The University of Nottingham

University of Nottingham, School of Computer Science

Predicting Key Presses over Video Calls using an Audio Side-Channel Attack and Machine Learning

Interim Report

Author
Adam Masters
COMP3003 - Computer Science with AI

Supervisor
Dr. Jamie Twycross(Computer Science)
pszjpt@exmail.nottingham.ac.uk

I hereby declare that this dissertation is all my own work, except as indicated in the text.

13th December 2021

Abstract

Due to COVID-19, an increasing number of Conferences and Meetings talking about potentially important topics are taking place on the Microsoft Teams Platform. Pairing this with the fact that background audio like keypresses can often be heard throughout these calls and now with the advancements in Machine learning techniques in the field of audio, it is worth asking if keypress data from background audio can be predicted and potentially obtained. This project will investigate this problem and evaluate the effectiveness of these techniques in real-world scenarios and if successful, will also aim to provide countermeasures that can be deployed to stop this form of cyber-attack. As of writing this interim report, early work has provided promising feasibility of this attack with a keypress prediction accuracy of approximately 30%. Although this is objectively low in the field of machine learning, this is significantly higher than randomly predicting a key and with many improvements planned for the future; it is a promising project.

Contents

1 Introduction

In almost all industrial sectors, the number of cyber-attacks in the modern computing world has been rising rapidly, and quantitative data suggests that the risk level of these attacks could be higher than ever due to COVID-19 [7]. As a consequence of COVID-19, the number of Microsoft Teams based video calls for sensitive and important conversations has also been increasing [13]. With many of these users not wearing high-quality headphones, background audio like the sound of keypresses can often be picked up within these calls. Additionally, considering the increasing amount of research into machine learning techniques for learning complex problems involving audio being undertaken and producing high-quality models that effectively solve these problems, it is worth investigating whether these can be applied to this audio to access and obtain personal data.

1.1 Background and Motivation

Over the past few years, there have been a rising number of cyber-attacks, utilising audio and acoustic eavesdropping, that have been demonstrated [1]. Ranging from using the sound of a physical lock turning to generate a replica key to recovering what a dot matrix printer processing English text is printing based on a record of the sound it makes [1] [3].

However, this form of attack is not just limited to devices that only some people have. For example, recently, several researchers have been investigating approaches to use the acoustics when a computer keyboard is interacted with to try and predict the key that was pressed to various degrees of success [5] [12] [6].

Now with the ongoing COVID-19 pandemic Microsoft Teams, a very popular video and telephony application, has grown to have over 145 million daily users, most of which communicate over video or audio calls every day [13]. Throughout these calls, often, a user is doing a secondary task like logging into accounts using a username and password. For example, in a recent study, 55% of workers have admitted to checking and responding to emails while on these calls [10]. As well as this, due to most users not wearing high-quality headsets keypresses and other background noise can often be heard while they are not on mute.

Considering all this, it is worth asking, with modern machine learning techniques, is it possible to extract personal information from just the background keypress sounds that occur in a Microsoft Teams call? And is an attack of this nature is feasible in the real world? Additionally, it is worth exploring what countermeasures can be used to prevent or block this attack and make users and readers aware of these.

1.2 Aims and Objectives

In particular, this project aims to apply and evaluate the effectiveness of modern machine learning techniques on keypress background audio from a Microsoft Teams call to investigate if this is a potential security vulnerability and what countermeasures can be implemented to stop this form of attack.

In order for this project to be successful and for the above aim to be achieved, the objectives are as follows:

1. To investigate the previous approaches to predicting keypresses based on their respective audio and consider how these can be applied to this project's aim and the steps required to implement these, if any are applicable.

2. Develop a key-logger that can generate a labelled data set of keypress audio files and their associated key from a Microsoft Teams recording.

3. Apply various machine learning models to the data set and perform statistical analysis on these approaches to determine if they are better than the proposed baseline "guess".

4. Investigate if external factors like using spellcheck and using LSTM can increase the accuracy and precision of the model [9].

5. Stretch Goal - Design and develop an application that, when executing, utilises the best-found machine learning approach and potentially external software to listen and predict keypresses outside of the data set in real-time.

6. Stretch Goal - Perform statistical analysis on the above stretch goal and evaluate its feasibility and identify what countermeasures can be implemented to stop this form of cyber attack.

2 Related Work

2.1 Previous Research

As mentioned in the previous section, research into acoustic attacks on keyboards has been undertaken before. One of the earliest examples of this was in 2004 in a seminal paper by Dmitri Asonov, and Rakesh Agrawal [2]. In this paper, they explored how training a neural network on keypress data recorded using a microphone from one metre away could be used to create a highly accurate model on specific keyboards. In some of their initial testing of distinguishing individual keys, they found that, on average, there were only 0.5 incorrect recognitions per 20 keypresses in one of their models, which is highly accurate and demonstrates high feasibility for an audio-based attack. Later on, in their final tests, when the user was typing multiple keys quickly, and the audio was

layered, the correct key was not found among the three candidates proposed by the network in only 12% of the tests presenting a high success rate of this attack concept.

This topic was later revisited in the "Keyboard Acoustic Emanations Revisited" paper, where instead of a labelled training data set, an unsupervised clustering algorithm was used based on the assumption that the targets typing would constrain to the rules of the English Language [15]. From a 10-minute inputted audio recording, the authors produced high accuracies retrieving 96% of typed characters at the end of their experiments. Although this approach is likely better than Asonov and Agrawal's approach, due to this project being focused on the Microsoft Teams platform, the chance of getting 10 minutes of clean typing audio to perform analysis on is near impossible and highly unlikely, so pre-trained models on labelled data sets will need to be used instead.

All this work was later built upon again in a 2017 paper, "Don't Skype & Type! Acoustic Eavesdropping in Voice-Over-IP", where this concept of an acoustic attack was applied to a Skype video call environment eliminating the need for physical proximity, which was a limitation in the previously discussed earlier work [5]. This paper discussed how they created a model based on audio received through Skype, which they then segmented based purely on the waveforms and extracted features using MFCC's. They then applied machine learning models and generated high initial accuracies of single keypress classification of above 80%.

2.2 Project Direction and Functionality

Leading on from the related work, in the "Don't Skype & Type!" the authors also have a section that discusses future work. In this section, they mention the need for this attack method to be applied to different environments, and with the rise of Microsoft Teams as a significant platform due to COVID-19, this seems like the perfect choice for further research.

Additionally, this project hopes to explore different data collection and pre-possessing approaches using keypress timestamps and onset detection and how these affect the final machine learning models performances. Furthermore, this project will also explore how spellcheck and other external tools affect the overall system's performance in a real-world setting and if it proposes a potentially even more significant threat to personal data.

Finally, a stretch goal of the project is to create a program that can use the best-found techniques to make predictions in real-time and to show the practical side of this kind of attack and demonstrate its success and to propose any countermeasures that can be implemented to avoid being a victim of it.

Functionally speaking, this stretch goal application should work by being a running program that when live or recorded Microsoft Teams audio is input, a

script of words is output to a file that the attacker can then read, and external tools like spellcheck can be applied to if the attacker wishes. Statistical analysis can then be performed on this by a structured series of controlled experiments to determine the feasibility of this kind of cyber attack in a real-world scenario.

3 Design, Implementation and Methodology

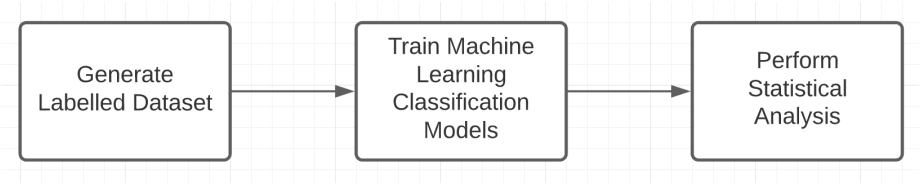

Figure 1: Overall workflow for semester 1 work

As discussed in the original work plan, this project could be split into several steps. In the first semester, the plan was to rapidly create initial models from a generated data set that could then be analysed and compared even if they were poor and then be used as a reference point to go back and improve each stage of the process. This agile, iterative approach to the process, visualised in figure 1, was chosen for the project as there are near-infinite optimisation and different approaches that could be taken or implemented at each step before the Machine Learning models are trained. However, by building out the whole pipeline quickly, the results initially obtained would provide a basis to build off and judge if future changes improved or compromised performance quickly and without making educated guesswork.

3.1 Generating the Data set

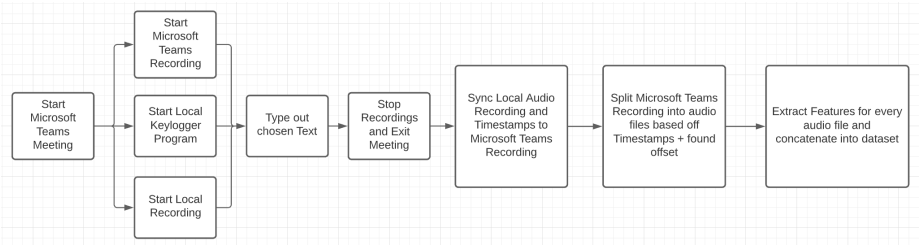

Figure 2: Generating the data set approach

Generating the data set to train the Machine Learning models is a critical part of the process; therefore, a good majority of the time spent during the first term was ensuring this data set was as good as possible. This process

aimed to generate a list of keypress labels with an accompanying list of features extracted from sound clips from a segmented Microsoft Teams recording where these segments were isolated single keypresses. Many methods could have been implemented to accomplish this utilising various audio analysis techniques, but the following approach was decided on, which can be seen in figure 2. The general idea behind this approach was as follows:

1. To start with, a Microsoft Teams meeting would be created, and the Start Recording button would be pressed to record the meetings audio.

2. Next, a participant in the meeting would begin typing out a chosen snippet of text. In the first "Test Bed" session, an act from Romeo and Juliet was chosen as it had a good balance of most characters and was a decent length. This choice is discussed in the next section and will most likely be changed for a "Test Bed Two" session in the following semester.

3. While this was happening, a local recording, as well as a key-logger would be running on the participant's machine. And as they were typing out the test set of data, timestamps of which keys were pressed and the time at which they were pressed would be recorded and stored.

4. Once the participant had finished typing all of the text, the meeting, local recording and key-logger would then terminate, producing two audio recordings and a list of timestamps.

5. Now the timestamps could be mapped up to the local recording, then this local audio could be mapped to the Microsoft Teams recording audio by a waveform comparison. By transitivity, this would then link the timestamps to roughly where they were in the Microsoft Teams meeting recorded audio.

6. Finally, the Microsoft Teams recording audio could then be split off these modified timestamps resulting in a list of audio files that linked to a list of keypresses.

7. Feature extraction could then be performed on these audio files and the results combined to create a labelled data set.

This approach was chosen over a purely audio analytical approach like what has been undertaken before in the "Don't Skype and Type" paper, as we believed it would lead to better wave isolation and fewer false positives existing in the data set. This was because, by having timestamps, sounds that sounded like a keypress in the Microsoft Teams Recording could be fact-checked with the timestamps, and if it did exist near the picked up audio, it could be confirmed as a valid keypress, but if it did not, then it was most likely a false positive and could be left out.

3.1.1 Timestamp and Audio Recording Generation

In order to undertake a timestamp driven approach, we first had to create a key-logger that created a mapping of times to the key that was pressed. This was relatively trivial to do in python using the keyboard module, creating a listener for the keys that, when pressed, created a list of tuples that were then saved to a file [4]. There were a few considerations that appeared during this stage that we had to take into account.

One of the first considerations was how the data should be stored and where it should be stored. It is common knowledge in machine learning that the more data there is, the generally better model that is produced and with keypresses being easily accessible and easy to produce, it is straightforward to generate a large set of data in a short amount of time which could lead to a large amount of storage being required. Additionally, with this problem being tied to the keyboard model, we wanted a key-logging system that could be used on other computers relatively easily and a storage place for this data retrieved by the key-logger where it could be easily accessed from other machines. Therefore, the key-logger used cloud storage or, more precisely, Firebase collections to store the data in a shared place that any machine with access could obtain, this also allowed good visualisation that helped with debugging and acquiring some basic information statistics of how much data we had.

Another Consideration was the precision of the timestamp. In one of the initial versions, the key-logger only saved the times of the keypresses to seconds. It was quickly realised that this was not precise enough after testing, but it then introduced the question of the minimum precision acquired. After doing some research into words per minute (wpm), on average, it was found that the majority of students age 17 - 27 had a wpm of 30-40 if we then take the upper bound of this and the average number of characters in a word which is 5.1 we can find the number of characters in a minute and thus the number of characters per second which was 3. [11] [14].

$$(40 * 5.1)/60 = 3.4cps$$

This backed up the theory that timestamps to the nearest second were nowhere near precise enough, but with python being able to produce differences in time to six decimal places by default, we were confident that the necessary level of precision could be reached.

With the key-logger now working and after numerous discussions with my project supervisor, we realised we would also need to record the local audio while the key-logger was running locally. Our final goal here is to use the audio data that comes from a Microsoft Teams Call; however, we quickly realised, due to delays in connection and meeting start times, that there was an offset between our keypress timestamps and the time it appeared in the meeting audio. To counter this, we decided to record locally as well. This then allowed us to

compare the waveforms, match up the timestamps to the right keys, and be confident we were not mismatching keypresses.

3.1.2 Audio Recording Synchronisation and Segmentation

Now that the key-logger above had been created, it was then used within a Microsoft Teams meeting to produce a five-minute audio recording of a participant typing out the second act of Romeo and Juliet. The second act of Romeo and Juliet was initially chosen as it has a decent spread of characters and leads to about 500 characters which was a reasonably large data set for initial testing. However, as later discussed, the choice of the text is a topic of future research as we realised there was not enough data when training the models with some keys not having many samples compared to others.

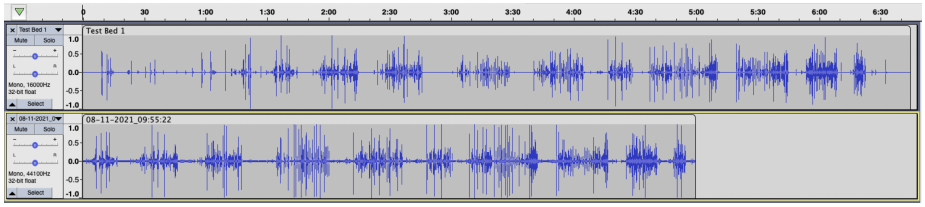

Figure 3: Microsoft Teams audio (Top) against local recording (Bottom)

After the recording was finished, we now had a local recording, a Microsoft Teams recording, and a set of timestamps mapping to the local recording that needed to be synchronised and segmented. To visualise this, we then loaded both of the recordings into Audacity, which can be seen in figure 3. From this, we can see that after an initial offset time in the Microsoft Teams recording, the audios appear to match up reasonably well, and so by dragging the local recording, we can get a good initial match and find an offset time(see figure 4).

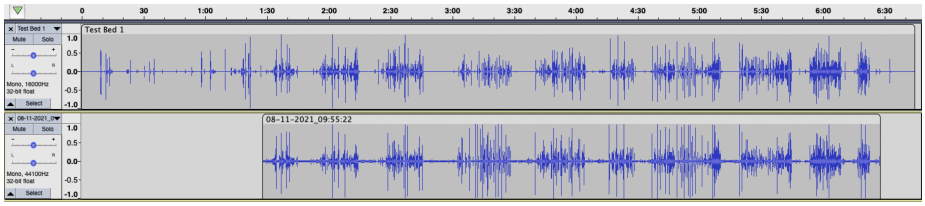

Figure 4: Audio recordings synchronised manually

With a timestamp offset now found and a Microsoft Teams recording, we could split the recording and produce a labelled data set. Therefore, using Sox and the list of timestamp letter pairings, we created a new audio file for each pairing which was a 0.5-second audio snippet from the timestamp given. 0.5 seconds

was initially chosen due to being a ballpark guess of how long a timestamp sound was from initial observations. This then resulted in 501 audio files being created. Using Mat lab plot, we then created some Mel-Frequency spectrograms to visualise these files and the waveforms they contained, and for most samples, these were positive visuals as can be seen in figure 5.

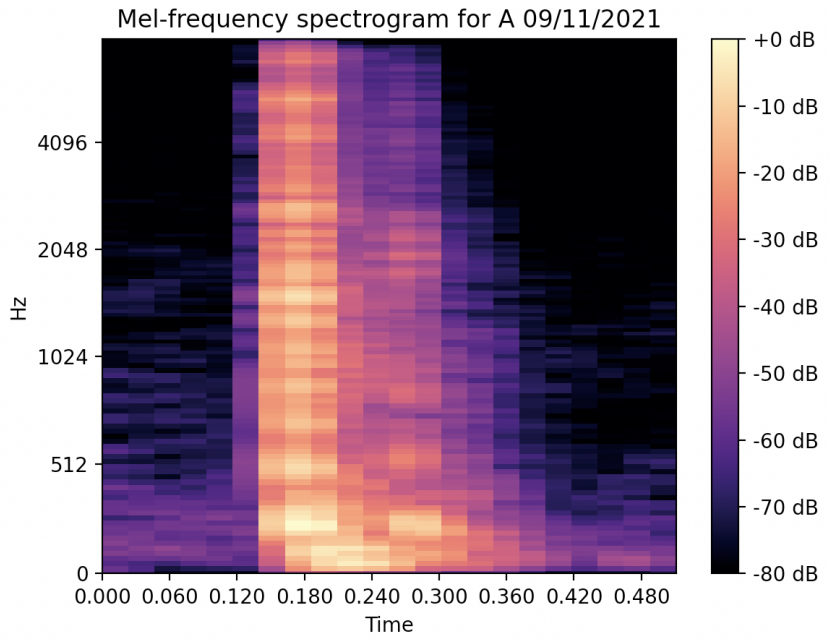

Figure 5: Successful Mel-Frequency Spectrogram of character A showing the waveform of a keypress

However, we quickly realised this was not the case later on for all audio files and that a fair number of them contained blank audio or noise that can be seen in figure 6. To figure out why, we turned back to the audacity waveforms of the initial recordings, and a problem we initially thought we would encounter became visible. When zoomed into the waveform that had been matched up previously, due to internet connection discrepancies, there was sometimes an offset between the local recording peak and the Microsoft Teams recording peak (see figure 7). This would lead to the blank files as if the offset was greater than the initial 0.5-second window; it would not be captured in the split audio file, leading to blank files.

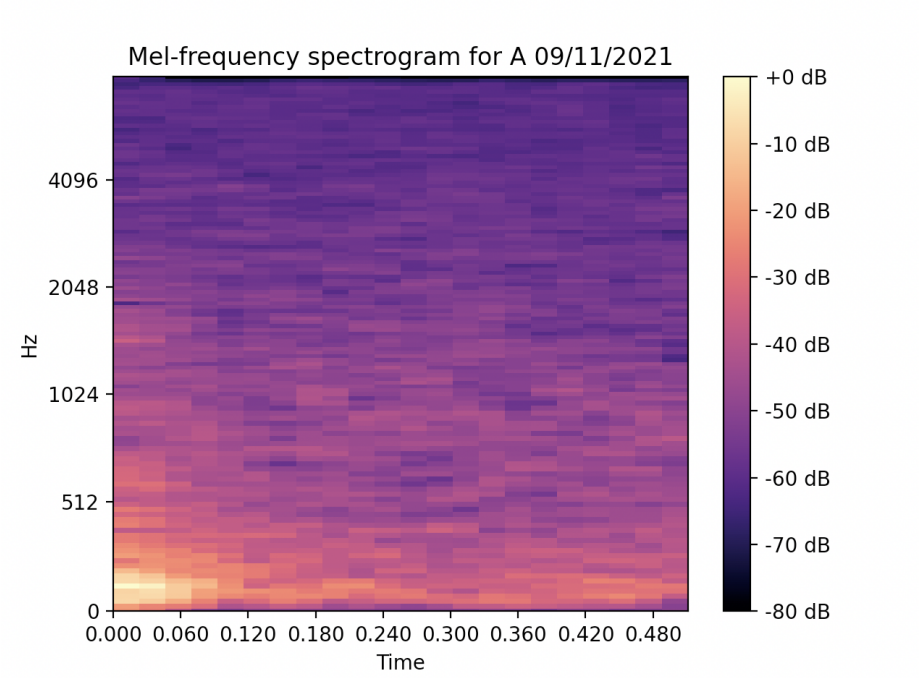

Figure 6: Empty Mel-Frequency Spectrogram of character A showing how occasionally the wave is not segmented properly

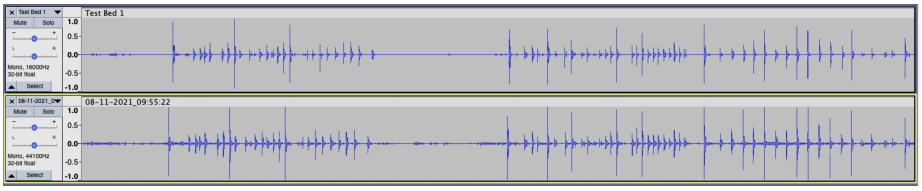

Figure 7: Potential manual audio syncing strategy problem

Consequently, in the following semester, the plan is to investigate solutions to this problem and implement some of them, for example, increasing the time window and implementing onset detection. Increasing the time window should allow us to capture the audios which do have a slight delay, and then by implementing onset detection, it should allow us to trim the file down so that it is only the found wave. By doing this and by implementing a function that removes empty files, if any still exist, it should improve the purity of the data set and increase the performance of the machine learning models.

3.1.3 Feature Extraction

Once the split audio files had been obtained, 20 features were extracted from each of these files using Mel-frequency cepstral coefficients (MFCCs) using Librosa [8]. We chose this feature extraction method as it was easy to implement, and in similar studies to this one, it was found to have the highest performance compared to other feature extraction methods like FFT features [5].

3.2 Machine Learning Models and Statistical Analysis

We now had a complete data set from the previous steps mentioned above and so could start training and evaluating classification models. For initial testing purposes, a K nearest neighbours model and a linear, polynomial and RBF support vector machine model were trained with 10 cross-fold validation and hyperparameter tuning. After the best parameters were found, the model was then trained on all the data and tested to produce the following results. As expected, the results were relatively poor due to the data set purity issues mentioned above.

	KNN	Linear SVM	Polynomial SVM	RBF SVM
Accuracy	18.4%	28.8%	23.2%	20.0%
Precision	15.3%	27.6%	21.1%	4%
F-Score	0.163	0.264	0.125	0.067

From this table, we can see that all the accuracies are relatively low; however, considering that we have 30 classes and a random guess approach would be approximately 3.3% accurate (found by dividing 1 by 30), this is better than a baseline guessing approach. We can also see that currently, the Linear SVM with a box constraint of 0.05 (found by hyperparameter tuning) performs the best after cross-validation on the data set. Since this was the case, we decided to analyse further the results obtained and, more specifically, the accuracy and precision of the specific labels. The result of this can be seen in figure 8. From this, we could quickly conclude that the main problem was a lack of data. For some classes like the letter 'E' and 'Space', we had a high number of samples in the test set; however, for others like 'B', we had two instances, and for some like 'Z', we even had 0, which is a clear indication we did not have enough data and that the data set was not balanced.

```
Best C found - 0.05
Now trained on full dataset...
              precision    recall  f1-score   support

           '       0.00      0.00      0.00         0
           ,       0.00      0.00      0.00         1
           A       0.50      0.71      0.59         7
           B       0.00      0.00      0.00         2
   Backspace       0.00      0.00      0.00         0
           C       0.00      0.00      0.00         2
           D       0.00      0.00      0.00         1
           E       0.29      0.40      0.33        15
           F       0.00      0.00      0.00         1
    FullStop       0.00      0.00      0.00         1
           G       0.00      0.00      0.00         5
           H       0.00      0.00      0.00         6
           I       0.09      0.33      0.14         3
           J       0.00      0.00      0.00         2
           K       0.00      0.00      0.00         0
           L       0.00      0.00      0.00         0
           M       0.00      0.00      0.00         3
           N       0.00      0.00      0.00         7
           O       0.29      0.36      0.32        11
           P       0.00      0.00      0.00         0
           R       0.00      0.00      0.00         6
           S       0.20      0.17      0.18         6
       Space       0.45      0.56      0.50        25
           T       0.50      0.36      0.42        11
           U       1.00      0.20      0.33         5
           V       0.00      0.00      0.00         1
           W       0.00      0.00      0.00         2
           Y       0.00      0.00      0.00         2

    accuracy                           0.29       125
   macro avg       0.12      0.11      0.10       125
weighted avg       0.27      0.29      0.26       125
```

Figure 8: Classification Report for Linear SVM on data set

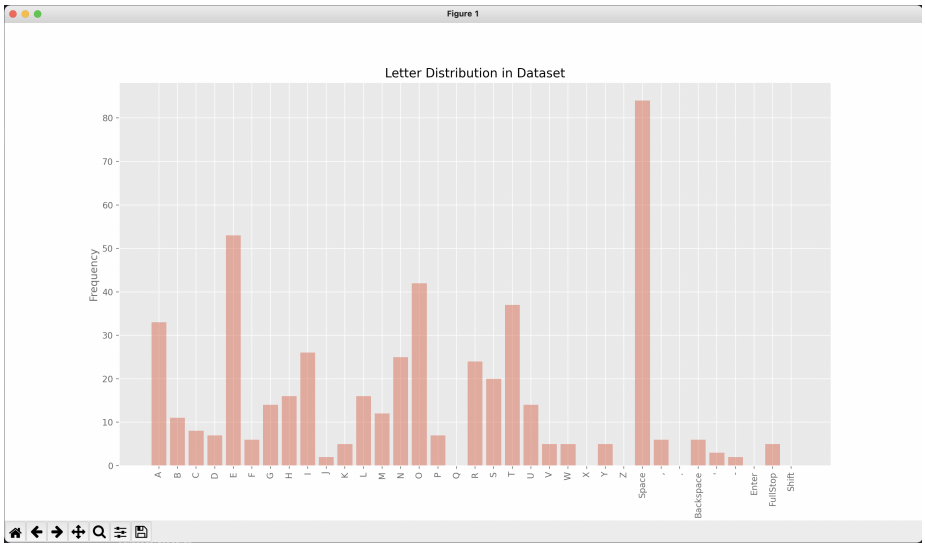

Figure 9: Class distribution in data set

To confirm this hypothesis, the spread of data against classes was then plotted to provide a visual of what the data set looked like as a whole, leading to the following graph shown in figure 9. From this, we can once again see that we do not have enough data as for some classes like 'X' and 'Q', we do not have any samples at all, and others have very few. This makes sense as the typed text came from Romeo and Juliet, which constrains to the English language, where certain letters occur more frequently than others due to the structure of words. So in conclusion, in the next term, one of the major plans is to collect more data and to collect a more balanced spread of classes by typing a specific number of each key instead of typing out a story that would conform to the constraints of the English language, this should lead to a more balanced data set and should therefore lead to better model performance.

4 Progress and Reflection

4.1 Project Management

As put forward in the initial Project Proposal, the goal for the first term was to create an initial version of the whole pipeline (from generating the data set to evaluating models). This approach was chosen over tackling one step of the process in depth at a time, as it would reduce the project risk of having nothing working by the end of the project if an issue that required prior steps to be reworked was encountered. This approach was effective as it allowed issues like the connection offset issue in the data set generation step to be encountered quicker than if the focus was purely on optimising the key-logger this term.

Additionally, weekly meetings took place to discuss the project's progress and any issues encountered, and/or improvements that could be made in the next few weeks. This worked really well as it enabled the discussion of problems early on, allowing progress and stopping any blocking factors, and it also made sure that development was constantly happening on the project and that there were no long periods of no development which could have hindered progress.

Furthermore, the GANTT Chart initially created during the project proposal, which can be seen in figure 10, was also really useful at tracking the progress made during the first semester, and where the project was meant to be at a specific point in time. Overall, the chart was stuck too; however, due to the data set impurity issues discovered in the first term and discussed above, a new data set will need to be created; hence the second term of the GANTT chart has now been altered in order to allow time for this.

4.2 Contributions and Reflections

The project so far has shown decent results for the approach taken. In particular, if considering a baseline guess approach has an accuracy of 3.3%, the fact that an accuracy of approximately 28% was achieved with a Linear SVM shows promising signs. If the data set impurity issues are also taken into account and other optimisation methods that will be implemented, the project has a high potential for much better performing models in the near future, which is an exciting prospect.

In terms of my own reflection, I am happy with the progress that has been made so far. With all aspects of the model creation process now having a working prototype, this has eliminated a lot of project risk related to a final functioning result. However, I think at the start of term, my communication with my supervisor could have been better. For the first few weeks, the progress was slow as I struggled to understand how I was going to collect the audio files from the Microsoft Teams meeting and tried a lot of different approaches, which all led to little success. If I had tried to arrange regular meetings earlier, I think more progress could have been made this term as I could have discussed ideas quicker and developed an approach that worked sooner. Therefore in the next semester, I hope to continue these weekly meetings from as early as I can as I believe they were vital to the success of the project so far.

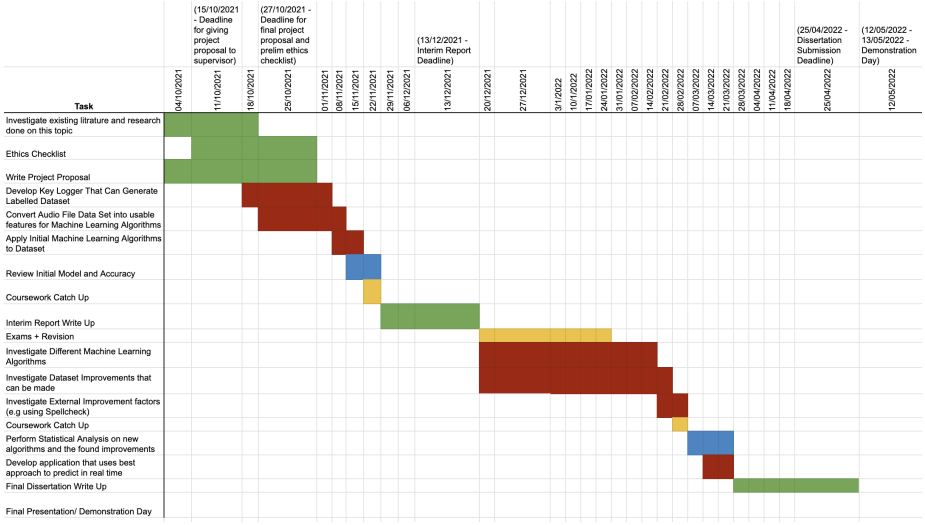

Figure 10: Original GANTT Chart Plan

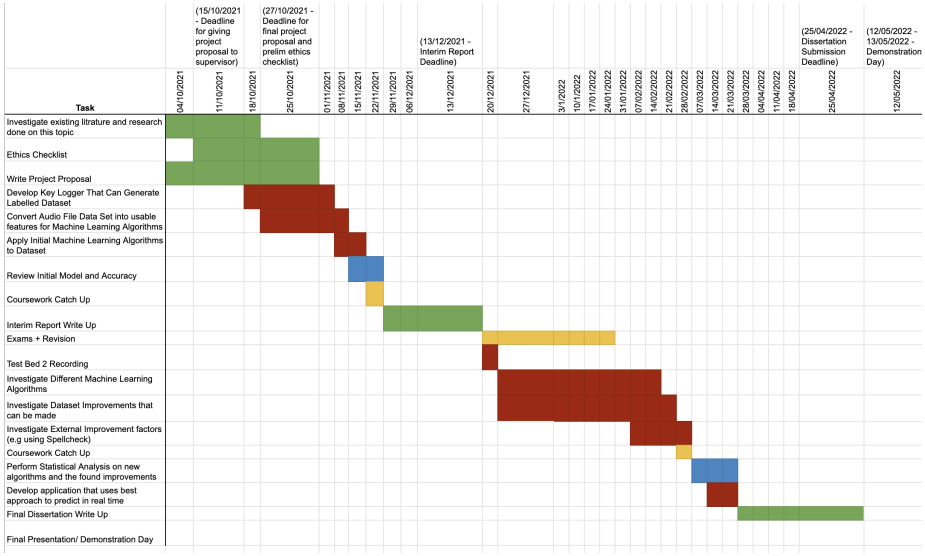

Figure 11: Updated GANTT Chart Plan

Key	Label
Reading/ Writing Deliverable Documents	
Development Time	
Catching Up / Free Time	
Reviewing Accuracy / Doing Analysis	

Figure 12: GANTT Chart Legend

References

[1] T. Anderson. Physical locks are less hackable than digital locks, right? maybe not: Boffins break in with a microphone., 2020. Last accessed 11 November 2021. https://www.theregister.com/2020/08/21/spikey_paper_acoustic_lock_pick/

[2] D. Asonov, R. Agrawal. Keyboard acoustic emanations. pp. 3 – 11, 06 2004. https://doi.org/10.1109/SECPRI.2004.1301311

[3] M. Backes, M. Durmuth, et al. Acoustic side-channel attacks on printers. pp. 307–322, 09 2010. https://doi.org/10.5555/1929820.1929847

[4] Boppreh. Python keyboard module documentation, 2021. Last accessed 28 November 2021. https://github.com/boppreh/keyboard#api

[5] A. Compagno, M. Conti, et al. Don't skype & type!: Acoustic eavesdropping in voice-over-ip. pp. 703–715, 04 2017. https://doi.org/10.1145/3052973.3053005

[6] G. Gerganov. Keytap: description and some random thoughts, 2018. Last accessed 08 November 2021. https://ggerganov.github.io/jekyll/update/2018/11/30/keytap-description-and-thoughts.html

[7] GOV.UK. Cyber security breaches survey 2021 official statistics, 2021. Last accessed 12 December 2021. https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2021/cyber-security-breaches-survey-2021

[8] Librosa. Mfcc's documentation, 2021. Last accessed 28 November 2021. https://librosa.org/doc/latest/generated/librosa.feature.mfcc.html

[9] S. M. Predict the next word of your text using (lstm), 2021. Last accessed 08 October 2021. https://www.analyticsvidhya.com/blog/2021/08/predict-the-next-word-of-your-text-using-long-short-term-memory-lstm/

[10] K. Morris. Virtual and zoom meeting distraction statistics [survey], 2020. Last accessed 03 December 2021. https://www.zippia.com/advice/virtual-meetings-zoom-survey/

[11] Online Typing. Average typing speed (wpm), 2021. Last accessed 3 December 2021. https://onlinetyping.org/blog/average-typing-speed.php

[12] C.D. Silva. Acoustic eavesdropping: Predicting keystrokes with google, 2020. Last accessed 08 November 2021. https://medium.com/swlh/acoustic-eavesdropping-predicting-key-strokes-with-google-automl-vision-b772b198d8ee

[13] J. Teper. Twitter status update, 2021. Last accessed 22 November 2021. https://twitter.com/jeffteper/status/1387141320519557120

[14] Wolfram Alpha. Average english word length, 2021. Last accessed 3 December 2021. https://www.wolframalpha.com/input/?i=average+english+word+length

[15] L. Zhuang, F. Zhou, J.D. Tygar. Keyboard acoustic emanations revisited. vol. 13, pp. 373–382, 01 2005. https://doi.org/10.1145/1609956.1609959