Acoustic Side-Channel Attack for PIN Retrieval from Google Pay

Wattamwar Akanksha Balaji (221214)
Mahaarajan J (220600)

Supervisor: Dr. Urbi Chatterjee

Abstract

This research investigates the feasibility of conducting an acoustic side-channel attack to infer UPI PINs entered on the virtual keyboard inside the GPay application. Using only the recordings from a smartphone's internal microphone, we analyze the unique acoustic signatures generated during PIN entry and demonstrate that these can be used to recover the digits entered. Our approach achieved 96% accuracy in classifying PINs using a Neural Network trained on recordings from the back microphone, establishing a strong proof of concept that PIN inference via acoustic side-channel is feasible under controlled conditions. Our work highlights significant security vulnerabilities in the virtual keyboard which could be exploited in mobile payment applications.

1 Introduction

Google Pay (GPay) is a widely used digital payment application that relies on UPI PIN-based authentication. Our project investigates the feasibility of conducting an acoustic side-channel attack to infer UPI PINs entered via the virtual keyboard on smartphones using nothing but the recordings from the phone's internal microphone.

1.1 Research Objectives

- Determine if the acoustic emissions during PIN entry recorded using internal microphones contain exploitable information
- Develop a method for extracting the digits entered from these recordings using Machine Learning
- Evaluate the effectiveness of the attack under various real-world conditions
- Identify which microphone is most effective for the attack

1.2 Contributions

- Built a dataset of 600 audio recordings of uniform PINs (e.g., 000000, 111111, etc.) recorded on iPhone15

- Extracted meaningful audio features from the recordings and trained several ML models on them
- Achieved 96% accuracy using a Neural Network trained on audio from the back microphone
- Identified the back microphone as the most effective for recording the audios due to its proximity to the internal screen wiring
- Established a strong proof of concept that PIN inference via acoustic side-channel is feasible under controlled conditions

1.3 Ethical Considerations

This research was conducted with strict adherence to responsible disclosure practices. We followed established ethical guidelines in security research, including:

- Conducting experiments only on our own devices and accounts
- Using only simulated PINs rather than real user credentials

2 Background

2.1 Digital Payment Security

Mobile payment applications like Google Pay implement several security mechanisms to protect user transactions. The primary security model relies on:

- Authentication through UPI PIN entry
- Device binding and verification
- Secure communication protocols

The UPI (Unified Payments Interface) system, developed by the National Payments Corporation of India, has become the backbone of digital payments in the Indian market. The security of this system heavily depends on the PIN authentication mechanism, which serves as the final authorization step for transactions.

Despite these protections, mobile payment systems remain vulnerable to various attacks like side channel attacks. These attacks target the PIN entry process itself to retrieve the digits entered.

2.2 Acoustic Side-Channel Attacks

Side-channel attacks exploit unintended emissions such as heat, electromagnetic radiation, or sound which are produced when any process runs on a device to extract sensitive information. In acoustic side-channel attacks, information leakage occurs through sound waves generated by internal components of the device even though these are inaudible to humans, they can be captured using built-in microphones or external acoustic sensors and later identified by ML models using various features.

2.3 Sources of Acoustic Leakage

The vibrations and electromagnetic emissions produced by screen interactions and voltage changes in capacitors generate distinct sounds which can be picked up by internal microphones, especially the back microphone which is closer to the hardware responsible for these emissions.

Several physical phenomena contribute to acoustic leakage during touchscreen interactions:

- **Screen Wiring and PCB Interactions**
  The screen sends and receives signals through the wires connecting it to the PCB which pass near the microphones. These microphones pick up the emissions produced by these wires when we type on the virtual keyboard. The sounds are produced by different vibrations that occur in the electronic elements such as capacitors on the PCB of the smartphone.

- **Electromagnetic Emissions**
  There are also electromagnetic waves which are produced due to charging and discharging of the residual capacitance in the wires. These signals are also captured by the microphones.

3 Threat Model

3.1 Adversary Capabilities

Our threat model assumes an adversary with the following capabilities and limitations:

- Access to a number of recordings from the target device's internal microphone during PIN entry for training the ML models
- Basic signal processing and machine learning capabilities
- No physical access to the target device
- No special privileges or root access on the target device

This threat model is realistic in several scenarios as it only requires acquiring the audio data from the internal microphones. Some scenarios to obtain this data could be:

- Malicious applications with microphone permissions
- Compromised legitimate applications that have microphone access
- Web-based attacks that gain temporary microphone access

The attack window is limited to the duration of PIN entry, typically a few seconds for a 6-digit UPI PIN. This brief window is sufficient for capturing the required acoustic data.

3.2 Attack Scenarios

We consider several realistic attack methods:

3.2.1 Malicious Application Attack

The adversary distributes a harmless application (such as a game, utility, or media player) that requests microphone permissions. Once installed, the application monitors audio in the background, specifically during periods when Google Pay is active, and captures the acoustic data during PIN entry events which could then be used by the ML models

In our experiments, we simulated this scenario using the dB Meter app running in the background while the user entered PINs in Google Pay.

3.2.2 Web-Based Attack

Modern browsers support audio capture via various APIs after user consent. A malicious website could obtain temporary microphone access and capture audio during PIN entry when the user interacts with both the website and Google Pay in close temporal proximity.

3.2.3 Physical Proximity Attack

In scenarios where physical proximity is possible (such as shared workspaces or public environments), an adversary can use a separate device to capture acoustic emissions from the target device during PIN entry as opposed to the device's internal microphone.

4 Experimental Setup

Our experiment to capture PIN entry data from GPay was conducted using an iPhone 15 and the dB Meter app was used to record audio from specific internal microphones. The PINs were entered on the GPay interface with a 1-second gap between digits to ensure distinct non-overlapping audio patterns for each digit.

4.1 Procedure

Here are the exact steps we took to record in detail:

1. Open dB Meter app and start recording after selecting the required microphone
2. Switch tabs to GPay (already opened in the payments page)
3. Ensure no other app is running in the background
4. Enter the PIN with 1-second gap between each click
5. Return to dB Meter to stop the recording

4.2 Dataset Details

We built an extensive dataset with the following characteristics:

- 600 recordings of uniform PINs (60 recordings per digit from 0 to 9)
- Each PIN was recorded 15 times using 4 different mic setups:
  - Front microphone

  - Back microphone
  - Bottom microphone
  - Voice Memos app (which uses all microphones)

- Only uniform PINs were used (e.g., 000000, 111111, 222222, etc.) to ensure that the models had the best chance of distinguishing between digits
- All recordings were made in trolled, quiet environments to minimize background noise
- It was ensured that no other app was running in the background to avoid any interference from audio signatures from other apps

5 Iterative Development

The project followed an iterative approach, evolving through a series of trial-and-error phases:

5.1 Initial Attempts

Initially, we had only 5 recordings per digit and attempted to identify correlating features by visualising them using MATLAB. However, we could not find any consistent features that would distinguish one digit from another.

5.2 First Machine Learning Attempt

We then extracted standard audio features from our small dataset and applied PCA (Principal Component Analysis) along with machine learning classifiers. However, these models (SVM, kNN, Random Forest) performed poorly, yielding accuracies between 10-20%.

5.3 Expanding the Dataset

Realizing that the size of our dataset was a limiting factor, we expanded it to 600 recordings. We initially experimented with features based on sound pressure level, but found that these did not give good accuracy during classification.

5.4 Refined Feature Set and Improvements

We then extracted a more robust set of audio features using the Librosa library, such as MFCC, Chroma Features among many . Using this set, we achieved significantly improved accuracy, which was further boosted by segmenting each recording into six distinct audio recordings each capturing one digit entry.

5.5 Microphone-Specific Training

Finally, we trained our models using recordings from individual microphones (front, back, bottom and voice memo app) and found that the back microphone consistently yielded the best performance. This could be attributed to its physical proximity to the screen wiring and internal components generating the acoustic signals.

6 Data Pre-processing and Feature Extraction

6.1 Audio Segmentation

To prepare the data for analysis, each audio recording was segmented to isolate individual keystrokes corresponding to PIN digits:

- The first 2.5 seconds of each audio recording were discarded to remove the initial part which involved switching the tabs and going to the PIN entry page
- The remaining part of each audio file was 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, which is treated as a separate audio for the ML model

Here is the function for splitting each audio recording with truncating the initial portion.

```python
import os
import librosa
import soundfile as sf

def split_audio(input_path, output_dir):
    """
    Split an audio file into 6 equal parts after skipping the first 2.5
    seconds.

    Parameters:
    input_path (str): Path to the input audio file
    output_dir (str): Directory to save the split audio files
    """
    # Create output directory if it doesn't exist
    os.makedirs(output_dir, exist_ok=True)

    # Load the audio file
    audio, sr = librosa.load(input_path, sr=None)

    # Calculate the start point after 2.5 seconds
    start_sample = int(2.5 * sr)

    # Trim the first 2.5 seconds
    trimmed_audio = audio[start_sample:]

    # Calculate the length of each part
    part_length = len(trimmed_audio) // 6

    # Get the base filename without extension
    base_filename = os.path.splitext(os.path.basename(input_path))[0]

    # Split and save the audio parts
    for i in range(6):
        start = i * part_length
        end = (i + 1) * part_length if i < 5 else len(trimmed_audio)

        part = trimmed_audio[start:end]

        # Create output filename with input filename prefix
        output_filename = os.path.join(output_dir, f'{base_filename}_{i
+1}.mp3')

        # Save the audio part
        sf.write(output_filename, part, sr)
        print(f'Saved {output_filename}')
```

Listing 1: Function for splitting the audios

6.2 Audio Feature Extraction

We extracted the following features using the Python's Librosa library:

- **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 such subtle differences thus enabling classification of digits based on their acoustic patterns.

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

- **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.

- **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.

- **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.

- **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.

Here is the code we used for extracting the audio features from each splitted sample.

```python
import librosa
import numpy as np
import pandas as pd

def extract_audio_features(files, fs=44100, n_fft=2048, hop_length=512):
    """
    Extracts multiple audio features from a list of audio files.

    Parameters:
    - files: List of paths to the audio files.
    - fs: Sampling frequency (default 44100).
    - n_fft: FFT size (default 2048).
    - hop_length: Hop length for frame splitting (default 512).

    Returns:
    - pd.DataFrame: A DataFrame containing audio features.
    """
    features_data = []

    for i, audio_file in enumerate(files):
        # Load the audio file
        y, sr = librosa.load(audio_file, sr=fs)

        # Extract various audio features
        mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
        chroma = librosa.feature.chroma_stft(y=y, sr=sr)
        spectral_contrast = librosa.feature.spectral_contrast(y=y, sr=sr
)
        zero_crossing_rate = librosa.feature.zero_crossing_rate(y=y)
        rmse = librosa.feature.rms(y=y)
        spectral_rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)

        # Flatten and concatenate features
        feature_vector = np.hstack((
            np.mean(mfccs, axis=1),
            np.mean(chroma, axis=1),
            np.mean(spectral_contrast, axis=1),
            np.mean(zero_crossing_rate, axis=1),
            np.mean(rmse, axis=1),
            np.mean(spectral_rolloff, axis=1)
        ))

        features_data.append(feature_vector)

    # Convert to DataFrame
    df = pd.DataFrame(np.array(features_data))

    return df
```

Listing 2: Final audio feature extractor that gave the highest accuracy

6.3 Feature Standardization

Since the duration of the audio segments varied slightly, each feature produced a time series of different lengths. To standardize this:

- The mean of each feature over time was computed and then stored in a pandas dataframe

- StandardScaler was then applied to normalize all feature values to ensure zero mean and unit variance and no single feature would dominate the classification due to scale differences

7 Machine Learning Models and Training

To classify the segmented audio samples, each segment was labeled with its corresponding digit (0-9) and fed to different machine learning models to learn the features and classify the test samples. The dataset was then split into 70% for training, 15% for validation, and 15% for testing.

7.1 Models Evaluated

We evaluated the following models:

- **k-Nearest Neighbors (kNN)**
  - Hyperparameters: $n\_neighbors = 5$
- **Random Forest Classifier**
  - Hyperparameters: $n\_estimators = 100$ and $random\_state = 42$
- **Support Vector Machine (SVM)**
  - Hyperparameters: $kernel = linear$ and $random\_state = 42$
- **Neural Network**
  A Neural network was used to capture the complex interconnections between the multiple features extracted amongst and between audios of different classes. Here is a code snippet describing the neural network used.

```python
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import Adam

NN_model = Sequential()
NN_model.add(Dense(128, input_dim=X_train_scaled.shape[1],
    activation='relu'))
NN_model.add(Dropout(0.3))
NN_model.add(Dense(64, activation='relu'))
NN_model.add(Dropout(0.3))
NN_model.add(Dense(32, activation='relu'))
NN_model.add(Dense(y_train_onehot.shape[1], activation='softmax'))
     # Output layer

# Compile the model
optimizer = Adam(learning_rate=0.0001)
NN_model.compile(loss='categorical_crossentropy',
                 optimizer=optimizer,
                 metrics=['accuracy'])
```

Listing 3: Keras Neural Network model for multi-class classification

The neural network was our best-performing model, likely due to its ability to learn and capture patterns in the audio features:

- The dense architecture with dropout helped it generalize well on intra-class variations by capturing the non-linear interactions.
- Training for 500 epochs allowed the model to converge effectively to a good accuracy and low loss.
- The dropout layer (rate = 0.3) helped prevent overfitting during the extended training by randomly deactivating 30% of the neurons in the previous layer

8 Results and Analysis

8.1 Model Performance Comparison

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 for this task. The following are the test accuracies when tested on the entire dataset.

Model	**Accuracy**
Neural Network	90.09%
Support Vector Machine	81.59%
Random Forest	85.89%
k-Nearest Neighbors	86.17%

Table 1: Performance comparison of different machine learning models

8.2 Feature Importance (Random Forest)

Using the Random Forest model, we were able to extract feature importance rankings which is indicative of how crucial a feature is in classifying the audio sample:

- RMS Energy ranked highest making it the most important feature.
- MFCCs, Chroma, and Spectral Contrast also contributed significantly to distinguishing frequency patterns

RMS energy's high importance indicates that the signal strength and clarity of each keypress is a crucial factor in distinguishing between different digits.
Here is the graph showing feature importance of various features.

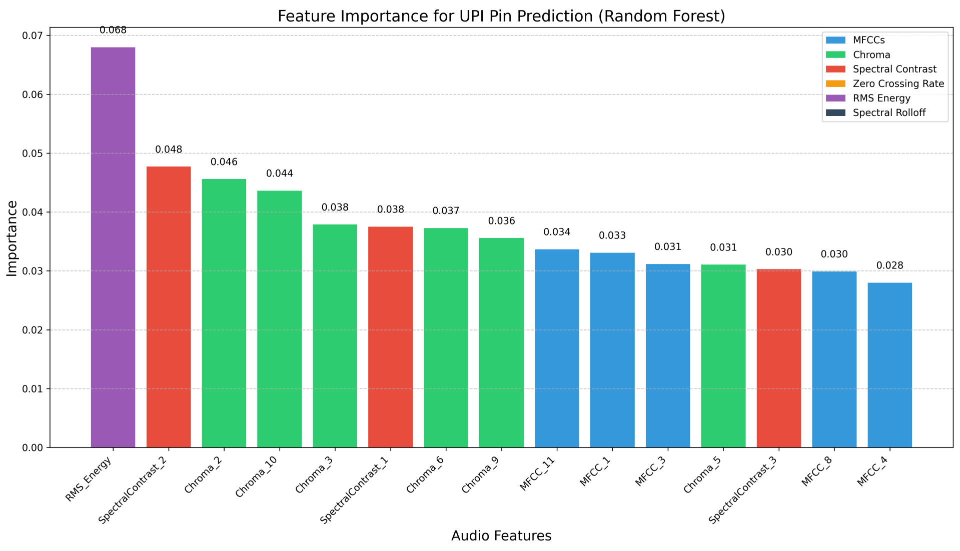

Figure 1: Feature importance plot with feature names.

8.3 Microphone Comparison

To understand which microphone captures audio signals most effectively, we trained and tested our models individually on the audios recorded from different microphones:

Microphone	**Accuracy**
Back Microphone	96.25%
Front Microphone	91.70%
Bottom Microphone	92.19%
All (Voice Memos app)	94.07%

Table 2: Classification accuracy by microphone position

The back microphone consistently outperformed other microphone positions. This is likely because:

- The display and touchscreen wiring are generally routed near the back side of the phone
- Back microphone is physically closer to the internal components that generate the acoustic emissions allowing it to capture discriminatory features more effectively

The improved accuracy using the back microphone reinforces the understanding that acoustic emissions from internal components are a significant source of information leakage in this attack.

9 Limitations and Challenges

9.1 Generalization Issues

When tested on fresh recordings (even in similar quiet environments), the model often misclassified PINs. The possible reasons include:

- Overfitting to environment-specific noise patterns present during our original data collection
- Background ambience changes, even if subtle, introduce variations which the model hasn't seen leading to misclassifications on the new data

9.2 Device Dependence

When tested on recordings from a different phone model (iPhone 6), the model couldn't perform well — highlighting its reliance on device-specific acoustic signatures. This could mean that:

- Different device models have different internal architectures affecting sound propagation and the audios received by the microphones making them have different combinations of feature values.
- The acoustic signatures produced by the clicks on the virtual keyboard are highly device-specific leading to misclassifications.

9.3 PIN Type Limitations

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.

9.4 Background App Interference

The experiments had only Google Pay and the recording app open. Keeping other apps running in the background might affect the audios recorded due to interfering noises

While our approach demonstrates a strong proof of concept, the current model lacks robustness across environments and devices, pointing to the need for improved generalization strategies and extension to real world scenarios like multiple apps running or non-uniform PINs

10 Attempts to deal with drawbacks

To overcome the identified limitations and improve the robustness of our acoustic side-channel attack, we explored several signal processing techniques focused on audio segmentation and noise reduction.

10.1 Autocorrelation-Based Audio Segmentation

We implemented an advanced segmentation algorithm based on autocorrelation to more precisely identify the boundaries between individual PIN digit presses. The rationale was to generalize to different digits appearing in the same PIN. The approach consisted of:

- Initial energy-based silence detection to identify potential boundaries between key-presses

- Frame-by-frame autocorrelation analysis to detect subtle transitions in the frequency domain along with peak detection
- Adaptive boundary selection that combines detected peaks, transitions with evenly distributed segments when necessary

The autocorrelation-based approach was designed to capture the natural rhythm of user input and adapt to variations in typing speed. However, despite the more sophisticated segmentation, this approach did not yield significant improvements in classification accuracy, suggesting that our initial uniform segmentation was sufficient for capturing the key acoustic signatures. This could be because in all the audios we do maintain a standard 1s gap between each digit press. And the model did not perform well on non-uniform PINs despite this new method.

10.2 Noise Reduction Techniques

We also investigated whether noise reduction might improve model performance by removing irrelevant acoustic information and hence reduce dependence on the environment and improve generalization. Our signal processing pipeline included:

- Notch filtering at 50 Hz to remove power line interference
- Low-pass filtering at 20 Hz to eliminate sub-audible noise
- High-pass filtering at 20 kHz to remove ultrasonic components

Contrary to our expectations, applying these noise reduction techniques resulted in a significant decrease in classification accuracy—from our original 96% down to approximately 20%. This substantial performance degradation provides important insight: the characteristic patterns that differentiate between digit keypresses are actually contained within the frequency components that were filtered out.

This finding supports our hypothesis that the most informative acoustic signatures come from electromagnetic emissions and electronic circuit responses rather than from audible mechanical interactions with the touchscreen. The frequencies eliminated by our filters—which we initially considered noise—actually contained the discriminative features essential for PIN digit classification.

These experiments reinforce our understanding that the attack relies on subtle electromagnetic and electronic emissions that manifest in the acoustic domain, rather than traditional audible sounds produced by physical interaction with the device.

11 Future Work and Improvements

11.1 Enhanced Dataset Diversity

To make the model robust across environments and devices we could build a 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

This allows the model to learn the invariant features characterising the digit entries irrespective of the environment or noise levels.

11.2 Transfer Learning Approach

Transfer learning represents a methodology where knowledge gained while solving one problem is applied to a different but related problem. This approach is particularly valuable for acoustic side-channel attacks where collecting extensive data across all possible devices and environments is tough.

- Transfer learning enables leveraging pre-trained models developed on source devices/environments to new target scenarios with minimal additional data collection
- By fine-tuning only specific layers of the neural network using a small set of samples from the target environment, the model can quickly adapt
- The pre-trained layers capture general acoustic features that remain consistent across different scenarios, while adaptation layers adjust to device-specific variations
- This approach could significantly reduces the data requirements from hundreds of samples to potentially 10-20 samples per digit
- Cross-device adaptation becomes feasible, allowing attacks to be rapidly deployed across different smartphone models after initial training
- Environmental factors (background noise, surface materials) can be better accommodated through transfer learning techniques

11.3 Advanced Deep Learning Techniques

Deep learning offers sophisticated approaches to feature extraction and pattern recognition that can significantly enhance acoustic side-channel attacks. Unlike traditional machine learning methods that rely on handcrafted features, deep learning models can automatically discover discriminative representations from raw or minimally processed data. Some examples are:

- Convolutional Neural Networks (CNNs) can process audio spectrograms as 2D images, identifying subtle patterns in both time and frequency domains that correspond to specific digit keypresses
- Recurrent Neural Networks (RNNs) model temporal dependencies in the acoustic signal, capturing the evolution of sound before, during, and after each keypress

The combination of transfer learning, advanced deep learning techniques and a large dataset holds significant promise for developing more robust acoustic side-channel attacks that work reliably across different devices, environments, and user behaviors. These approaches directly address the primary limitations identified in our current implementation, potentially elevating this attack from a highly monitored experiment to a practical security threat.

12 Conclusions

This research demonstrates that acoustic side-channel attacks pose a significant threat to PIN-based authentication in mobile payment applications like Google Pay. We established several key insights:

- The back microphone captures the most informative acoustic signals due to its proximity to internal hardware
- RMS Energy is the most discriminative feature for PIN classification
- Current models show limited generalization across devices and environments

Our project provides a strong proof-of-concept that PINs entered via virtual keyboards can be inferred using acoustic side-channel analysis. While accurate under controlled conditions, generalization to real-world settings remains a challenge, paving the way for future research in improving model robustness.

The success of these attacks highlights the need for improved security measures in mobile payment applications, particularly enhanced protection against acoustic leakage, and better monitoring of microphone access during sensitive operations.

References

[1] Giannakopoulos, T., & Pikrakis, A. (2014). *Introduction to Audio Analysis: A MATLAB Approach.* Academic Press.

[2] Qurthobi, A., Damaševičius, R., Barzdaitis, V., & Maskeliūnas, R. (2025). Robust forest sound classification using Pareto-Mordukhovich optimized MFCC in environmental monitoring. *IEEE Access*, *13*, 20923–20929. https://doi.org/10.1109/ACCESS.2025.3535796

[3] Zaman, K., Sah, M., Direkoglu, C., & Unoki, M. (2023). A survey of audio classification using deep learning. *IEEE Access*, *11*, 106620–106627. https://doi.org/10.1109/ACCESS.2023.3318015