Final Presentation: Keyboard Snooping

An Le, Eugene Chu, Zixuan Zhong

Overall Project Goals and Specific Aims

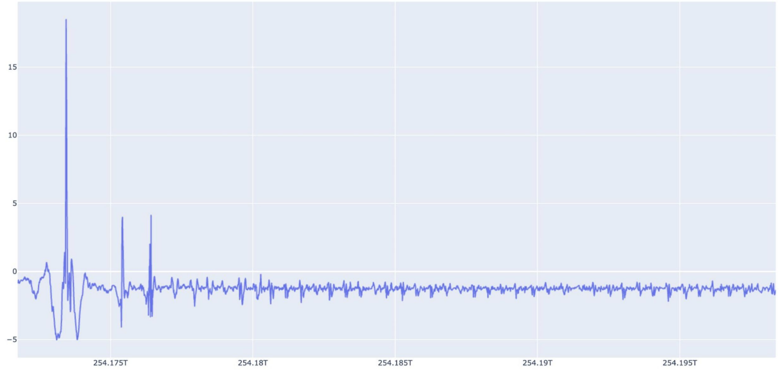
- Perform keyboard snooping with microphone and smartwatch (motion sensors)
- Given audio and motion sensor (accelerometer, gyroscope, or linear accelerometer) data
    - Identify specific keys
    - Identify 6-digit PINs
    - Identify tasks involving keyboard

Technical Approach

- Algorithms
    - Key identification
        - Audio: lossless wave ⟶ extract peaks ⟶ FFT ⟶ NN classifiers
        - Motion Sensor: extract displacement manually? direct training?
    - Task identification
        - Neural network classifier
- Datasets
    - Task 1: single keys, each has an audio wave file
    - Task 2: 6-digit PINs, each has an audio wave file and five accelerometer data
    - Task 3: each sample is at least 30 seconds long and has both audio data and accelerometer data
- Platform
    - Data collection:
        - Audio Recording: A Python Program (software), The built-in microphone of macbook pro (hardware)
        - Motion Sensor: Android Physical Toolbox Suite (app), any android phone (simulating a smart watch)
    - Training and inference:
        - Keras

Task 1 Implementation: Recording and Keylogging

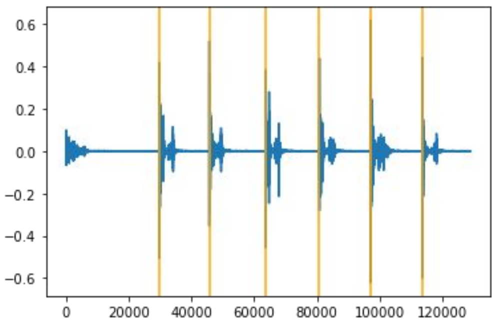
- A python program instruct the user to type
  - Audio: Macbook built-in stereo microphone (.wav)
  - Keylogger: Python keyboard module (.csv)




```
Current devices:

> 0 Built-in Microphone, Core Audio (2 in, 0 out)
< 1 Built-in Output, Core Audio (0 in, 2 out)

Choose mic (index):  0
Recording parameters set.
Started microphone . . .
Please start actual recording by pressing [SPACE] . . .

Start actual recording:

1) space 2 enter

2) enter 3 enter

3) enter 4 enter

4) enter 5 enter

5) enter 6 enter

Stop recording.

Writing key logs to outputs/0320074758.csv . . .
```

Task 1 Implementation: Preprocessing and Architecture

- Audio processing
  - Peak extraction
    - Detect peaks given threshold and minimum time between keystrokes
      - Ensure threshold large enough to filter out release peaks
    - Grab audio from 5ms before peak to 20ms after peak
  - Feature extraction
    - Sliding window of length 20ms and stride of 1ms
    - Take average of FFTs of sliding windows
    - Each feature normalized
- Neural network
  - MLP with two hidden FC layers (64, 64), dropout (0.5), FC softmax output (36)
  - Output layer size 36 (a-z, 0-9)



Task 1 Implementation: Machine Learning and Result

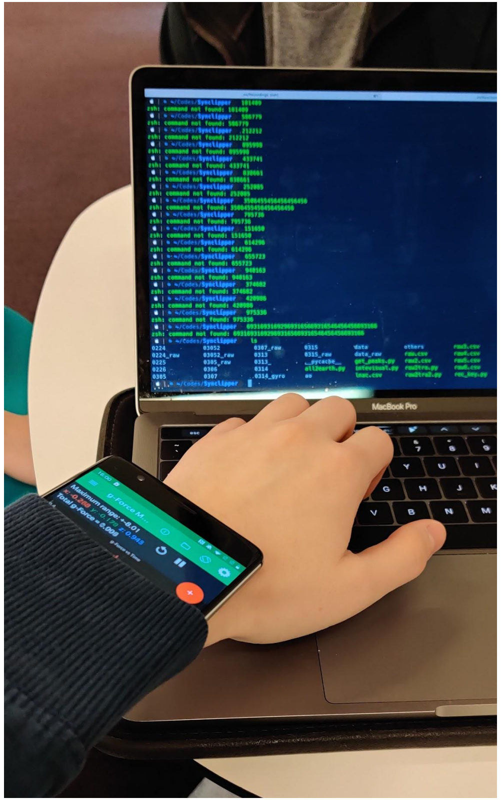
- Dataset
  - 4807 total keystrokes (a-z, 0-9)
  - Train-val-test split: 70%-15%-15%
- Training
  - Batch size 128
  - 100 epochs
  - Save model with lowest validation loss
- Results (top-1 accuracy)
  - Single model: **96-98%**
  - Ensemble (averaging) of 5 models: **99%**
  - cf. Keyboard Acoustic Emanations (2004): 79%

Task 2 Implementation: Which sensor should we use and how?

- Can we extract displacement manually?
  - Conversion from the device's frame to the world's frame, and Double integration from accelerometer to displacement (high school physics)
    - Need linear accelerometer data and orientation data (low sampling rate)
    - Large error even the Kalman FIlter is used
  - We lose details when doing manual extraction
    - Previously: a sequence of sensor data; After: only one value
- Which motion sensor to use, accelerometer, gyroscope, or linear accelerometer?
  - Trained three model (accelerometer, gyroscope, or linear accelerometer)
  - Gyroscope: hard to find peaks
  - Accelerometer V.S. linear accelerometer: 80% > 70%

Task 2 Implementation: Hardware and Software

- ZenWatch 2 & AndroidSensorDemo
  - NESL has a app for ZenWatch 2 https://github.com/nesl/AndroidSensorDemo which can dump GRAVITY, ACCELEROMETER, and LINEAR_ACCELEROMETER sensor data
  - Sampling rate not high enough
  - Some logging issues
- Android phone & Physics Toolbox Sensor Suite
  - High sampling rate: 400 Hz
  - All sensors
  - Single sensor or customized combinations of multiple sensors



Task 2 Implementation: Synchronization and Labeling

Synchronization

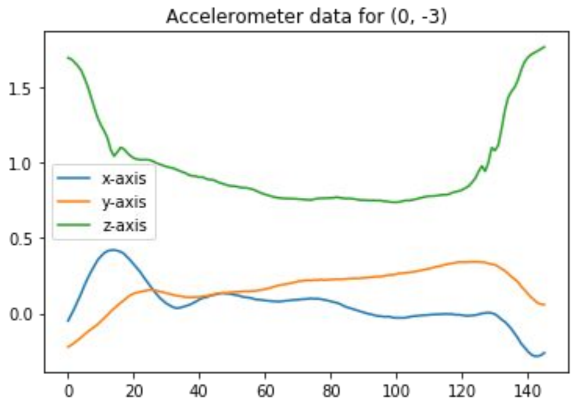
1. Find the peak from the first X seconds of the audio data
2. Find the peak from the first Y seconds of the accelerometer data
3. Based on keylogs, cut a recording into groups
   a. Each group contains a single key (Task 1)
   b. Each group contains 6 digits (Task 2)
4. For the PIN task, Find 6 peaks for each group and cut the accelerometer data into 5 parts

Labeling

- Labels of audio data: from key logs
- Labels of accelerometer data: equivalent displacements
  - Encode each key to 2D coordinates: 0: (0, 0), 1: (-1, 1), 2: (0, 1), 3: (1, 1), 4: (-1, 2), 5: (0, 2), 6: (1, 2), 7: (-1, 3), 8: (0, 3), 9: (1, 3)
  - If the PIN is “104582”, there are 5 key pairs in this PIN, which are: <1, 0>, <0, 4>, <4, 5>, <5, 8>, <8, 2>
  - Each key pair is labeled with a vector: (1, -1), (-1, 2), (1, 0), (0, 1), (0, -2)

Task 2 Implementation: Preprocessing and Architecture

- Audio processing
  - Same as Task 1
- Accelerometer data processing
  - Use segments between push peaks
  - Tried FFT but accuracy low
  - Instead, we resample segments to fixed length of 256
- Neural network
  - Audio
    - Same as Task 1 except with 10 output classes (0-9)
  - Accelerometer
    - Similar architecture to audio
    - Linear output layer representing 2D coordinates
    - MSE loss function



Task 2 Implementation: Machine Learning and Result

- Dataset
  - 718 PINs (0-9) of length 6
    - 2 people, 2 keyboards
    - For each PIN, 6 audio samples and 5 accelerometer samples
  - Train-val-test split: 70%-15%-15%
- Training
  - Batch size 128 for both models
  - 100 epochs for both models
  - Save models with lowest validation loss

- Key model evaluation
  - Individual key accuracy: **84.1%**
    - Top-3 accuracy: **95.1%**
- Displacement model evaluation
  - Accuracy (after rounding output): **80.8%**
  - Loss (MSE): **0.172**

Task 2 Implementation: Maximum Likelihood Tree Search (MLTS)

- Main idea: find the most probable PIN given key probabilities, displacement estimates assuming that
  - Length of PIN is known
  - Distribution of displacement estimate is bivariate normal with σx = σy
- Objective function (log likelihood)
  - **Every subterm is effectively non-positive**

$$LL(G) = \sum_{i=1}^{|G|} \log(p(k_i = g_i)) - c \sum_{i=2}^{|G|} |d_i - \hat{d}_i|^2$$

$$\begin{aligned}
G &= \text{PIN subguess with } g_i \text{ as the } i^{\text{th}} \text{ key} \\
K &= \text{True PIN with } k_i \text{ as the } i^{\text{th}} \text{ key} \\
p &= \text{Softmax outputs of key estimator} \\
c &= \text{Reliability constant of displacement estimator} \\
d_i &= \text{True displacement from } g_{i-1} \text{ to } g_i \\
\hat{d}_i &= \text{Estimate of above}
\end{aligned}$$

Task 2 Implementation: Maximum Likelihood Tree Search (MLTS)

- How to find maximum likelihood
  - Naive search: Try all PIN combinations (1 million!)
  - **With heuristic: Set baseline to LL of 6-key PIN guess from key classifier**
    - Perform DFS starting with 1-key subguesses
    - If LL(subguess) less than or equal to maximum LL so far, **do not go deeper**
      - Otherwise, add one more key to subguess
    - Update maximum LL if subguess has 6 keys and its LL is greater than the current max

- **Example run**
  - Set baseline
    - Key classifier guesses [6, 9, 5, 0, 9, 9]
    - LL([6, 9, 5, 0, 9, 9]) = **-7.424**
  - Prune bad guesses
    - LL([1]) = **-7.688 ≤ -7.424**
    - Skip all guesses starting with 1
  - Update max LL and best guess if better full guess is found
    - LL([8, 9, 5, 0, 9, 9]) = **-5.963 > -7.424**
    - Imagine d1 = [1, 0]
  - Turns out the true PIN is [8, 9, 5, 0, 9, 9]

Task 2 Implementation: Evaluation after Audio and Accelerometer Fusion

- Individual keys
  - Key classifier only: **84.1%**
  - With fusion + MLTS: **91.8%**
- Entire PIN (5 attempts)
  - Key classifier only:
  - With fusion + MLTS: **83.3%**

- **Additional information from accelerometer reduces error rate**

Task 2 Implementation: Evaluation on Additional Test Set

- Different person, different keyboard
- Displacement estimator
  - Accuracy (after rounding output): **51.8%**
  - Loss (MSE): **0.420**
- Key classifier
  - Key classifier only: **13.6%**
  - With fusion + MLTS, c = 1: **13.6%**
  - With fusion + MLTS, c = 100: **33.3%**
- Performance drop of displacement estimator not as drastic as that of key classifier since hand movement is fairly consistent across different people
- **Some information gain from displacement despite poor key classification**

Task 3 Implementation: Hardware and Software

In this task we collect sensor fusion data from three user tasks (gaming, messaging, emailing) for classification

- Same setup as Task 2 (Android phone & Physics Toolbox Sensor Suite)
- Dataset: 30 seconds/sample, 30 samples/task
- Main idea: find the most probable task given, accelerometer and audio information

Task 3 Implementation: Preprocessing and Architecture

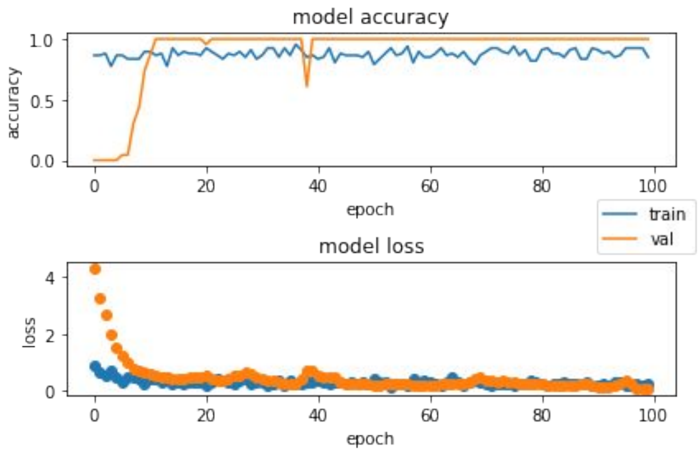
- For this task, we preprocess accelerometer and audio data to reduce feature size:
  - Calculate mean and standard deviation of acceleration for every 300 samples
  - Fourier Transform on audio waveform with window size of 20ms and stride of 1ms
  - Final feature size: 631 per sample
  - Train-val split: 85%-15%
  - Batch size: 10
  - Epochs: 100
- Neural network
  - Audio
    - FFT sample features (1D)
  - Accelerometer
    - Flatten input for FC layer

  - FC layer with batchnorm and dropout
  - Categorical cross-entropy

Task 3 Implementation: Machine Learning and Result

- Results:
  - High accuracy, but may be have overfitted
  - Model easily converges because of small dataset

- Task classification is also dependent on the key press intervals (Intuition):
  - Gaming tasks have constant key presses limited to certain keys (little acceleration change)
  - Messaging has multiple keys pressed clustered in short intervals (requires the other person to reply)
  - Emailing tasks are have more constant typing over a longer period (longer sentences)
  - Dataset is too small



Prior work examples and Relative Novelty: Task 2

- Keyboard Acoustic Emanations (D. Asonov et al., 2004)
  - First work to develop key classifier based on sound of keypress
  - Works because **the keyboard plate is struck at different spots, producing different timbres like a drum**
- Keyboard Acoustic Emanations Revisited (L. Zhuang et al., 2005)
  - Utilizes clustering and language model to predict keys and words without prior labeling
  - Analyzes key identification attack as substitution cipher
- Don't Skype & Type! Acoustic Eavesdropping in Voice-Over-IP (A. Compagno et al., 2017)
  - Considers different attack scenarios such as knowing or not knowing actual keystrokes or laptop manufacturer
  - Relies on a database of sounds from different keyboards to infer victim's laptop
- **Our work: Fusion of audio and motion sensor data**
  - Exploits estimated displacement from accelerometer to improve key prediction
  - No language model required

Prior work examples and Relative Novelty: Task 3

- Activity classification using a single wrist-worn accelerometer (S. Chernbum roong et al., 2011)
    - Two models: “Decision Tree C4.5” and “1-layer FC network” for classification of 5 tasks
        - sitting, standing, lying, walking, and running
    - Using **Correlation-based Feature Selection** gives the best accuracy of 94.13% and F-1 score of 0.91

- A Comprehensive Study of Activity Recognition Using Accelerometers (N. Twomey et al. 2018)
    - The paper surveys and evaluates methods of activity recognition using accelerometers.
    - A window length of 5 to 7 seconds gave the highest prediction accuracy using a single wristband accelerometer.

- **Our Work: Fusion of audio and motion sensor data to infer tasks with relatively similar movements**

Strengths, Weakness, and Future Directions

- Strength
  - Multiple scenarios (3 tasks)
  - Multiple ways of sensor fusion
  - High accuracy
- Weakness
  - Applicability. Samples from limited number of hardware.
  - The model requires high sampling rate which is hard to be achieved
  - The Victim may not use the hand with watch to type
- Future Directions
  - More data samples
  - Data samples from different hardware devices
  - To identity the user’s emotion while typing

Contributions

An Le	Survey of Prior Work, Machine Learning (Task 1 & 2), Maximum Likelihood Search Algorithm (Task 2)
Eugene Chu	Dataset Generation (Task 1 & 2 & 3), Machine Learning (Task 3)
Zixuan Zhong	Explorations on Motion Sensors (Frame Conversion, Double Integration, and Sensor Selection), Dataset Generation (Task 1 & 2 & 3), and Dataset Synchronization (Task 1 & 2 & 3)

GitHub Repo and Project Website

- Github Repo
  - https://github.com/binhanle/keyboard-snooping
- Project Website
  - https://binhanle.github.io/keyboard-snooping/

References

- Android Physical Toolbox Suite: https://www.vieyrasoftware.net
- Python3: https://www.python.org
- Keyboard: https://github.com/boppreh/keyboard
- Keras: https://keras.io