Keylogging Using Acoustic Signature

Group 9

Group Members

Zhe Xu, JianWen Su, JingTian Qiu, Sheng Zhang

Abstract

Input devices, such as laptops with built-in keyboards, produce sounds with certain characteristics when tapped by the user due to subtle differences in their internal mechanics. Our work shows that when the relative distance between the input device and the pickup device is determined, and the text is entered by the same person using a uniform keyboard, the text content of the user's input can be predicted more accurately by a trained Machine Learning (ML) model from the keyboard tapping sound if the effect of ambient noise is ignored. The project is an exercise in information attack by leveraging the physical properties of the event itself. The input content at the information level is revealed by collecting its objective features, such as sound side channels. Our attack model consists of audio signal processing and ML algorithms that effectively improve the accuracy of object reconstruction. The basic process includes audio acquisition, audio segmentation, wave alignment, spectrogram generation, noise sample filtering, and model training. The main idea is to convert the audio signal into a log spectrogram, and then use convolutional neural networks to identify data samples with different labels (corresponding to different characters). The method significantly reduces the complexity of the problem and achieves better predictions by converting audio information with higher dimensions (time domain, frequency domain, and amplitude) into a two-dimensional planar graph and achieves better prediction results.

Keywords: Keylogging, Acoustic Signature, Spectrogram, ML

Introduction

This section focused on the motivation, background, and problem statement. Then approaches applied and expected results would be mentioned. Finally, how the project relates to the course (ECE647), and the organization of this passage would be properly presented.

Motivation, Background, and Problem Statement

After attending the ECE647 course, we have found that information security is threatened by all kinds of attacks including both main and side channels.

Motivation & Background

The importance of information security in the Internet era has long been self-evident. Since personal computers, mobile smart devices, and other infrastructures have become popular in daily life, there have been numerous incidents of information theft for unlawful gain, with a variety of methods. As the average user becomes more aware of information security, conventional attacks such as keyloggers have long been abandoned by crafty miscreants in favor of more insidious and difficult-to-guard methods. For example, Abe Davis mentioned in his March 2016 TED talk about an eavesdropping device implemented with a high-precision camera to capture object vibrations, which can complete the eavesdropping before the target of the attack is aware of it.

Problem Statement, including Approaches, Expected Results, and Connection with ECE647

Based on the above background, this paper aims to explore a keylogger that is more easily ignored by the victim and can significantly reduce the cost of the attack. Using this tool, an attacker can obtain discrete characters (e.g., passwords) entered by the user by simply using the built-in microphone (or other kinds of pickup devices) and embedded keyboard widely available in modern laptops. According to the course, sound as a "side-channel information" has the quality of being more imperceptible. So, predicting the input text based on the keyboard tapping sound might lead to a valuable experiment. The basic principle of this method is to find the connection between the sound made by the user tapping the keyboard and the input content, and subsequently to predict the input content based on the keyboard tapping sound. Due to the differences in keystroke sounds caused by the internal mechanics of the keyboard, the relative position of the computer microphone to the keyboard, etc., there may be many limitations in the practical application of this method from objective factors. As an exploration of the initial stage of the method, this paper will focus more on the feasibility of the method itself, rather than on the realization of a practical product with high usability and high interference immunity.

Organization of the report

In this report, some Bibliographical entries, and pdfs (or links) would be first introduced. Then we would explain the Methodologies and Tools we have used. Next, the Partitioning of Work and Schedule would be presented. Finally, we will show our Experiments and the Results.

Related Work

1. Don’t Skype & Type! Acoustic Eavesdropping in Voice-Over-IP

This paper describes the related work in the field of keyboard input eavesdropping so that we can quickly understand the background of the project. Also, this paper mentions one method to build the data-collecting function model to collect sounds made by a specific type of keyboard. We can improve this method and apply it to the project.

2. Keyboard Acoustic Emanations Revisited

This article has made experiments on the different sounds of different keys on the keyboard and verified that the idea is feasible. At the same time, it provides a reference for the data collection format, cleaning method, and model selection and construction. In addition, factors that may affect the accuracy of the results are analyzed, such as keyboard model, keyboard usage time, experimental environment noise, typing personnel, etc.

3. Fine-tuning deep convolutional neural networks for distinguishing illustrations from photographs

When crawling the network for images, systems for aggregating illustrations require a method for automatically identifying illustrations from photographs. The accuracy of a previous attempt to create this functionality by defining fundamental features that were deemed useful for classification was only around 58 percent. Deep neural networks, on the other hand, had fared well in computer vision tasks, and convolutional neural networks (CNNs) had done well at extracting such useful image information automatically.

4. The ML Approach for Analysis of Sound Scenes and Events

The Machine
Learning Approac

This article provides a more detailed and complete description of the process and methods for processing sound data using ML techniques. Its ideas can help build the functional modules of data acquisition, processing, and ML model training for this project, which is of high guiding significance for this project.

5. Anomalous sound event detection: A survey of ML-based methods and applications

In this paper, a comprehensive survey of anomalous sound event detection is presented, covering various aspects of the topic, i.e., feature extraction methods, datasets, evaluation metrics, methods, applications, which can help us to select, process sound data, and give us a reference for evaluating sample methods.

6. Hearing your touch: A new acoustic side-channel on smartphones

The paper steps further into acoustic side-channel attacks. Their targets are expanded to devices containing touch screens. While fingers type on the virtual keyboard, audios of tapping on glass could be used to predict user input as well. It is an interesting way of exploring.

Methodology and Tools

In this section, the threat model, libraries, and tools built by them such as the audio splitter, noise filter, and how the ML model is designed would be introduced. Meanwhile, the predictions of user input are quite sensitive to the environment noise and lots of other objective factors such as keyboard model, keyboard typing habits, etc. Thus, data collection has become a challenge for the project. It is hard to find a given dataset to cover all the possible samples. So, we are going to keep collecting data and re-train the ML model for each attack (the exact process will be shown in the threat model to ensure the logical self-consistency of the project).

Threaten Model

We would explain the relationship between Attacker, Victim, Web Interface, and Application Server.

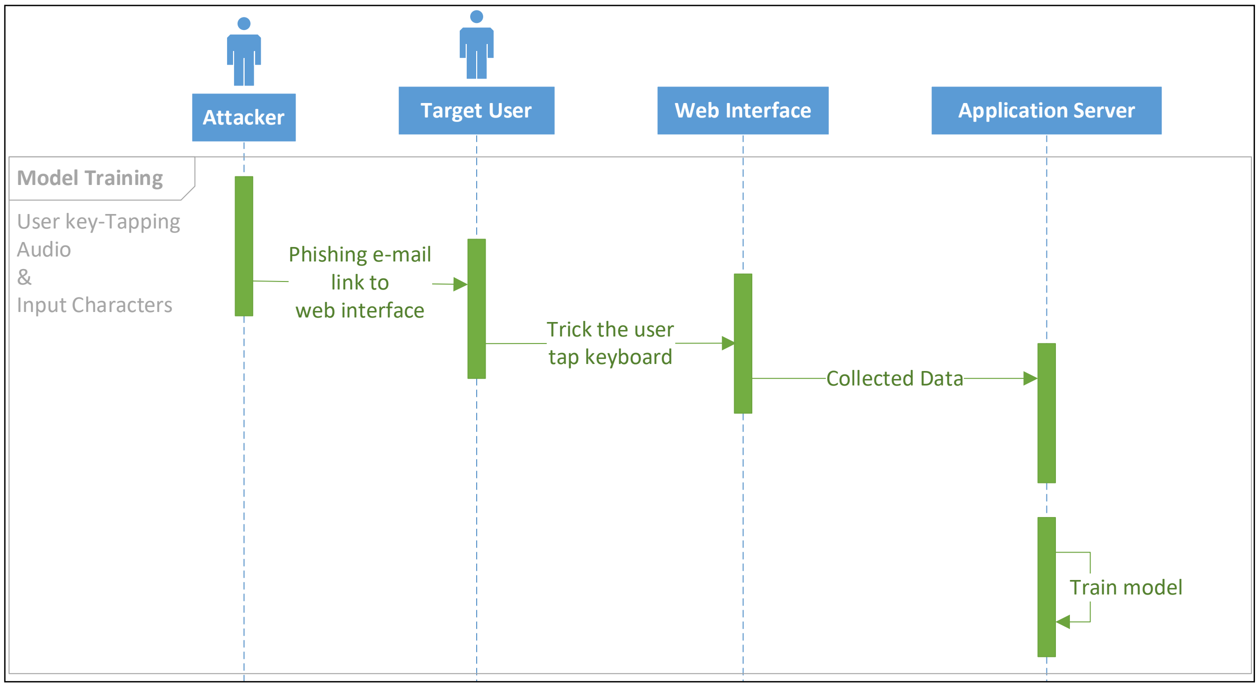

Attacker
Target User
Web Interface
Application Server
Model Training
User key-Tapping Audio & Input Characters
Phishing e-mail link to web interface
Trick the user tap keyboard
Collected Data
Train model

Figure 1, steps of data-collection and ML model training

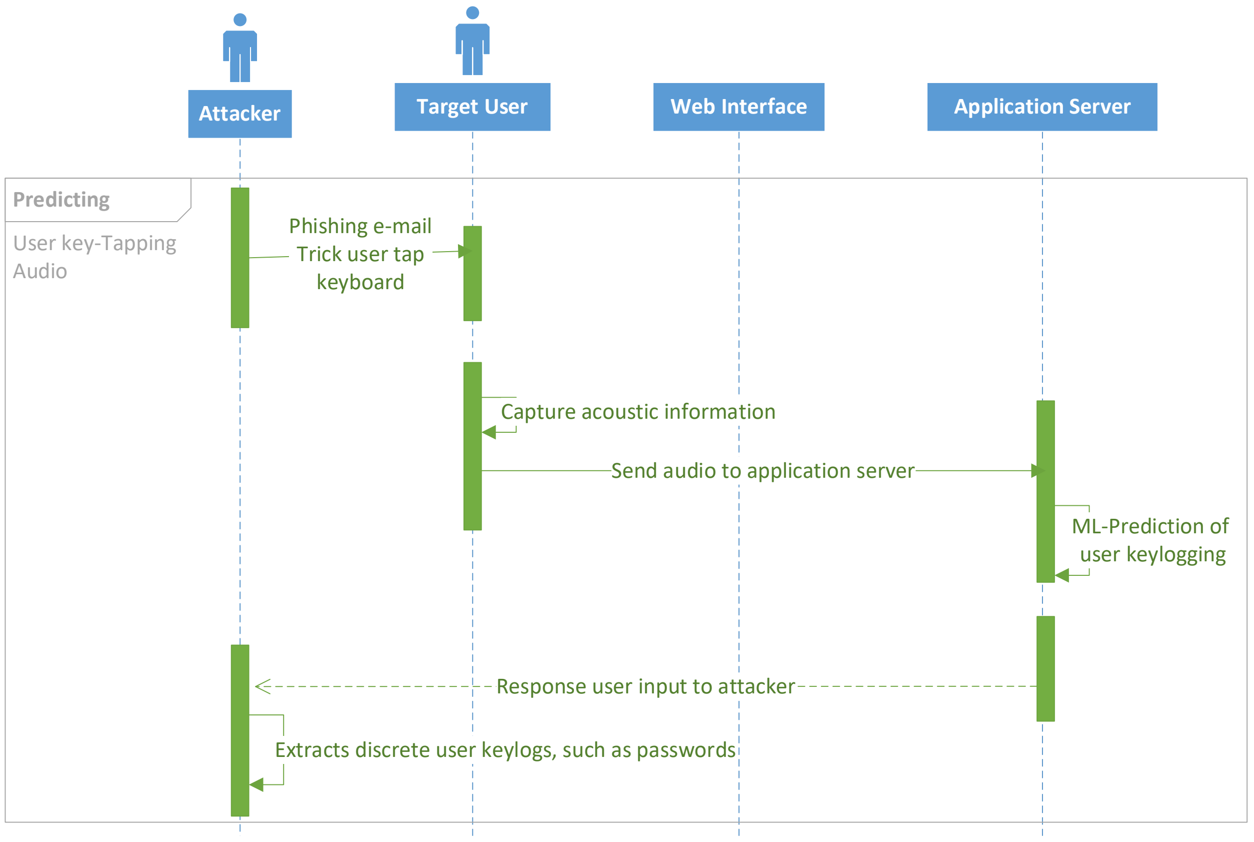

Attacker
Target User
Web Interface
Application Server
Predicting
User key-Tapping Audio
Phishing e-mail Trick user tap keyboard
Capture acoustic information
Send audio to application server
ML-Prediction of user keylogging
Response user input to attacker
Extracts discrete user keylogs, such as passwords

Figure 2, user key-tapping prediction

Description

The threat model is divided into two parts.

First, the attacker entices the user to access the Web Interface. The character's input and the corresponding audio of key tapping would be recorded. At the same time, the Web Interface stores the time stamp for each typing as the splitting basis. For the last step, the Application Server process the data passed by Web Interface and load it as the training set to feed into the ML model.

Next, the attacker should lead the user to expose their acoustic information for later prediction. And after receiving the audio data, the Application Server should accept them as the test set for the ML model to make the speculation. In the end, the attacker might disclose the discrete content based on what the Application Server responded to.

Libraries and Tools

To achieve what was planned in the threaten model, we need to build several tools by using code libraries from the python community. First, a brief introduction of some of the most important libraries and frameworks would be presented.

Librosa

Librosa is a library in python that offers useful functions to process audio signals. In this project, it is used for generating spectrograms from audios. To install it, check if 'ffmpeg' is available then use the 'pip' command shown below:

Code block 1, install Libros

```
pip install Libros
```

There are three important methods used in the program to load an audio file and generate wave gram, and spectrogram, respectively.

Code block 1, key methods

```
# load audio file, sr represent for sampling rate
y, sr = librosa.Load(audioFilePath)

# show wave gram
Librosa.display.waves how(y, sr)

# show spectrogram
melspec = librosa.feature.melspectrogram(y, sr, n_fft=1024, hop_length=512,
n_mels=128)
logmelspec=librosa.power_to_db(melspec)
librosa.display.specshow(logmelspec)
```

PyTorch

PyTorch is an open-source ML framework that offers various kinds of structures and functions to allow developers to build and use their ML models and run the code below to install.

Code block 3, install PyTorch

```
pip install torch torchvision
```

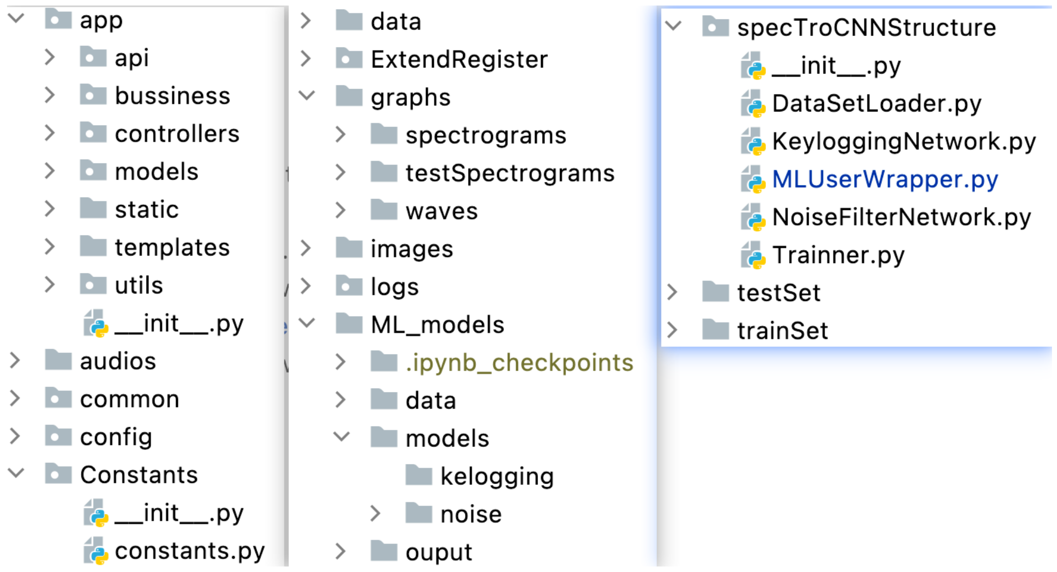
We built the ML model through PyTorch, and some of the most important methods are shown below.

Code block 4, key functions to build ML model

```
# inherit class 'Dataset' to define logics to load datasets
class DataSetLoader(Dataset):

# inherit class 'nn.Module' to define own ML network structure
class KeyloggingNetwork(nn.Module):

# use Adam as the optimizer
optimizer=torch.optim.Adam(self.net.parameters(),lr=self.LR,weight_decay=sel
f.penalization)

# use CrossEntropy as a loss function
loss_func = nn.CrossEntropyLoss()
```

Flask

Flask allows developers to build an application server easily. In this project, it is used to hold all functions including dataset making, ML model training, label predicting, etc.



```
app
  api
  bussiness
  controllers
  models
  static
  templates
  utils
  __init__.py
audios
common
config
Constants
  __init__.py
  constants.py
```

```
data
ExtendRegister
graphs
  spectrograms
  testSpectrograms
  waves
images
logs
ML_models
  .ipynb_checkpoints
  data
  models
    kelogging
    noise
  ouput
```

```
specTroCNNStructure
  __init__.py
  DataSetLoader.py
  KeyloggingNetwork.py
  MLUserWrapper.py
  NoiseFilterNetwork.py
  Trainner.py
testSet
trainSet
```

Figure 3, the directory structure of the Flask-based application server

Datasets

The audio data (sound of key tapping) could be easily affected by reasons listed below,

1. Environment noise.
2. The mechanical structure of a specific model of keyboard.
3. Different specifications of sound pickup equipment made by different manufacturers.
4. Distance between keyboard and microphone.
5. The keystroke habits of the person hitting the keyboard, such as the fingers corresponding to the keys and the angle and force of the keystrokes.
6. Etc.

In summary, generating an interferential immunity requires humongous samples. But unfortunately, we did not find any dataset that meet the need. Thus, we will perform data acquisition when the user, the keyboard used, the microphone model, the relative position of the keyboard to the microphone, and the ambient noise level are all certain.

Meanwhile, to complete a minimum achievable model as complete as possible in a limited time to verify the effectiveness of the method, we limit the value of the label to the range of twenty-six lowercase English letters.

Samples would be stored as spectrograms and labeled by file name. The example is shown below.

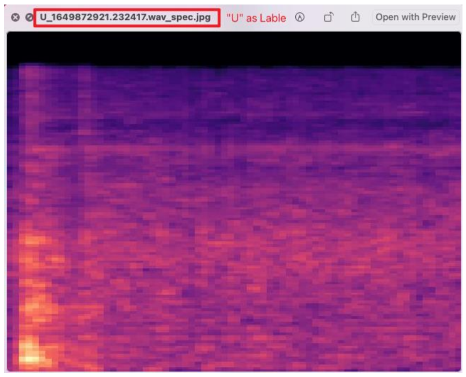


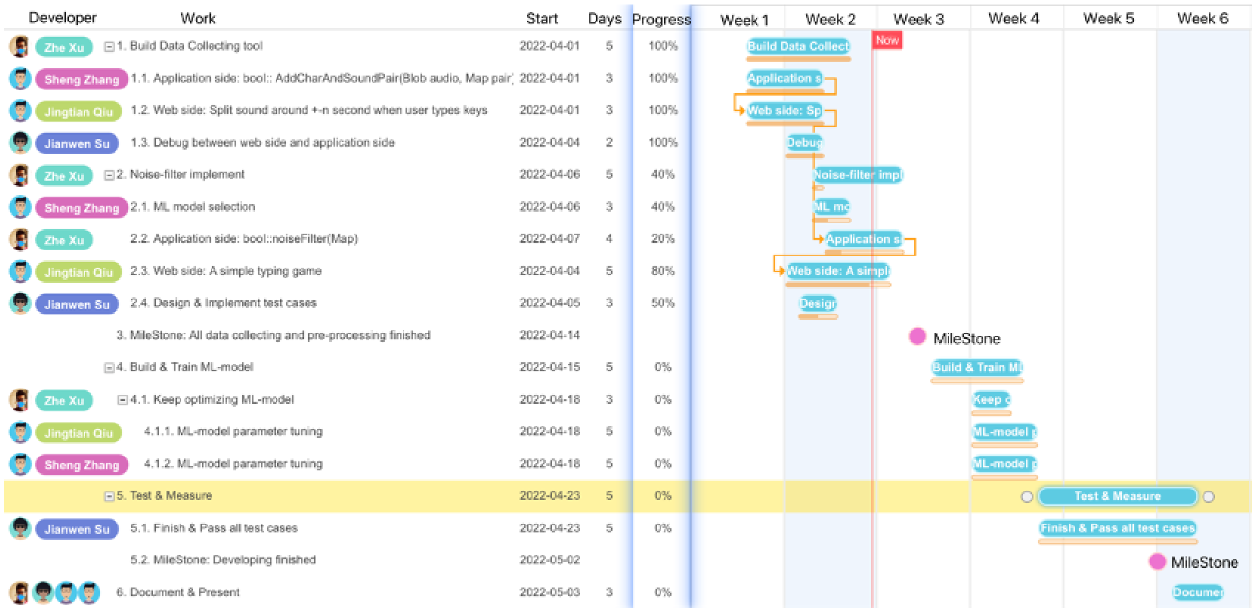
Figure 4, an example of sample "U"

Partitioning of Work and Schedule

This section would emphasize the division of labor and progress schedule within the group and work plan.

Gantt Chart

The schedule is shown in the Gantt chart below:



Developer	Work	Start	Days	Progress
Zhe Xu	1. Build Data Collecting tool	2022-04-01	5	100%
Sheng Zhang	1.1. Application side: bool:: AddCharAndSoundPair(Blob audio, Map pair)	2022-04-01	3	100%
Jingtian Qiu	1.2. Web side: Split sound around +-n second when user types keys	2022-04-01	3	100%
Jianwen Su	1.3. Debug between web side and application side	2022-04-04	2	100%
Zhe Xu	2. Noise-filter implement	2022-04-06	5	40%
Sheng Zhang	2.1. ML model selection	2022-04-06	3	40%
Zhe Xu	2.2. Application side: bool::noiseFilter(Map)	2022-04-07	4	20%
Jingtian Qiu	2.3. Web side: A simple typing game	2022-04-04	5	80%
Jianwen Su	2.4. Design & Implement test cases	2022-04-05	3	50%
	3. MileStone: All data collecting and pre-processing finished	2022-04-14		
	4. Build & Train ML-model	2022-04-15	5	0%
Zhe Xu	4.1. Keep optimizing ML-model	2022-04-18	3	0%
Jingtian Qiu	4.1.1. ML-model parameter tuning	2022-04-18	5	0%
Sheng Zhang	4.1.2. ML-model parameter tuning	2022-04-18	5	0%
	5. Test & Measure	2022-04-23	5	0%
Jianwen Su	5.1. Finish & Pass all test cases	2022-04-23	5	0%
	5.2. MileStone: Developing finished	2022-05-02		
	6. Document & Present	2022-05-03	3	0%

Figure 5, work schedule Gantt chart

Work Division

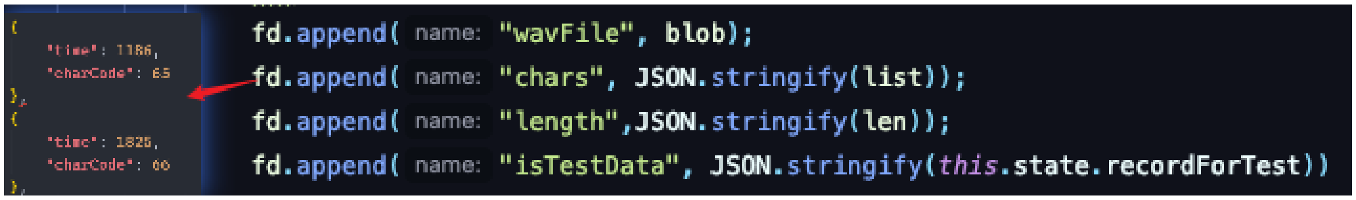
- Zhe Xu: Application Server building and functions embedding. Noise-Filter building. Keylogging ML model building, training, and tunning. Work plane making.
- JianWen Su: Functional testing. Keylogging ML model building, training.
- JingTian Qiu: Web Interface building. Keylogging ML model training, and tunning.
- Sheng Zhang: Sample pre-processing. Keylogging ML model training, and tunning. Additional theoretical support.

Experiments and Results

A complete introduction to this project would be presented here following the order of development including how whole audio is spliced into parts, steps to train the noise filter and keylogging ML model, and flask application to host and organize all the functions. The demonstration video could be found HERE.

Audio splitting

In the first stage of the threat model mentioned earlier, the user will first access the Web Interface and leave their tapping audio logging. The Web Interface would also record the time stamp and characters for each key stroking as the basis for slicing. The request message structure is shown in Figure 6.



```
{
    'time': 1186,
    'charCode': 65
},
{
    'time': 1826,
    'charCode': 66
},
```

```
fd.append( name: "wavFile", blob);
fd.append( name: "chars", JSON.stringify(list));
fd.append( name: "length",JSON.stringify(len));
fd.append( name: "isTestData", JSON.stringify(this.state.recordForTest))
```

Figure 6, Structure of request

Then, API "audioCharPairsAPI" would take the data and pass them to method "SampleHandler.splitSound" to split into audio slices for the next steps later. A simple descriptive diagram of the process is as follows (Figure 8).

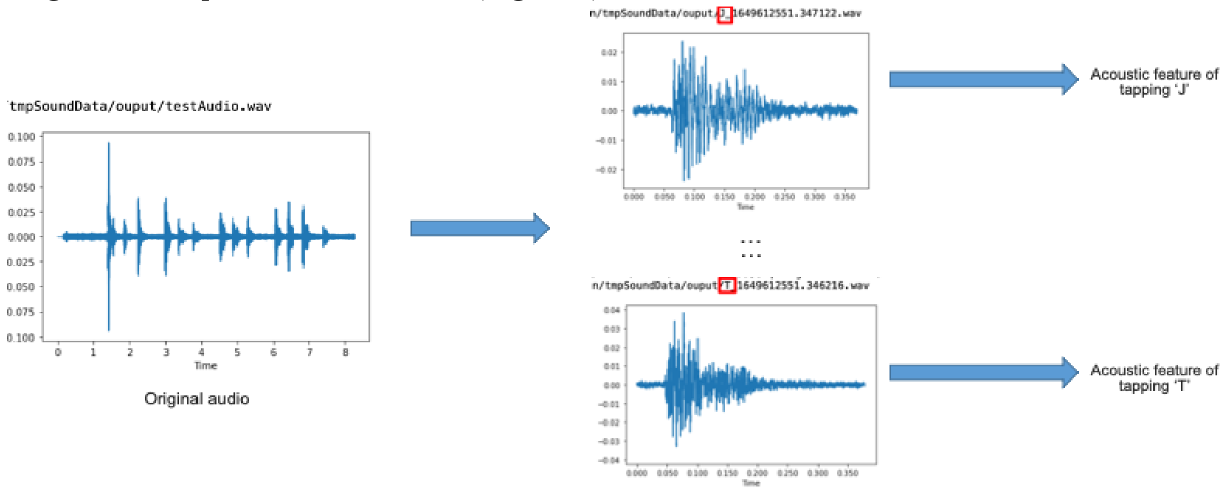

tmpSoundData/ouput/testAudio.wav
Time
Original audio
n/tmpSoundData/ouput/J_1649612551.347122.wav
Time
Acoustic feature of tapping 'J'
n/tmpSoundData/ouput/T_1649612551.346216.wav
Time
Acoustic feature of tapping 'T'

Figure 8, Generating audio slice

After that, the method “SampleHandler.generateGramFromSplitedAudio” would load audio slices to librosa and find the push peak for them.

According to “*keyboard Acoustic Emanations Revisited”* (Zhuang, L., 2009), a key tapping sound should contain two peaks (Push Peak is significantly larger than Release Peak) as Figure 8 shows.

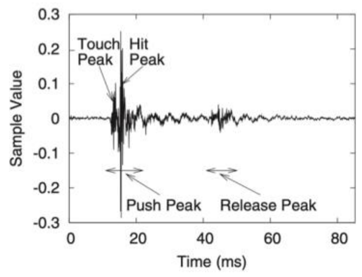


Figure 8, quoted from “*keyboard Acoustic Emanations Revisited”*

Zhuang also claims that typically every tapping sample would last about one hundred milliseconds and the gap between tapings is always larger than one hundred milliseconds. Thus, to align the samples by Push Peak on the time axis, it is necessary to find the peak with the largest amplitude and take the middle one hundred milliseconds. While we observed about five hundred samples, we decided to take seventy-five milliseconds before the Push Peak and one hundred after (shown in Figure 9). It helps us to capture a complete wave without irrelevant items.

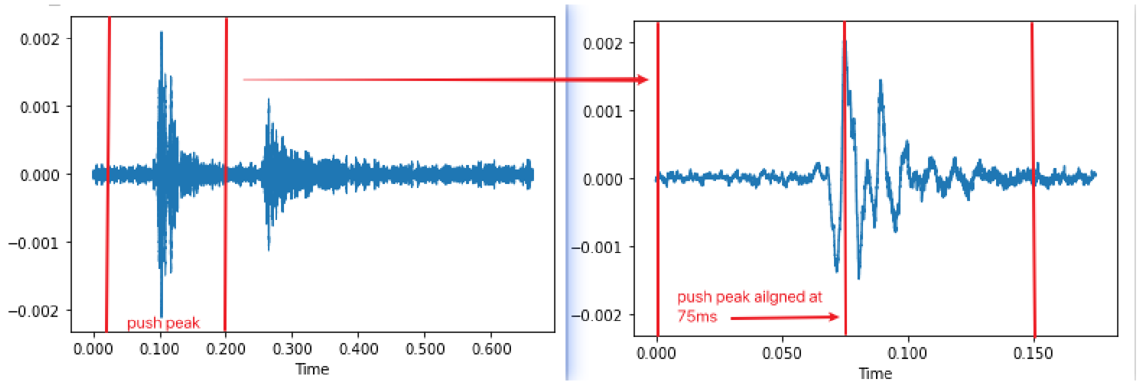


Figure 9, Align push peak

Key code is shown as code block 5,

Code block 5, the key code of aligning push peak

```
y=ys[int(maxStart)-int(miliSecondLength*75):int(maxStart)+int(miliSecondLength*100)]
```

Noise Filter & Keylogging ML model

In the last step, audios have been converted to figures. So, the acoustic problem now becomes a vision problem. Both noise filter and keylogging prediction are trying to classify the graphs. Thus, Convolutional Neural Network (CNN) might be a desirable choice.

The noise filter is a binary classifier (labeled as “is noise” or “not noise”) so the structure of CNN could stay simple. While the keylogging ML model might contain twenty-six labels at most the neural network should be more complex.

CNN Structure Experiment

For the noise filter, the first try reaches average accuracy of 99.35483870967742% in 5 folds cross-validation on the dataset. The structure shows in Figure 10.

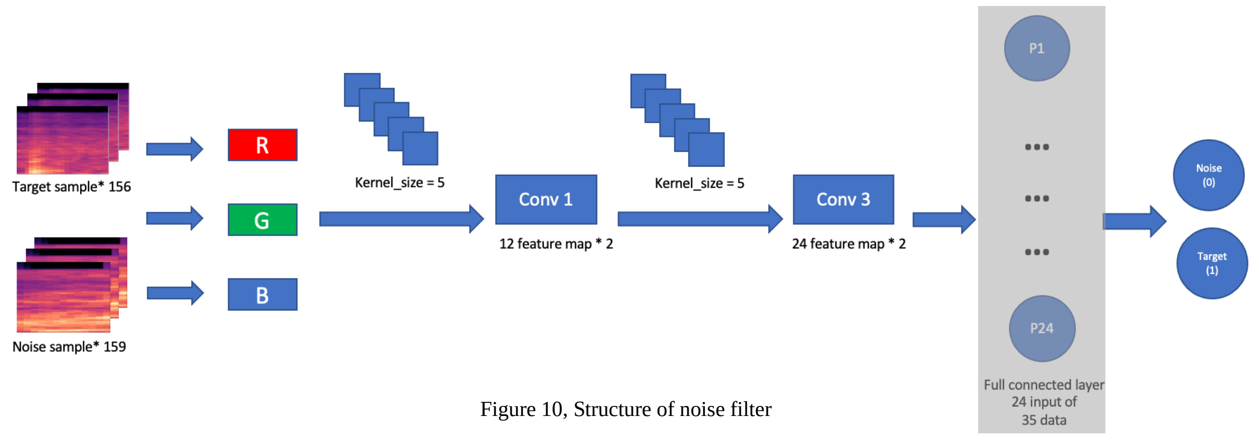
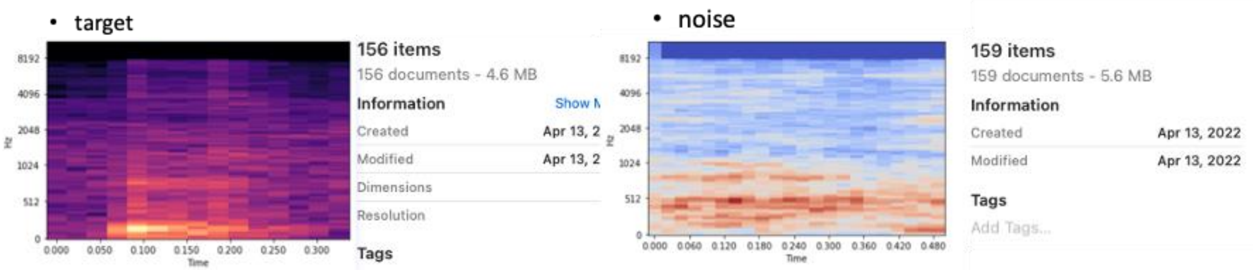


Figure 10, Structure of noise filter

The noise filter was trained with a learning rate of 0.001 and a penalization value of 0.1. The target sample was collected from keystrokes by the same person using the same keyboard in a quiet environment. More information shows in Figure 11.



• target

156 items
156 documents - 4.6 MB
Information Show M
Created Apr 13, 2
Modified Apr 13, 2
Dimensions
Resolution
Tags

• noise

159 items
159 documents - 5.6 MB
Information
Created Apr 13, 2022
Modified Apr 13, 2022
Tags
Add Tags...

Figure 11, details of noise filter dataset

The keylogging ML model contains much more labels than the noise filter, so it needs to explore a better ML model structure. We tried from three convolutional layers to six and tried to reach a balance between predict accuracy and time consumption. Figure 12 shows the test result.

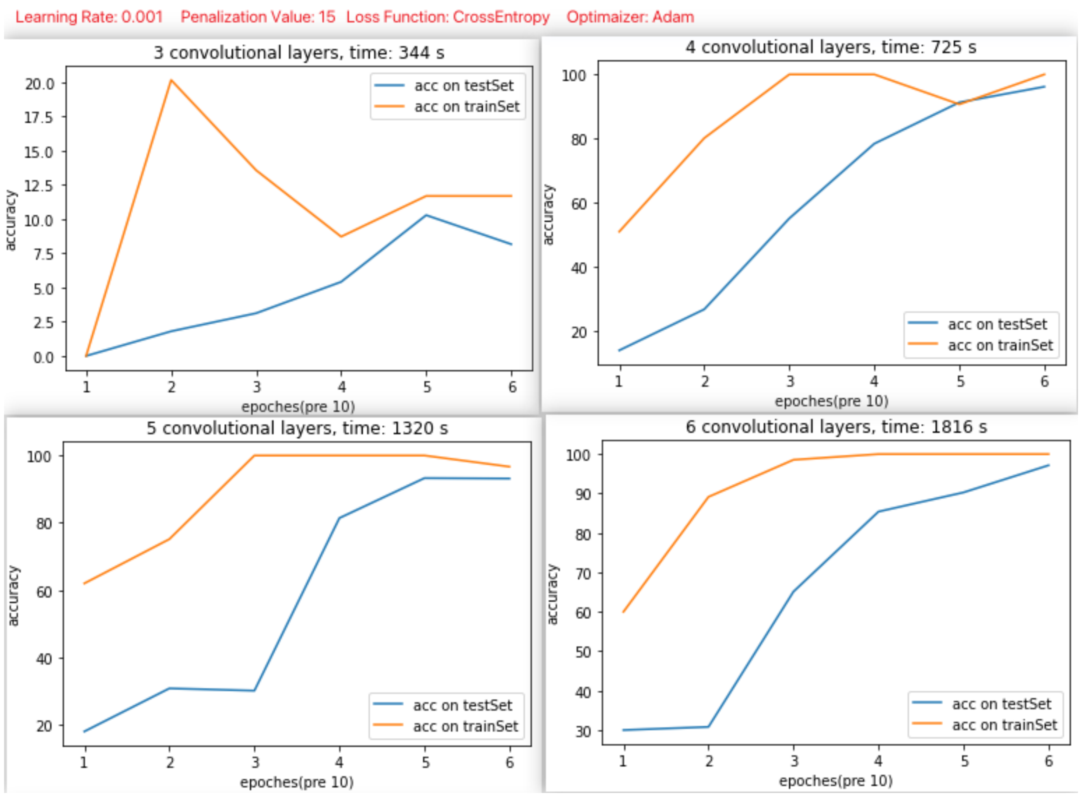


Figure 12, Layer test

A model with two layers was too slow to converge. The 5 or 6 layers model takes a quite long time to train and shows a little bit tent to overfitting. So, four layers would be a viable choice.

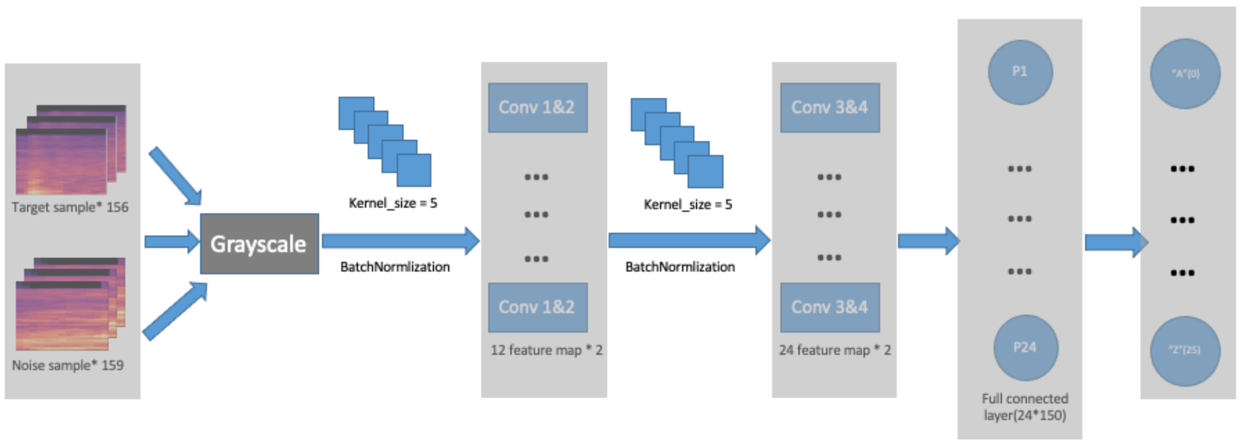


Figure 13, Model structure

Then, to find out how the number of the sample affects the result, we measured when the limit number of samples for each class to the range of 3, 6, 9, and 12, respectively. Turns out, three would lead to obvious overfitting. With the increase of samples to nine, the accuracy of prediction raises while falling when the sample limit reaches twelve. Thus, the number of samples should not be too small. But if noise or abnormal samples are mixed in the dataset, an additional part of the samples might not help at all.

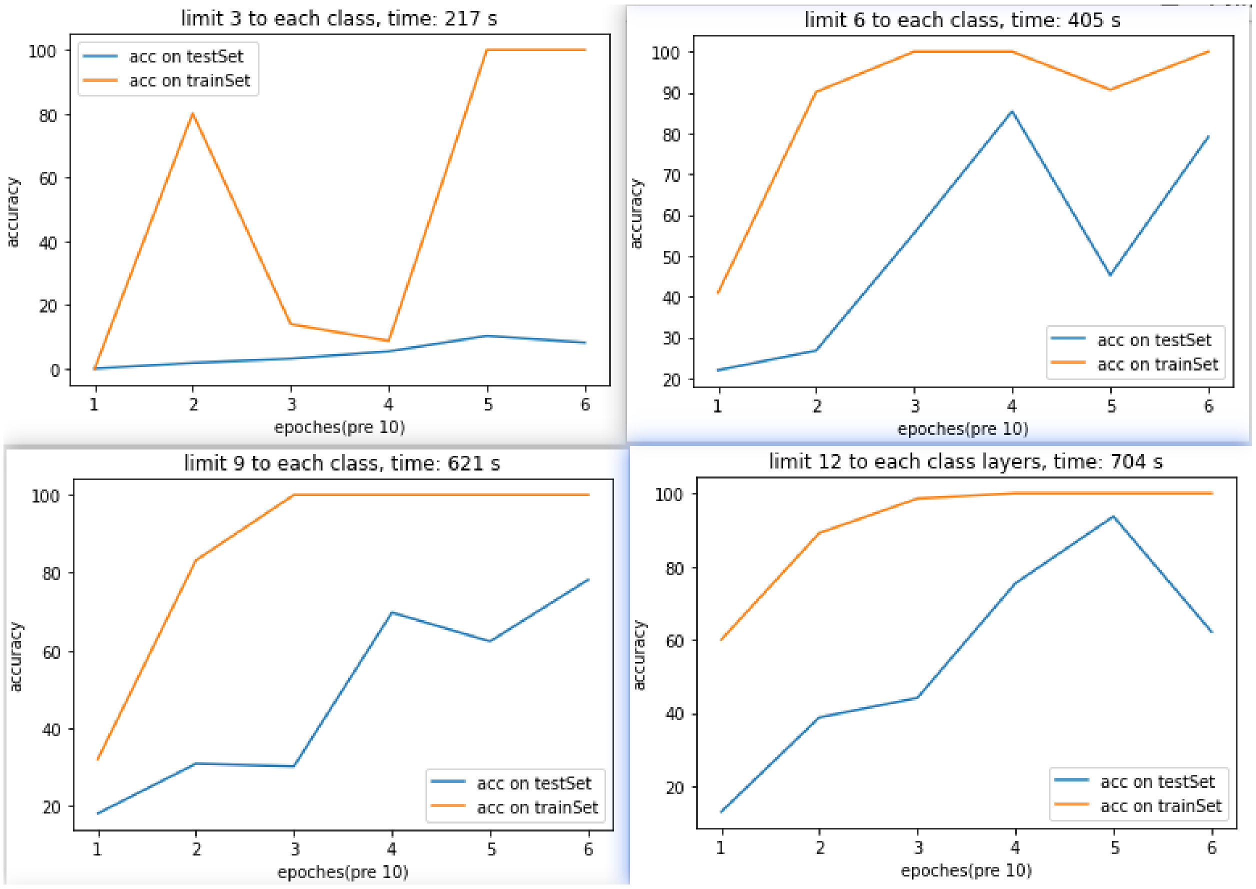


Figure 14, test sample numbers

We also explored how prediction accuracy changes with the growth of label numbers up to 26. The result shows in figure 15.

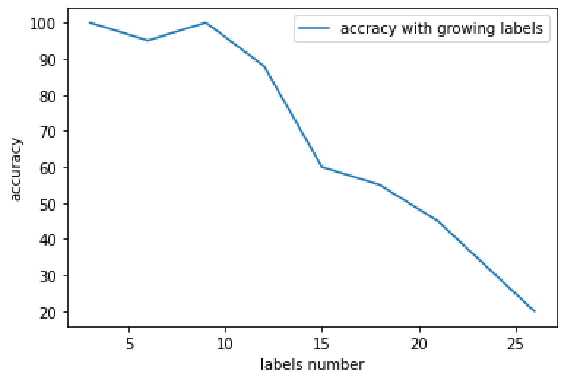


Figure 15, prediction accuracy changing

Flask Application & Web Interface

This chapter will give a brief description of how to run the demo if anyone wants to try to reproduce the project.

Flask Application

To start the Application Server, the first step is to install pipenv by using the command shown in code block 6:

Code block 6, install piping

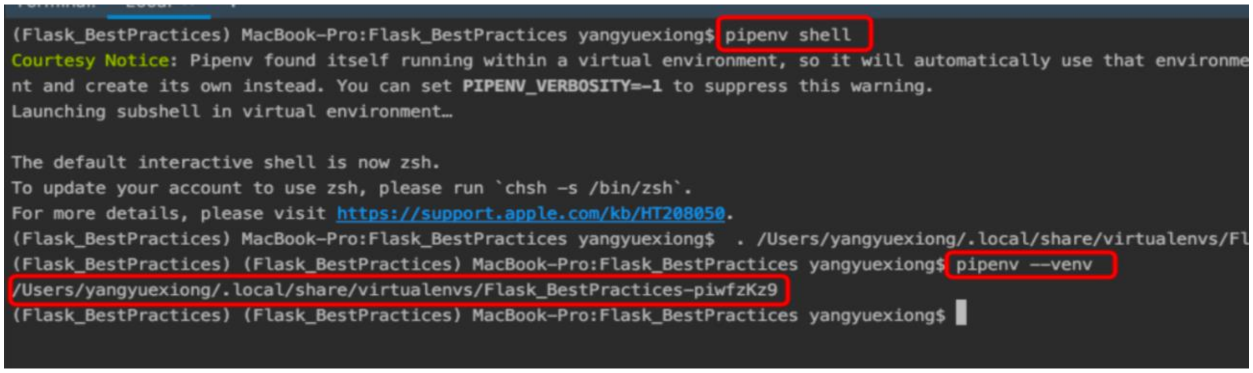
```
pip3 install pipenv
```

Then install dependencies,

Code block 7, install dependencies

```
cd $<project_dir>$

pipenv install

pipenv shell

pipenv --venv
```

Check whether the virtual environment is configured successfully:



```
(Flask_BestPractices) MacBook-Pro:Flask_BestPractices yangyuexiong$ pipenv shell
Courtesy Notice: Pipenv found itself running within a virtual environment, so it will automatically use that environme
nt and create its own instead. You can set PIPENV_VERBOSITY=-1 to suppress this warning.
Launching subshell in virtual environment…

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(Flask_BestPractices) MacBook-Pro:Flask_BestPractices yangyuexiong$  . /Users/yangyuexiong/.local/share/virtualenvs/Fl
(Flask_BestPractices) (Flask_BestPractices) MacBook-Pro:Flask_BestPractices yangyuexiong$ pipenv --venv
/Users/yangyuexiong/.local/share/virtualenvs/Flask_BestPractices-piwfzKz9
(Flask_BestPractices) (Flask_BestPractices) MacBook-Pro:Flask_BestPractices yangyuexiong$
```

Figure 16, Config virtual environment

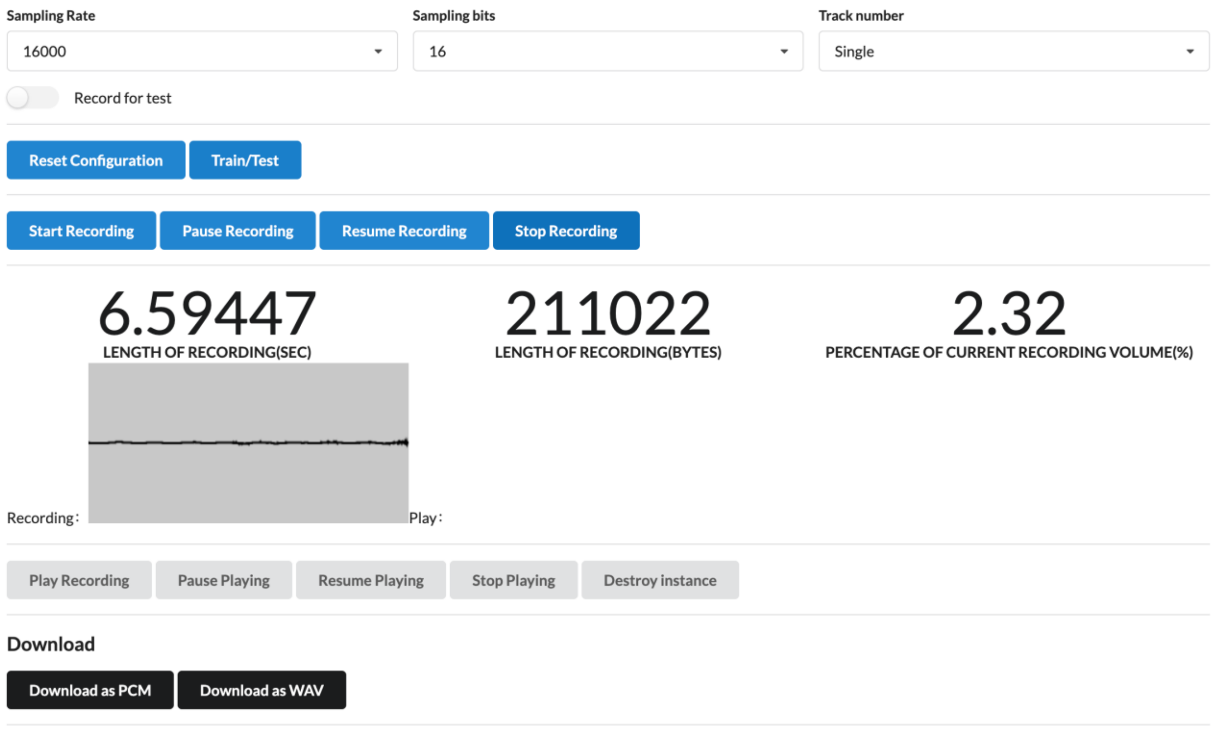
Web Interface

To start Web Interface, first-run code in code block 8.

Code block 8, Run web interface

```
npm install & npm run dev
```

The web interface should look like the screenshot shown in Figure 17.



Sampling Rate: 16000
Sampling bits: 16
Track number: Single

6.59447 LENGTH OF RECORDING(SEC)
211022 LENGTH OF RECORDING(BYTES)
2.32 PERCENTAGE OF CURRENT RECORDING VOLUME(%)

Figure 17, Screenshot of Web Interface

Analysis and Reflection

In this section, what have we learned through this project would be concluded. And some additional possibilities to improve the performance would also be mentioned.

What have we learned?

Through this project, we have gained a deeper understanding of side-channel attacks (especially acoustic channels). We also have a clearer understanding of acoustic signal processing and machine learning applications in the field of information security.

More Samples

As mentioned in the ML model above, the inclusion of more non-noisy samples will drive the machine learning model in a better direction. However, due to the limitations of the means of collecting user data in the threat model, it is not possible to obtain a larger amount of data in a single data collection.

This leads us to focus instead on generating more clean data using machine learning. That is, two machine learning models are trained simultaneously, one for keylogging and the other for generating more data that can be used for augmented learning. If the project deadline had been delayed by a month, we might have focused more on GAN (Generative Adversarial Network) to achieve the above goal.

Better Way to Avoid Noise

Another idea to improve the training efficiency is to try to shield the training samples from the uncontrollable entropy increase caused by noise. The current noise removal method is to brute-force remove samples identified as noisy by the noise filter module, which somehow further reduces the number of samples available for training of the keylogging prediction model.

If we could find some way to filter the samples containing noise, it might greatly improve the current problem of an insufficient number of training samples. If additional work time is available, we may look for more potential solutions in that direction.

Stricter Align of Push Peak

Due to slight computational errors and occasional burr signals collected by the microphone, sometimes individual samples have problems with the accurate alignment of the push peak. By applying the "peaks" method provided by librosa, there is a greater chance of improving this problem.

Conclusions and Future Work

We experimented with CNNs to classify audio data that had been push-peaks-aligned and transcoded to spectrograms, and with a small number of labels (typically under 10), the model could still make good predictions using only three to five samples per category. However, once the number of labels increases (e.g., to a maximum of 26), the model becomes hungrier for larger sample data to combat the noise. We will explore additional means to optimize this project as conditions allow.

References

Compagno, A., Conti, M., Lain, D., & Tsudik, G. (2017, April). Don't skype & type! acoustic eavesdropping in voice-over-IP. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (pp. 703-715).

Zhuang, L., Zhou, F., & Tygar, J. D. (2009). Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security (TISSEC)*, *13*(1), 1-26.

Heittola, T., Çakır, E., & Virtanen, T. (2018). The machine learning approach for analysis of sound scenes and events. In *Computational Analysis of Sound Scenes and Events* (pp. 13-40). Springer, Cham.

Mnasri, Z., Rovetta, S., & Masulli, F. (2021). Anomalous sound event detection: A survey of machine learning-based methods and applications. *Multimedia Tools and Applications*, 1-50.