Mechanical Keyboard Keystroke Predictions(Draft)

Saleh, Basem
Johns Hopkins University
Engineering for Professionals
Baltimore, US
bsaleh2@jhu.edu

Abstract-This project seeks to experiment with the paper titled "A Practical Deep Learning-Based Acoustic Side Channel Attack on Keyboards." The goal of the paper was to show how keystrokes of a keyboard can be identified through a neural network using the audio information grabbed. Members of Durham University detailed a process of extracting keystrokes from a laptop keyboard after using a phone microphone only a few inches away to record audio. For the project within this paper, the focus was placed on recreating this same experiment with a similar data collection process, but to focus on diversifying the dataset to include different mechanical keyboard switches of tactile and linear switches. In addition to training on different mechanical switches, the CoAtNet model used was modified to become a fusion model, so it can accept additional keyboard information that is non-auditory to help further training. Results shown indicate reliable performance for single key prediction, but poor across multiclass predictions and with a limited dataset collected.

Keywords—CoAtNet, fusion, keystroke audio, mechanical keyboards, classification

I. Introduction

Cybersecurity remains a very important field within the technology field. Opportunities consistently arise for new methods that hackers of various backgrounds employ in order to steal data from unsuspecting victims. These methods normally include phishing emails or nefarious links that users will open or click that allows access to a user's personal information. In the form of access to their computer or, if the hacker wanted to take information, they would create forms or emails that told users their personal information was required[1].

As decades pass, there has been a stronger emphasis on password security. Making passwords longer, creating two-factor authentications, and in addition, password managers to help better secure passwords. Companies like BitWarden or LastPass offer these capabilities, but are still prone to cybersecurity risks. LastPass[2] had a massive breach that caused password vaults to be leaked and if the vault wasn't given a secure enough password, it could be cracked and all passwords for various sites could be stolen.

While these are good for general security, an old method of finding passwords is still s risk: key-logging. This type of program will consistently record all keystrokes performed by a user once the file reaches the computer[3]. That data is then fed back to a source in real time over the internet or if the malicious user knows of this computer, can grab the recorded data off of it. Recent developments have occurred that now been discovered that allows keystroke prediction without the need getting into the computer itself but rather utilize just an external microphone.

II. Hypothesis

This project will focus on predicting keystrokes using recorded data from a MacBook Scissor keyboard, a linear mechanical keyboard, and tactile mechanical keyboard to attempt to prove the alternate hypothesis to accurately predict correct key presses using a machine learning model trained on different key switch keystroke data with an 80% accuracy or greater. Otherwise, reject the null hypothesis of training a model that is no better than random guess of keyboard strokes. This will check to see if the model requires further understanding of the keystroke audio, such as keyboard material and keyboard sizes to better predict keys. In addition, experiments will vary by limiting the scope to only one linear keyboard and completing binary classification as well rather than on multiple keys on the keyboard. A fusion model will ingest both audio data, through Mel-spectrograms passed through the CoAtNet, combined with tabular data of keyboard sizes and switch types.

III. Related Work

A. Deep Learning on Acoustic Side Channels

The paper[4] this project focuses on accomplished the ability of predicting keyboard strokes using a trained neural network. The paper focused on data that was collected from a single MacBook Pro keyboard while recording keystrokes from a phone microphone only a few inches away to the side. In an effort to better preserve the controlling nature of the project, the phone was kept on a thick cloth and each desired key was pressed 25 times at varying pressures. This allowed them to create many data points for all of the keys required.

In addition to recording right next to the laptop, the model also predicted keystrokes over a Zoom call after recording data from keys in the same Zoom environment. Accuracy was very high, around 92% over zoom and 95% without zoom.

One part of this paper that stuck was that this was only done on a MacBook Pro keyboard. This keyboard has a specific scissor design that is different than traditional membrane keyboards found on laptops or desktops, and also mechanical keyboards. While this paper proved useful in predicting keystrokes on a scissor keyboard, the chance to see if mechanical keyboards can have keystrokes predicted in a similar fashion would prove a bigger challenge.

B. SoK: Keylogging Side Channels

The paper SoK: Keylogging Side Channels[5] provides a comprehensive overview of the evolution and techniques of keylogging side channel attacks. This paper outlines how keylogging methods have diversified, leveraging various side channels including acoustic and electromagnetic emissions, user finger and hand movements, and microarchitectural features of

host computers. The paper categorizes these attacks into spatial and temporal side channels, with spatial channels revealing physical key locations or key pair similarities, and temporal channels exploiting key press and release timings.

The study evaluates the effectiveness of idealized spatial and temporal keylogging channels, showing that significant information gains can be achieved even with considerable measurement errors. It emphasizes that for temporal side channels, the information obtained is closely related to typing speed and style. The paper also highlights the persistence and evolution of keylogging attacks, noting the increasing complexity and sensing capabilities of computing devices. It underscores the challenge of mitigating such attacks and advocates for further empirical research to enhance device security and understand the implications of human-computer interaction on privacy.

C. *Robust Keystroke Transcription from the Acoustic Side-Channel*

This paper[6], by Two Six Labs, demonstrates that achieving accurate keystroke detection is crucial for effective keystroke classification and that traditional methods, such as power thresholding, fall short in realistic scenarios. These conventional approaches struggle with overlapping keystrokes and common office noises, including ambient sounds and mechanical disturbances. In contrast, the proposed deep learning system, which employs an end-to-end audio-to-keystroke model, shows superior performance by integrating detection and classification into a single framework. This model reduces the end-to-end character error rate (CER) on English text from 36.0% to 7.4% for known typists and from 41.3% to 15.41% for unknown typists, outperforming previous methods even when ground truth keystroke timings are available.

The novel system does not require precise keystroke timing information for training, relying instead on transcripts of keystroke sequences, which alleviates challenges related to timing precision due to keyboard polling latency and other delays. Although the approach assumes the audio input pertains to a typing session and does not handle non-typing gaps in the dataset, the model's robustness to noise and overlapping keystrokes remains notable. Future work should focus on expanding the dataset to include a variety of keyboards, environments, and microphone placements to enhance generalization. Additionally, exploring the use of multiple microphones and incorporating new deep learning architectures could further improve the system's adaptability and performance in diverse conditions.

The three papers train various models to help with keystroke prediction. There is emphasis on tackling work across different types of keyboards, microphones, and environments. So, a focus of this paper is developing a recording setup that can be easily distributed and used to record data from different users who are enthusiasts of different keyboard types. Since recording different keyboards from different users includes different environments, this also tackles the issue of different microphones being utilized within different rooms as well.

In addition, keyboards will have various material compositions, switch types, and may introduce background noise in order to better classify them.

IV. DATASETS

There are various keyboards available to consumers, such as mechanical switches with clicky, tactile, and linear type switches from different brands[7]. Clicky switches are bumpy and are very loud, making a clicking sound on the press and release of a button. This includes switches similar to a Cherry MX Blue or Razor Green switches. Tactiles are slightly less noisy, depending on the chosen brand of switch. They can emit a loud sound on the press, but may not emit a sound on the release. And lastly, linear switches boast no sound as there is no NumPy or clicky feeling when pressing the switch. It solely relies on features such as the material the switch is made of, the material of the keyboard base, and many other factors. However, when it comes to sound, those are the most important features.

In addition, non-mechanical keyboards mostly mimic general computer keyboards that are considered membrane-type keyboards. Instead of individual switches, a connection is made through a plastic, membrane like layer between the key and the computer keyboard itself[8]. Most laptops and office peripherals would have this keyboard. In addition, MacBook's have the Scissor keyboards that, while not fully membrane, have a unique leveling arms that press into a lighter membrane but return a light, mechanical feel.

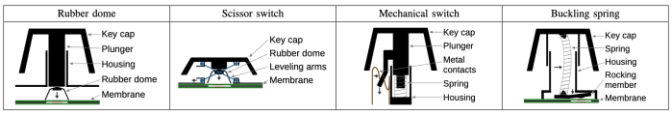

Fig. 1. Common Key switches[5]

A. *DeepKeyAttack*

The DeepKeyAttack dataset is the public repository that houses the same data used in the original paper[1]. There are two sources of data: MBPWavs and Zoom. Both folders have files that are named with just the key the data was recorded from with each one containing twenty-five presses evenly spaced within the program. Data was recorded on a 16-inch MacBook Pro M1 Pro on top of a microfiber cloth to reduce vibration.

B. *Keyboard Recorder*

The Keyboard Recorder Dataset[9] is a dataset collected using a personal repository from Johns Hopkins University's(JHU) student under the name bsaleh2. This repository houses the data collection scripts and a link to the recorded data within a JHU OneDrive folder. This repository aimed to repeat the same recording experiment from the DeepKeyAttack dataset.

The recording scripts instruct the user to place the microphone next to the keyboard, and then use two scripts. The first one to record general information, such as the type of key switch used, keyboard material, keyboard size, and the brand name of the keyboard. The second script allows user to select the key they want to record, and then record said data. After completing a key, users are prompted to move to the next key to continue recording.

The completed dataset used for this paper includes a combination of the DeepKeyAttack's MBPWavs non-Zoom audio and two additional users from utilizing the Keyboard Recorder repository. Together, it is a combination of a Linear switch recording, a Tactile switch recording, and a scissor keyboard recording(from MBPWavs). There are In total, 48 leys were recorded, consisting of all of the letters on a keyboard, all of the numbers above the letters, punctuation(without holding the shift key), and the space bar. The MBPWavs inclusion does not include the punctuation and instead focused only letters and keyboards.

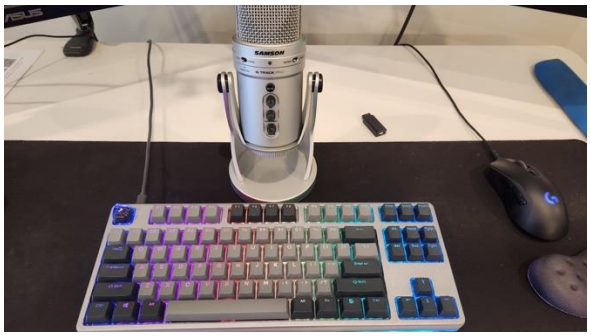

Fig. 2. Recording setup of Tactile portion of Keyboard Recorder dataset.

The Linear and Tactile recordings utilized an older method of recording one key press at a time, where the recording would start and stop after every pulse, ensuring collection was accurate. MBPWavs was already recorded with 25 pulses in a single file for each key. In total, after processing(detailed in later sections), 1840 samples of key presses are present. The linear and tactile switches include 480 each and MBP wavs include 880 for the scissor switches.

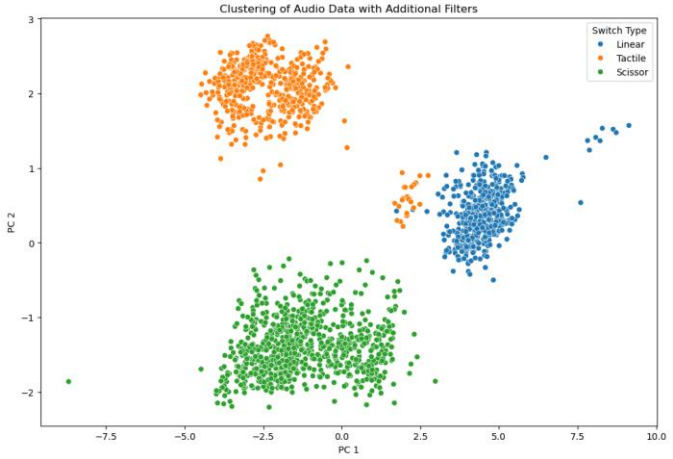

Fig. 3. Clustering of audio features, reduced to two PCA components for plotting. Hue is the type of switch from the Keyboard Recorder dataset.

To get a better understanding of the recorded data, each extracted pulse was placed through processing to extract thirteen Mel-frequency cepstral coefficients. This was chosen to extract features of the pulses while reducing dimensionality[10]. This is a commonly used feature in automatic speech recognition. After returning the coefficients, they were put through a PCA algorithm to extract two components[11], which were then grouped into 3 clusters through Kmeans clustering[12]. The figure above shows clear distinction between the different audio pulses.

However while the pulses look separated from each other due to different switch types, this does not account for the different microphones used for the various recordings of each dataset[13]. So, instead of the pulse itself, there is potential that this is just the microphones being separated into different clusters and the clusters also just happen to have the different switches for each recording. A more substantial recording dataset could help further get around this.

V. Experiment Overview

The first part of the experiment relied on the creation of the Keyboard Recorder dataset. There existed no common library or repository that gives the ability to record these keystrokes. So, the first part of this experiment relied on the creation of this data recorder.

In addition to the recording, proper audio file preprocessing was required to take place. Each audio file was a recording of twenty-five keystrokes. To better isolate the strokes to automatically, each file had its energy calculated, and then normalized across all of the data using a sliding window of size 50 samples. Then, two thresholds were placed on to the average sample: a lower threshold and an upper threshold. The lower threshold is set to .01%(0.001) to ignore noise that may enter the signal instead of a key. Then, an upper threshold is set to 1%(0.01). These were tested across various keys to get a threshold that can best get all of the keys from an audio file without accidentally grabbing unknown noise artifacts. And values within a continuous range of the signal where it exceeds the noise and upper threshold is considered a pulse. Sample locations are calculated from the original signal and cut as separate files for processing. Since this method doesn't grab everything properly and the keystrokes are given at varying pressures, not all 25 samples are collected per key using this method.

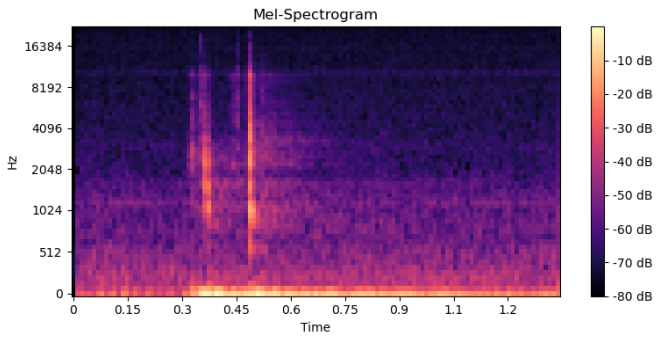

Fig. 4. Mel-Spectrogram of the Z key from the Tactile keyboard recording.

Audio was then passed through the Mel-spectrogram[14]. A Mel-spectrogram is a version of a general spectrogram[15]. A spectrogram, in digital signal processing, is a visual view of the signal wave representing all frequencies present within the signal. Depending on how the axis is oriented X time and Y represents the frequencies within the signal. When plotting, the more red a spectrogram is, the bigger the amplitude is of the captured frequency.

This concept is then built-upon to form Mel-spectrograms, in which the unit of frequency is adjusted to Mels: a logarithmic scale more representative of how humans hear sound. After pulse extractions from the recorded keystrokes are complete, they are placed through a function to produce Mel-spectrograms with the number of Mel coefficients at 64, a window size of 1024, and a hop length of 500. This was done to mimic the results of DeepKeyAttack paper preprocessing.

In addition to the audio, each audio file is given addition data of the encoded keyboard size, encoded switch type, and keyboard material. This data is initially created in a tabular form and once completely encoded, will be used as additional input to the model, along with the Mel-spectrogram information.

The model used in this experiment is the Convolution and Self Attention Neural Network(CoAtNet)[16]. CoAtNet is a neural network architecture that combines convolutional layers and transformer encoders, aiming to leverage the strengths of both CNNs and transformers. The architecture is designed to improve efficiency and performance on various computer vision tasks by integrating local feature extraction capabilities of CNNs[17] with the global context modeling of transformers[18]. It's the same model as was done in the reference paper and will also be utilized this paper.

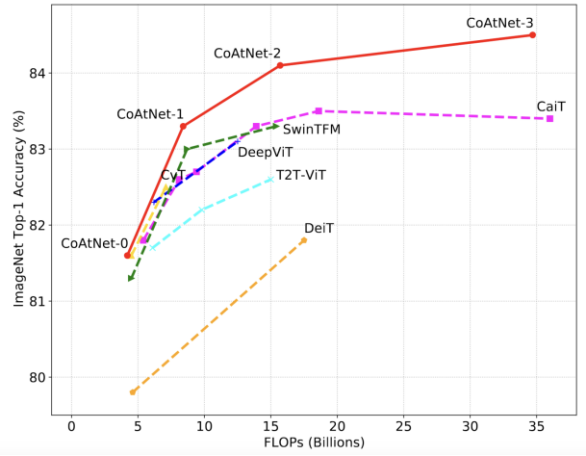

Fig. 5. Accuracy-to-FLOPS versus Top-1% Accuracy trained on ImageNet with images sized at 224x224[18].

In addition to the basic architecture that is provided by this model, it was also converted to a fusion format[19]. After the base model performs processing on the audio data and learns it, it's merged with results of another, smaller multi-layer perceptron[20] that only ingests the encoded features. Once the output from both models are done, they are concatenated to a final output linear layer before moving to a classifier for all of the given keys.

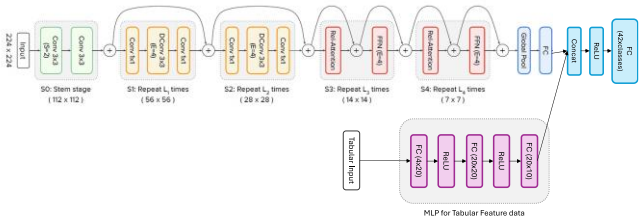

Fig. 6. Architecture of the CoATNet Fusion Modal. Top section is the CoAtNet before the fusion portion. Bottom is the Multi-layer Perceptron that processes feature data before concatenation. After concatenation, a final linear layer combines both outputs.

By combining both of these models together to format one output, this accomplishes the goal of a network both learning the audio distinction of individual keys on a keyboard based on location and at the same time, develops filters between different types of encountered keyboards, based upon what it's initially trained on.

In running these experiments, the most important piece is to check if the model is able to correctly identify a single label. By doing this, it will show the model is capable of predicting the correct letter from all keys. So, accuracy was chosen as the metric. Papers in the Related Work section showed the same metric as their top measurement for indicating a model's success in keystroke prediction.

An additional note is that after the collection was done for these experiments, certain non-audio features were dropped, such as keyboard material and an encoding between mechanical and membrane. Due to a smaller recording sampling, instead, the features passed through the MLP of the network are an encoding between tactile, linear, and scissor switches. The keyboard sizes were different, so that was ordinally encoded and kept in as well. All other features were dropped.

A. *Experiment #1*

Experiment #1's goal will include all recorded data from the Keyboard Recorder dataset. This will be the main focus of the hypothesis as it encompasses tactile and linear mechanical keyboard switches, along with the scissor keyboard switches from MBPWavs. All keys encompass a total of 48 classes. Successful accuracy metrics from this will show that the model can predict between various switches and under different, recorded keyboards.

B. *Experiment #2*

Experiment #2 will recreate the same experiment, but this time will only include the tactile switches from the Keyboard dataset. This experiment will see if predictions on tactile switches are successful, but also limit the space to only one recorded microphone from the user who recorded this dataset.

C. *Experiment #3*

Experiment #3 will now reperform Experiment #1, but rather than have 48 classes, it will perform binary classification of a single letter, 'H'. 'H' was chosen as the letter as it is the center of the keyboard. Any potential undetected features that may arise from being at the edge of the keyboard and a more common letter to hit than punctuation and numbers will show if the model can accurately see at least one key from a keyboard. It will also be able to distinguish between the three different microphones and keyboard switch types.

D. *Experiment #4*

Experiment #4 will combine the class reduction of Experiment #3 with the reduction in dataset of Experiment #2. Only the tactile switches will be used to predict the letter 'H' on the keyboard. This will help further remove the microphone

feature that may be present among the data, along with reducing the classification space.

E. *Experiment #5*

Experiment #5 will be similar to Experiment #4, but instead will train to learn the spacebar instead. Since the spacebar is a much more differently shaped key on all keyboards and has the most distinct area for users to press on, it has the biggest potential to produce different source based on pressure and pressure on certain areas. This will work with Experiment #4 to see if a space bar is just as easy or hard to predict as a normal letter on a keyboard and if size affects predictions.

VI. Training Procedure

The training procedure utilizes modified training scripts from the Deep Key Attack repository to train the CoAtNet Fusion model. Edits include the dataset modified to also ingest tabular data in the form of a pickle file in conjunction with loading in each respective audio file. The training loop was also modified to incorporate tabular features in addition to the labels and data points from the batches of data. This was the same model used for all of the experiments. All experiments initially attempted to mimic the DeepKeyAttack paper by training on 1100 epochs with a learning rate of 5e-5(Adam) and a batch size of 16. However, with this unique dataset, 1100 epochs proved to be too much. The graphs shown below were rerun with the same random state before overfitting occurred using new epoch sizes of 250 epochs, 60 epochs, 500 epochs, 300 epochs, and 500 epochs respectively for each experiment. All experiments were run on an NVIDIA 3080 GPU.

Data was split with 80% training and 20% testing. Accuracy was calculated from the test sets. Cross Entropy Loss was used as the loss function.

VII. Results and Analysis

TABLE I. Keyboard Recorder Experiment Accuracy Metrics

Experiments	Classes	Keyboard Recorder Dataset	Accuracy
DeepKeyAttack	0-9, a-z	MBPWavs	95.00%
Exp. #1	0-9, a-z, punctuation	Full	23.64%
Exp. #2	0-9, a-z, punctuation	Tactile	1.04%
Exp. #3	‘h’	Full	98.64%
Exp. #4	‘h’	Tactile	96.87%
Exp. #5	‘space’	Tactile	94.79%

The table here highlights the accuracy of each experiment performed, the classes used during the experiment, and which portions of the Keyboard Recorder dataset used. The experiment from *A Practical Deep Learning-Based Acoustic Side Channel Attack on Keyboards* is included here for comparison purposes.

A. *Experiment #1*

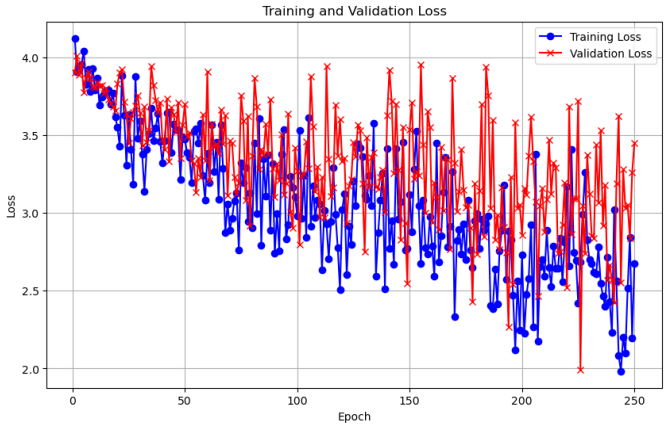

Fig. 7. Experiment #1 training and validation curves on the CoAtNet Fusion model. Using all of the Keyboard Recorder dataset

While convergence was continuing on the training and validation curves, after 250 epochs, the overfitting was then present as the validation curve began to increase in error while training decreased. Table I indicates the accuracy after training to be close to 24%. While more training epochs show it fitting better on the data, total accuracy never exceeded beyond what’s shown on Table I.

B. *Experiment #2*

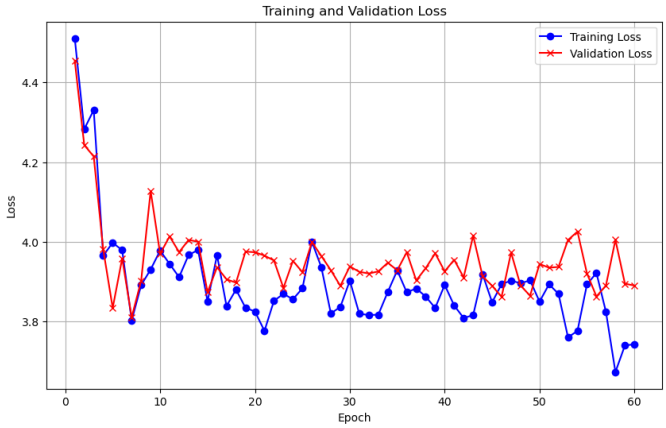

Fig. 8. Experiment #2 training and validation curves on the CoAtNet Fusion model. Using only the tactile keys of the Keyboard Recorder dataset.

Training on just the tactile keys of the dataset prove to be very difficult. After the 60 epochs shown here, there was overfitting present on the data with the validation curve slowly growing over time. The tactile keys making up only 26% of the data that’s present. Due to the smaller set, and because no data augmentation was used for the data, it proved to be much more to learn on without additional keyboard datapoints. This is not sufficient enough to show if data from only one microphone proved useful when the sampling of this keyboard was too small.

C. Experiment #3

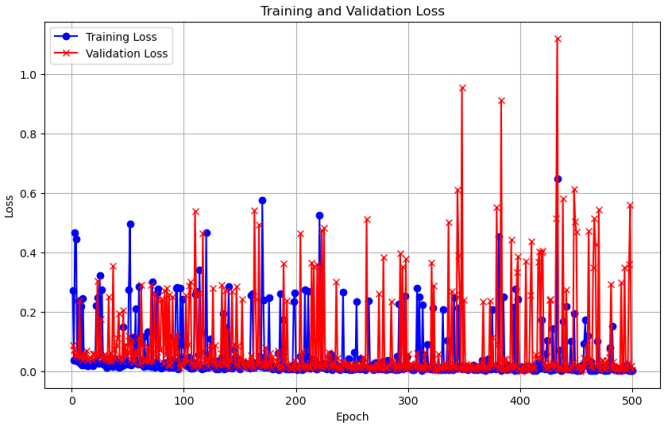

Fig. 9. Experiment #3 training and validation curves on the CoAtNet Fusion model. Using all of the Keyboard Recorder dataset for binary classification of the letter 'h' on the data

This model trained fairly quickly when working to identify only one letter rather than all 48 classes of letters on the keyboard. Table I shows the accuracy performance to be around 98%. This shows that rather than predicting all of the classes, only one letter proves to be well. Also, because of the inclusion of all three keyboards, more datapoints were provided and the extraneous feature of different microphone recordings was filtered away by the continuous learning of the model.

D. Experiment #4

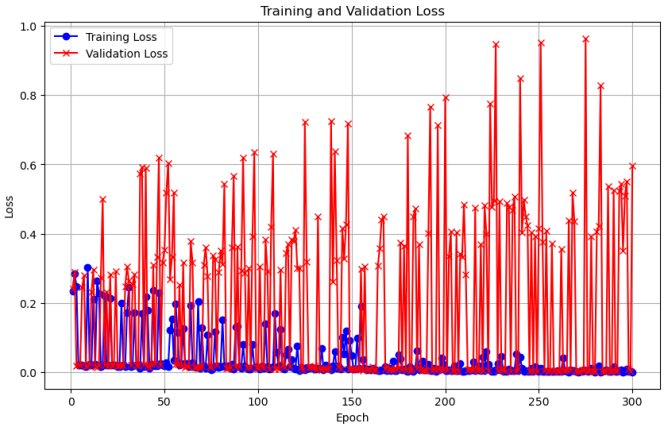

Fig. 10. Experiment #4 training and validation curves on the CoAtNet Fusion model. Using only the tactile keys of the Keyboard Recorder dataset for binary classification of the letter 'h' on the data

This model also trained fairly quickly when working to identify the letter 'h' within the dataset. While not all classes were used and the letter 'h' only posted 48 samples from 480 total samples of the tactile set, the sounds produced from the tactile keys proved to still be different from each other when only looking to predict one single label. This, however, does not take into account distinguishing between different microphones.

E. Experiment #5

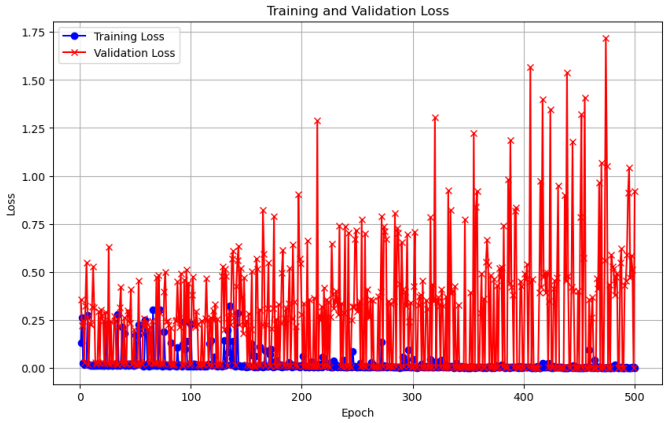

Fig. 11. Experiment #5 training and validation curves on the CoAtNet Fusion model. Using all of the Keyboard Recorder dataset for binary classification of the space bar on the data

The expectation was to have this prediction be much higher than the single letter 'h' from the previous experiment due to how different the space bar sounds compared to all other keys. However, the lower score of ~94%, versus the previous experiment getting roughly 97%, may be due in part to the different sounds produced from different points of the spacebar being hit during data collection, causing differing sounds. Overfitting was happening with the initial epoch size of 1100, so to reduce this, epochs were reduced to 500 with minimal validation error increases, but continuous, sporadic increases in validation error. Regardless, accuracy was still a very good metric.

VIII. Conclusion

With the completion of all experiments, with the collected dataset, the alternate hypothesis was to prove that the keystrokes between various keyboard switches could be learned. However, given Experiment #1 was designed to fully answer this, instead, the null hypothesis is accepted and the alternate hypothesis is rejected.

The tabular portion of the fusion model may have proved useful in helping to separate out different keyboard audio features to better classify labels. While not powerful for all 48 classes, it proved successful for binary classification for the letter 'h' when doing just one dataset. When not using multiple keyboards, the fusion model may not prove entirely useful as it is redundant information that does not change between audio pulses.

In regards to the poor performance of the first experiment, tying directly to the alternate hypothesis, there are a few reasons for this that may be answered through future research efforts. First, Further datapoints are required across different keyboards. While the full dataset used contained 1840 points, these are direct pulse recordings without any additional augmentation of the data. If further augmentations were done, more data could have been generated to allow for better accuracy across the various experiments.

All recordings were done on separate microphones. This introduces additional error as the model may instead be learning

features that were not intended, as shown by Figure 3. Future efforts should ensure either the dataset is all recorded under the same microphone and environment setup, or that the dataset is bigger and more diverse in recordings that the microphone may not be present as a significant learning feature.

More data is available for the scissor data versus the linear and tactile switches. This is due in part to how the linear and tactile recordings were initially recorded in with 10 pulses for key while the MBPWavs produced around 25 pulses per key. So, data is more skewed towards the scissor switches and future efforts should try to grow closer towards a more even dataset and also include more switch types, such as clicky type switches(i.e. Cherry MX Blue)[21].

REFERENCES

[1] J. Wayburn, "Two-Step Phishing Campaign Exploits Microsoft Office Forms," *Perception Point*, Jul. 25, 2024. https://perception-point.io/blog/two-step-phishing-campaign-exploits-microsoft-office-forms/ (accessed Aug. 11, 2024).

[2] Karim Toubba, "Security Incident December 2022 Update - LastPass - The LastPass Blog," *Lastpass.com*, 2022. https://blog.lastpass.com/posts/2022/12/notice-of-security-incident

[3] "What is a Keylogger? | How to Detect Keyloggers," *Malwarebytes*. https://www.malwarebytes.com/keylogger#:~:text=Keyloggers%20are%20a%20particularly%20insidious

[4] J. Harrison, E. Toreini, and M. Mehrnezhad, "A Practical Deep Learning-Based Acoustic Side Channel Attack on Keyboards," 2023. Available: https://arxiv.org/pdf/2308.01074

[5] John V. Monaco (2018). SoK: Keylogging Side Channels. *2018 IEEE Symposium on Security and Privacy (SP)*, 211-228.

[6] Slater, D., Novotney, S., Moore, J., Morgan, S., & Tenaglia, S. (2019). Robust keystroke transcription from the acoustic side-channel. In *Proceedings of the 35th Annual Computer Security Applications Conference* (pp. 776–787). Association for Computing Machinery.

[7] "Switch Types - Mechanical Keyboard," *Mechanical-Keyboard.org*, Nov. 26, 2015. https://www.mechanical-keyboard.org/switch-types/

[8] "Membrane Keyboards: Types, Uses, Features and Benefits," *www.iqsdirectory.com*. https://www.iqsdirectory.com/articles/membrane-switch/membrane-keyboards.html

[9] B. Saleh, "Keyboard Recorder," *Github*, Aug. 11, 2024. https://github.com/bsaleh524/keyboard_recorder (accessed Aug. 11, 2024).

[10] U. Kiran, "MFCC technique for speech recognition," *Analytics Vidhya*, Jun. 13, 2021. https://www.analyticsvidhya.com/blog/2021/06/mfcc-technique-for-speech-recognition/

[11] Z. Jaadi, "A Step by Step Explanation of Principal Component Analysis," *Built In*, Feb. 23, 2024. https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[12] P. Sharma, "The Most Comprehensive Guide to K-Means Clustering You'll Ever Need," *Analytics Vidhya*, Aug. 19, 2019. https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

[13] P. Sharma, "The Most Comprehensive Guide to K-Means Clustering You'll Ever Need," *Analytics Vidhya*, Aug. 19, 2019. https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

[14] "Mel spectrogram - MATLAB melSpectrogram," *www.mathworks.com*. https://www.mathworks.com/help/audio/ref/melspectrogram.html

[15] "What is a Spectrogram?," *Pacific Northwest Seismic Network*. https://www.pnsn.org/spectrograms/what-is-a-spectrogram

[16] Z. Dai, H. Liu, Q. Le, and M. Tan, "CoAtNet: Marrying Convolution and Attention for All Data Sizes." Accessed: Aug. 11, 2024. [Online]. Available: https://arxiv.org/pdf/2106.04803

[17] IBM, "What are Convolutional Neural Networks? | IBM," *www.ibm.com*. https://www.ibm.com/topics/convolutional-neural-networks

[18] A. Vaswani *et al.*, "Attention Is All You Need," Jun. 2017. Available: https://arxiv.org/pdf/1706.03762

[19] W. Li, Y. Peng, M. Zhang, L. Ding, H. Hu, and L. Shen, "Deep Model Fusion: A Survey." Accessed: Aug. 11, 2024. [Online]. Available: https://arxiv.org/pdf/2309.15698

[20] S. Abirami and P. Chitra, "Multilayer Perceptron - an overview | ScienceDirect Topics," *www.sciencedirect.com*, 2020. https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron

[21] "MX2A BLUE," *Cherry-world.com*, 2024. https://www.cherry-world.com/mx2a-blue#:~:text=CHERRY%20MX%20BLUE- (accessed Aug. 11, 2024).