Exploring Acoustic Keystroke Recognition: A Machine Learning Signal Processing Experiment

Peiheng Lyu, Michael Ortiz, Hanqing Wang

[a]*University of Illinois Urbana-Champaign, The Grainger College of Engineering*

Abstract

This study presents the development and evaluation of a machine learning model capable of predicting keystrokes from audio data using convolutional neural networks (CNNs). The model is designed to interpret the acoustic signatures produced by typing, with a focus on robust performance across various noise environments. Data collection involved capturing keystrokes under different conditions, including varying loudness levels and background noise. The model was trained and tested in environments categorized by noise levels ranging from minimal to moderate. We experimented with spectral factorization for denoising and discuss the limitations and key considerations of denoising methods in such scenarios. Our findings demonstrate that the model maintains accuracy in different acoustic settings, although challenges arise in high-noise environments.

Keywords: Computer security, signal analysis, keyboards, Deep learning

1. Introduction

This project explores the intersection of acoustics and machine learning, specifically focusing on using sound to predict keystrokes. Central to our research is the question: Can the nuanced sounds of typing, often subtle and easily overlooked, be accurately interpreted by a machine learning algorithm? This investigation leads us into a detailed exploration within signal processing and computational linguistics.

A significant challenge we faced was environmental noise. Differentiating the specific sounds of keystrokes from various background noises, such as everyday conversations, ambient sounds in different environments, or even the low hum of computer equipment, required precise noise filtering and sound isolation strategies. Developing an algorithm capable of effectively capturing and interpreting keystroke sounds in such diverse auditory conditions was a complex task.

Our research is motivated by an interest in understanding the limits and capabilities of machine learning in interpreting and predicting human-computer interactions based on auditory data alone. We delve into the intricacies of keyboard sounds, where each keystroke produces a distinct acoustic signal, and examine how these can be transformed into meaningful data.

The potential applications of this research, particularly in the field of accessibility, are noteworthy. For individuals with visual impairments or motor disabilities, technology that can interpret auditory cues from keyboard interactions could lead to the development of more adaptive and user-friendly interfaces. This could enhance the usability of technology for a broader range of users, making it more inclusive.

This paper will detail our methodologies, the specific solutions we devised to address the challenges of environmental noise and variability, and the implications of our findings.

2. Implementation

In this section, we outline the practical aspects of our project, focusing on three main areas: data collection, model training, and real-time application. The data collection process is the foundation of our research, where we describe the methods used for gathering and preparing the acoustic data necessary for our model. The quality of this data is crucial, as it directly impacts the model's accuracy. Following this, we discuss the model training phase, including our choice of algorithms, training methods, and the challenges we faced in optimizing the model for keystroke prediction. The final part of this section details the development of a real-time application, demonstrating how our model functions in an operational setting. This section aims to provide a clear and concise overview of the steps we took in implementing our research, illustrating how each stage contributes to the overall functionality and effectiveness of our model.

2.1. Data Collection

In the data collection phase of our project, we focused on capturing a comprehensive range of acoustic data to ensure the robustness of our model. The primary challenges in this phase were related to the loudness of sound, the variety of sounds produced by the same key, device consistency, and the influence of device location.

1. **Loudness of Sound**
   Recognizing the significance of varying loudness levels in keystroke sounds, we meticulously collected data across a spectrum of these levels. It was crucial to include a broad range of loudness in our dataset, as encountering an unfamiliar loudness level during model deployment could lead to inaccurate predictions. By encompassing a diverse set

of loudness levels, we aimed to enhance the model's adaptability and accuracy in real-world scenarios.

2. **Variation in Keystroke Sounds**
   A unique aspect of our data collection was accounting

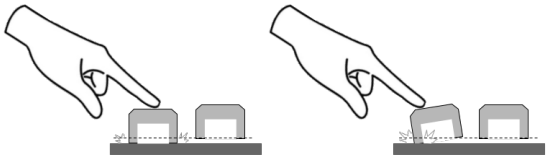

Figure 1: Different sound when pressing with different direction

for the different sounds produced by the same key when pressed with different fingers. This variation, primarily due to the differing force and direction of finger presses, was an essential factor in our data collection strategy. We systematically recorded keystrokes of the same key using various fingers to capture these subtle yet significant acoustic differences.

3. **Device Consistency**
   Considering the variation in sound production and capture by different keyboards and microphones, we emphasized device consistency in our data collection. We acknowledged that a model trained on specific hardware might not perform optimally with different devices. To address this, we aimed to train our model in a way that would be less dependent on the specific characteristics of particular keyboards and microphones.
4. **Device Location**
   The location of the recording device plays a crucial role in the characteristics of the captured sound. If the neural network has not been trained using data from diverse device locations, there is a high probability that the result will alter in scenarios where the device's location changes.

In practice, our data collection process involved using two microphones to record stereo audio of the keystrokes. This setup allowed us to capture a richer, more spatial sound profile. We conducted recordings under various conditions, including different finger presses and varying levels of background noise, to ensure our dataset was as comprehensive and representative as possible of the real-world conditions the model would encounter.

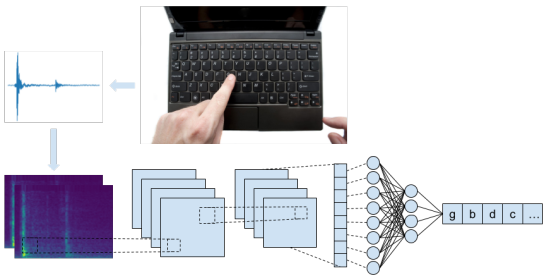

Figure 2: Program structure

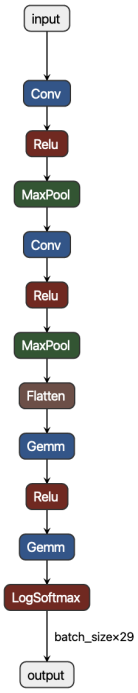

Figure 3: Network architecture

2.2. *Model Training*

We use a convolutional neural network (CNN) to predict keystrokes from audio data. The choice of a CNN was driven by its superior ability in capturing spatial and temporal patterns in data, which is essential for identifying unique sound patterns of keystrokes. In comparison to recurrent neural networks (RNNs), which we also considered, CNNs offer a simpler yet effective solution for processing our audio data. While RNNs are adept at handling sequential data, our experiments indicated that CNNs perform equally well in capturing the necessary patterns in keystroke sounds, but with a more straightforward architecture and lower complexity.

The network begins with two convolutional layers dedicated to feature extraction. These layers are pivotal in identifying the distinct acoustic signatures associated with different keystrokes. Following the convolutional layers are max pooling layers, which serve to reduce the dimensionality of the data. This reduction not only enhances computational efficiency but also aids in maintaining the robustness of the key features extracted by the convolutional layers.

To mitigate the risk of overfitting, we have incorporated dropout layers into the architecture. These layers randomly deactivate a subset of neurons during the training process, promoting the model's ability to generalize to new, unseen data rather than memorizing the training dataset.

The network proceeds with fully connected layers, which play a critical role in interpreting the features extracted by the earlier layers. These layers culminate in a final output layer that probabilistically represents different keystrokes. The use of non-linear activation functions throughout the network allows for the capture of complex patterns within the data. Specifi-

cally, the softmax function in the final layer facilitates precise multi-class classification, essential for distinguishing between various keystrokes.

Overall, this CNN architecture effectively translates the acoustic signatures captured from typing into identifiable keystrokes. It showcases the capability of machine learning, particularly CNNs, in interpreting auditory data, providing a balance of efficiency, simplicity, and accuracy.

2.3. Data Preprocessing

In our study, we preprocessed and transformed the time series data into two forms: Spectrogram and Cepstrum. We compared these two feature types in our experiments. Our audio data were collected at a 44.1kHz sample rate, with each keystroke sound stored in a 1100ms .wav clip. During preprocessing, we trimmed the clip to 9000 data points (approximately 204ms), which suits the length of most keystrokes in our tests. For the spectrogram, we employed Short-Time Fourier Transform (STFT) with the following parameters: DFT size = 256, hop size = 64, zero padding = 256, and applied a Hamming window to each segment. This yielded data with a shape of (137, 257), which best represents the frequency feature of a keystroke in our experiments. Considering we have two channels for each keystroke sound, the actual shape of a single data point is (2, 137, 257).

For the cepstrum, we utilized Mel-frequency cepstral coefficients (MFCCs), resulting in data with a shape of (2, 20, 18), which is simpler than the spectrogram. Although MFCCs require slightly more time for preprocessing, their transformation into a simpler form means that the deep learning process is much faster compared to spectrogram data. In our experiments, training on MFCCs data was about five times faster than training on spectrogram data.

2.4. Real-time Application

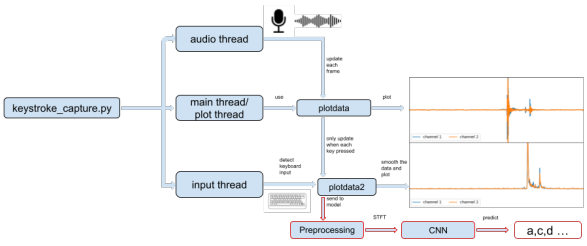

Figure 4: Real-time system architecture

To enhance the functionality of our pipeline, we have prioritized optimizing our program for real-time operation.

The real-time system operates using three concurrent threads:

1. **Audio Thread**: Captures new audio data streams and places them in a queue. The denoising process starts in this thread, preparing the data for further analysis.
2. **Plot Thread**: Reads denoised audio data from the queue and updates 'plotdata'. It renders the data visually for real-time monitoring and verification.
3. **Input Thread**: Captures keyboard inputs and associates them with the corresponding denoised audio data. In the data collection phase, this involves displaying the pressed key and saving the signal data. In the prediction phase, it involves sending the denoised signal to the deep learning model for prediction.

In our real-time signal capture process, we continuously visualize the signals from both microphone channels. This approach helps us identify unique features of the sound. When a

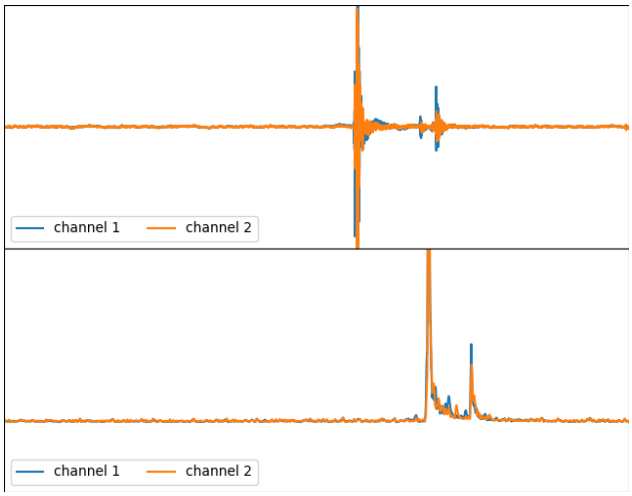

Figure 5: Example keystroke showing separate channels

keystroke sound originates from a key near the left, we typically observe a higher average magnitude in channel 1. Conversely, a key closer to the right tends to produce a higher average magnitude in channel 2. This implies that our system is also capable of capturing the positional information of the keys.

2.5. Denoise

We experimented with spectral factorization in our program and found its denoising effect to be ideal. However, integrating it with our neural network posed a challenge. We applied the following functions:

$$\mathbf{V} = \frac{|\mathbf{F}|}{\mathbf{W} \cdot \mathbf{H} + \epsilon}$$

$$\mathbf{H} = \mathbf{H} \odot \left[\mathbf{W}^{\top} \cdot \mathbf{V}\right]$$

$$\mathbf{W} = \mathbf{W} \odot \left[\mathbf{V} \cdot \mathbf{H}^{\top}\right]$$

We trained and acquired $\mathbf{W}_{key}$ on data without noise. Then, for each noisy audio clip, we learned $\mathbf{W}_{noise}$ from the non-keystroke parts of the clip. Subsequently, we extracted the denoised sound using the dot product of $\mathbf{W}_{key}$ and corresponding $\mathbf{H}$.

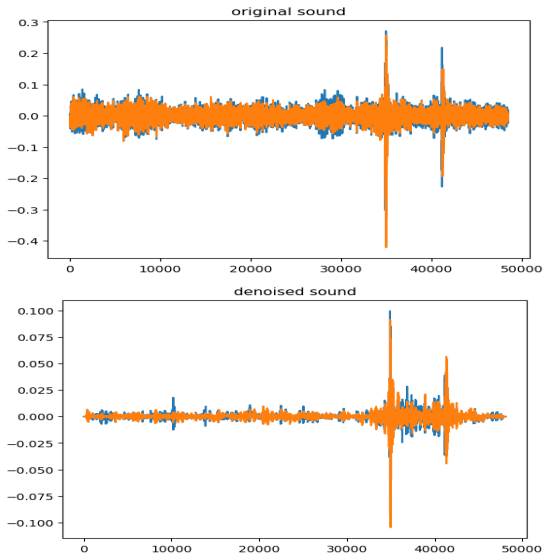

Figure 6: denoise example

A significant issue with this approach is its instability in producing consistent results for the same keystroke. The denoised sound's quality heavily depends on the noise characteristics. Similarities between the noise and keystroke features lead to alterations in the original keystroke sound, significantly affecting the CNN's performance. Upon analysis, we concluded that the neural network struggles to predict how the sound will change based on the cancelled noise. After applying this denoising method to the same dataset, the accuracy of our CNN model dropped from 95% to 65% even on dataset with minimal noise . Further experiments revealed that both the CNN model and the MFCCs are robust at reasonable noise levels. Including noisy data in the training set improved the CNN model's performance on similarly noisy test cases. Consequently, we modified our denoising approach: we now use the denoise function solely to precisely locate the keystroke, and then feed that keystroke data to the CNN directly. This method improved accuracy by about 10% on datasets where the noise is louder than the keystroke itself.

3. Results

In the evaluation phase of our project, we trained and tested our model in a variety of environments, each characterized by different noise levels. To systematically assess the impact of ambient noise on our model's performance, we categorized noise levels into five distinct categories:

- Level 1: Almost no noise.
- Level 2: Minimal noise, such as a fridge or a mini-fan in the room.
- Level 3: Noticeable noise, akin to an office environment.
- Level 4: Moderate noise, like background music.
- Level 5: Loud noise, The noise is louder than keystroke itself.

Spectrogram(STFT)			
Train scale	**Test scale**	**Noise level**	**Accuracy**
Small scale	Small scale	1	71%
median scale	median scale	2	78%
Large scale	median scale	3	86%
Large scale	Large scale	4	84%
Large scale	Small scale	5	15%

Table 1: Experiments result for Spectrogram

Cepstrum(MFCCs)			
Train scale	**Test scale**	**Noise level**	**Accuracy**
Small scale	Small scale	1	89%
median scale	median scale	2	94%
Large scale	median scale	3	95%
Large scale	Large scale	4	94%
Large scale	Small scale	5	27%

Table 2: Experiments result for Cepstrum

4. Discussion

The results of our experiment indicate that Mel-frequency cepstral coefficients (MFCCs) outperform Spectrograms by approximately 10% in terms of accuracy. MFCCs not only enhance training speed and reduce neural network complexity but also offer advantages in accuracy. Therefore, similar tasks should consider using MFCCs for feature representation instead of Spectrograms.

For the denoise part, through our experiments, we recognized that developing a denoise function for a real-time system and combine with deep learning classification is challenging. Such a function must satisfy two conditions:

- The algorithm needs to be stable. If the same keystroke sound is mixed with different noises, the extracted keystroke should be as similar as possible.
- The algorithm must be fast, it needs to be efficient enough to deal with real-time streaming content

An alternative approach could be to employ another neural network specifically for denoising. Although the output of a neural network can be variable, it may be feasible for the classification network to adapt to the workings of the denoising network, thereby managing this uncertainty. An example of such a system could be a Generative Adversarial Network (GAN).

The current model is only useful to specific type of keyboard with two microphones placed at the same fixed locations as the experiments. In another words, the model is not yet generalized. The practicality of such device still needs to be further explored. One of the most potential application could be on the tablet keyboard. The only arguable question is that is it worth putting two microphones on each end of the tablet than wiring the keyboard up.

For generalization development, although we don't have enough time to do it, we gave some thoughts about that. To generalize the model, we must consider two more variants– keyboard layout and microphones' position. To train the model to adept for different keyboard layouts, we can simply change the output from key prediction to coordinate prediction. With the correct coordinates and keyboard layouts known, we can easily extrapolate which key was stricken. For microphones' position, we can use prompt tuning technique to fine-tune the existing model. We can encode microphones' position into prompt input and train the model to adapt for different microphone's position.

Although this model might not be developed into a commercial product, it reveals a security threat for computer users. It is possible for the hackers to record everything you typed without hacking into your system. Such privacy invasion leaves no trace. Most of sensitive information can be easily extrapolated. So, the most valuable part of this project is the introduction of new vulnerability of cybersecurity.

5. Summary and conclusions

In this study, we developed a real-time system capable of efficiently capturing the sound features of a keyboard. This system facilitates the building of datasets, training of CNN models, and enables real-time prediction of pressed keys using streaming keystroke audio data. When the same devices (keyboard and microphone) are used in same relative position, our method demonstrates considerable accuracy (94%) in real-time keystroke sound prediction, provided the noise level is moderate or lower, akin to background music. However, noise levels exceeding the keystroke volume pose a significant challenge to prediction accuracy. Our denoising function, while effective in noise cancellation, also alters the original keystroke features, leading to substantial negative impacts on the CNN model's performance. We believe that a more suitable denoising approach for this task exists, and future efforts should focus on leveraging the inherent robustness of neural networks against noise.

References

Harrison, J., Toreini, E., Mehrnezhad, M., 2023. A practical deep learning-based acoustic side channel attack on keyboards, in: 2023 IEEE European Symposium on Security and Privacy Workshops (EuroSamp;PW), IEEE. URL: http://dx.doi.org/10.1109/EuroSPW59978.2023.00034, doi:10.1109/eurospw59978.2023.00034.

Zhuang, L., Zhou, F., Tygar, J.D., 2009. Keyboard acoustic emanations revisited. ACM Trans. Inf. Syst. Secur. 13. URL: https://doi.org/10.1145/1609956.1609959, doi:10.1145/1609956.1609959.