Acoustic Side-Channel Attacks using Keystrokes: Classifications and Real-World Applicability

Candidate Numbers: 24883, 29662, 34462, 80406

Abstract

This paper explores the comparative performance of different deep learning architectures, including Basic CNN, Denoising Autoencoder with Classifier, and ResNet for the task of identifying keystrokes from their Mel Spectrogram transformed audio files. This project is based on an original study which attempted the same task, but only tested out the CoAtNet architecture. We believe there is rationale for other architectures to perform just as well. The results showed that all models but our 'Author's CoAtNet' are promising, with ResNet, Basic CNN, and the Autoencoder two-stage all achieving at least 80% test accuracy. Even though the images have noise, the preprocessing and set up parameters of each model allow the structure of the input to be retained and are translated through with classifications being determined in the final layer.

1. Introduction

In this task we are seeing how a range of models compare in the task of identifying keystrokes based on melspectogram transformed audio recordings, specifically building off of Harrison et al. (2023). This project builds upon a previous study that focused solely on testing the CoAtNet architecture for the given task. However, we posit that there are valid reasons to explore the performance of other architectures that also take images as inputs and give classifications for outputs. Rationales for why each architecture was chosen are developed in the specific sections for those architectures in section 4.

The models we are using are the original CoAtNet from the aforementioned paper, ResNet18, a more basic model using a Convolutional Neural Network, and an Autoencoder with Classifier. We have chosen these are a range of models that exemplify different layer structures while still being comparable. Our main test accuracy results are: ResNet 80%, Author's CoAtNet 27%, CNN 85%, Autoencoder with classifier 93%.

2. Literature Review and Related Works

The history and evolution of Acoustic Side Channel Attacks (ASCAs) have been shaped by the development of technologies used for communication, as actors search for a way to intercept and understand them. ASCAs are a subset of Side Channel Attacks (SCAs) that exploit signals emitted by devices, such as electromagnetic waves, speech reverberations, acoustic emissions. They pose a very high risk, being both easy to implement and effective alongside the broad availability of data due the difficulty of totally eradicating exploitable audio or emission leakage (15), alongside the ubiquity of vulnerable devices and recording opportunities. SCAs and ASCAs have evolved into prevalent attack methods deployed on a diverse range of devices, including laptops, ATM pin pads (6), 3D printers (2), smartphones (7), and wearable devices (11). The field gained significant research attention in the mid-1990s when researchers began to demonstrate practical attacks that could exploit side channel emissions for cryptanalysis, though having been previously explored by classified governmental departments (14). Notably, Kocher's (1996) work on timing and power-consumption-based side channel attacks highlighted the potential and the risks of these methods(10).

The development of ASCA using keystrokes begin with the foundational Asimov and Agrawal (2004), who successfully applied ASCA techniques to interpret keystroke data from "PC keyboards, notebook keyboards, telephone and ATM pads" (4). Since then, ASCA research has diversified, with different projects employing various model structures and approaches. For instance, the interpretation of printer noises has achieved significant accuracy improvements by integrating models with language models for contextual information (from 72% to 95%)(5). Keyboard geometry affects the triangulation of sound by the recording system, and allows for analysis without relying on linguistic context (16)(13).

Other approaches have sidestepped individual keystroke identification and instead used solely linguistic constraints to decipher a 10 minute sound, with general success (15). In contrast, Harrison et al. (2023) have highlighted the effectiveness of direct recordings in achieving high accuracy rates in keystroke exploration directing this paper's approach (8).

3. Data

The dataset utilised in this study was derived from the original research conducted by Harrison et al. (2023) titled ”A Practical Deep Learning-Based Acoustic Side Channel Attack on Keyboards” - ’JBFH-Dev/Keystroke-Datasets’ used under the MIT access license (9). The experiments were carried out on a MacBook Pro 16-inch whose keyboard shares identical switch design with models from previous years, ensuring the relevance of the findings across multiple device iterations (see Zhu et al., 2014; Taheritajar et al., 2023).

In the experimental setup, the following recording mode was employed: phone-recording mode. In the phone-recording mode, an iPhone 13 mini was positioned 17cm away from the leftmost side of the laptop on a folded micro-fiber cloth to minimise desk vibrations. Stereo recordings were made at a sample rate of 44100Hz with 32 bits per sample, emphasising the acoustic aspects of keystroke detection.

Each keystroke was recorded in WAV format, with a single file containing 25 consecutive presses for each of the 36 keys used in the experiments. Notably, the absence of shift keys, caps lock, or space bar usage in the experiments highlights the focus on individual key press dynamics. Thus, the total unaltered dataset gives 900 observations from the phone-recording mode.

The experimental setup, although capturing keystrokes through microphone recordings, aims to simulate real-world scenarios. However, it’s important to note that the controlled nature of the experiments may introduce limitations regarding the true variability and complexity of everyday typing environments. Nonetheless, by leveraging these data sources, the study endeavours to approximate real-world conditions, thereby facilitating the development of robust and applicable deep learning models for keystroke analysis and security applications.

3.1. Pre-processing

Data pre-processing plays a pivotal role in preparing raw keystroke recordings for subsequent deep learning analysis. This section outlines the key preprocessing steps involved in isolating keystrokes and extracting relevant features from the recorded data. The preprocessing code was guided by that done by Harrison et al., 2023 (9).

Keystroke Isolation: The first preprocessing step involves isolating individual keystrokes from the recorded audio files. A common method employed in recent literature involves performing the fast Fourier transform (FFT) on the recording and summing the coefficients across frequencies to compute ’energy’. An energy threshold is then defined and applied to identify the presence of a keystroke. Notably, keystrokes isolated from this data were standardised to a fixed length of 14400 (0.33s) whilst ensuring that all 25 presses from each file were coming through clearly.

Feature Extraction: Following keystroke isolation, features are extracted from the audio data to facilitate deep learning model training. Various methods for audio feature extraction exist, including the Fast Fourier Transform (FFT), Mel-Frequency Cepstral Coefficients (MFCC), and Cross Correlation (XC). In this study, we propose the use of mel-spectrograms as a feature extraction method for deep learning models as they proved most successful in the original study and their ability to capture essential acoustic characteristics of keystrokes in a frequency-time domain, which aligns well with human auditory perception. A mel-spectrogram is a modified version of a spectrogram that represents sound waves as a map of coloured pixels, with frequency on the Y-axis and time on the X-axis. Unlike traditional spectrograms, mel-spectrograms utilise a logarithmic scale called mels. Through initial experiments, mel-spectrograms were found to visually represent sound in a recognisable manner, leveraging deep learning’s efficacy in image classification tasks.

3.2. Data Augmentation

Data augmentation is a crucial technique used to enrich the training dataset by generating additional observations through various transformations applied to the original samples. It is also especially important in this study since the original dataset is only 900 observations, which for these models tends to be quite a small number. In this study, we employ four alteration processes: addition of Gaussian noise, time masking, frequency masking, and time shifts.

Addition of Gaussian Noise: Gaussian noise is introduced to original audio samples to simulate real-world variability and enhance the model’s robustness against noise interference. The noise is generated with a standard normal distribution, and its amplitude is controlled by the noise_factor. This factor adjusts the intensity of the added noise, which introduces random fluctuations in the signal. By mimicking environmental noise, this process helps improve the model’s ability to generalize to unseen data.

Time Masking and Frequency Masking: Time masking and frequency masking are employed to further diversify the dataset and improve model generalization. Time masking directly modifies the audio signal by setting segments of it to zero, effectively obscuring temporal information. This is controlled by the mask_duration parameter, which determines the fraction of the total signal duration that will be masked. Similarly, frequency masking is applied in the frequency domain representation of the signal, obscuring consecutive frequency bands. This is governed by the frequency_width parameter, which specifies the proportion of total frequencies to be masked. Both parameters can be adjusted to increase the severity of the masking, thereby enhancing the model's robustness to variations in audio data. A value of 10% of the total signal length for both time and frequency masking was chosen, to most closely align with Harrison et al.'s (2021) approach.

Time Shifting: Additionally, time shifting is applied to the augmented audio samples to introduce temporal displacement. This process involves randomly shifting the audio signal either forwards or backwards in time. The extent of this time shifting is controlled by the shift_max parameter, which determines the maximum proportion of the signal's length that can be shifted. A 10% time shift randomly forward or backwards was implemented, similarly based on Harrison et al. (2021). By simulating variations in the temporal alignment of audio patterns, this technique enhances the model's resilience to temporal variations in audio recordings, thereby improving its ability to generalize across different contexts and conditions.

In process_audio_files function, the argument 'num_augmented_versions' is by default set to 2, and can be increased to increase the size of the dataset as part of the data augmentation process. This is useful when working with large networks with many parameters. Thus, each the clean and the noisy version were doubled the size of the original isolator.

The final dataset includes 3460 observations, including 1730 clean X values and the same many noisy. These were split into training and test data with a 80% and 20% split.

4. The Model

4.1. The architectures

This project explores how four different model structures approach keystroke analysis. Every model's final layer is a dense layer with 36 nodes, the number of classes. The activation of this layer is softmax function which is a common neural network to predict the class of an input image.

Basic CNN model

The basic Convolutional Neural Network (CNN) model is specifically designed to assess the performance of straightforward CNN architectures on our dataset. It serves as an initial testing ground to understand how simple convolutional networks handle the data without the added complexities found in deeper or more sophisticated architectures.

This basic model utilizes a sequence of convolutional and pooling layers, culminating in a fully connected layer for classification. The network starts with a convolutional layer of 32 filters of size 3x3 and ReLU activation to capture basic features. This is followed by a 2x2 max pooling layer to reduce spatial dimensions and prevent overfitting. The architecture progresses with additional convolutional layers of 64 and then 128 filters, each followed by its own max pooling layer to further condense the feature maps. After flattening the output, it feeds into a dense layer of 128 units. To improve generalization and prevent overfitting, a dropout layer with a rate of 50% is introduced after the dense layer. The model concludes with a softmax classifier for multi-class prediction, effectively categorizing the images into different classes based on the learned features.

By employing this minimal configuration, the basic CNN not only evaluates the essential performance of CNNs but also establishes a baseline for comparison with more complex models in subsequent experiments.

ResNet

The Basic Architecture: Residual Neural Networks (ResNets) are a specific family of CNN architectures designed to address the gradient vanishing problem encountered in deep CNNs. This problem arises because weights in the earlier layers of the network cannot be efficiently updated using the gradient, which is too small in these layers. Consequently, training very deep networks becomes complex due to the vanishing gradient issue.

To mitigate this problem, ResNets incorporate shortcut connections that skip over some layers in the network. These shortcuts allow for the construction of deep networks without experiencing vanishing gradients. Typically, ResNets employ residual blocks, which consist of convolutional layers followed by a shortcut connection to the block's output. This shortcut connection enables the gradient to flow through, effectively resolving the gradient vanishing problem. It is often implemented as a connection without any intermediate operations or as a convolutional layer with kernels of size one to match the number of filters for addition.

ResNet18 Architecture: The input shape for the network is (180, 180, 1), indicating grayscale images with dimensions of 180x180 pixels. The input images undergo initial convolutional processing with a convolutional layer having 64 filters and a kernel size of 7x7. This operation is applied with a stride of 2, resulting in downsampled feature maps. Batch normalization and ReLU activation are applied after the convolution operation to normalize and introduce non-linearity into the network. Following the initial convolution block, max-pooling with a kernel size of 3x3 and strides of 2x2 is applied to further downsample the feature maps. ResNet18 consists of a series of residual blocks. Each block contains two convolutional layers with 3x3 kernels and batch normalization. ReLU activation functions are applied after each convolutional layer. The residual connection, or ”shortcut”, is implemented via an ”Add” layer that combines the output of the convolutional layers with the original input of the block. If necessary, a 1x1 convolutional layer is used to match the dimensions of the shortcut connection with the output dimensions of the residual block. After the series of residual blocks, global average pooling is applied to generate a fixed-length feature vector. This operation computes the average value of each feature map across all spatial locations, resulting in a compact representation of the input data. The global average pooled features are fed into a fully connected layer with softmax activation. This layer produces the final output logits, representing the predicted probabilities for each class in a classification task.

CoAtNet

The Basic Architecture: CoAtNet, short for Convolutional Attention Network, represents a novel approach that combines the strengths of convolutional neural networks (CNNs) with self-attention mechanisms. This hybrid architecture is designed to effectively handle diverse datasets with large feature sets, striking a balance between local and global feature modelling. CoAtNet effictively combines convolution and attention mechanisms to handle datasets of varying sizes - offering promise for the expansion of datasets and future work. Through comprehensive experimentation presented in Advances in Neural Information Processing Systems, 34 (2021), the authors showcase the adaptability and robustness of CoAtNet across diverse machine learning tasks, including side-channel analysis. It was the architecture used by Harrison et al. (2023) and so included as a 'baseline' for comparison to the other models - this paper's CoAtNet model was too initially based on it (12).

CoAtNet Architecture: The CoAtNet architecture is characterized by its modular design, consisting of convolutional and transformer blocks organized into stages. These blocks are iterated based on user-defined parameters, allowing for flexibility and adaptability in capturing both local and global features of the input data. The convolutional blocks leverage traditional spatial feature extraction, while the transformer blocks enable long-range dependencies to be captured through self-attention mechanisms. The number of epochs used for CoAtNet is significantly less than for the other models (10) because of the limitations of computing power alongside the scale of the model.

Code for CoAtNet: The implementation of CoAtNet involves defining custom layers that encapsulate the functionalities of both convolutional and transformer blocks. The MBConv class represents a mobile inverted bottleneck convolutional block, known for its efficiency in extracting spatial features. The MultiHeadSelfAttention class implements multi-head self-attention, facilitating the capture of long-range dependencies in the input data. The ConvTransformer class combines convolutional and transformer operations into a unified block, offering a seamless integration of both feature extraction methods. Finally, the CoAtNet function orchestrates the assembly of the complete CoAtNet architecture based on user-defined parameters, providing a versatile and powerful tool for various machine learning tasks.

Autoencoder with Classifier

This model integrates a denoising autoencoder with a subsequent classification model, designed specifically to remove noise from images before they undergo classification, thereby enhancing prediction accuracy. The architecture is divided into two principal components: the encoder-decoder framework for image denoising and the classifier for subsequent image categorization.

Autoencoder Framework: The encoder compresses the input image into a lower-dimensional latent space using convolutional layers, effectively capturing essential data attributes while suppressing noise. The encoder begins with a convolutional layer consisting of 32 filters, followed by batch normalization and ReLU activation to stabilize learning and introduce non-linearity. A subsequent 2x2 max pooling layer reduces the spatial dimensions while retaining critical features, thus suppressing irrelevant noise. This process is repeated with a 64-filter convolutional layer to further enhance feature extraction.

The decoder reconstructs a noise-free image from this compressed latent representation. It employs transposed convolutional layers that incrementally increase the spatial dimensions back to their original size, each layer paired with batch normalization and ReLU activation for efficient feature restoration. The reconstruction process concludes with a sigmoid activation layer that ensures the final denoised image is properly scaled between 0 and 1.

Classifier Framework: Following the denoising process, the classifier uses the encoder's output as its input, leveraging the cleaned and enhanced feature set for classification. The classifier architecture builds upon the encoder by adding a series of fully connected layers. After flattening the encoder output, a dense layer with 256 units and ReLU activation is utilized to maintain non-linearity in decision-making. Dropout layers with a 60% dropout rate are strategically placed after each dense layer to prevent overfitting, ensuring the model generalizes well on unseen data. The classification process culminates with a softmax activation layer that provides a probability distribution across multiple classes, facilitating precise multi-class prediction.

Training and Testing Configuration: In the initial training phase of the autoencoder, noisy images are input to the model, while the corresponding clean images are the desired outputs. This setup allows the autoencoder to learn effective noise reduction techniques tailored to the specific characteristics of the dataset's noise, aiming to reconstruct clean images from their noisy counterparts. Subsequently, during classification, the encoder's output, representing the denoised images, is used as input to the classifier, ensuring that the classifier operates with high-quality data for accurate categorization.

For testing, the model includes both noisy and clean images in the test set. This method enables a direct comparison with other models and ensures consistent evaluation metrics across different models. The inclusion of both noisy and clean images in the testing phase provides a more comprehensive evaluation of the model's capabilities, assessing not just the noise reduction success but also the classifier's robustness across different levels of image quality. This approach validates the model's performance against a broader spectrum of image conditions, demonstrating its practical utility in real-world applications where noise levels can significantly vary.

Hyperparameter Tuning

Details of the hyperparameters are outlined in Table 1. As this paper focuses on a comparative approach, we aimed to normalise these across all models. The Author's CoAtNet uses both a different number of epochs and a different Optimizer - the former is due to computational limits, and the latter is due small attempts to improve the model in any way. There was an opportunity here to maximise the accuracy of the individual models, but our focus meant the sacrifice of accuracy for the proper comparitivity. Similarly, the number of epochs was much less than in the original paper (8) also due to computational limits. We feel this is not as much of a limitation as too many epochs would've resulted in overfitting with a large amount of data augmentation.

4.2. Numerical Evaluation Results and Discussion

As we were unable to completely replicate Harrison et al., 2023's CoAtNet model (having converted it from PyTorch) - there is no specific baseline model, instead we take a comparative approach across the models. As the labels are one hot encoded, the sparse categorical cross entropy (SCCC) was the most appropriate loss function to calculate all models' loss function, with the exception of the autoencoder wherein the denoising section uses the Mean Squared Error, while the classifier returns to SCCC.

Basic CNN model

The Basic CNN model was tasked with establishing a baseline for performance on a mixed dataset that included both noisy and clean images. This setup was intended to assess the model's capability to generalize across varying conditions typical in real-world scenarios.

The Basic CNN model showed a strong capacity to adapt and learn from the mixed dataset as evidenced by the rapid decrease in training and validation loss during the initial epochs. The training loss showed a consistent downward trend, demonstrating the model's effective learning from the dataset. The validation loss paralleled this trend with minor fluctuations, indicative of the model's ability to generalize to unseen data, which includes both noisy and clean samples.

Across 100 epochs, the model achieved a training accuracy nearing 86% and a validation accuracy of about 88% (see Table 1). These results are indicative of the Basic CNN's ability to manage both noise and clarity in images effectively, despite its simple architecture. However, the test accuracy was reported at approximately 85.26%, suggesting some challenges when the model encountered new, possibly more varied data in the test set.

The performance on the combined dataset of noisy and clean images indicates that the Basic CNN, while generally robust, might still benefit from additional specialized preprocessing for noise or enhancements in architecture to better handle variability in image quality.

The Basic CNN's straightforward structure with layers of convolution and pooling followed by dense layers demonstrates sufficient capability for basic classification tasks under varied input conditions. The dropout at 50% was effectively utilized to prevent overfitting, maintaining close training and validation accuracies. However, the close yet slightly lower test accuracy compared to the validation accuracy could suggest that while the model generalizes well, there might be room for incremental improvements. These could potentially come from tuning existing layers or adding noise-specific preprocessing steps to further adapt the model to varied real-world scenarios, especially under conditions not fully represented in the training set.

In conclusion, while the Basic CNN serves as a useful benchmark, indicating the baseline capabilities necessary for image classification, its simplicity also points to limitations in handling complex scenarios without additional enhancements. For environments with higher noise levels or greater data variability, more complex architectures or preprocessing strategies would likely yield better performance.

Autoencoder with Classifier

Denoising Autoencoder: The denoising autoencoder training sessions indicated an encouraging trend in loss reduction, evident from the rapid decrease in loss values from 0.1505 to 0.0595 within the initial ten epochs (Figure 1, Appendix). This swift adjustment demonstrated the model's efficacy in filtering out noise and refining image quality. The validation losses, which converged to 0.0603, paralleled the training losses, underscoring a stable performance across both the training datasets and unseen validation datasets. This performance trajectory attests to the autoencoder's proficiency in mastering noise reduction techniques applicable across similar datasets.

Classifier: Regarding classification, it demonstrated significant enhancements over the training period, achieving

Final Model	Baseline CoAtNet (8)	ResNet	Author's CoAtNet	CNN	Autoencoder & Classifier
N. of Parameters		11,203,108	17,603,332	6,651,044	8,407,461
Loss function	Cross Entropy	Cross Entropy	Entropy	Cross Entropy	Mean Squared Error (autoencoder) Crossentropy (classifier)
Dropout	None	None	0.5	0.5	0.6 (classifier)
Epochs	1100	20	10	100	10 (autoencoder), 100 (classifier)
Learning rate	0.0005	0.001	0.01	0.001	0.001
Batch size	16	32	32	32	32
Optimizer	Adam	Adam	SGD	Adam	Adam
Validation Accuracy	0.96 (PVA)	0.98	0.16	0.88	0.88
Test Accuracy		0.8	0.27	0.85	0.93
Test F1	0.95	0.05	0	0.53	0.6

Table 1. Model configurations and results

a peak validation accuracy of approximately 88%. This robust training process allowed the classifier to perform exceptionally well in real-world scenarios, as evidenced by its test accuracy reaching 92.77% (see Table 1). This superior performance on the test set, compared to the validation set, can be attributed to the inclusion of dropout layers in the network architecture. Dropout helps in preventing overfitting during training by randomly disabling a proportion of neurons, thus ensuring the network learns more generalized features.

Combined Performance: The combined use of a denoising autoencoder followed by a classifier effectively managed the challenges posed by noisy data. The autoencoder improved the quality of input data for the classifier, which in turn, was able to utilize its robustly learned features to make accurate predictions. The integration of these components resulted in a powerful model that exhibited excellent adaptability and accuracy across both noisy and clean datasets in the test phase.

Overall, the denoising autoencoder and classifier model demonstrated a potent approach to handling image data with noise, achieving notable accuracy and robustness, making it highly suitable for applications where dealing with noise is a significant challenge. This model stands out for its ability to perform reliably under different conditions, validating its effectiveness and practical utility in real-world scenarios.

Author's CoAtNet model

This model performed the worst out of the five. From Figure 4 (Appendix) it is evident that the small number of epochs alongside the huge parameter size (17603332) significantly impacted the performance of the model (Figure 11). The training loss decreases by 0.002 over the 10 epochs, while the validation loss increases by the same amount. The accuracy of the model fluctuates hugely, and unfortunately means little by way of interpretation can be explored.

ResNet

The ResNet model, specifically the ResNet18 architecture, was utilized to evaluate performance on a diverse dataset involving both noisy and clean images. This model is known for its ability to handle deep learning tasks efficiently due to its residual blocks that facilitate learning by adding shortcut connections.

During the training process, the ResNet model quickly adapted to the mixed dataset. The training loss decreased steadily from the start, showcasing the model's capability to effectively learn from the data. This is reflected in the training loss reaching a low of 0.3013 after 20 epochs (Figure 3, Appendix). Similarly, the validation loss showed a corresponding decline, indicating that the model was not only learning but also generalizing well to unseen data. These results were mirrored in the validation accuracy which closely tracked with the training accuracy.

Over the course of 20 epochs, the ResNet model achieved a commendable training accuracy of approximately 90.46%, with a validation accuracy nearly aligning with this figure. The test accuracy confirmed the model's robust performance, reaching 93%, which is indicative of the ResNet's capability to effectively handle both noise and clarity in images (see Table 1). This performance is particularly notable given the simplicity of the training duration and the architecture's ability to maintain high accuracy across varied input conditions.

The ResNet model's performance on this mixed dataset of noisy and clean images illustrates its strong generalization ability, likely aided by its deep layered structure and residual connections which help mitigate the vanishing gradient problem that deeper networks often face. Despite the model's high accuracy, the slight variance between training and validation metrics suggests there could be potential for further optimization, possibly through more targeted data augmentation or advanced regularization techniques.

In conclusion, the ResNet model serves as a powerful benchmark in the realm of image classification, demonstrating high efficacy and robustness, particularly in handling complex scenarios that include variable image quality. For environments that feature high levels of noise or greater variability in data, the ResNet architecture provides a strong foundation, with scope for further enhancement to achieve even better performance.

5. Interpretation

By comparing the Grad-CAM outputs between the basic CNN model and the autoencoder with a classifier, it can be observed that both models focus on regions with distinct audio intensity. However, a notable difference between them lies in the size of the attended regions. The basic CNN tends to highlight larger areas, appearing as square-shaped blocks in the visualization. Conversely, the autoencoder with a classifier exhibits smaller, pinpointed areas of focus distributed across what the model deems as crucial. Perhaps, this capability of the autoencoder with a classifier to minimize the size of its attended regions allows it to capture subtle nuances within the patterns, thereby achieving higher accuracy than the basic CNN model.

6. Conclusion

The results from our examination of four different model architectures for keystroke analysis show promising potential, especially considering the limited hyperparameter tuning conducted in this study.

Basic CNN model, autoencoder with classifier model, and ResNet all showed significant promise. All effectively filtered out noise from input images and achieved high accuracy in image categorisation, suggesting its potential utility in noise-sensitive real-world applications.

Unfortunately, the Author's CoAtNet did not obtain similar positive results. We hypothesize this to be because of the excess number of parameters given the relatively small dataset size which leads to slow learning and fast overfitting (Figure 11).

Overall, while the models performed well given the constraints, further optimization and exploration are warranted. Future research could focus on fine-tuning hyperparameters, exploring more sophisticated architectures, or investigating additional preprocessing techniques to enhance performance. By building upon these promising results, we can continue advancing the state-of-the-art in keystroke analysis and related image classification tasks.

6.1. Potential future works

There are incredible opportunities for future work in this space, specifically expanding on the data, the approach, or the angle of research. A key limitation in this project was the small amount of data, which lead to wide variation in accuracy in each training system, especially when working with models with a huge number of parameters. An initial project would be hugely expanding the dataset to include keystrokes from wide range of keyboard geometries from different devices, structured and recorded in different ways. This would be extremely valuable in industry to identify what makes a device's keyboard less secure, and to explore new design potential. It would complement research by Adhin et al., (2021) that works to identify devices based on acoustic emissions, potentially creating a powerful attack process with integrated device identification into keystroke identification (1).

This expanded dataset should also explore recordings made in a much larger range of environments. Harrison et al. (2023) worked with Zoom and phone recorded data adjusted with synthetic noise, though we chose to only work with phone recorded data. This expanded dataset could potentially give a better understanding of how realistically risky these attacks are based on your environment. This segues into the next potential future angle: mitigation strategies. The inclusion of white noise, alongside prerecorded keystrokes, can "effectively cloak" the actual user's keystrokes (3). Strategies of how best to approach cloaking of device emissions is an interesting area. Some models (specifically ResNet) show promise for keystroke analysis: and potentially future work could focus on maximising its potential with further hyperparameter tuning (potentially using keras tuning).

7. Statement about individual contributions

24883: Translation of CoatNet, GradCam initialisation, Evaluation

29662: Set up and development of ResNet18, GradCam initialisation, Evaluation

34462: Data preprocessing, Set up and development of Base CNN, Autoencoder with Classifier, GradCam initialisation and interpretation

80406: Data pre-processing, Translation of CoAtNet, GradCam initialisation, Evaluation

References

[1] V S Adhin, Arunjo Maliekkal, K Mukilan, K Sanjay, R Chitra, and A. P. James. Acoustic side channel attack for device identification using deep learning

models. In *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 857–860, 2021.

[2] Mohammad Abdullah Al Faruque, Sujit Rokka Chhetri, Arquimedes Canedo, and Jiang Wan. Acoustic side-channel attacks on additive manufacturing systems. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, pages 1–10, 2016.

[3] S Abhishek Anand and Nitesh Saxena. Speechless: Analyzing the threat to speech privacy from smartphone motion sensors. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 1000–1017, 2018.

[4] D. Asonov and R. Agrawal. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 3–11, 2004.

[5] Michael Backes, Markus Dürmuth, Sebastian Gerling, Manfred Pinkal, and Caroline Sporleder. Acoustic side-channel attacks on printers. pages 307–322, 09 2010.

[6] Kiran Balagani, Matteo Cardaioli, Stefano Cecconello, Mauro Conti, and Gene Tsudik. We can hear your pin drop: An acoustic side-channel attack on atm pin pads. In *Computer Security – ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part I*, page 633–652, Berlin, Heidelberg, 2022. Springer-Verlag.

[7] Gabriel Goller and Georg Sigl. Side channel attacks on smartphones and embedded devices using standard radio equipment. pages 255–270, 04 2015.

[8] Joshua Harrison, Ehsan Toreini, and Maryam Mehrnezhad. A practical deep learning-based acoustic side channel attack on keyboards. In *2023 IEEE European Symposium on Security and Privacy Workshops (EuroSPW)*, pages 270–280, 2023.

[9] JBFH-Dev. Keystroke datasets repository. https://github.com/JBFH-Dev/Keystroke-Datasets, 2023.

[10] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[11] Ani Nahapetian. Side-channel attacks on mobile and wearable systems. pages 243–247, 01 2016.

[12] Soheil. Deepkeyattack: Deepkeyattack for Python, 2023.

[13] Alireza Taheritajar, Zahra Mahmoudpour Harris, and Reza Rahaeimehr. A survey on acoustic side channel attacks on keyboards, 2023.

[14] COMINT United States. National Security Agency. A history of us communications security (the david g. boak lectures). https://media.defense.gov/2021/Jun/29/2002751403/-1/-1/0/HISTORY_COMSEC.PDF, 1973.

[15] Yongbin Zhou and Dengguo Feng. Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. *IACR Cryptol. ePrint Arch.*, 2005:388, 2005.

[16] Tong Zhu, Qiang Ma, Shanfeng Zhang, and Yunhao Liu. Context-free attacks using keyboard acoustic emanations. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 453–464, 11 2014.

8. Appendix

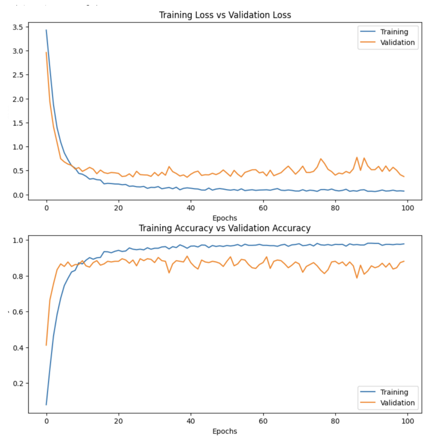

Figure 1. Loss graphing across training and test for the autoencoder with classifier

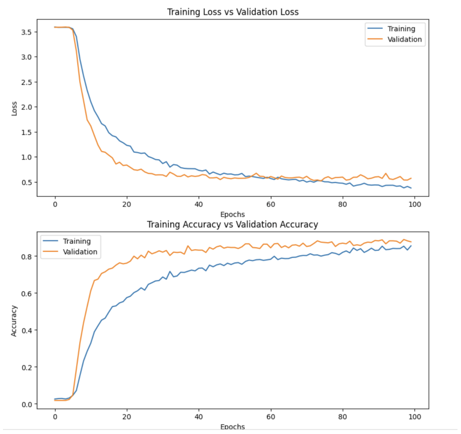

Figure 2. Loss graphing across training and test for the CNN

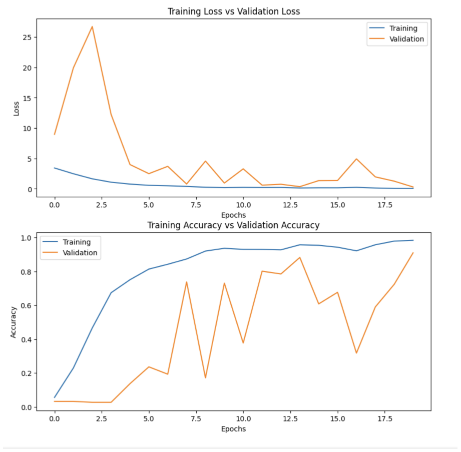

Figure 3. Loss graphing across training and test for ResNet18

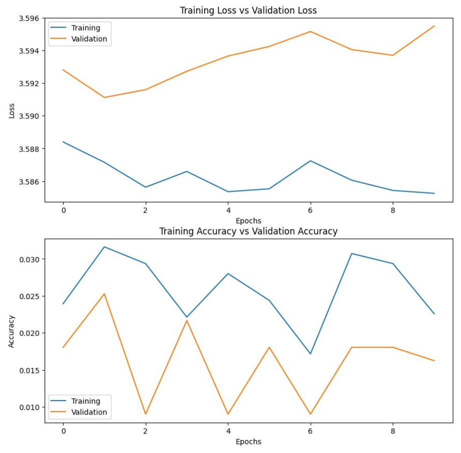

Figure 4. Loss graphing across training and test for the Author's CoAtNet

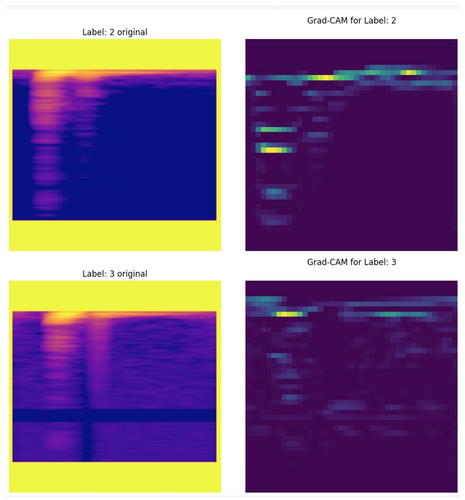

Figure 5. GradCAM 1 for the Base CNN Model

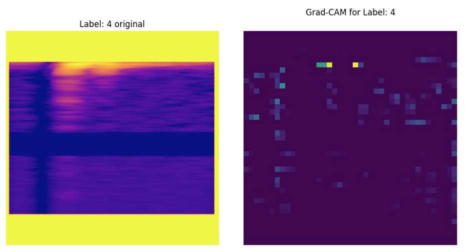

Figure 6. GradCAM 2 for the Base CNN Model

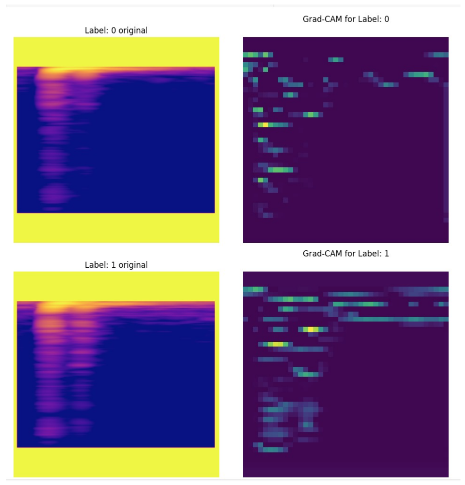

Figure 7. GradCAM 3 for the Base CNN Model

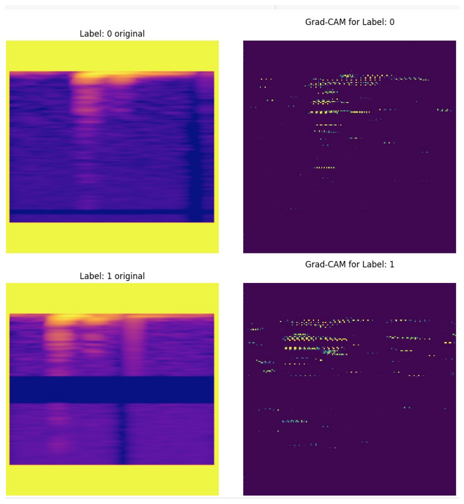

Figure 8. GradCAM 1 for the AutoEncoder

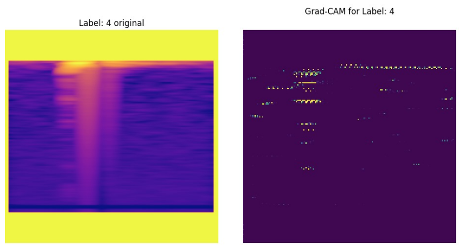

Figure 9. GradCAM 2 for the AutoEncoder

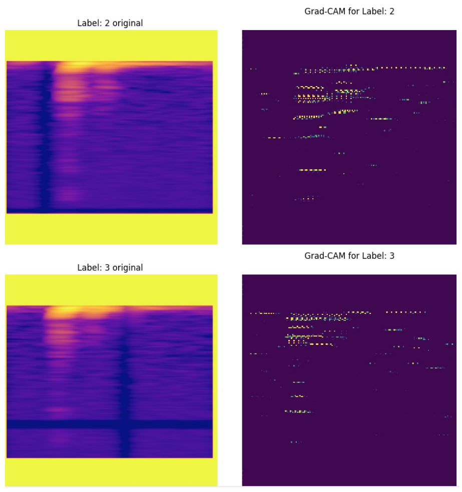

Figure 10. GradCAM 3 for the AutoEncoder

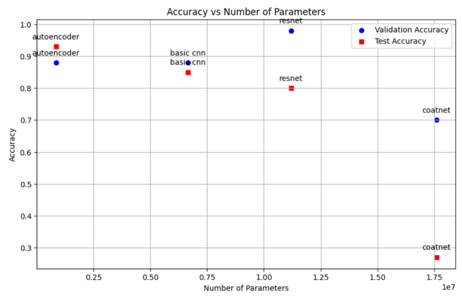

Figure 11. Scatter Plot of Number of Parameters per model to their Validation and Test Accuracy