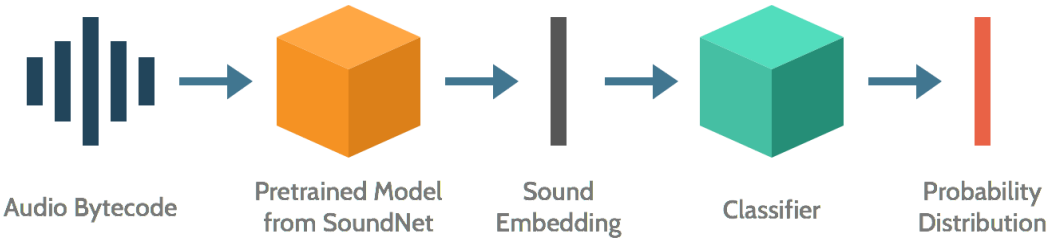


Figure 3. High-level structure of sound recognition model

2.2.1 Design

We reimplemented part of SoundNet as a feature extractor, but trained a new classifier to fit the AudioSet[15] dataset. After trying different classifiers, including SVM, Random Forests[16], XGBoost[17], k-NN[18], MLP[19] and LSTM, LSTM is the most performant classifier (Detail comparison is shown in Section 2.2.3 Testing). The pretrained model and classifier are used for sound prediction.

A. Data Collection

We handpicked 9 types of useful sound for classification. They include baby crying, cheering, finger clicking, raining, glass breaking, metro, indoor, natural and urban. We obtained samples from the AudioSet dataset. In addition, we also obtained some videos from Youtube for sound not included in AudioSet.

Interestingly, the AudioSet website does not provide readily available data, but a CSV file containing all Youtube video IDs, and the corresponding labels, start time and end time. We thus designed a crawler to download and preprocess the Youtube videos listed in AudioSet (the details of the crawler is discussed in Section 2.2.2 Implementation).

---

[14] Raw audio, or bytecode representation, refers to uncompressed audio in a sample-by-sample format without any header information (sampling rate, bit depth, endian, or number of channels).
[15] AudioSet is a dataset by Google, containing 2 million clips from YouTube with 527 labels.
[16] Random Forests is an ensemble learning method that constructs a multitude of decision trees at training time and outputs the class that is the mode of the classes of the individual trees.
[17] Extreme Gradient Boosting (XGBoost) is an implementation of gradient boosted decision trees designed for speed and performance.
[18] k-Nearest Neighbors (k-NN) is a non-parametric method for data classification and regression. It finds the k closest training examples and performs majority vote for the final output label.
[19] Multilayer perceptron (MLP) is a neural network with at more than two layers. Except for the input nodes, each node is a neuron that uses a nonlinear activation function.

B. Feature Extraction

As previously mentioned, we used part of SoundNet CNN as the feature extractor for our model. An audio passes through various CNN layers, in which 5 of the intermediate output layers (8, 14, 17, 18, 21, 24) are selected as the candidate features to be input to the classifier, based on its performance reported by the literature. These features are in the form of a 2D array describing the sound in different time period of the audio, for example, the shape of layer 24 of a 10-second audio is [15, 1024], meaning that the audio is split into 15 segments (number of features in Figure 4), each represented by 1024 numerical values (feature dimension in Figure 4). Then it remains as a multi-label classification task for the classifier.

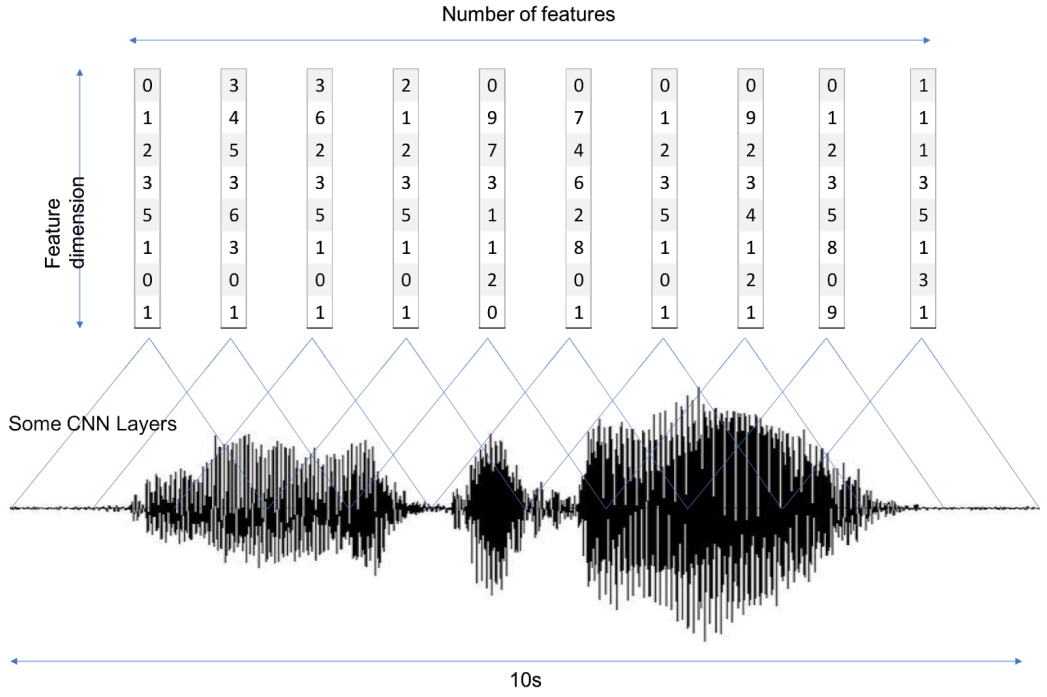


Figure 4. Illustration of SoundNet feature extraction

C. Classification

We used two approaches to classify the sound features, namely, cross-sectional approach and time-series approach. The first approach treats the features independently, while the second one treats the sound features as time-series data.

Cross-sectional approach

Probability aggregation

During training stage, the feature vectors are treated as individual training samples, i.e. for 15 feature vectors generated by a 10-second audio, 15 feature-label pairs are added to the training set individually.

During prediction, the output of a classifier is the probability of each class. An audio with 15 feature vectors will generate 15 such corresponding probability distributions. The label with the averaged probability across the 15 distributions would be output if it is larger than a user-given threshold.

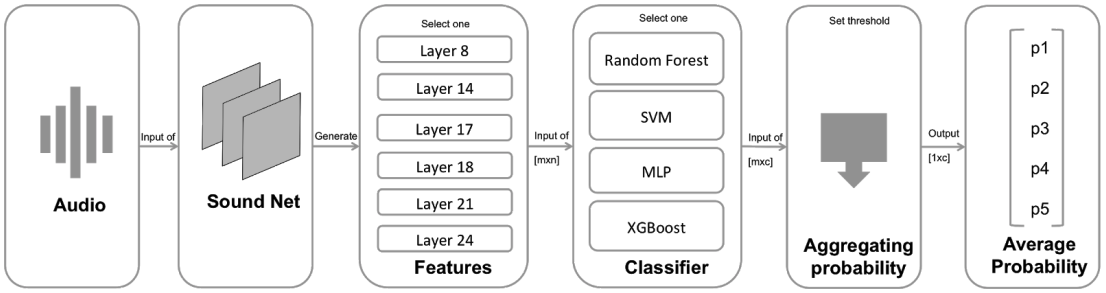


Figure 5. Steps of classification

Feature averaging

Instead of averaging the probability, we may also consider averaging the features to get a more global view of the entire 10s audio. So, the 15 feature vectors generated by a 10-second audio are averaged and feed into the model as one sample point. Same averaging procedure applies to the prediction.

Time series approach

Long short-term memory

The cross-sectional approach does not consider the time dependency between the sound in preceding timestamps. By using long short-term memory (LSTM) recurrent neural network, we can capture the time information to make a global decision on what type of sound the entire sound clip belongs to. The accuracy is expected to be higher, but the drawback is more training samples and data padding is required for audios of different length.

2.2.2 Implementation

A. Data Collection

We implemented a YouTube crawler in Python with a library called 'YouTube-dl'. We used the crawler to download YouTube videos and convert them into MP3 files. The MP3 files are then trimmed in batch according to the start and end time provided by AudioSet CSV files with AudioSegment, a library for audio editing.

B. Feature Extraction

The reimplementation of SoundNet is built on top of TensorFlow[20], instead of Torch[21] which was used in the original model. We also implemented a batch version to process multiple audio clips at the same time for faster computation.

Design Justification - Why TensorFlow?

Tensorflow supports automatic differentiation of tensors in huge dimensions, enabling us to design complex neural networks in graphs structure. It is also widely used in production environment by large companies like Google and Netflix so it provides better documentation, tooling and community support than other

[20] TensorFlow is an open-source software library for dataflow programming across a range of tasks.
[21] Torch is a machine learning framework based on the programming language Lua.

frameworks.

C. Classification

We implemented an SVM, a XGBoost Tree, and a Random Forests model using the Scikit Learn library. Random Search and Grid Search is applied to look for the optimal hyper-parameters: slack variables, kernels, tree span, number of trees, etc. The neural network classifier is implemented using TensorFlow as it provides more flexibility on adjusting the optimization parameters. Lastly, the LSTM model is implemented using Keras, for quick and more intuitive implementation. We also visualized several metrics, like ROC[22], AUC[23] and confusion matrix, to assist our evaluation on classifiers' accuracies (Details in Section 2.2.3 and 2.2.4).

Design Justification - Why using these ML tools?

Scikit Learn is a python machine learning library. It covers most of the popular statistical data science and machine learning methods and offers a wide range of preprocessing and data utilities.

Keras is a wrapper of Tensorflow. It offers consistent and simple API, making it easy and intuitive to use, allowing fast implementation of complicated model.

After building the model, the model is expected to be compiled into a single module for backend service to use its functions. The module should be capable of (1) load a pre-trained model given its path, (2) output the probability distributions for each label given the sound features and (3) output the one-hot encoding label given the sound features and threshold.

For the computing resource support, the implementations are first deployed in local machines with Docker. However, due to limited computing power, we moved the training phase to Databrick and the AWS Cloud Computing Platform.

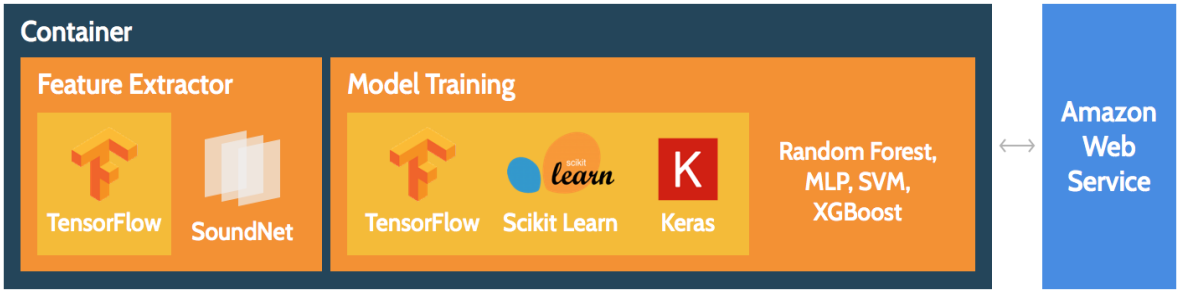


Figure 6. Implementation setup

The performance in our first few trials was not satisfying, potentially due to the noise in the training data. Originally, we only took the vector in the middle of the output layer, i.e. in a 10 time-segmented audio record, we only used the 5th segment and discarded all the others. It missed a lot of useful information in the training data and reduced the

[22] Receiver Operating Characteristic curve, i.e. ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.
[23] AUC stands for Area Under the Receiver Operating Characteristic curve.

model robustness. So, we introduced a majority vote algorithm to calculate the average probability of each label across different segments and output the more likely ones with probabilities higher than a given threshold. This improved the accuracy from 75% to 85% for prediction of 5 types of sound. However, with the increase of number of sounds the accuracy drops as expected. Still, we are able to achieve a 76% accuracy up to 9 types of sound with the use of time-series classification.

2.2.3 Testing

To validate our design, we first reproduced and validated the result of SoundNet using ESC-10 dataset. We then used our own classifier setup and additional datasets to test various classifiers and parameters.

A. Datasets

In addition to the previously mentioned AudioSet dataset, we also used a open-source dataset called ESC-10 for testing.

ESC-10

ESC-10 contains 10 types of clean, single-labeled environment sound. Each audio is of 5 seconds. There are altogether 400 data with 40 sound clips for each class.

AudioSet

AudioSet, as mentioned, contains 632 classes of 2,084,320 multi-labeled 10-second sound clips extracted from YouTube videos. However, in this dataset, there is a class imbalance problem yet to be solved.

We used 80% of the data for training and the remaining 20% for testing. Performance is evaluated based on the two metrics - Jaccard similarity score (multi-label prediction accuracy) and F1 score (optimal value balancing precision and recall). The cross-sectional classifiers we used are Support Vector Machine (SVM), Multilayer Perceptron (MLP), Random Forests and XGBoost. The time series classifier is Long Short Term Memory (LSTM). The performance of these models are compared below.

Classifier	Metrics	SVM	MLP	Random Forests	XGBoost	k-NN	LSTM
ESC-10 (10 Classes)	**Accuracy (%)**	**85.3**	65.2	72.3	82.5	57.0	75.0
	F-Score	**87.6**	67.8	78.4	83.2	61.8	76.8
Subset of AudioSet (5 Class)	**Accuracy (%)**	70.3	80.0	62.7	63.4	45.7	**91.7**
	F-Score	70.7	83.2	65.9	68.0	48.4	**92.3**

Table 3. Performance of different models

From the experiments, we validated our concept of using neural network approach to classify sound. SVM performed the best on the clean ESC-10 dataset (Table 3). This can be explained by the complexity of the problem. Esc-10 is a rather clean data set with

nearly no noise in the audio signal, complex models like MLP and LSTM may easily overfit the problem.

For the actual AudioSet dataset, LSTM outperforms other models, and MLP performs the second best and is the best model among other cross-sectional classifiers (Table 3).

Layer	8	14	17	18	21	24
Accuracy (%)	43.4	56.8	57.0	**80.0**	74.3	20.0
F-Score	53.1	61.8	61.0	**83.2**	76.8	33.3

Table 4. Performance of different SoundNet layers (MLP on AudioSet subset)

Among all layers, we found that using layer 18 generates the best results, so we further compared the performance of different threshold values with layer 18 (Table 4).

Threshold	0.25	0.3	0.35	0.4	0.45	0.5
Accuracy (%)	59.3	**80.0**	75.1	69.5	61.5	39.5
F-Score	66.4	**83.2**	77.3	73.2	64.5	55.6

Table 5. Performance of different thresholds (Layer 18 of MLP on AudioSet subset)

To generate multiple label output, we can set different threshold for the model to recognize the sound if its corresponding predicted probability is higher than this threshold. The optimal threshold of our model is around 0.3. (Table 5)

# of Classes	4	5	6	7	8	9
Accuracy (%)	92.8	91.7	90.6	88.4	80.1	73.9
F-Score	93.4	92.3	91.0	88.8	80.6	75.2

Table 6. Performance of LSTM on various number of sound

B. Empirical Testing

To further verify our model, we also tested the model performance using real time prediction via smartphone and IoT device we developed. The device is exposed to wild sound. From the experiments, the model performs well in certain types of sound, including cheering, finger ticking, indoor (silent). However, for other types, like rural, nature and raining are less accurate.

2.2.4 Evaluation

A. Model Selection

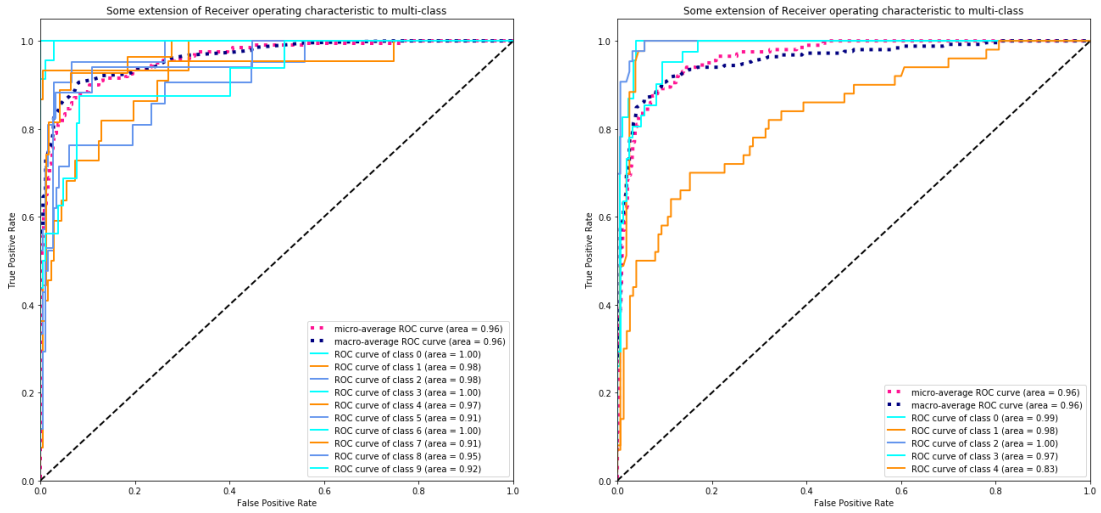
The hyper parameters of SVM, XGBoost and Random Forests were selected by 10-fold cross validation with Grid Search[24] and Random Search[25]. For MLP, we kept track of the

[24] Grid Search tries all the parameters specified by users and select the best combinations

training loss and validation loss curve. Based on our performance metric, we evaluated our model based on the accuracy, F-score and also the ROC curve.

B. ROC Curve Evaluation

The model performs well on ESC-10 dataset, which is better than the pure classifier on raw sound data suggested by the ESC community (85%). However, it has an accuracy close to the SoundNet paper suggested. From the ROC curve, most of the classes have high AUC, indicating that both precision and recall are high. This shows that the performance of our model on clean sound data is satisfying. Noticed that more generic sounds, like “indoor”, are harder to predict because there are fewer distinct features. Therefore, the AUC is low for these classes Figure 7(b).



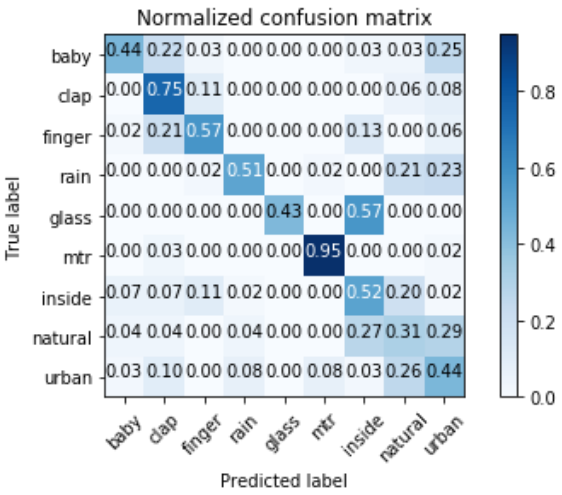
(a) (b)

Figure 7. ROC of (a) ESC-10 and (b) AudioSet

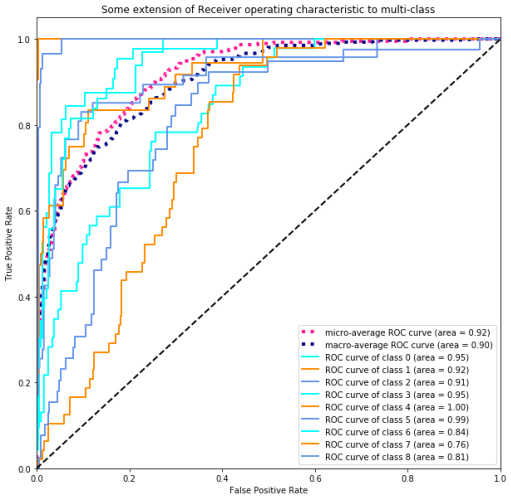
C. Evaluation on LSTM

Nine types of sound are chosen to be our final goal based on their distinguishability and usefulness. The sounds include: baby crying, cheering, finger tapping, raining, glass breaking, inside metro, indoor, outdoor natural and outdoor urban. The accuracy, F-score is (55.1%, 60.9) for MLP and (73.9%, 75.2) for LSTM. LSTM performs better than MLP significantly. The breakdown is illustrated as below.

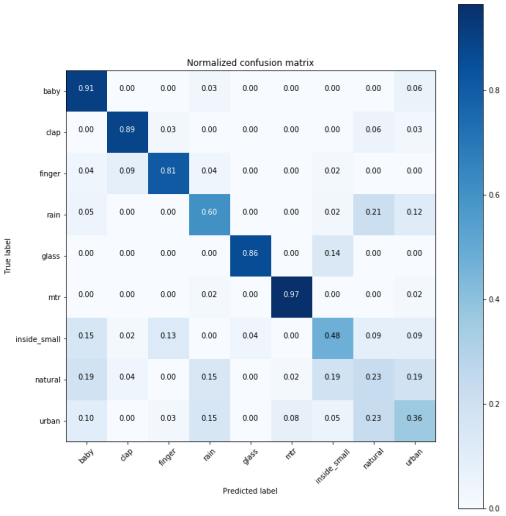
[25] Random Search randomly tries the parameters within the range set by users and select the best combinations



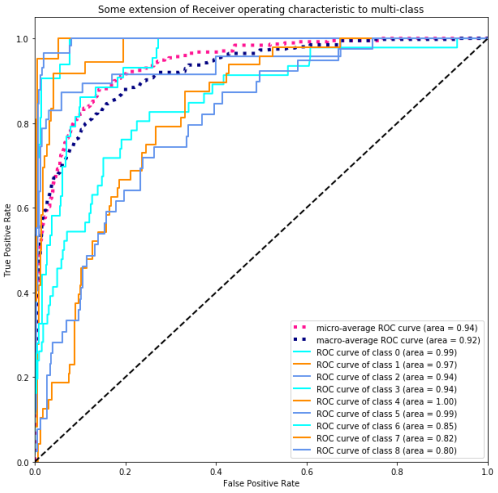
Confusion Matrix of MLP



ROC of MLP



Confusion Matrix of LSTM



ROC of LSTM

Figure 8. Performance for MLP and LSTM on 9 types of sound

Despite concluding LSTM performs better than MLP, it can be observed that more distinct sounds, for example, crying, finger clicking and metro background sound, in general have a higher accuracy (darker colour along the axis), the more context-related sound like urban, natural are indoor often classified as some other sounds, probably due to more noise and lack of distinctive characteristic of these sounds.

For demonstration and usability purpose, accuracy is more important than number of sound. So, the performance on different number sound is also investigated to get the demo work well.

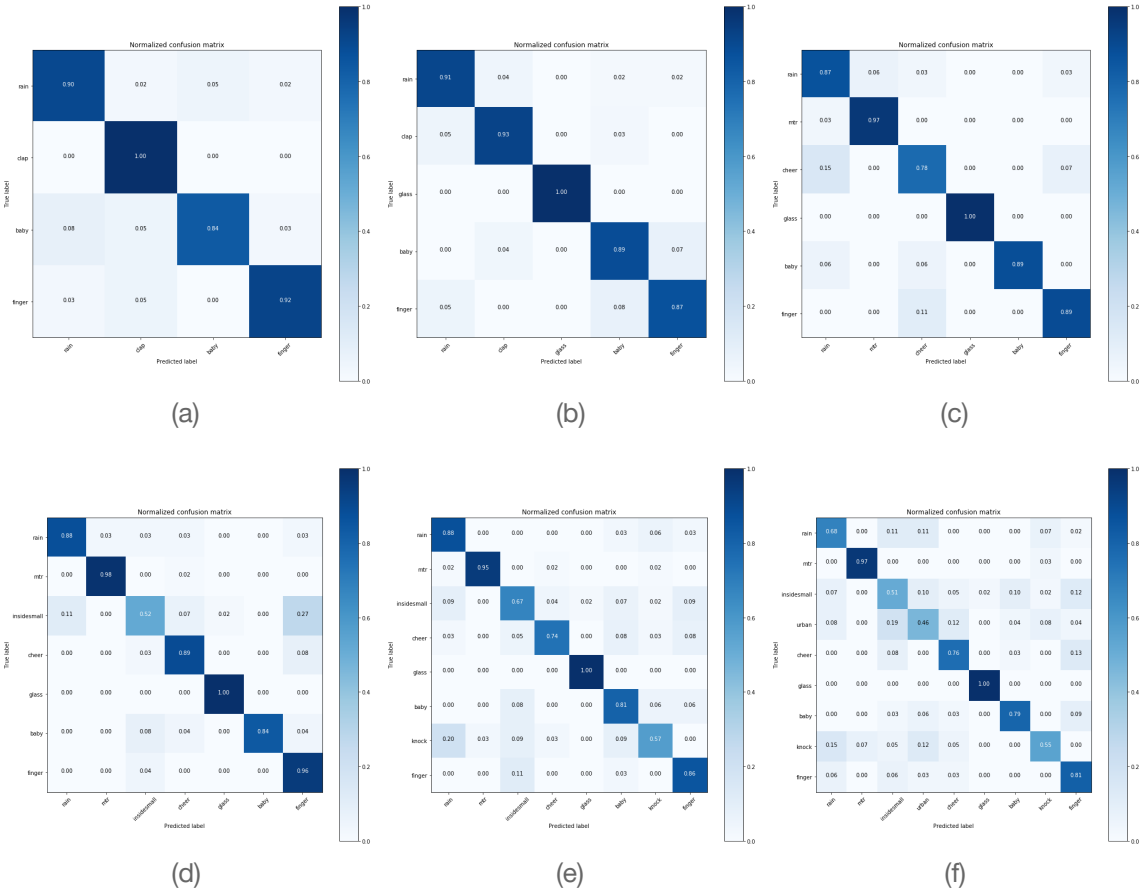


Figure 9. Confusion matrices of LSTM model on different numbers of sound

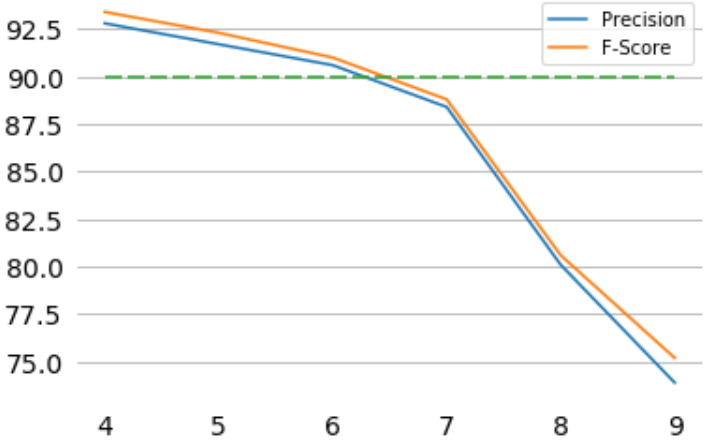


Figure 10. Optimal number of sound to achieve 90% accuracy

According to the result from Table 6 and Figure 9 (a-f), the accuracy drops with increased number of sound. As specified from the proposal, the classification is expected to give around 90% accuracy for it to be useful in application. Therefore, a 7-sound model with test accuracy 88.4% is delivered in our demo (Figure 10).

2.3 API

The role of Zamplify API is to handle recognition requests from client applications, and provide complementary application functionalities, such as user system, push notification, 3rd party integration, and so on. In fact, it is a collection of APIs that deal with different aspects of Zamplify. Its functionalities include but is not limited to the following.

- Preprocess audio in .mp3, .ogg, .aac or .m4a format from the requesting device
- Pass preprocessed sound data to the sound recognition model for prediction
- Output the prediction in JSON format
- Store and manage user data (profile and authentication details of 3rd-party apps)
- Integrate IFTTT to support third party actions to be triggered
- Integrate Firebase Cloud Messaging[26] for push notification of client applications
- Provide device pairing API to pairing up different IoT devices with user account

The communication among API and other applications is shown below.

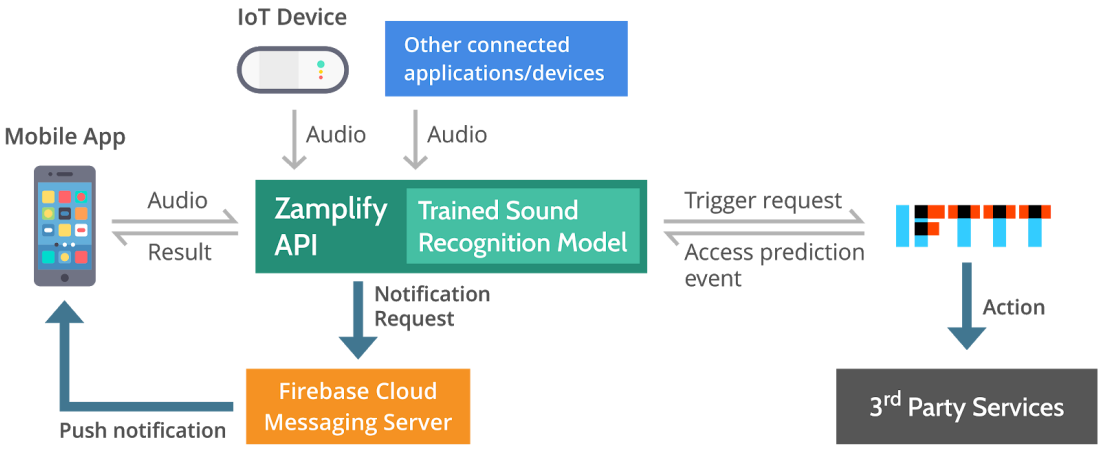


Figure 11. Integration and workflow for the backend

2.3.1 Design

To enable real-time prediction, our trained sound recognition model is wrapped in Flask, a Python RESTful[27] framework, to provide an endpoint for prediction. The framework also supports user management with MongoDB[28].

Design Justification - What is Flask, and why using it?

Flask is a micro framework for building RESTful API in Python. It provides minimal server functionalities. This facilitates agile development without breaking existing modules while developing new features.

[26] Firebase Cloud Messaging is a service provided by Google for sending push notifications to android devices

[27] Representational State Transfer (REST) is a communication architecture which describes a system that transfers data between a client and server.

[28] MongoDB is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemas.

In order to stimulates the development and deployment process, we adopted the idea of containerization[29] to abstract the runtime environment from different hardware infrastructure so as to reduce platform-dependent bugs and achieve unlimited and automatic scaling[30]. A Docker image[31] was built to pack the code into a container to further simplify the deployment. Also, to make the API service scalable, Kubernetes[32] is used to manage and scale Docker containers depending on network traffic. It creates more containers when the server experiences massive amount of network requests.

Request-to-container assignment is managed by load balancer. It makes sure the amount of work assigned to each container is even and fair. For demonstration and development purpose, the entire stack is hosted on Google Kubernetes Engine[33] for public access.

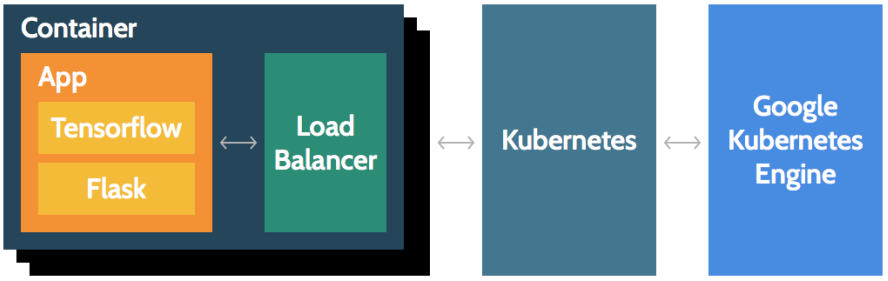


Figure 12. Architecture of backend

The API is currently available at https://zamplify.ml. To secure our API service and prevent suspicious attack, our service sits behind Cloudflare reverse proxy[34]. All requests will be redirected to our server Cloudflare if they are not spam or attack.

[29] Containerization refers to an operating system feature in which the kernel allows the existence of multiple isolated user-space instances. Instances are called containers.
[30] Container is designed to be independent of each other. When a container is under pressure, any number of containers belonging to the same application can be automatically spawned to share the workload.
[31] Docker is a popular containerization service. A Docker image refers to the original copy of an application. Each container is created by cloning its corresponding image.
[32] Kubernetes is an open-source system for automating deployment, scaling and management of containerized applications.
[33] Google Kubernetes Engine (GKE) is a Kubernetes for Docker container and container clusters that run within Google's public cloud services.
[34] Cloudflare is an security service that provides a content delivery network, DDoS mitigation, Internet security services and distributed domain name server services, sitting between the visitor and the Cloudflare user's hosting provider, acting as a reverse proxy for websites.

2.3.2 Implementation

A. Prediction API

To make a prediction, the server has to undergo three processes: (1) Conversion of audio to bycode; (2) Extraction of sound feature from SoundNet; (3) Classification of sound features.

Specifically, the server accepts requests with the 'multipart/form-data' header, a parameter 'file', and a file with .mp3, .m4a or .acc extension. The uploaded files are stored in a temporary folder and converted into bytecodes using the Librosa[35] library.

The bytecodes are then passed to the SoundNet for feature extraction. The resultant features are fed into our classifier, which returns a probability vector of different labels. After that, the server constructs the following JSON structure and sends to the users.


```
{
    "confidences": {
        "BABY_CRYING": 0.0011171671794727445,
        "CHEERING": 0.012127059511840343,
        "FINGER_CLICKING": 0.0012957437429577112,
        "GLASS_BREAKING": 0.0006360174156725407,
        "INDOOR": 0.0014964675065129995,
        "NATURAL": 0.061673641204833984,
        "ON_TRAIN": 0.0006696179043501616,
        "RAIN": 0.0012489261571317911,
        "URBAN": 0.919735312461853
    }
}
```


Figure 13. Response of the Prediction API

Apart from raw sound files, the server also accepts features generated by SoundNet as input. These features can be locally extracted from the client side using Tensorflow Lite. This reduces the size of data sent over the network and improves the response time for actions that require low latency, like taking photo when it detects finger clicking.

Classes with probabilities higher than a certain threshold (0.3 in our case) are chosen to be the predicted context. For example, the prediction is "URBAN" in the above JSON.

B. User Management API

The user management API allows individuals to associate their Zamplify accounts with IFTTT, as well as records their paired IoT devices. It uses JWT for authentication and Facebook OAuth[36] Login. Once a user has successfully logged in to our system, he is given an access token to access the API resource for 8 hours. If the access token is expired, a refresh token can be sent to the server to retrieve a new access token. With this design, we ensure that no personal information, like password or Facebook access token, is saved, which greatly decreases the risk of leaking sensitive information.

[35] Librosa is a python library for music and audio analysis

[36] OAuth is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords.

Design Justification - Why JWT?

JWT stands for JSON Web Tokens. It is an open industry standard for securely transmitting information between parties as a JSON object using HMAC encryption algorithm. It guarantees integrity and security of data.

C. IFTTT Integration API

As mentioned, IFTTT is a popular trigger-action platform. By integrating with IFTTT, we expand the number of available actions to over 360 services. The IFTTT Integration API transforms prediction result of a user into a format that is compatible with IFTTT. OAuth2 is used to ensure that the user authorizes IFTTT to access his prediction records.

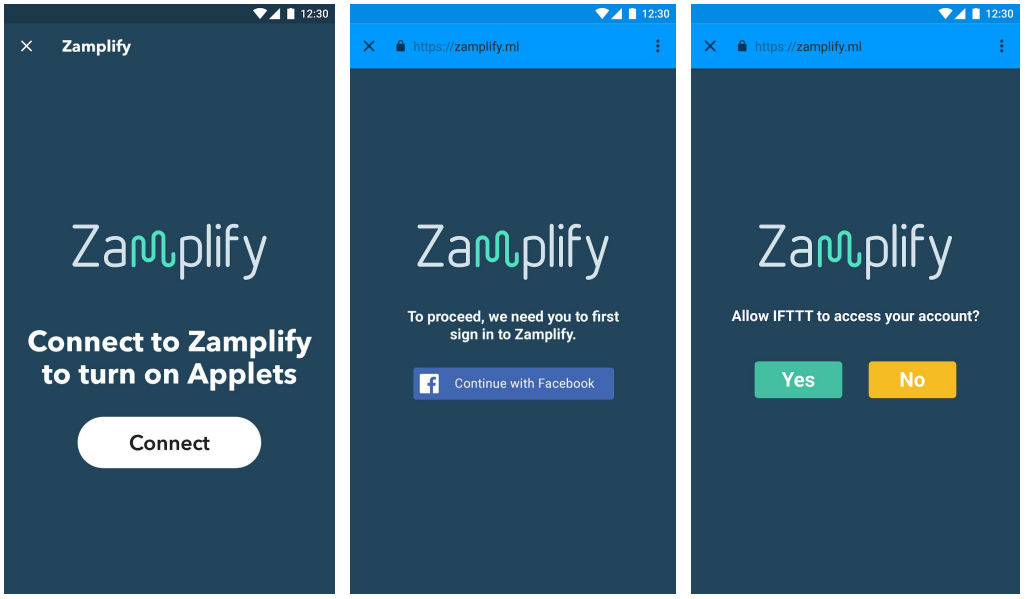


Figure 14. Screenshots of IFTTT OAuth flow

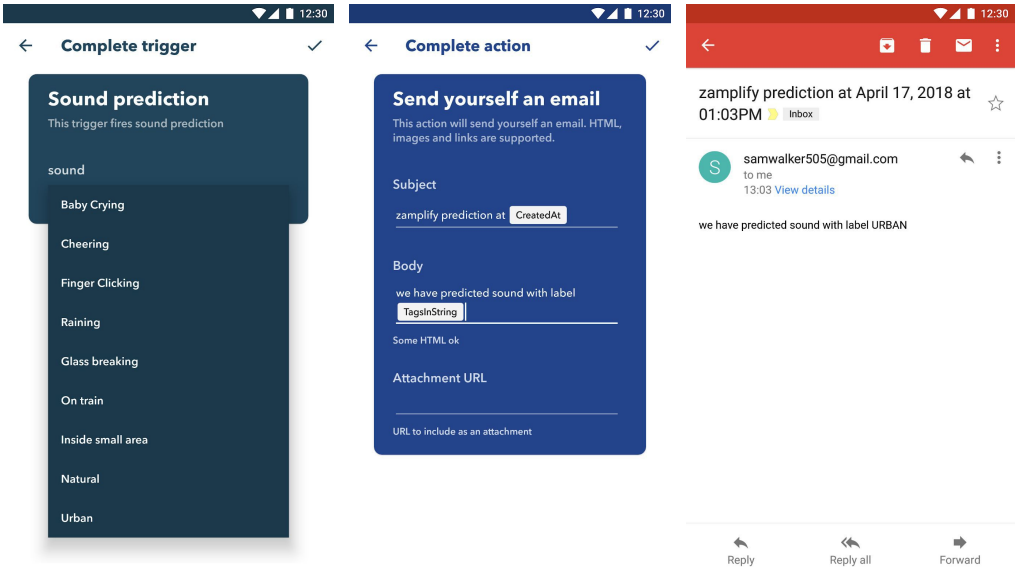


Figure 15. Screenshots of IFTTT usage

D. Device and push notification API

To associate Iot device with a particular user, the device and push notification API is developed. Each device is given a unique ID and a “secret”. Users are required to input both the ID and “secret” to pair-up a device. Our server will prompt an error if the device has been associated up with another user, or upon an incorrect device secret.

To make a prediction, the IoT device sends the recorded the sound file with its device id to the server. A push notification with the prediction result will be delivered to the owner’s smartphone through a pre-registered firebase cloud messaging token.

2.3.3 Testing

The API passed all unit tests with the JavaScript framework Mocha (Figure 16). These tests ensures that the system catches missing file exception, invalid format exception and provides a valid probability vector given a valid request. This verifies the stability and robustness of our API.

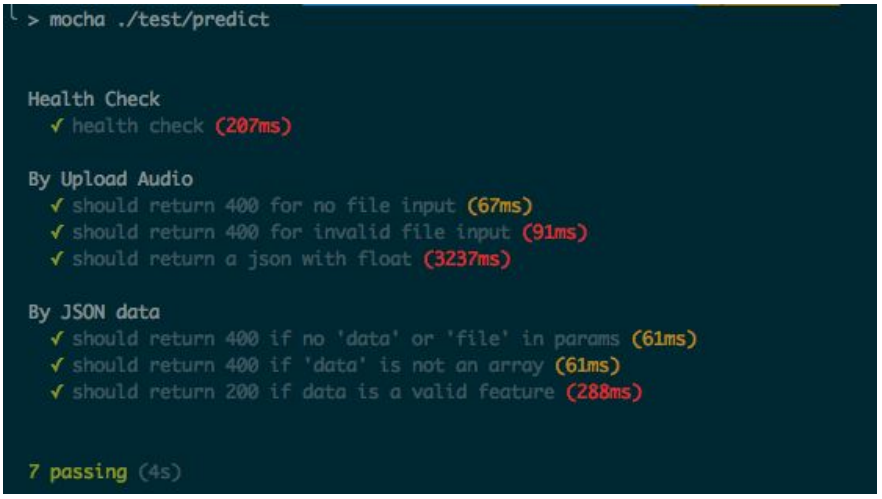


Figure 16. Result of Prediction API test

Other than our own tests, IFTTT also provides integration test for connected services. Our server passes both the connection test and endpoint test. For the device and push notification API, to the best of our understanding, there are no available standard testing tool. So only empirical test is carried out for ensuring its correctness.

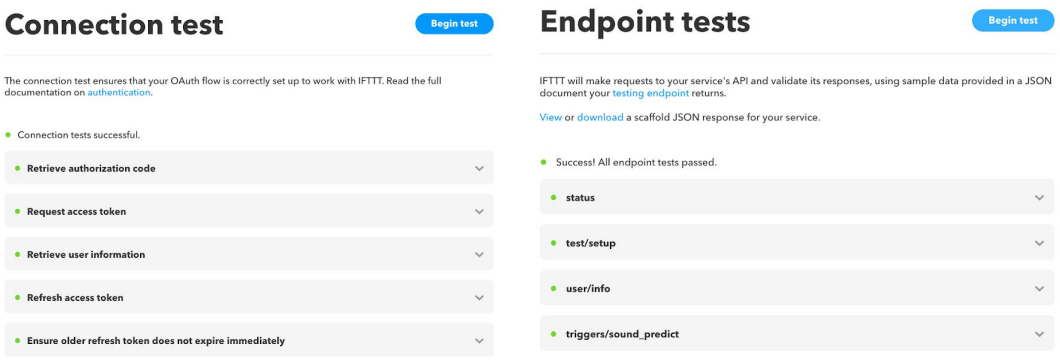


Figure 17. Results of IFTTT connection test and endpoint test

2.3.4 Evaluation

The major metric for evaluating the API performance is the response time. At the beginning, every module was developed separately, therefore, individual module like SoundNet model had to be initialized from a local numpy file, which caused redundant disk IO for saving used and intermediate features extracted from SoundNet. It took 16 - 20 seconds to perform a prediction.

To improve the response time, we converted the weights of SoundNet to TensorFlow checkpoint file (ckpt), which makes the restoration much faster as it is natively supported by TensorFlow.

As explained, our classifier only consider the output from SoundNet layer 18. So, we only perform the convolution until layer 18 and pass it to the classifier. This avoids redundant computation for processing the preceding layers and unnecessary file IO for saving and reloading the features.

Secondly, we improve the response time by applying the singleton design pattern. This avoids instantiating multiple machine learning model instances over time, which saves lots of computational resource and reduces the latency.

2.4 Client-side Sound Recognition Model

Due to the limited computational power of smartphones and IoT devices, we also explored the possibility of transforming the trained model to a faster and lighter version for running in a portable environment like Android and Raspberry Pi Zero W. Although this model was mainly for experimentation and was not actually deployed due to some technical issues, it has improved our understanding about the capabilities of TensorFlow Lite.

2.4.1 Design

We aimed at converting our server-side model into the lite version that can be run efficiently with Android NDK[37]. The communication between Android SDK[38] and NDK is shown below.

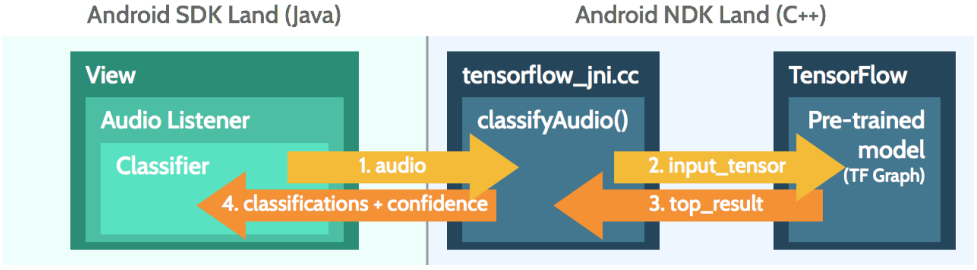


Figure 18. Communication between Android SDK and NDK

Android stores the audio file from the `AudioRecord` class into a buffer, which can then be read by the `TensorFlowInferenceInterface` class to fetch the frozen model output. [16]

In the server-side version, the Librosa library is used to convert audio into bytecode representation. However, Librosa is not part of TensorFlow and can not be ported to the client side easily. Thus, tensorflow.ffmpeg module is used in replacement of librosa (librosa is based on AudioRead which is also build on top of ffmpeg) on Android. [15] (Note that the server-side model still uses librosa while the client-side version uses tensorflow.ffmpeg)

2.4.2 Implementation

The raw model was first converted into TensorFlow checkpoint files, stored periodically at training time as a frozen graph [14] that can used for inference. The workflow is as follows:

[37] Android Native Development Kit (NDK) is a toolset that lets developers implement parts of their apps in native code, using languages such as C and C++. This allows them to run performance-critical part of their application in a bare-metal environment that has minimal runtime overhead.

[38] Android System Development Kit (SDK) is a toolset that provides developers the API libraries and developer tools necessary to build, test, and debug apps for Android. In contrast to NDK, it only runs code in JVM-compatible languages such as Java, Kotlin, etc.

1) Export 3 checkpoint files: .chkp.meta (ML graph and metadata), .chkp.data (weights) and .chkp.index (key-value pair between graph node name and the weight)
2) Export .pb / .pbtxt graph_def file (the ProtoBuf definition of our graph)
3) Open a new Session, combine the graph and the weights (convert variables to constants) into a frozen graph .pb file

The frozen graph can then be used by applications on smartphones and other devices without Internet connection. However, we still need to port the majority vote algorithm and model bridging codes to Android in order to make the client-size model fully functional.

2.4.3 Testing

A test script was written to simulate how an Android app runs the frozen model. The app will load 1) the frozen graph into a new session and provide 2) an audio file to obtain a SoundNet feature extraction tensor (2 inputs and 1 output).

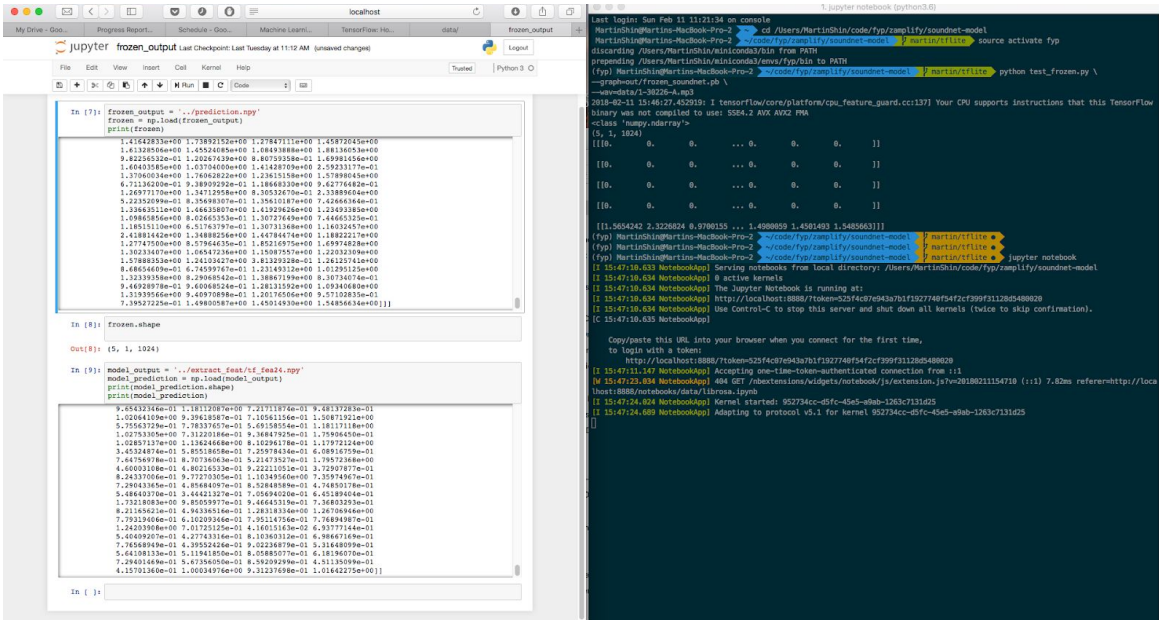


Figure 19. Screenshots of test summary

2.4.4 Evaluation

We can evaluate the performance of our client-side model using prediction latency. Despite the fact that computation in mobile devices is usually slower than in servers, we still expect that the client-side implementation will be faster than the server-side because high network latency is the major bottleneck of the server-side approach.

However, this client-side sound recognition model is not integrated with our core system. Although we successfully ported the SoundNet model to the android operating system, some other data processing and classification libraries , like Librosa, are not supported. Therefore, it is technically challenging to overcome the compatibility issues between different platforms and it is not integrated into our system due to limited time.

2.5 Mobile App

Our Android app serves as a channel of audio data collection and, at the same time, demonstrates the use cases of our API. It records sound automatically every minute and sends it to the server for context recognition. It also allows users to define trigger-action pairs like in IFTTT.

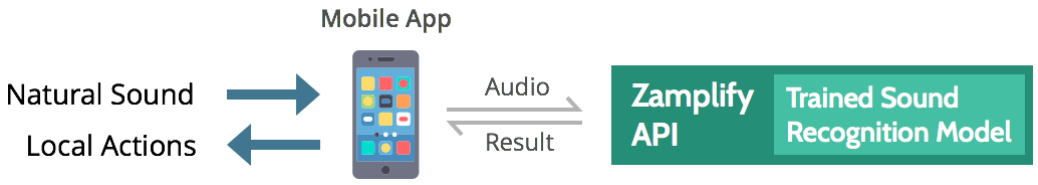


Figure 20. End-to-end data flow

2.5.1 Design

A. User interface and experience design

Like most apps and websites, our design cycle starts with low-fidelity prototyping, followed by visual design, high-fidelity prototyping and actual implementation.

Figure 21. Design workflow of our mobile app

Based on the feedback from users and testers, we iterated the entire process a number of times. Such approach can force us to first think about how the users interact with our app and what functions they really need. The cycle can restart at any time to achieve a fail-fast design and development process.

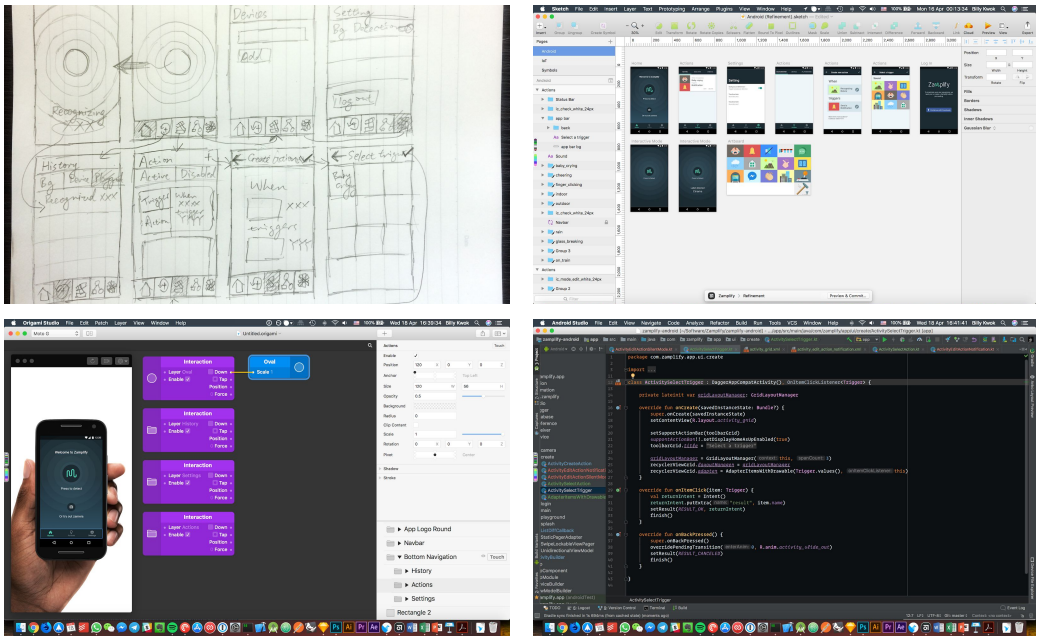


Figure 22. Different stages of design

Design Justification - What are these tools, and why using them?

Marvel is a site that turns sketches on papers to prototype on screen. Compared with similar prototyping tools, we can more conveniently upload photos of user interface sketches and arrange them into different navigation flows with Marvel. This helps us to test and evaluate the design of our application quicker.

Sketch is de facto tool for visual design in the recent years. It provides advanced but easy-to-use functions for vector graphic editing. Compared with other design applications like Photoshop, it has much stronger built-in support for designing user interface and provides convenient one-click Android image asset export.

Origami Studio is a high-fidelity prototyping tool for designing user interface and user experience, built and used by designers at Facebook. Compared with other prototyping tools, it allows us to quickly import design from Sketch, and easily define interaction and animation that simulate the real app experience.

After a number of iterations, we successfully came up with the design of our mobile app, as well as a color scheme for the entire project.

Color scheme

The color scheme is used to maintain the visual consistency among interface elements. Colors are selected to create visual coherence and harmony, while keeping high contrast for visually impaired users. In addition, this color scheme also applies to other materials like reports and presentation slides.

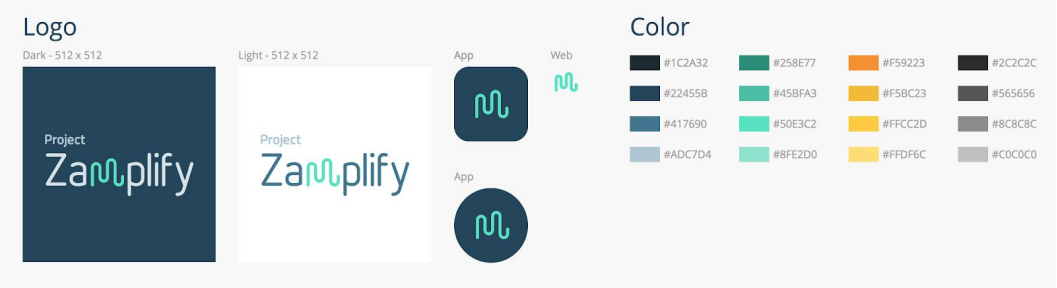


Figure 23. Color scheme

User Interface

With the color scheme in mind, we designed the user interface of our app. The app adopts the popular tab-based bottom navigation design with five tabs - Home, History, Actions, Devices and Settings. These are all the common functionalities of our app. Such design ensures that users can reach the page that we want to navigate to with a single click.

The app supports two modes of audio recognition - background recognition mode and interactive recognition mode. In either case, a 3-second audio sample is recorded and sent to the server for context recognition.

The functionalities of each page are listed below.

Page	Function
Home	Allow users to enter the interactive detection mode, which is designed for testing and demonstration. Users can also click the Zamera button to enter the camera mode that allows them to take photo using sound as a trigger.
History	Display three lists of recognized context. Each corresponds to sound recognized in background, IoT devices and interactive mode.
Actions	Display two lists of trigger-display pairs - one for active pairs and one for disabled pairs. Users can also create new trigger-display pairs on this page.
Device	Manage device associated with the current user. Users can also pair up new device on this page.
Settings	Allow users to change certain behaviors of the app, for example, toggling background sound recognition, etc.

Table 7. Main pages

Apart from the above five pages and their corresponding sub-pages, there are two additional pages that complement the app functionalities.

Page	Function
Splash	Indicate loading state of the app when it is launched or resumed.
Sign up/in	Allow user to sign up or sign in to Zamplify.

Table 8. Function of splash screen and login page

Screenshots of each page are shown in the following.

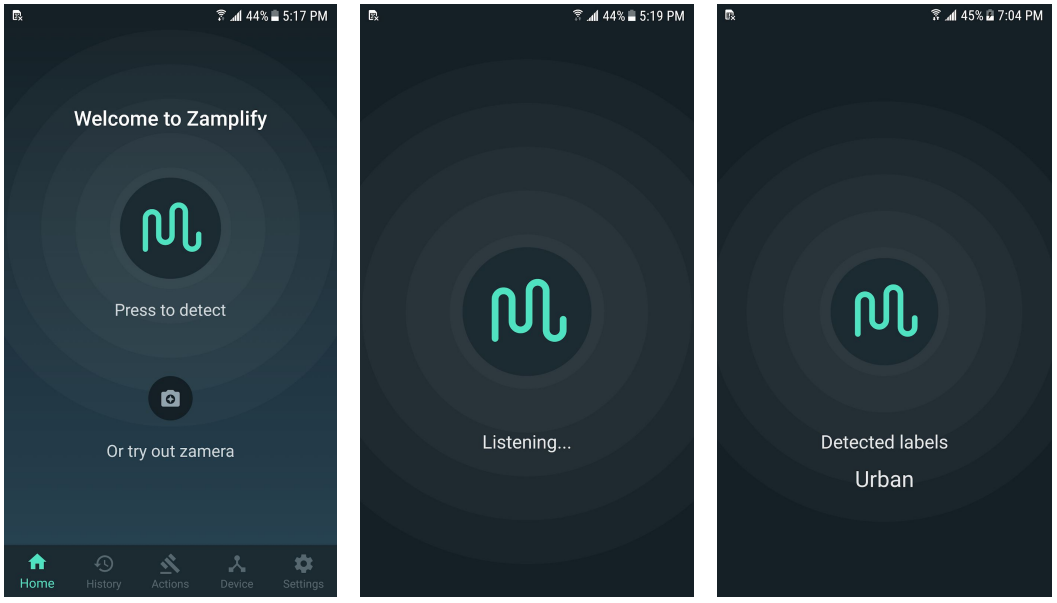


Figure 24. Screenshots of entering interactive detection mode from home screen

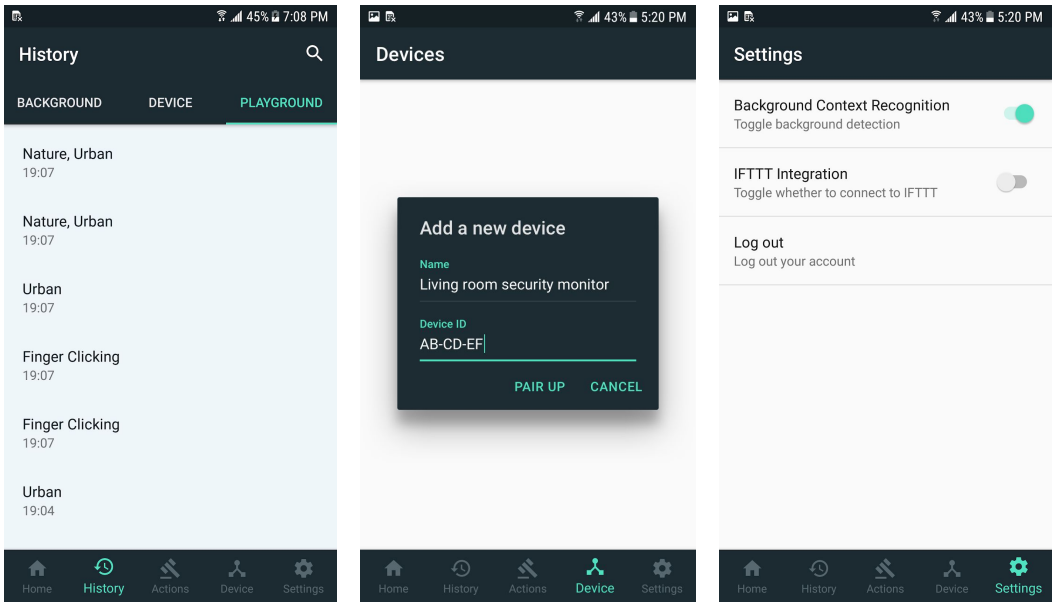


Figure 25. Screenshots of recognition history, device and settings

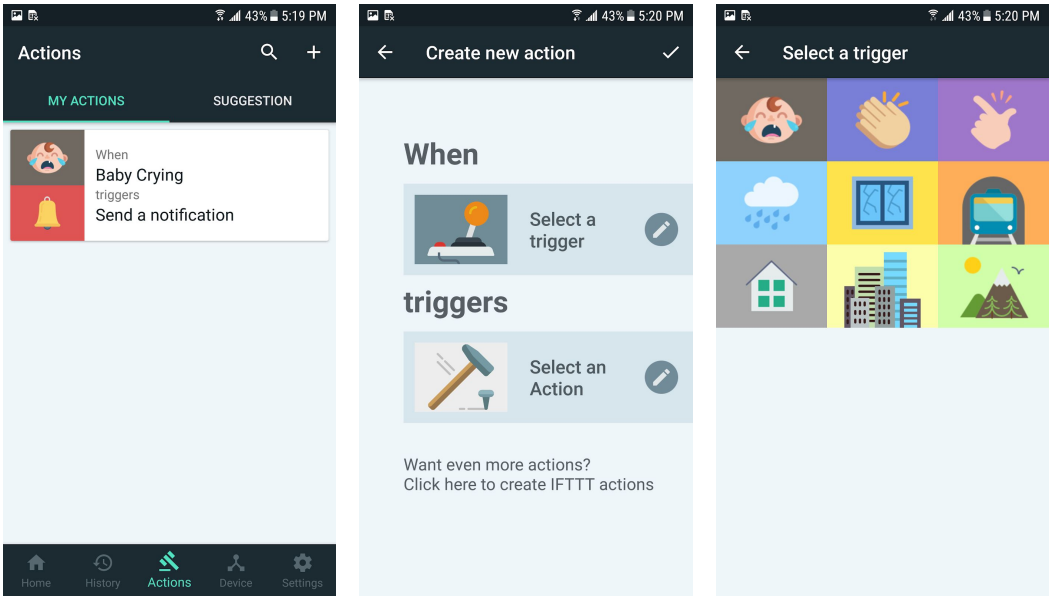


Figure 26. Screenshots of trigger-action pair list and action creation flow

B. Technical Design

The app is written for Android because of the availability of compatible devices and its looser permission control.

Modes of context recognition

In background recognition mode, recording is done roughly every minute. Actual frequency depends on real-time conditions such as computational workload and battery life of the device. Actions are not performed when the corresponding trigger is invoked in the interactive recognition mode. This prevents accidental invocation of certain actions, such as sending emails, that may spam the user.

Supported triggers

When users define trigger-action pairs using our app, they can choose among the list of triggers that our sound recognition model currently supports (refer to Section 2.2). However, they also use activities as triggers in addition to sound. Currently, the supported activity types are running, walking, biking, and driving. This complements our context awareness mechanism and improve overall user experience. Such support is provided by the Google Awareness API, which will be discussed in Section 2.6.2.

Supported actions

The currently supported actions are toggling silent mode, sending notification, ringing alarm. These are the actions that run locally. A wide range of additional actions are supported through the IFTTT integration (refer to Section 2.3).

Design for privacy

To protect privacy, we do not store any recorded audio after recognition so as to minimize the risk of leaking personal data.

2.5.2 Implementation

We successfully built a fully native app in Kotlin instead of a webview-based app like Ionic[39] or hybrid app like React Native[40], because our app requires advanced access to the Android operating system and efficient code execution. Over 200 classes were implemented in the process.

Design Justification - What is Kotlin, and why using it?

Kotlin is a Java-compatible statically-typed programming language running on the Java Virtual Machine. Kotlin has all the features of Java while providing more concise syntax and supporting more functional programming features. This greatly reduced the development overhead and potential bugs in our application.

We used Gradle[41] as our package manager and build tool because of its popularity and great support in Android Studio[42], the IDE that we used to write app. There are two build types in our Android project, test and release. The test build includes extra debugging framework and monitoring tools, while the release build only keeps the minimal monitoring codes for crash report. Test builds are used by ourselves testers while release builds are used for the public.

[39] Ionic is a JavaScript framework for web developers to build mobile apps. It essentially acts as a browser and wraps a web application in a webview to simulate native application experience.
[40] React Native is a JavaScript framework for building native mobile apps, developed and used by Facebook and Instagram. It runs JavaScript in the background thread while keeping the UI thread completely native, resulting in a better performance than webview-based application.
[41] Gradle is an open-source build automation system that builds upon the concepts of Apache Ant and Apache Maven and introduces a Groovy-based domain-specific language (DSL) instead of the XML form used by Apache Maven for declaring the project configuration.[3] Gradle uses a directed acyclic graph ("DAG") to determine the order in which tasks can be run.
[42] Android Studio is the official Android IDE developed by Google. It provides powerful code formatting, autocomplete, compilation, testing, and debugging capabilities that no alternative can provide. It also extensively supports Kotlin, the language that our app is written in.

A couple of libraries were used to speed up our development.

Library	Function
Google Architecture Component	MVVM Architecture
RxJava	Communication between components
Room	Local database management and DAO[43]
Dagger 2	Dependency Injection
Stetho	Debugging
Fotoapparat	Camera
Retrofit 2	HTTP Client

Table 9. Major libraries used and their functions

In the following, we will discuss each function with their corresponding library.

MVVM Architecture

Our app follows the Android MVVM architecture proposed by Google. Under such architecture, the app is broken down into models (M), views (V), and view models (VM) to achieve separation of concern[44]. Models are the representations of different types of data, for example, user, recognition record, and preference. Views are only responsible for rendering data into visual elements and handling user interaction. View Models are lifecycle-aware components[45] that observe the change in data and transform them into suitable structure consumable by views.

Inspired by the React[46] library and Flux[47] architecture, we further improved the original MVVM architecture to achieve a unidirectional data flow. In particular, a view model in our app encapsulates its own state. If other components need to change the state, they need to first create and dispatch a corresponding action to the view model. It then updates components that subscribe to its change.

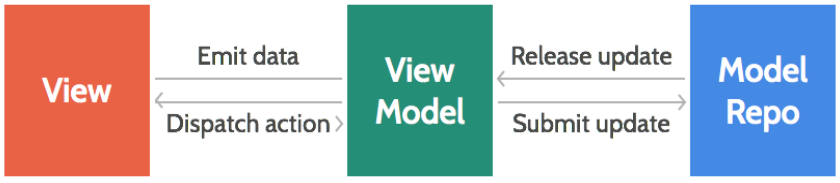


Figure 27. Data architecture of the Zamplify mobile app

[43] Data access object (DAO) is an object that provides an abstract interface to some type of database or other persistence mechanism.
[44] Separation of concerns (SoC) is a design principle for separating a computer program into distinct sections, such that each section addresses a separate concern.
[45] In Android, lifecycle-aware component means that the component is bound to a user interface and shares the same lifecycle with it.
[46] React is a JavaScript library used in the development of single-page sites and mobile apps. It is known for its one-way data flow design that defines user interface as a pure function of data.
[47] Flux is the application architecture that Facebook uses for building web applications. It complements React's composable view components by utilizing a unidirectional data flow.

Communication between components

To facilitate the communication between each layer, we adopted the idea of functional reactive programming[48]. Components send data to each other by subscribing to the data provider. A data item being subscribed is called an observable, which is essentially a continuous stream of data. Each subscriber observes the stream and does custom transformation if needed.

With the observable design, we designed the following application structure. We expect this structure to be more scalable and maintainable than the official Android app architecture suggested by Google because it provides clearer specialization and better isolation among components, and make sure that data transformation as pure (without side effect) as possible.

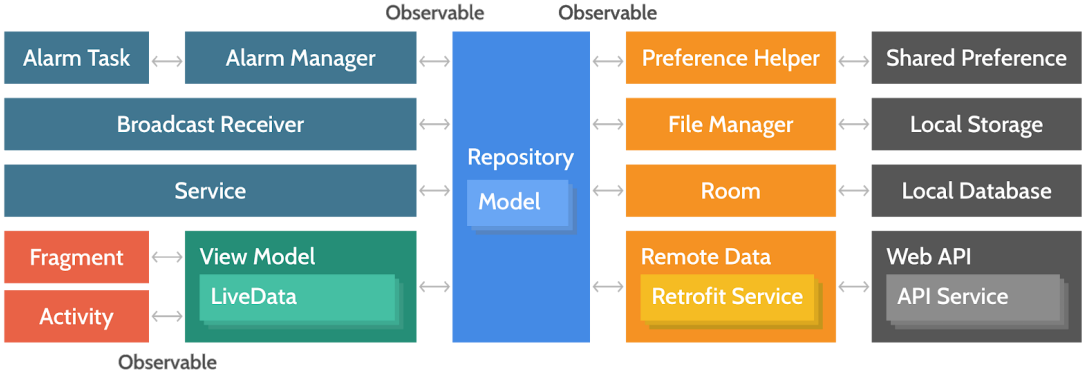


Figure 28. Communication between components

Local database management

Using SQLite in Android is traditionally difficult due to the steep learning curve. Luckily, Google released a Room, an Android library that provides an abstraction layer over SQLite to allow fluent database access while harnessing the full power of SQLite. Room also supports the observable pattern out of the box.

One challenge we faced when using local database in Android is that SQLite, as a relational database, only supports limited primitive types. They are handy for storing simple object but this is not the case in our app. The triggers and actions in our app differ from each other very much. For instance, sending a notification requires a title and a body message, while toggling silent mode only requires the targeted state (on or off). Since they are stored to the same table, we designed a way to store complex object into Android SQLite database.

When the app needs to store triggers or actions into the database, it serializes them into a JSON object in string format before inserting them. When selecting the items from database, the app deserializes the corresponding entry from JSON to a Kotlin object. This is virtually equivalent to using SQLite as a NoSQL database and solves the problem mentioned above.

[48] Functional reactive programming (FRP) is a programming paradigm for reactive programming (asynchronous dataflow programming) using the building blocks of functional programming (e.g. map, reduce, filter).

Dependency injection

Managing dependencies between components can be error-prone. For instance, if component A needs component B in order to perform its function, B must be instantiated before being used by A. The difficulty in handling such dependency increases significantly when the project gets large. While some design patterns such as Singleton[49], can partially solve the problem, they increase the code complexity and decrease readability.

To solve the problem, we use dependency injection framework to automatically manage dependency using a dependency graph. The graph is statically created and analyzed by dependency injection framework Dagger[50] upon compilation. The corresponding initialization code of components is then generated. An example of the dependency graph is shown below.

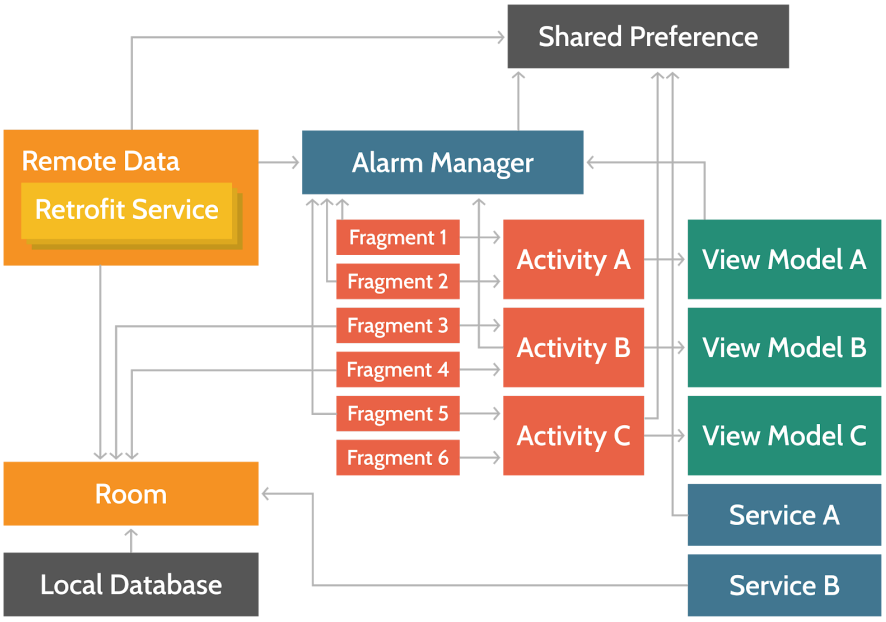


Figure 29. Example of dependency graph

Debugging

Debugging in Android can be painful because some parts of the system are difficult to inspect. When writing website, we can simply open the developer tool and look at every single piece of data stored in the browser. This is almost impossible in Android. Luckily, Facebook has written a library, called Stetho[51], that simulates the developer tool of a website. Once Stetho has been installed, we can open Google Chrome and use its developer tool to inspect the shared preference and database of our app. This greatly reduces the time we spent on debugging.

[49] The singleton pattern is a software design pattern that restricts the instantiation of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system.
[50] Dagger 2 by Google is a compile-time fast dependency injector for Android and Java.
[51] Stetho is a sophisticated debug bridge for Android applications. When enabled, developers have access to the Chrome Developer Tools feature natively part of the Chrome desktop browser.

Camera

Developing camera app in Android is also a challenging task, not mentioning that Android has two camera APIs. We used a popular camera library called Fotoapparat to achieve the camera function in a few lines of codes.

HTTP Client

Sending JSON request and reading JSON response can be difficult in Android. Retrofit is a library that solves this problem. It is a HTTP client that provides fluent API for building HTTP request body and parsing response from server. It also has a handy feature called interceptor that allows us to add JWT to every request header.

A problem that cannot be solved by library is that Android only allows a single piece of code to use the microphone at a time. It means not only that two applications cannot share access to microphone at the same time, but also that the same application cannot have two `MediaRecorder` instances that access the microphone at the same time. This becomes a problem because when users enable background sound recognition, and open the interactive mode at the same time. We need to prevent the background recording class from using the microphone when the interactive mode class is doing recording. We ended up with implementing a `BlockingQueue` that accepts record requests from both classes, and cancels the currently background recording if a new request from the interactive mode class is sent.

Another big problem is that Android bans apps from running long-standing background services. This makes background recording much more difficult than expected. We experimented with several workarounds, such as using Job Scheduler[52], Alarm Manager [53], and Foreground Service[54]. Job Scheduler is not a desirable choice because it only supports repeated task execution down to 15 minutes, which is much longer than what we need. Alarm Manager is a good choice but it does not provide exact scheduling for repeated tasks and the minimum execution interval is one minute. In our experiment, when the device is idle, one-minute execution is achieved. But when the device is busy, the actual scheduling interval can be as long as 10 minutes. This is undesirable due to its unpredictability. We eventually took reference from the Shazam[55] app background song recognition function and used Foreground Service to achieve exact scheduling with interval less than a minute.

[52] The Android JobScheduler API performs tasks asynchronously on behalf of apps, outside the direct flow of user interaction. It optimizes task execution so as to increase system performance and save battery life.
[53] The Android AlarmManager API allows apps to schedule tasks to be run at some point in the future. It provides exact or almost exact scheduling, which is not possible with JobScheduler.
[54] An Android foreground service is a long-running service that the user is actively aware of and is not a candidate for the system to kill when low on memory.
[55] Shazam is the Zamplify for music, an app that identify music around the users.

2.5.3 Testing

We have conducted a series of tests to ensure the stability of our mobile app.

A. Unit Test

Ideally, we wish to use Test-driven Development[56] throughout the project. Unfortunately, due to the need for rapid prototyping, we applied it only in selected components. Still, each component still underwent unit test with JUnit[57] and Mockito[58]. This ensures that all component behave as expected down to the class level, and significantly reduces the chance of crashing and data inconsistency.

B. Integration Test

The integration test of our app focuses on the communication with Zamplify backend and third-party services. The communication should be stable in different network conditions and server status, especially when the server is down. In specific, the app should maintain the HTTP connection with a long enough timeout since prediction mode takes time to run. Even when the timeout is expired, the app should not crash but show a message of no Internet connection or server failure.

C. User Acceptance Test

User acceptance test was conducted to determine if the whole application meet the requirements from users' perspective. Criteria such as mobile app power consumption, response time, ease of use, invasiveness towards daily life were assessed. To conduct the test, we invited our friends and schoolmates to join our internal alpha testing program via Google Play. They provided valuable feedback in the process.

[56] Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: Requirements are turned into very specific test cases, then the software is improved to pass the new tests, only.

[57] JUnit is a unit testing framework for the Java programming language. It has been important in the development of test-driven development. It is linked as a JAR at compile-time.

[58] Mockito is an open source testing framework for Java released under the MIT License.[1][2] The framework allows the creation of test double objects (mock objects) in automated unit tests for the purpose of test-driven development (TDD) or behavior-driven development (BDD).

2.5.4 Evaluation

We are using a number of metrics to quantify the performance and user-friendliness of our mobile app. Currently, most targets are met but there are few items to be further improved.

Metrics	Target	Result	Test
Line Coverage	At least 90% of the lines in testable components at the end of project	91%	UT
Branch Coverage	At least 70% of the lines (including branches) in testable components at the end of project	73%	UT
End-to-end Integration	The app should be able to communicate with the backend smoothly for any endpoint	Yes	IT
Build Size	The release .apk should be less than 150MB	Less than 100MB	UAT
Frame per Second	The rendering of user interface should achieve 60 fps in most parts of the app and at least 30 fps in all parts of the app	Satisfied	UAT
Maximum Navigation	All app features should be accessible with at most three clicks, or gestures	Satisfied	UAT
Battery Usage	The app should not reduce the battery change of regular device by more than 10%	Satisfied	UAT

Table 10. Evaluation on mobile application

2.6 IoT Device

The IoT device that we built is an Internet-connected, always-on sound recognizer that can be used in homes and offices. It demonstrates the power of sound recognition in the context of a smart home or smart office.

2.6.1 Design

The device was developed using Raspberry Pi Zero W microcontroller and a ReSpeaker Mic Hat microphone array. A case was 3D-printed to protect the electronic components and to improve the aesthetics. The device constantly records environmental sound and triggers actions specified by its users. Users can use the Zamplify mobile app to configure their trigger-action pair for sounds collected from the IoT devices.

Figure 30. Initial outlook of the 3D-printed case without the microphone array

2.6.2 Implementation

A. Microcontroller

Raspberry Pi Zero W has been successfully setup. This model of Raspberry Pi comes with WiFi and bluetooth module which can be configured to automatically join a recognized network once booted up. To work with the microcontroller, the official Raspbian OS with GUI interface is flashed into a micro-SD card and inserted into the corresponding slot of the device. After initial setup for WiFi connection, account password configuration, the device can be accessed in headless mode via SSH connection. The device is then soldered with female GPIO ports to allow pin connection to other devices/ modules.

B. Microphone Array

ReSpeaker microphone array module is connected to the 40 GPIO pins of the Raspberry PI to provide access to the microphone input and speaker output. After running the official driver installation script provided by Respeaker, the microphone and speaker card will be visible to the Raspberry Pi using the `arecord --list` and `aplay --list` command respectively.

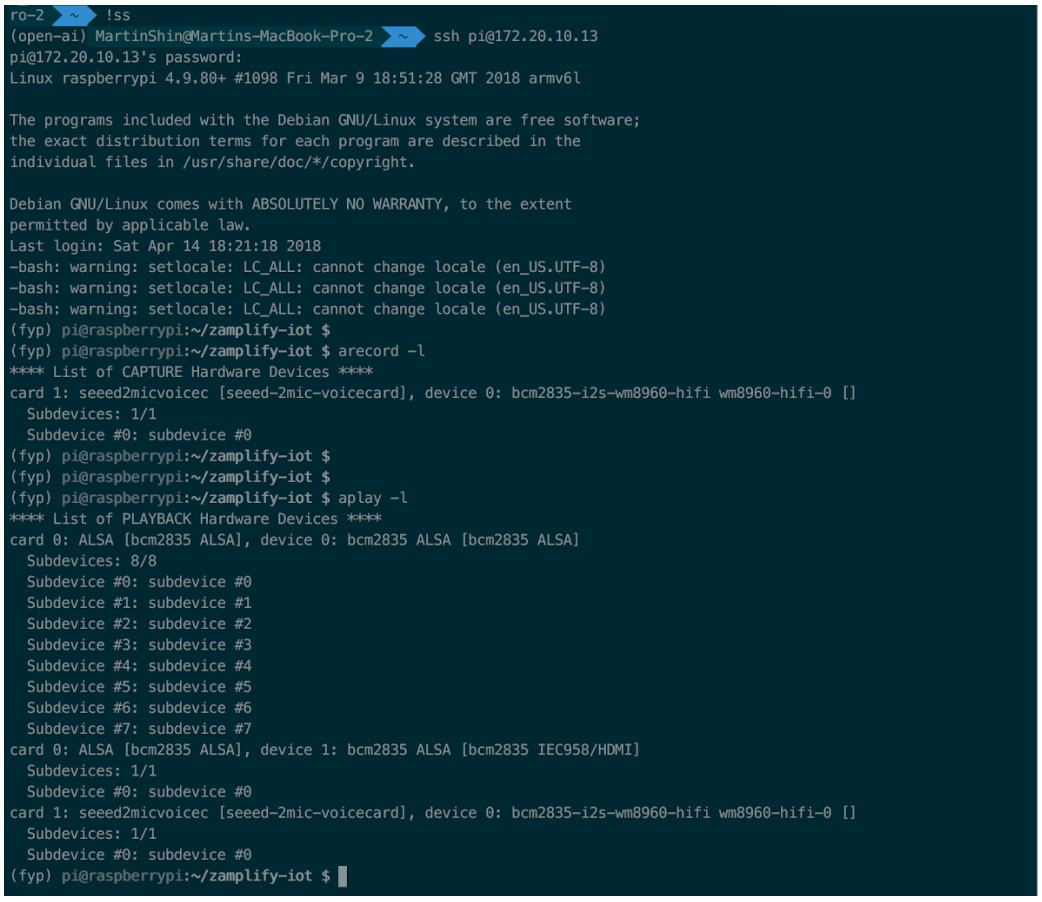


Figure 31. Screenshot of microphone input and speaker output from ReSpeaker

C. IoT Python Script

A cron job[59] is set up to execute the python script for recording a 3-second audio from the environmental every 5 seconds. The audio is then sent to the backend server for prediction via POST request. The cron job is configured to run in background once the IoT device is booted up and connected to the internet. Predictions result will then be pushed to the user's app to handle the user-defined trigger-action pair logic. Configurations of sound recording is implemented to resolve the issue of significant white noise of the microphone by default.

Attribute	Target
File Encoding	S16_LE (Stereo 16 bit Little Endian)
Duration	3 seconds
File Format	WAV file
Sampling Rate	22050 Hz

Table 11. Configuration of IoT device

[59] Cron job is a time-based job scheduler in unix-like computer operating systems.

D. 3D Printing

We have printed the case with a web-based CAD software OnShape. Once the CAD model is completed, it will be exported into .STL file which represent the print model in triangles. The .STL files is then transformed into a .gcode file via Cura, a slicing software provided by the 3D printer manufacturer as instructions code of how the printer will print out the design layer by layer through movements of printer core extruder head in XYZ space.

2.6.3 Testing

Empirical testing is employed to test performance and accuracy that simulates the environment close to real life settings. Audio files of the trained sound categories are played for the IoT device to listen to, where prediction results are returned from the API. Latency is also recorded, where API response is roughly 1.70 seconds for predicting a 3-second clip which amounts to around 5.33 seconds total time for a cycle.

2.6.4 Evaluation

We will evaluate the performance of our IoT devices based on the following criteria.

Metrics	Target	Results	Test
Latency	The mobile app should be able to receive the notification from IoT device in 15 seconds	Passed (5 seconds)	UAT
Sound Quality	The device should be able to hear sound larger than 40 dB in 20m	Passed	UAT

Table 12. Evaluation result of IoT device

3 Discussion

3.1 Project Review

Our project successfully delivered an end-to-end system consisting of a context awareness API powered by a sound recognition model, an Android app, and an IoT device. It is capable of recording surrounding sound, detecting context, setting the trigger-action pairs, and performing actions with our Android app or through IFTTT. We have succeeded in accomplishing most of the objectives in this project.

Objective 1

> **Design a cutting-edge machine learning model for context recognition using sound, either based on existing research and self-invented ideas**

With the trails of multiple machine learning models, we built the machine learning model using the novel approach of LSTM classifier on CNN features to recognize 7 to 9 types of sound. The number of sounds that our model is able to predict is slightly less than the expectation. This is possibly due to insufficient and noisy training samples, making it hard to capture some of the hidden features in sound. Another possible problem is that we model is trained with time-domain information, which is more difficult to model than frequency-domain information.

Objective 2

> **Transform trained machine learning model from a research demonstration to a production-ready system that is scalable and computationally maintainable**

Our system successfully transform the sound recognition model into a production-ready system that is scalable and computationally maintainable. By adapting suitable design patterns and state-of-the-art tools, for example, model simplification, cloud computing, Kubernetes, IFTTT and Firebase Cloud Messaging service, our backend server is able to deliver various complementary services, such as trigger-action mechanism, account authentication, making a machine learning model find its application in daily use.

Objective 3

> **Explore and evaluate the feasibility and practicality of inferring context from non-speech audio content**

By finishing the project and achieving an acceptable result, we have successfully demonstrated the feasibility of using natural sound to infer context. Our prediction has an accuracy of 80% and takes only 3 seconds, which is very practical for real-life usage. Yet, there are still model and hardware improvements to be made before it is launched. For instance, a bad microphone drags down the performance of our model severely. Also, empirical test suggests that the model is not general enough to detect all possible forms of sound. More source of sound samples are required.

Objective 4

Apply machine learning to software and/or hardware that is/are used in practical daily life

A fully functioning Android app and IoT device are developed in our project. They are able to detect user context and perform useful daily-life actions, such as sending push notification, toggling silent mode, and even triggering IFTTT actions with convenient setup.

Objective 5

Gain solid experience in the agile development cycle and project management while using software that involves machine learning

Throughout the project, we gained solid software development and project management experience. Appropriate tools and libraries were used to speed up the development process. We are able to deliver all planned deliverables on time, including proposals, monthly reports, posters and progress report.

3.2 Implications and limitations

Our project result demonstrates that a sound-based context recognition system has huge potential, either as a standalone service or a complementary part of an integrated context awareness API. We cannot wait to see further application of our system. However, our system is not without its limitations. In fact, there are a few restrictions to be aware of.

Limitations of the Sound Recognition Model

First of all, although our model can predict 7 to 9 types of sound with adequate accuracy, the number of classes is still not enough to cover common context in daily life. Also, the accuracy tends to decrease faster than our expectation when the number of classes increases.

LSTM generally requires a fixed length of input. The training samples are of 10 seconds. For audio less than 10 seconds, we pad the audio by duplicating itself. However, the model did not handle the audio that is longer than 10 seconds as it is not our primary goal to predict long audio. Besides, processing long audio takes longer time and creates pressure on the network.

We tried to achieve multi-label prediction using by setting a threshold on the probability. However, the shortcoming is that adding more labels to model may decrease the confidence of predicting existing labels. As a result, the current model still generates single label prediction most of the time.

Due to limited computing power and data, we did not train the feature extraction but took the pre-trained model from SoundNet, which have a different choice of labels from ours and may result in inaccuracy.

Limitations of the API

It takes 3 - 5 seconds for the system to make a prediction, thus, there is a slight delay on the actions being delivered.

Docker virtualization technology is not mature for GPU programming, so it is hard to encapsulate the software from hardware layer if we want to switch to GPU environment.

Tensorflow prohibits the use of multiple graphs at a time. To process the data in two graphs (Soundnet and our classifier), redundant computation is consumed in initializing the two graphs alternatively.

Limitations of the Android Application

A huge limitation of the Android app is that it offers limited built-in triggers (currently only sending push notification and toggling silent mode). Also, triggering of actions is done on device instead of on the cloud. This is less scalable and reliable because the Android app may not always be ready to trigger actions. For example, when Internet connection is suddenly lost after sending the sound to the server, the corresponding actions may be skipped. Also, mobile device has lower computation power than server, and every instruction it runs still consumes battery power (although not significant in our case). This reduces the performance and efficiency of our service.

Limitations of the IoT Device

The hardware microphone records sound in relatively poor quality. This affects the prediction accuracy and requires trial-and-error to find the best encoding format.

The current system architecture requires sending the sound collected to the API. Alternatively, the IoT device may perform sound classification by running the Tensorflow Lite model locally, which adds a layer of protection to data privacy.

4 Conclusion

4.1 Summary

The previous work from SoundNet has provided a clever way to extract features from sound. However, it did not model the time sequence. In this project, we used the neural network approach which comes with flexibility and adaptability, combining the CNN of SoundNet to an extra LSTM classifier to model the sequential property of audio signal and successfully applied it to daily life ambient sounds, without sacrificing the accuracy.

The system extends the current context awareness capabilities to include sound as a powerful signal for context inference. With the trigger-action mechanism inspired by IFTTT, an end-to-end system is implemented to demonstrate the application of our model and the potential of sound-based context cognition.

Throughout the project, we have learned and strengthened various skills. First, we went through the entire machine learning process, from data collection to transforming model into production-ready system. We also had a great learning experience in solving data-related problems, such as missing values, noise and label imbalance. Also, we have not done a project in such large scale before. It involves the use of data science, mobile application development, cloud computing, cybersecurity and Internet of Thing. Not only does it helps us to revisit the knowledge that we have learned in our university life, but also gives us to a good opportunity to practice agile development.

4.2 Future work

Our project proved that sound can be useful for context awareness and we successfully built a system to use make use of such capability to perform useful tasks. In the future, the model and system can be further improved in three aspects.

Sound Recognition Model

To improve the accuracy and the number of sound categories, we suggest collecting cleaner sound samples and increasing the number of samples to obtain better model output. For the LSTM classifier, we also suggest adding an attention algorithm to capture the important moments of an audio sequence for prediction, because sometimes the major events only happen in a sub-frame of an audio clip. With the increased number of samples and complexity of our model, we believe that the system can handle more sound types and achieve a higher accuracy.

API

For the backend system, it can be further improved by merging the SoundNet model with our classifier to reduce the initialization time. Also, migrating the server from single container to a cluster with better CPU and GPU resources also reduces the response time. We could even separate the feature extraction and prediction process into two servers, avoiding reinstanciation of the weight variables. Concerning the API service, we supports Facebook login to skip email verification and forgot password issue. In long term, the system can extend to a email-password system or other third party OAuth system.

Mobile App

To increase the stability of our Android app, we suggest moving the triggering of actions into the cloud. Moreover, although alternative data sources, like GPS, Wifi connection, proximity, and device motion, cannot infer generic context, we can still include them in our system to cover cases that certain context cannot be recognized just by sound. With the integration of multiple data sources, the application can get a more complete view of the surrounding environment.

IoT Device

To enhance the usability of the application, we suggest applying noise filter to reduce noise of the recorded audio in the IoT device. Another workaround is to try different sound recording hardware modules to achieve a better sample quality. Also, personal assistants, like Google Assistant or Amazon Alexa, may be integrated into the IoT device so that it becomes an all-in-one smart home solution.

5 References

1. A. K. Dey et al., "iCAP: Interactive Prototyping of Context-aware Applications," *Proc. Pervasive Computing* (PC 06), 2006, pp. 254-271.
2. B. Ur et al., "Practical Trigger-Action Programming in the Smart Home," *Proc. SIGCHI Conference on Human Factors in Computing Systems* (CHFC 14), 2014, pp. 803-812.
3. C. Chaey, "Foursquare Now Knows Where You Are, What To Do There, Even If You Don't," *Fast Company*, 29 Aug. 2013; https://fastcompany.com/3016555/foursquare-now-knows-where-you-are-what-to-do-there-even-if-you-dont.
4. D. Barchiesi et al., "Acoustic Scene Classification: Classifying Environments from the Sounds They Produce," *IEEE Signal Processing Magazine*, Vol. 32, no.3, 2015, pp. 16-34.
5. H. Lu et al., "SoundSense: Scalable Sound Sensing for People-centric Applications on Mobile Phones," *Proc. MobiSys. ACM* (MOBISYS 09), 2009, pp. 165-178, 2009; http://pac.cs.cornell.edu/pubs/mobisys09_soundsense.pdf.
6. H. Parry, "Introducing Our New VPN Feature: SurfEasy Wi-Fi Security on Android," *SurfEasy*, 12 Aug. 2015; https://surfeasy.com/blog/surfeasy_new_vpn_feature_surfeasy_wifi_security_android.
7. M. Rossi. Et al., "AmbientSense: A Real-Time Ambient Sound Recognition System for Smartphones," *International Workshop on Impact of Human Mobility in Pervasive Systems and Applications*(IHMPSA 13), 2013.; http://ieeexplore.ieee.org/document/6529487.
8. M. W. Newman, "Now We're Cooking: Recipes for End-User Service Composition in the Digital Home," *CHI 2006 Workshop IT@Home*, 2006; https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.4013&rep=rep1&type=pdf.
9. M. W. Newman, A. Elliott, and T. F. Smith, "Providing an Integrated User Experience of Networked Media, Devices, and Services through End-User Composition," *Proc. Pervasive 2008* (PC 08), 2008, pp. 213-227; https://link.springer.com/chapter/10.1007/978-3-540-79576-6_13.
10. R. Arandjelovic, and A. Zisserman. "Look, Listen and Learn," *International Conference of Computer Vision* (ICCV 07), 2017; https://arxiv.org/abs/1705.08168
11. S. Lui et al., "UbiEar: Bringing Location-independent Sound Awareness to the Hard-of-hearing People with Smartphones," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol*, (IMWOT 17), 2017, Article 17; https://doi.org/10.1145/3090082.
12. Y. Aytar, C. Vondrick, and A. Torralba, "SoundNet: Learning Sound Representations from Unlabeled video," *Neural Information Processing Systems* (NIPS 16), 2016; http://soundnet.csail.mit.edu.
13. Y. Dahl, and R. M. Svendsen, "End-User Composition Interfaces for Smart Environments: A Preliminary Study of Usability Factors, " *In Design, User Experience,*

and Usability. Theory, Methods, Tools and Practice, 2011, pp. 118-127; https://link.springer.com/chapter/10.1007/978-3-642-21708-1_14

14. M. Giraud, “TensorFlow: How to Freeze a Model and Serve It with a Python API”, 26 Nov 2016; https://blog.metaflow.fr/tensorflow-how-to-freeze-a-model-and-serve-it-with-a-python-API-d4f3596b3adc
15. E. Erwitt, Machine Learning WAVE Files with TensorFlow, 28 Nov, 2017; https://becominghuman.ai/machine-learning-wave-files-with-tensorflow-5a990385fb3e
16. B. Thumar, “Speech Recognition Using TensorFlow”, 10 Sep, 2017; http://androidkt.com/speech-recognition-using-tensorflow/
17. S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
18. R. Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. CoRR, abs/1411.2539, 2014.
19. Google. Awareness API, 2018; https://developers.google.com/awareness/

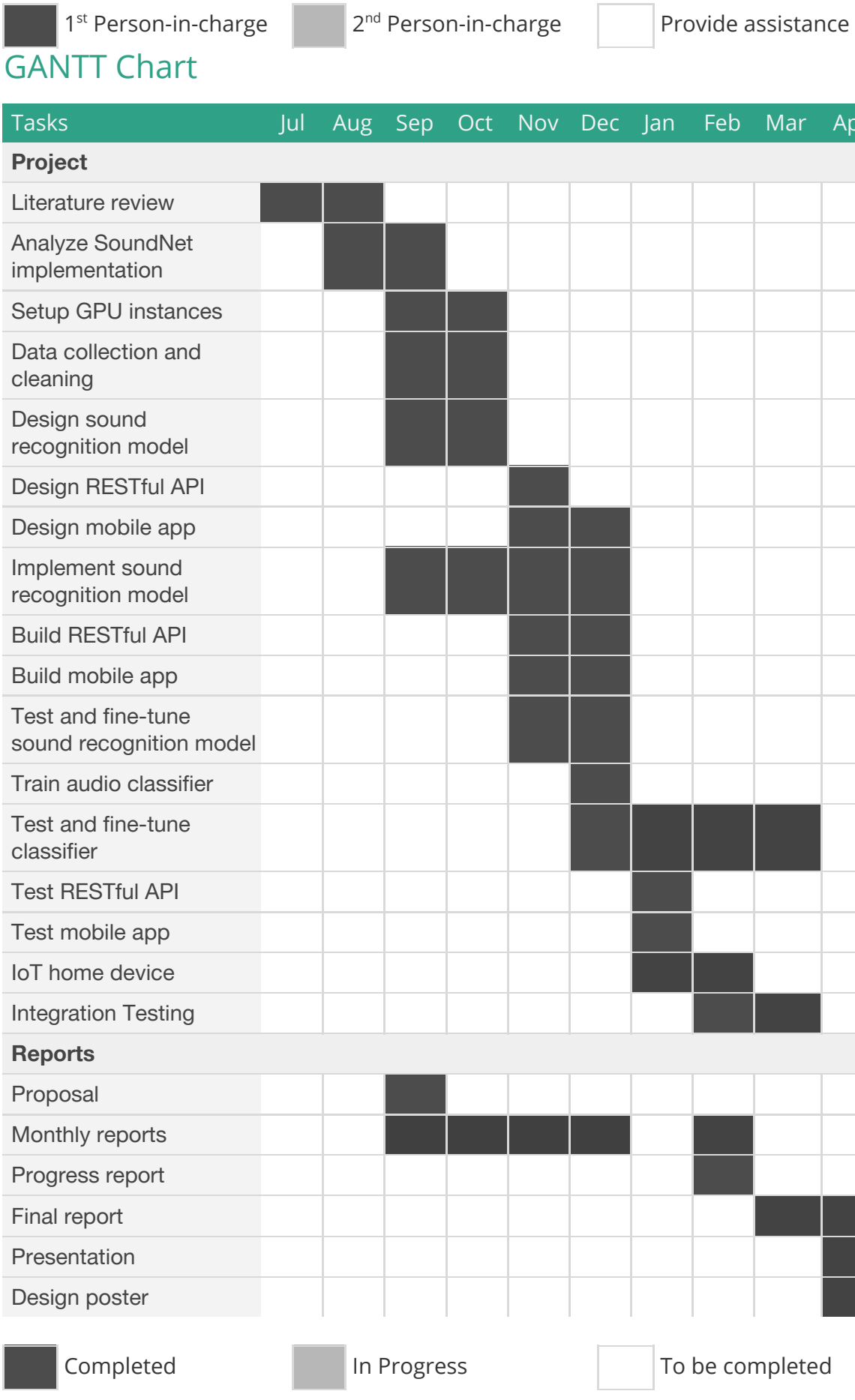
6 Appendices

A. Project Planning

Distribution of Work

The following chart shows the person-in-charge (PIC) of each task. By default, all other 3 members should provide assistance when PIC needs help.

Tasks	Sam	Billy	James	Martin
Project				
Literature review				
Analyze SoundNet implementation				
Setup GPU instances				
Data collection and cleaning				
Design sound recognition model				
Design RESTful API				
Design mobile app				
Implement sound recognition model				
Build RESTful API				
Build mobile app				
Test and fine-tune sound recognition model				
Train audio classifier				
Test and fine-tune classifier				
Test RESTful API				
Test mobile app				
IoT home device				
Integration Testing				
Reports				
Proposal				
Monthly reports				
Progress report				
Final report				
Presentation preparation				
Design poster				



B. GANTT Chart

Tasks	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr
Project										
Literature review										
Analyze SoundNet implementation										
Setup GPU instances										
Data collection and cleaning										
Design sound recognition model										
Design RESTful API										
Design mobile app										
Implement sound recognition model										
Build RESTful API										
Build mobile app										
Test and fine-tune sound recognition model										
Train audio classifier										
Test and fine-tune classifier										
Test RESTful API										
Test mobile app										
IoT home device										
Integration Testing										
Reports										
Proposal										
Monthly reports										
Progress report										
Final report										
Presentation										
Design poster										

C. Hardware and Software Requirements

Hardware Requirements

A. Server

Virtual machine specifications are listed below.

	Development	Production
Processor	1 vCPU	3 x 1 vCPU
Memory	3.75 GB RAM	3 x 3.75 GB RAM
Storage	60 GB SSD	120 GB SSD + Google Cloud Storage

B. Mobile Phone

Any Android smartphone with sound recording capability.

C. IoT Device

IoT Device Specification	
Platform	Raspberry Pi Zero W
Input	Microphone Array
Connectivity	Wifi / Bluetooth

Software Requirements

A. Server

Server Specification	
Operating System	Ubuntu ≥ 16.04 LTS
Languages	Python = 2.7
Machine Learning Framework	TensorFlow ≥ r1.6.0
Container Manager	Docker (Community Edition)
GPU Support Library	CUDA Toolkit 9.0 (cuDNN ≥ v5)

B. Mobile Application

Mobile Application Specification	
Operating System	≥ Android KitKat (4.4)
Languages	≥ Java 8 ≥ Kotlin 1.1.50
Development Kit	≥ JDK 1.8 ≥ Android SDK 26.0

C. IoT Device

IoT Device Specification	
Operating System	Raspbian Stretch with Desktop (Debian based)
Languages	Bash Python
Development Kit	Vim, VSCode text editor

D. Meeting Minutes

Minutes of the 1st Project Meeting

Place: Skype
Time: 1330 - 1500 23/04/2017
Attendee: Billy, Sam, Martin, James

1. Approval of minutes
   This was the first formal group meeting, so there were no minutes to approve.

2. Report on Progress
   All members have read through the guidelines on CSE department FYP page.

3. Discussion Items
   - 3.1. We picked sound as the area of focus for our proposed FYP topic and come up 3 ideas:
     - 3.1.1. Extracting feature and do a object recognition for a audio. We will train a machine learning model and build simple app / interface to record sound and show the composition of it.
     - 3.1.2. Autosuggestion for special terms in speech. it will be app that record audio, analysis the context, if there are some special terms, it will automatically suggest the information of it, like a smart AI to auto google the content for you.
     - 3.1.3. Real-time presentation slides generator based on speech content. It will summarize the speech for bullet points real time and display in the slides, showing the image of the context, also the slide may change due to the audience's action(like if the audiences clap hand, the slide would auto show thank you)
   - 3.2. Professor choice
     - 3.2.1. As sound is the main theme of our FYP, We will ask Prof. Brian Mak for advice and be our advisor first, as he had relative experience of speech recognition.
     - 3.2.2. Other professors that we might approach: Sung Kim, Andrew Honor and James kwok.
   - 3.3. Goals for the coming week
     - 3.3.1. Billy would draft an email for ask about the topics and send by Today (24-04-17), please help proofread.
     - 3.3.2. Submit our information and form a group in the FYP system.

4. Meeting adjournment and next meeting
   The Next meeting will be at 1300-1400 13/05/2017.

Minutes of the 2nd Project Meeting

Place: Skype
Time: 1300-1400 13/05/2017
Attendee: Billy, Sam, Martin, James

1. Approval of minutes
   The minutes of last meeting were approved without amendment.
2. Report on progress
   2.1. Meeting with Prof. Mak have been scheduled on 08/06/ 2017 at 1-2pm
3. Discussion Items
   3.1. Project Timeline:
   Jul - Aug: Environment setup, test SoundNet pretraining model
   End of Aug: Draft Proposal.
   3.2. The agenda for the meeting with Prof. Mak:
   - Ask Prof for what resources he can provide
   - Discuss application scenario of our 3 proposed ideas (e.g. whole-day tracking, home security).
4. Meeting adjournment and next meeting
   The next meeting will be at 1330 - 1400 08/06/2017

Minutes of the 3rd Project Meeting

Place: Skype
Time: 1330 - 1400 08/06/2017
Attendee: Billy, Sam, Martin, James, Prof. Mak

1. Approval of minutes
   The minutes of last meeting were approved without amendment.
2. Report on progress
   2.1. The team present the general idea of the 3 proposed topic and the research paper & relevant information that backup the idea.
3. Discussion items
   3.1. Feedbacks on the 3 proposed topic were given (e.g. pros & cons, feasibility, application scenario)
   3.2. The team conclude that Topic 1 seems more preferable
   3.3. Prof. Mak suggest we can explore the possibility of integrating the technology in smart home device (e.g. Google Home, Amazon Echo).
   3.4. Regarding resources available, CSE department and his own research group have GPU farms he can provide
   3.5. We also discuss possibility of apply Google's TensorFlow Research Cloud
4. Goals for the coming week
   4.1. Come up with detailed timeline
   4.2. Come up with Division of labor
   4.3. Implementation result of replicating the SoundNet paper **
   4.4. Come up with solid application scenario (so that he can assess what we have done)
   4.5. Look into google home / echo API if we're going down that direction
5. Meeting adjournment and next meeting
   The next meeting will be at 1600-2000 16/07/2017

Minutes of the 4th Project Meeting

Date: 1330-1500 04/09/2017
Place: IPO Common
Attendee: Billy, Sam, Martin, James

1. Approval of minutes
   The minutes of last meeting were approved without amendment.
2. Report on progress
   - 2.1. Testing of the current SoundNet source code
   - 2.2. Write TensorFlow Prediction
3. Discussion items
   - 3.1. Objective of FYP
     - 3.1.1. Enable the possibility the of using sound,
     - 3.1.2. Sound to tag, tag to action
   - 3.2. Scope of the FYP
     - 3.2.1. Machine Learning core (implement Soundnet in TensorFlow)
     - 3.2.2. API access of the Machine Learning
     - 3.2.3. Mobile App (Android only)
     - 3.2.4. IoT (if time allow)
   - 3.3. Whole year schedule of the FYP
     - 3.3.1. Sept - Testing for lua code(Sam), Understand lua and TensorFlow code , write prediction in TensorFlow, Audio - label Pair, Proposal
     - 3.3.2. Oct - Prepare data set (Training set, Validation Test, Test set), Complete existing TensorFlow version of SoundNet (with small set of data), Tensorboard
     - 3.3.3. Nov - Finetune model, Build an API access toward the model
     - 3.3.4. Dec - Explore external API integrate, App Design, Prototyping
     - 3.3.5. Jan - App Development, Backend sound preprocess
     - 3.3.6. Feb - Bridge App and Backend
     - 3.3.7. March - Testing, Evaluation and Debug, Poster, Report
   - 3.4. Division of labour this month
     - 3.4.1. Sam and Billy will familiarize themselves with Torch code, James and Martin will familiarize themselves with TensorFlow code
   - 3.5. To discuss about the workflow of the team
     - 3.5.1. Source code
     - 3.5.2. Document management
4. Goals for the coming week
   - 4.1. Continue Testing the SoundNet source code on Torch and TensorFlow
   - 4.2. Set up GPU instance for training the model.
   - 4.3. Meeting with Prof. Mak scheduled on 7/9/2017 for progress update.

5. Meeting adjournment and next meeting
   The next meeting will be at 1645-1730 09/07/2017

Minutes of the 5th Project Meeting

Date: 1645-1730 09/07/2017

Place: Prof. Mak's office

Present: Billy, Sam, Martin, James, Prof. Mak

1. Approval of minutes
   The minutes of last meeting were approved without amendment.
2. Report on progress
   - 2.1. The team present the finalized idea and a detailed timeline by every month
3. Discussed items
   - 3.1. Idea of the application
     - 3.1.1. It would be like a IFTTT app by using background sound
     - 3.1.2. Sound will trigger different action (e.g. concert would auto reply email)
   - 3.2. Scope of the FYP
     - 3.2.1. Machine Learning core (implement Soundnet / Related Paper in TensorFlow)
     - 3.2.2. API access of the Machine Learning
     - 3.2.3. Mobile App (Android only)
   - 3.3. To talk about current findings
     - 3.3.1. Soundnet paper
     - 3.3.2. Look, Listen and Learn by Deepmind
   - 3.4. Different resource we can use
     - 3.4.1. computer / vm with CUDA
     - 3.4.2. Storage limit
     - 3.4.3. TFRC
4. Goals for coming week
   - 4.1. Make sure training data contain scenario we want to classify
   - 4.2. Apply google TPU
   - 4.3. Apply CSE UG cloud
   - 4.4. Apply AWS cloud for research (if exist)
5. Meeting adjournment and next meeting
   The next meeting will be at 1330-1500 12/09/2017

Minutes of the 6th Project Meeting

Time: 1330 - 1500 12/09/2017

Place: IPO Common
Attendee: Billy, Sam, Martin, James

1. Approval of minutes
   The minutes of last meeting were approved without amendment.
2. Report of progress and discuss question related to it
   - 2.1. Ask MIT: Mp3 label pair, Standard env of the github code
   - 2.2. Ask Soundnet - TensorFlow: What is the output of the layer mean
3. Discussed items
   - 3.1. Date to meet communication tutor
   - 3.2. Name of the product and conclude to call it Zampify
   - 3.3. FYP workshop details
   - 3.4. Possibilities of applying funding
4. Goals for coming week
   - 4.1. Setup Trello for task management
   - 4.2. Look for details and deadline for applying funding and discuss next time
   - 4.3. Send email to soundNet TensorFlow implementation author
   - 4.4. Send email to MIT authors
5. Meeting adjournment and next meeting
   The next meeting will be at 1330-1500 19/09/2017

Minutes of the 7th Project Meeting

Time: 1330 - 1500 21/09/2017

Place: IPO Common
Attendee: Billy, Sam, Martin, James

1. Approval of minutes
   The minutes of last meeting were approved without amendment.
2. Report of progress and discuss question related to it
   - 2.1. To report the progress
     - 2.1.1. Billy have already installed and test the original Soundnet code
     - 2.1.2. After testing the source code, billy discovered that it is not very accurate as we want.
     - 2.1.3. The owner of Soundnet - TensorFlow said the final layer should output a probability vector after softmax layer
     - 2.1.4. Develop classifier classifers to enhance the accuracy
3. Discussed items
   - 3.1. Logistics
     a. Monthly report should be written before next meeting with Prof Mak
4. Goals for coming week
   - 4.1. Add a softmax layer to SoundNet TensorFlow and test the accuracy
   - 4.2. Containerize the SoundNet TensorFlow code for development purpose
5. Meeting adjournment and next meeting
   The next meeting will be at 1330-1500 26/09/2017

Minutes of the 8th Project Meeting

Time: 1330 - 1500 26/09/2017

Place: IPO Common
Attendee: Billy, Sam, Martin, James

1. Approval of minutes
   The minutes of last meeting were approved without amendment.
2. Report of progress and discuss question related to it
   2.1. James have already finished writing the extract_prediction.py
   2.2. Billy have containerize the SoundNet-TensorFlow code
3. Discussed items
   3.1. Sam would in charge of downloading AudioSet data
   3.2. Billy would develop KNN and MLP classifier
   3.3. Martin would develop XGBoost classifier
   3.4. Jams would develop SVM and RF classifier
4. Goals for coming week
   4.1. Dataset with 10 labels from AudioSet would be downloaded
   4.2. All classifier working code will be developed.
5. Meeting adjournment and next meeting
   The next meeting will be at 1330-1500 12/10/2017

Minutes of the 9th Project Meeting

Date: 1645-1730 12/10/2017

Place: Prof. Mak's office

Present: Billy, Sam, Martin, James, Prof. Mak

1. Approval of minutes
   The minutes of last meeting were approved without amendment.
2. Report on progress
   - 2.1. All classifiers mentioned in the previous meeting have
   - 2.2. 10 labels from Audiost have collected for the purpose of training classifier
3. Discussed items
   - 3.1. Sequential Problem
     - 3.1.1. Professor suggest that we can try to revert the audio clip inorder to get more features of the sound in order to avoid misclassification
   - 3.2. Merging Problem
     - 3.2.1. More data needed to use for training the classifier in order to solve the sound which have multiple labels
4. Goals for coming week
   - 4.1. To use esc 50 dataset to test the existing classifier and try to update the weight with the sound
5. Meeting adjournment and next meeting
   The next meeting will be at 1330-1500 31/10/2017

Minutes of the 10th Project Meeting

Time: 1330 - 1500 31/10/2017

Place: IPO Common
Attendee: Billy, Sam, Martin, James

1. Approval of minutes
   The minutes of last meeting were approved without amendment.
2. Report of progress and discuss question related to it
   2.1. Sam have prepared the audio feature and the info would be in json
   2.2. James have tuned RF to have around 70% of accuracy
   2.3. Martin said Esc-50 dataset is too large and push floyd-hub for testing
3. Discussed items
   3.1. Billy suggest a no pass filter layer maybe needed
   3.2. A utility function needed to implement to get features and tag for traine
4. Goals for coming week
   4.1. Develop the utility function
   4.2. Finetune other classifier
5. Meeting adjournment and next meeting
   The next meeting will be at 1330-1500 28/11/2017

Minutes of the 11th Project Meeting

Time: 1330 - 1500 28/11/2017

Place: IPO Common
Attendee: Billy, Sam, Martin, James

1. Approval of minutes
   The minutes of last meeting were approved without amendment.
2. Report of progress and discuss question related to it
   2.1. The utility function has been developed but not fit into current classifier.
3. Discussed items
   3.1. The utility function for reading the audio needed to be refactor as for majority vote.
   3.2. A crawler machine needed to setup for batch process the youtube video clip.
4. Goals for coming week
   4.1. Modify the utility function to fit in the classifier.
   4.2. Set up the crawler in cloud
5. Meeting adjournment and next meeting
   The next meeting will be at 1330-1500 21/12/2017

Minutes of the 12th Project Meeting

Date: 1645-1730 21/12/2017

Place: Prof. Mak's office

Present: Billy, Sam, Martin, James, Prof. Mak

1. Approval of minutes
   The minutes of last meeting were approved without amendment.
2. Report on progress
   - 2.1. Handed in the monthly report of November
   - 2.2. Reported the graph of SVM representation
   - 2.3. Reported will explore the possibility of using tensorflow-lite in android
3. Discussed items
   - 3.1. The difficulty of using tensorflow-lite in android
     - 3.1.1. Convert audio into representation by the use of tensorflow ffmpeg
     - 3.1.2. Freezing the model and enable it on the side of android
4. Goals for coming week
   - 4.1. Choose 5 scenario to train for detection
   - 4.2. Finetune the classifier with multiple labels
5. Meeting adjournment and next meeting
   The next meeting will be at 1330-1500 17/01/2017

Minutes of the 13th Project Meeting

Time: 1330 - 1500 19/1/2018

Place: IPO Common
Attendee: Billy, Sam, Martin, James

1. Approval of minutes
   The minutes of last meeting were approved without amendment.
2. Report of progress and discuss question related to it
   - 2.1. The classifier have F-score of 80 %
3. Discussed items
   - 3.1. The mobile app UI
   - 3.2. The scenario to trigger, including:
     - 3.2.1. Clapping
     - 3.2.2. Finger Tapping
     - 3.2.3. Speech
     - 3.2.4. Raining
     - 3.2.5. Baby Cry
4. Goals for coming week
   - 4.1. Completion of API for prediction
   - 4.2. Completion of mobile application connecting the API
5. Meeting adjournment and next meeting
   The next meeting will be at 1330-1500 6/2/2018

Minutes of the 14th Project Meeting

Time: 1330 - 1500 6/2/2018

Place: IPO Common
Attendee: Billy, Sam, Martin, James

1. Approval of minutes
   The minutes of last meeting were approved without amendment.
2. Report of progress and discuss question related to it
   - 2.1. The mobile application (end to end) demo is completed, but the response time of the server is slow
3. Discussed items
   - 3.1. Ways to improve the speed of prediction
     - 3.1.1. Initializing the SoundNet and classifier at same time
     - 3.1.2. Reduce redundant file IO
     - 3.1.3. On phone feature extraction
4. Goals for coming week
   - 4.1. Minimize the response time to 5 seconds
5. Meeting adjournment and next meeting
   The next meeting will be at 1330-1500 20/02/2018

Minutes of the 15th Project Meeting

Time: 1330 - 1500 5/3/2018

Place: IPO Common
Attendee: Billy, Sam, Martin, James

1. Approval of minutes
   The minutes of last meeting were approved without amendment.
2. Report of progress and discuss question related to it
   5.1. Start User System
   5.2. Demo android application

3. Discussed items
   5.3. Ways to improve the speed of prediction
   5.4. Improve of android application
   5.5. Trails of hardware, IoT
   5.6. Improve model
   5.7. Study IFTTT

4. Meeting adjournment and next meeting
   The next meeting will be at 1330-1500 21/3/2018

Minutes of the 16th Project Meeting

Time: 1330 - 1500 12/4/2018

Place: IPO Common
Attendee: Billy, Sam, Martin, James

3. Approval of minutes
   The minutes of last meeting were approved without amendment.
4. Report of progress and discuss question related to it
   5.8. LSTM classfier
   5.9. Demo android application

3. Discussed items
   5.10. IFTTT demo
   5.11. Improve of android application
   5.12. Improve model
   5.13. Hardware API call
   5.14. Final Report writing

4. Meeting adjournment and next meeting
   The next meeting will be at 1330-1500 19/4/2018