

UGP Presentation

Acoustic Side-Channel Attack for PIN Retrieval from Google Pay

Speakers:

Wattamwar Akanksha Balaji (221214)

Mahaarajan J (220600)

Supervisor:

Dr. Urbi Chatterjee

Overview

- Introduction
- What are Acoustic Side-Channel attacks and How do they work
- Our experiment using GPay
- Audio Features extracted
- Pre-processing
- Machine Learning models used
- Results
- Drawbacks
- Future Work and improvements

Introduction

Aim:

To investigate whether acoustic side-channel attacks can be used to infer UPI PINs entered on GPay using only audio recordings from the smartphone's internal microphones.

Contributions:

- Achieved 90% accuracy in classifying the PINs using a Neural Network trained on the recordings.
- Identified that the back microphone captured the most informative signals due to its proximity to internal hardware.
- Established a strong proof of concept that PIN inference via acoustic side-channel is feasible under controlled conditions.

Acoustic Side-Channel Attacks

- Every process consumes some energy and may generate noise, vibrations or produce heat which leads to leakage of information. Exploitation of this information from side effects of a process is called Side-Channel attack. When unintended sound emissions are extracted, we call it Acoustic SCA.
- Although these sounds are often inaudible to the human ear or masked by ambient noise, they can still be captured by specialised devices or even the internal microphones.



Sources of these Sounds

These sounds are produced by different vibrations that occur in the electronic elements such as capacitors on the PCB of the smartphone. There are also EM waves which are produced due to charging and discharging of the residual capacitance in the wires.



The screen sends and receives signals through the wires connecting it to the PCB which pass near the microphones which pick up the sound produced when we type on the virtual keyboard.

Experimental Setup

We recorded the PIN entries in the Gpay using dB Meter application, which allows us to extract audio recorded by a single internal microphone, running in the background.

Here are the detailed steps of how we recorded the audios on iPhone 15:

Step-I:

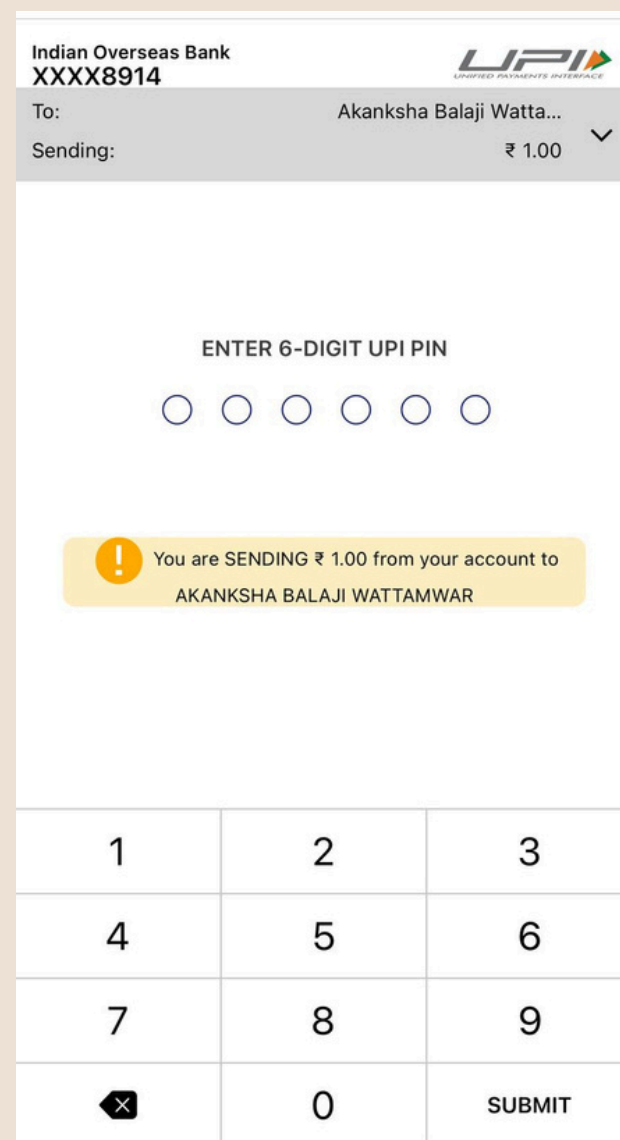
Open dB Meter app and start recording after selecting the required microphone.



Experimental Setup

Step-2:

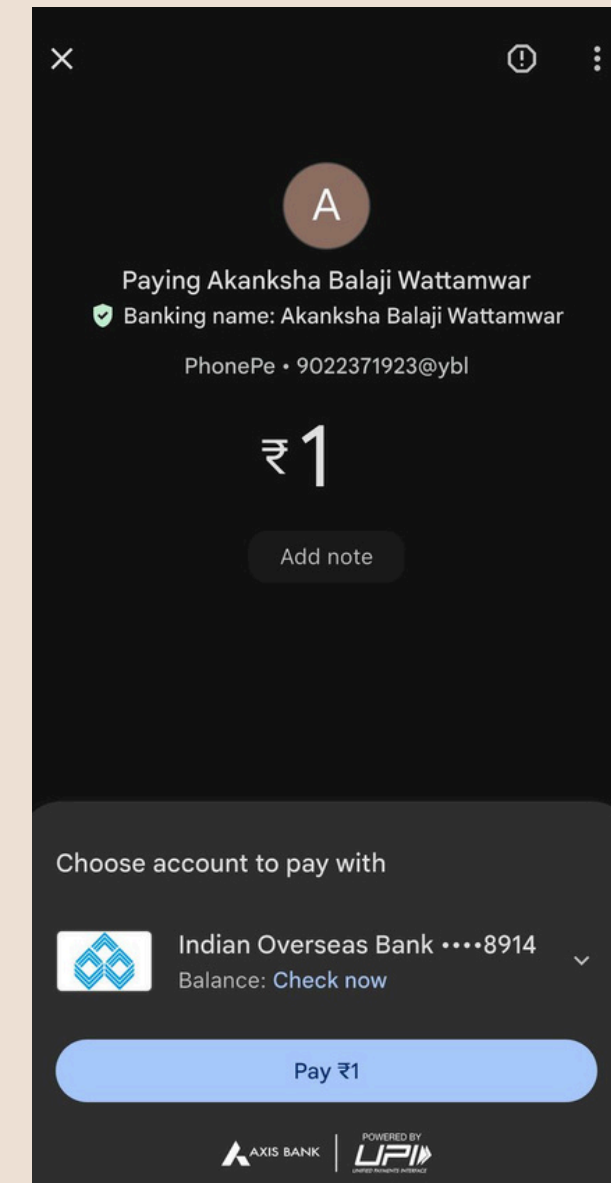
Switch tabs to GPay which is already opened in the payments page. Also ensure that no other app is running in the background.



1	2	3
4	5	6
7	8	9
✕	0	SUBMIT

Step-3:

Enter the PIN with 1 second gap between each click then go back to dB Meter to stop the recording.



More Details...

In the experiment we recorded audios for uniform PINs alone i.e 000000, 111111, 222222 etc., We did this to ensure that the models trained have the best chance of distinguishing between PINs to start with.

In this way, we prepared an extensive dataset of 600 recordings consisting of 60 recordings for each of the uniform PINs (0 to 9) 15 each from 3 microphones (front, bottom and back) and 15 from the voice memo app which uses all of the above microphones.

Audio Features Used

We analysed various audio features for finding the similarities between recordings of pins of same number and simultaneously differences between those of different numbers.

Following are the features that gave us significant results:

- MFCC Coefficients
- Spectral contrast
- Root Mean Squared Energy
- Chroma features
- Zero Crossing Rate
- Spectral Roll-Off

1. MFCC (Mel-Frequency Cepstral Coefficients):

- MFCCs capture the short-term power spectrum and spectral shape of audio.
- Each virtual keypress generates a slightly different frequency signature and MFCCs help capture these subtle differences, enabling classification of digits based on their acoustic patterns.

2. Chroma Features:

- Chroma features represent the energy distribution across 12 pitch classes (like musical notes) as a vector.
- Useful for distinguishing tonal qualities in keypresses which are different for each digit.

3. Spectral Contrast:

- This feature captures the difference in energy between crests and troughs in the frequency spectrum across sub-bands.
- Taps on different keys might result in different degrees of variation in sound which can be captured using spectral contrast.

4. Zero Crossing Rate:

- ZCR measures how often an audio signal crosses the zero amplitude level within a given time frame. A high value represents a sharper sound, as high-frequency content typically causes more zero-crossings.
- Different keypresses may result in slightly sharper or duller sound which is highlighted by ZCR.

5. Root Mean Squared Energy:

- Measures the energy of the signal over a short frame.
- Taps at different positions may result in variations in energy which can be captured by RMSE.

6. Spectral Roll-off:

- Indicates the frequency below which a set percentage (usually 85–95%) of total spectral energy is concentrated.
- Some PINs could result in more low-frequency or high-frequency energy that can be captured by spectral roll-off.

Pre-Processing

Segmentation

To prepare the data for analysis, each audio recording was segmented to isolate individual keystrokes corresponding to PIN digits. Here's how we segmented the audio:

- The first 2.5 seconds of each audio recording were discarded.
- The remaining part of each audio file were evenly divided into 6 segments, each intended to capture one keystroke (PIN digit entry).
- Each segment was then saved as a separate audio file with appropriate label.

Pre-Processing

Audio Feature Extraction

- We used Python's librosa library to extract key audio features from each keystroke segment.
- Since the duration of audio segments varied slightly, each feature produced a time series of different lengths. To standardise this, we computed the mean of each feature over time and stored them in a pandas DataFrame.
- We applied StandardScaler to the DataFrame to normalize all feature values to ensure that all features have zero mean and unit variances so that no single feature dominates due to scale differences.

ML models & Their Results

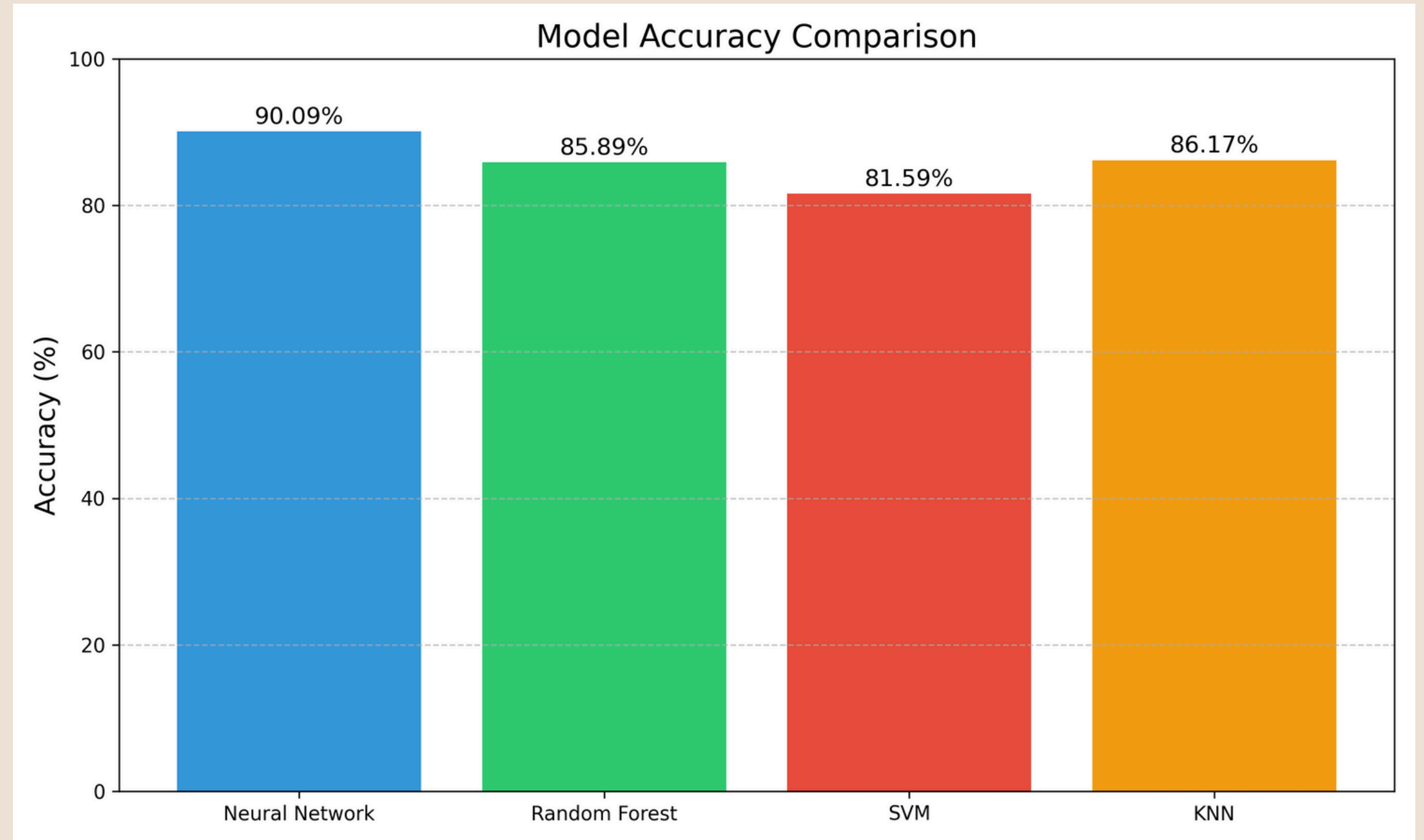
To classify the segmented audio samples, each segment was labeled with its corresponding digit (0-9). The dataset was then split into 70% for training, 15% for validation, and 15% for testing to effectively train and evaluate the machine learning model.

Here is a list of ML models we used with their hyperparameters:

1. k-Nearest Neighbors (kNN) (n_neighbors = 5)
2. Random Forest Classifier (n_estimators = 100)
3. Support Vector Machine (SVM) (kernel = 'linear')
4. Neural Network

Model Accuracies

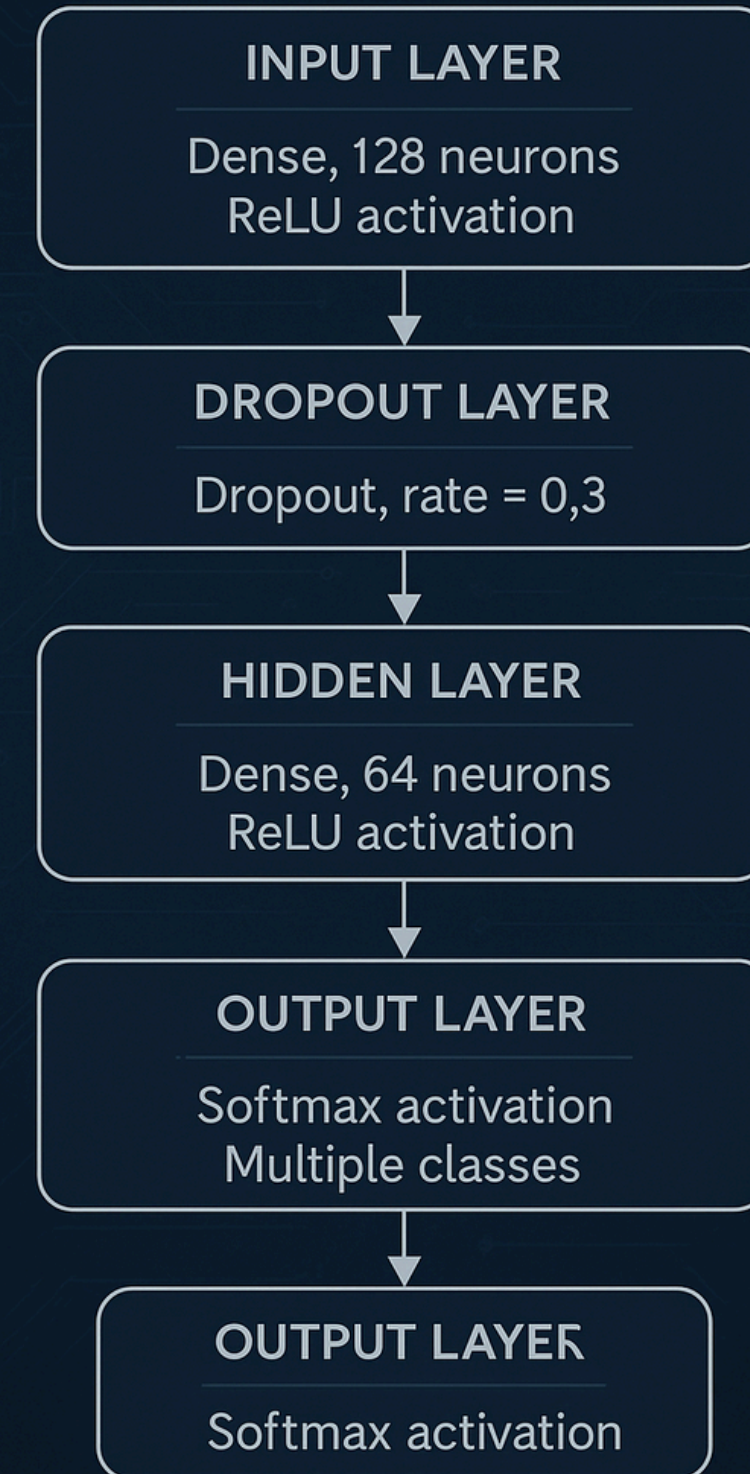
Models like SVM, kNN, and Random Forest rely on simpler decision boundaries or distance metrics, which fall short for our noisy, overlapping audio features, whereas the Neural Network learns complex patterns making it better suited.



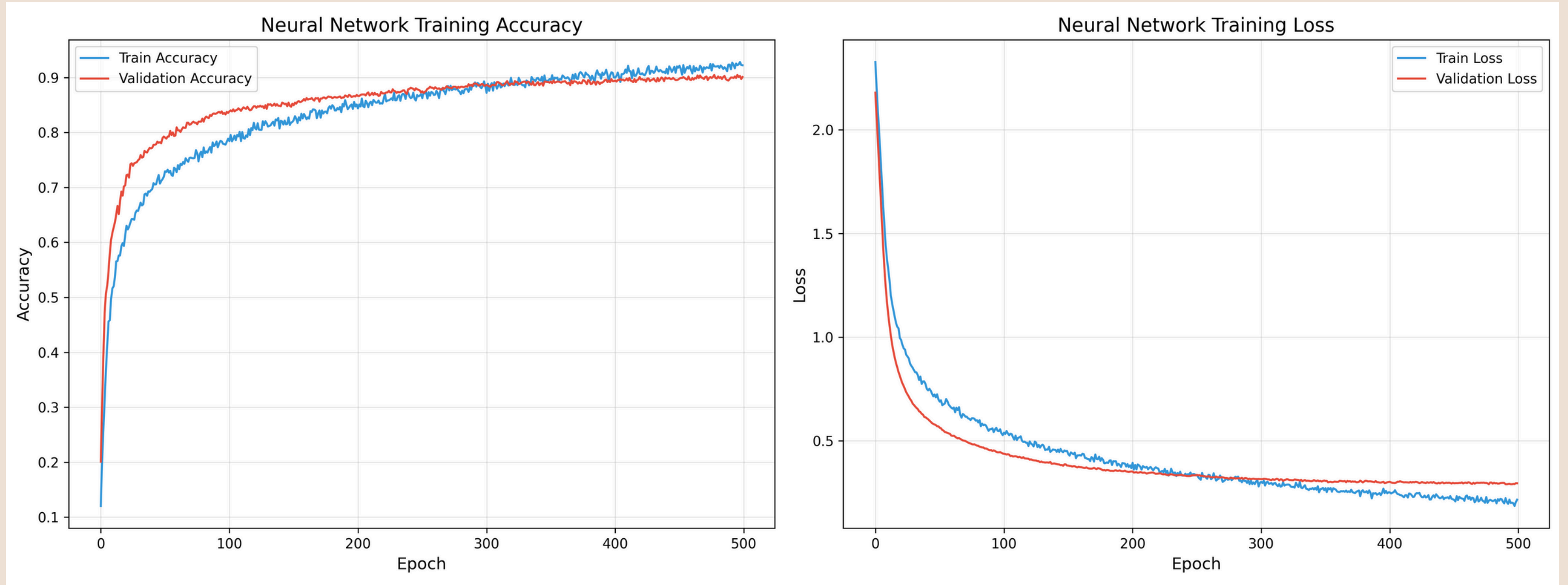
Neural Network

- The neural network was trained over 500 cycles (epochs) using Adam as optimiser with categorical_crossentropy as loss.
- The dense architecture with dropout helped it generalize well on intra-class variations and capture non linear patterns and interactions.

NEURAL NETWORK ARCHITECTURE

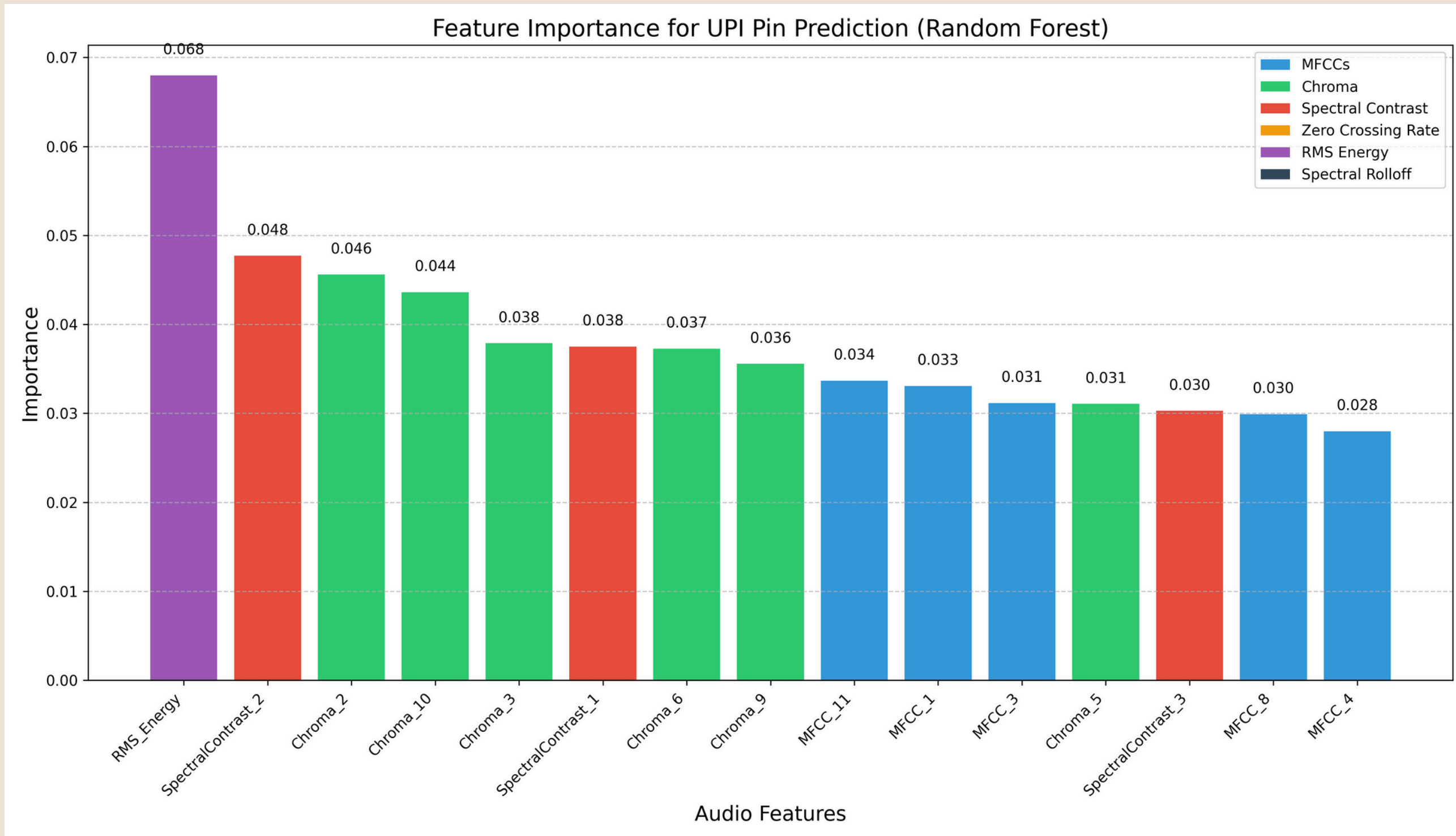


Training During Neural Network



Finally we got a training accuracy of 91.82% and a validation accuracy of 90.67%

Feature Importance (Random Forest)



RMS Energy ranks highest as it captures the clarity and signal strength of each keypress, helping distinguish digits well.

Testing for different mics

To understand which microphone captures audio signals most effectively, we tested our models individually on audio from different microphones.

The microphones used are:

- Front Microphone
- Back Microphone
- Bottom Microphone
- Combined Audio (from Voice Memos app in iPhone)

Testing for different mics

- The display and touchscreen wiring are generally routed near the back side of the phone.
- The improved accuracy using the back mic reinforces this fact.



Drawbacks

- When tested on fresh recordings (even in similar quiet environments), the model often misclassified PINs. The possible reasons could be:
 - Overfitting to environment-specific noise patterns present during our original data collection.
 - Microphone sensitivity or positioning changes between recordings, affecting signal clarity.
 - Background ambience changes, even if subtle, introduces variations the model hasn't seen leading to misclassifications.
- When tested on recordings from a different phone model (iPhone 6), the model failed — highlighting its reliance on device-specific acoustic signatures.

Drawbacks

- The model was trained exclusively on uniform PINs (e.g., 111111, 222222). It may fail to classify real-world, non-uniform PINs, where digit transitions introduce more complexity.
- The experiments done only had GPay and the recording app open. Keeping other apps running in the background might affect the audios recorded due to interfering noises.

Our approach demonstrates strong proof of concept, but current model lack robustness across environments and devices, pointing to the need for improved generalization strategies.

Future Work & Improvements

To make the model robust across environments and devices:

1. Larger and more diverse dataset:

- Include recordings from varied ambient conditions, background noise levels, and user behaviors.
- Extend the dataset to include non-uniform PINs (e.g., 195203), reflecting real-world usage.
- Incorporate recordings from different smartphone models to reduce hardware dependency.

Future Work & Improvements

2. Transfer Learning Approach:

- Fine-tune a pre-trained model trained on a known environment using a small set of new samples from the new environment.
- This enables the model to adapt quickly and provide correct classifications to devices or environments not seen before

3. Deep Learning for Feature Learning:

- Automatically learn discriminative, hierarchical features from raw or spectrogram audio data.
- Capture temporal patterns and local variations that simpler models might miss.

Thank you !!