MAYIBONGWE Moyo

R214568M_THESIS_f.pdf

University of Zimbabwe

Document Details

Submission ID
trn:oid:::6524:291840110

Submission Date
Jun 29, 2025, 6:49 PM GMT+2

Download Date
Jun 30, 2025, 8:16 AM GMT+2

File Name
R214568M_THESIS_f.pdf

File Size
5.7 MB

97 Pages

37,085 Words

225,973 Characters

5% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Submitted works

Match Groups

78 Not Cited or Quoted 2%
Matches with neither in-text citation nor quotation marks

3 Missing Quotations 0%
Matches that are still very similar to source material

73 Missing Citation 2%
Matches that have quotation marks, but no in-text citation

2 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

5% Internet sources
0% Publications
0% Submitted works (Student Papers)

Integrity Flags

1 Integrity Flag for Review

Replaced Characters
37 suspect characters on 18 pages
Letters are swapped with similar characters from another alphabet.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

UNIVERSITY OF ZIMBABWE

Dynamic Audio Watermarking Algorithm using PCA and Distinctive Detection

by

MAYIBONGWE MOYO

R214568M

Project submitted for review for the
COMPUTER ENGINEERING
in the Faculty of Computer Engineering Informatics and Communications at the University of Zimbabwe

Supervisor: Mr P Worsnop
June 27, 2025

Declaration

I, MAYIBONGWE MOYO, declare that this project, entitled "Dynamic Audio Watermarking Algorithm using PCA and Distinctive Detection," is my own work and has not been submitted for any other degree or professional qualification. All sources are properly acknowledged.
I affirm that this thesis was completed in accordance with the guidelines and requirements set forth by both the Faculty of Computer Engineering Informatics and Communications and the University of Zimbabwe.

Approval

This certifies that the project titled *'Dynamic Audio Watermarking Algorithm using PCA and Distinctive Detection'*, submitted by **Mayibongwe Moyo**, has been approved as part of the requirements for the *Bachelor of Science Honours in Computer Engineering (HCE)*:

Head of Department _____

Project Guide _____

Supervisor _____

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Mr. P. Worsnop, for his invaluable guidance, encouragement, and expertise throughout this research journey. His insights and feedback played a critical role in shaping the direction of this work. I am also deeply thankful to the Faculty of Computer Engineering, Informatics, and Communications at the University of Zimbabwe for providing the resources and support necessary to undertake this project. A special mention goes to my family and friends for their unwavering encouragement and patience during the challenging times of this endeavor. Finally, I am grateful to the broader academic and open-source communities, whose contributions have laid the foundation for this research. Specifically, I appreciate the creators of the VoxPopuli dataset and the tools such as TensorFlow and Librosa, which were integral to the technical implementation of this project. To everyone who has supported me, directly or indirectly, in the completion of this work—thank you.

Abstract

The need for digital audio traceability in today's world where content can be recycled, remixed, and manipulated through multiple stages is increasingly pressing. The available audio traceability schemes such as the blockchain-based systems and cloud hashing systems are in many cases not resilient enough adding this to the complexity they entail. While audio watermarking appears as a hopeful solution to robustness, traditional techniques tend to emphasize static, single-layer watermarking, which makes it impossible to track audio in a complex, multi-level setting. This study directly addresses these challenges by proposing a novel, deep learning-based audio watermarking system. It features a generator-detector neural network model operating on a Parallel Frequency Band (PFB) paradigm, specifically designed for multi-level traceability. Unlike conventional approaches, the system incorporates multiple distinct watermarks concurrently into the audio's frequency domain. The crucial innovation lies in leveraging learned basis vectors, conceptually inspired by Principal Component Analysis (PCA), to dynamically establish optimal, separable embedding spaces for each watermark. This enables distinctive detection capabilities, ensuring traceability through transformations while meticulously balancing imperceptibility, robustness, and capacity. The system's performance was rigorously evaluated using the Vox Populi database. Quantitative benchmarks, including Bit Error Rate (BER), Signal-to-Noise Ratio (SNR), and Mean Squared Error (MSE), demonstrated groundbreaking results. The refined 4-watermark system achieved an exceptional 33 dB SNR and a 0.0014 BER in clean conditions, signifying near-perfect message recovery with high auditory transparency. Its robustness was further validated against standard adversarial distortions, such as Additive White Gaussian Noise. While challenges persist in maintaining full message integrity under aggressive frequency-destructive attacks like low-pass filtering, these findings precisely delineate current frontiers for future research. The developed technology, embodied in a functional application, substantially fills the existing gap in audio traceability and watermarking, providing a highly imperceptible, scalable, and robust solution applicable in copyright protection, forensic analysis, and digital content security.

Contents

1 Introduction **7**
- 1.1 Introduction . . . 7
- 1.2 Background of the study . . . 7
- 1.3 Problem definition . . . 8
- 1.4 Aim . . . 8
- 1.5 Research Objectives . . . 8
- 1.6 Scope and limitations of the project . . . 8
- 1.7 Feasibility Study . . . 9
  - 1.7.1 Technical Feasibility . . . 9
  - 1.7.2 Economic Feasibility . . . 9
  - 1.7.3 Social Feasibility . . . 9
  - 1.7.4 Operational Feasibility . . . 10
- 1.8 Significance and motivation for the project . . . 10
- 1.9 Work Plan . . . 10
- 1.10 Conclusion . . . 10

2 Literature Review **11**
- 2.1 Introduction . . . 11
- 2.2 Review of relevant literature: . . . 11
  - 2.2.1 Traceability of data in audio watermarking . . . 11
  - 2.2.2 Frequency Domain Watermarking . . . 12
  - 2.2.3 Frontier watermarking techniques: Deep Learning based watermarking . . . 12
  - 2.2.4 Principal Component Analysis . . . 13
- 2.3 Discussion of similar systems . . . 15
  - 2.3.1 Blockchain-Based . . . 15
  - 2.3.2 Cloud-Based Hashing . . . 15
- 2.4 Identification of gaps . . . 16
- 2.5 Conclusion . . . 16

3 Methodology **17**
- 3.1 Introduction . . . 17
- 3.2 Algorithm Overview . . . 17
  - 3.2.1 Generator: . . . 18
  - 3.2.2 Detector: . . . 20
- 3.3 Training Pipeline . . . 22
- 3.4 Dataset . . . 23
- 3.5 Evaluation of Hypotheses . . . 24
  - 3.5.1 Hypotheses . . . 24
  - 3.5.2 Adversarial Attacks . . . 24
  - 3.5.3 Evaluation Metrics . . . 25
- 3.6 Tools for Implementation . . . 26
- 3.7 Conclusion . . . 27

4 System Analysis and Design **28**
- 4.1 Introduction . . . 28
- 4.2 Detailed analysis of the problem domain and user requirements . . . 28

4.2.1 Functional Requirements . . . 29
4.2.2 Non-functional Requirements . . . 30
4.3 Identification of system components and functionalities . . . 30
4.3.1 Use-Case diagram/s . . . 32
4.3.2 Sequence diagram . . . 33
4.4 System architecture and design considerations . . . 33
4.4.1 Context Diagram and DFD Diagram . . . 34
4.4.2 Architectural Design (Software Component Interaction & ML Model Integration) . . . 35
4.4.3 Physical Design . . . 36
4.4.4 Database Design . . . 36
4.4.5 Interface design . . . 36
4.4.6 Security design . . . 39
4.5 Conclusion . . . 39

5 Results **41**
5.1 Introduction . . . 41
5.2 Presentation of Findings . . . 41
5.2.1 Phase 1: Evaluation of State-of-the-Art Tools for Incremental Watermarking . . . 41
5.2.2 Phase 2: Development of a Custom DSSS Algorithm . . . 44
5.2.3 Phase 3: Development and Refinement of a Custom Learned PFB System . . . 45
5.3 Conclusion . . . 52

6 Discussion **53**
6.1 Introduction . . . 53
6.2 Summary of Findings . . . 53
6.3 Model Evaluation and Analysis . . . 54
6.3.1 Imperceptibility: The Achievement of Auditory Transparency . . . 54
6.3.2 Robustness: Near-Perfect Message Recovery and Identified Vulnerabilities . . . 54
6.3.3 Detection Sensitivity and Specificity: Broader Evaluation Context . . . 55
6.3.4 Evolution of Model Architecture: Blueprint vs Realization . . . 55
6.4 Analysis and Evaluation of Hypotheses . . . 57
6.4.1 Hypothesis 1: Watermarking vs. Other Traceability Frameworks . . . 57
6.4.2 Hypothesis 2: PCA-based FDW vs. 'Simple' Techniques . . . 57
6.5 Comparison with Existing Literature . . . 58
6.5.1 Validation of Initial Hypotheses and Problem Statement . . . 58
6.5.2 Advancing Frequency-Domain Watermarking (FDW) . . . 58
6.5.3 Innovation in Distinctive Detection and PCA's Role . . . 59
6.5.4 Comparison with State-of-the-Art (SOTA) Audio Watermarking . . . 59
6.5.5 The DSSS Baseline in Context . . . 59
6.6 Theoretical Implications . . . 59
6.7 Practical Implications . . . 60
6.8 Validation and Reliability . . . 61
6.9 Limitations and Methodological Reflections . . . 62
6.10 Conclusion . . . 63

7 Conclusion and Future Work **64**
7.1 Introduction . . . 64
7.2 Summary of the Project . . . 64
7.3 Key Findings and Contributions . . . 65
7.4 Evaluation of Objectives . . . 65
7.5 Reflection on the Project Process . . . 66
7.6 Future Work and Recommendations . . . 67
7.7 Conclusion . . . 67

Bibliography **68**

A Code and Supplementary Materials **71**
A.1 Core Watermarking Algorithms (`pca_prime_watermarking.py` excerpts) . . . 71
A.1.1 PFBGenerator Architecture . . . 71
A.1.2 PFBDetector Architecture . . . 72
A.1.3 Incremental Embedding Logic . . . 74
A.1.4 Detection Logic . . . 76
A.2 Backend API Orchestration (`app.py` excerpts) . . . 78
A.3 Frontend User Interface Excerpts (`Dashboard.tsx`) . . . 82
A.4 Testing and Evaluation Script Excerpts (`experiment_y_prime_visual_audio_test.py`) . . . 86
A.5 Environment Setup (`requirements.txt`) . . . 87
A.6 Training Pipeline Parameters and Sample Logs . . . 87
A.6.1 Hyperparameters and Constants . . . 87
A.6.2 Sample Training Log Excerpt . . . 87

B User Manual for Audio Watermark Lab **89**
B.1 Introduction to the Audio Watermark Lab . . . 89
B.2 System Requirements and Setup . . . 89
B.2.1 Prerequisites . . . 89
B.2.2 Installation Guide . . . 89
B.3 Application Modes Overview . . . 90
B.3.1 Application Mode (Dashboard) . . . 90
B.3.2 Research Mode . . . 90
B.4 Using the Application Mode (Dashboard) . . . 91
B.4.1 Embedding Watermarks . . . 91
B.4.2 Detecting Watermarks . . . 91
B.5 Using the Research Mode . . . 91
B.5.1 Configuring Watermarks and Attacks . . . 91
B.5.2 Running Experiments and Analysis . . . 92
B.6 Troubleshooting and FAQs . . . 92
B.6.1 Common Issues . . . 92
B.6.2 Tips for Optimal Performance . . . 92

List of Figures

1.1 Gantt Chart . . . 10

3.1 Watermarking Generator architecture with added PCA layer . . . 18

4.1 Main System Use-Case Diagram . . . 32
4.2 System Context Diagram . . . 34
4.3 Level 0 Data Flow Diagram . . . 35
4.4 Software Component Interaction Diagram . . . 36
4.5 Database Entity-Relationship Diagram . . . 36
4.6 Application Mode Navigation Menu . . . 37
4.7 Research Mode - Watermark Configuration Panel . . . 37
4.8 Application Mode - Embed Watermark Input . . . 38
4.9 Research Mode - Performance Metrics Display . . . 38
4.10 Application Mode - Watermark Information Output . . . 39
4.11 Application Mode - Watermarking History/Ledger Display . . . 39

5.1 Performance Trends Across Multiple Watermarks: SFA, SDA, and Early PFB Comparison. This figure illustrates (top-left) BER, (top-right) SNR, (bottom-left) Detection Probability, and (bottom-right) False Positive Rate as sequential watermarks are embedded. It visually highlights the high BER variability and detector saturation in SFA/SDA, contrasted with PFB's more stable (albeit high) SNR and low detection in early attempts. . . . 43
5.2 Conceptual Illustration of Average Final SNR and BER for SFA, SDA, and PFB (Early Attempts). This figure visually represents the data from Table 5.1, emphasizing the comparative performance across methods. . . . 43
5.3 Comparative Performance: Experiment X vs. Experiment Y-Prime (Overall BER and SNR). This figure visually illustrates the significant improvements achieved by the refined architecture. . . . 49
5.4 Waveform of the original audio. . . . 49
5.5 Spectrogram of the original audio. . . . 49
5.6 Waveforms and spectrograms of the clean and noise-added audio after applying WM1. **Clean Audio:** SNR = 36.18 dB, Average BER = 0.0000 **Noisy Audio:** SNR = 19.87 dB, Average BER = 0.2039 . . . 50
5.7 Waveforms and spectrograms of the clean and noise-added audio after applying WM2. **Clean Audio:** SNR = 36.16 dB, Average BER = 0.0000 **Noisy Audio:** SNR = 19.89 dB, Average BER = 0.3224 . . . 50
5.8 Waveforms and spectrograms of the clean and noise-added audio after applying WM3. **Clean Audio:** SNR = 36.14 dB, Average BER = 0.0000 **Noisy Audio:** SNR = 19.87 dB, Average BER = 0.3509 . . . 51
5.9 Waveforms and spectrograms of the clean and noise-added audio after applying WM4. **Clean Audio:** SNR = 36.12 dB, Average BER = 0.0000 **Noisy Audio:** SNR = 19.88 dB, Average BER = 0.3289 . . . 51

List of Tables

5.1 Transposed Performance Metrics for SFA and SDA Methods 42
5.2 DSSS with Simplified BER (Average Results) 45
5.3 Experiment 2: Learned PFB System Performance with Learnable Strength and High Imperceptibility Penalties 46
5.4 Experiment X: Robustness Evaluation for 4-Watermark Parallel PFB System (Baseline Scaling) 47
5.5 Experiment Y-Prime: Robustness Evaluation for 4-Watermark Parallel PFB System (Refined) 48

A.1 Experiment Y-Prime Training Hyperparameters 87

Nomenclature

PCA - Principal Component Analysis
DRM - Digital Rights Management
SOTA - State Of The Art
BER - Bit Error Rate
MSE - Mean Square Error
SNR - Signal-to-Noise Ratio
COC - Chain of Custody
MPP - Music Production Pipeline
AI - Artificial Intelligence
DL - Deep Learning
PESQ - Perceptual Evaluation of Speech Quality
ViSQOL - Virtual Speech Quality Objective Listener
AUC - Area Under Curve
ACC - Accuracy
FDW - Frequency Domain Watermarking
LSB - Least Significant Bit
DFT - Discrete Fourier Transform
DWT - Discrete Wavelet Transform
WOLA - Weighted Overlap-Add
STFT - Short-Time Fourier Transform
STCT - Short-Time Cosine Transform
MLPCA - Multi-Layered Principal Component Analysis
MDCT - Modified Discrete Cosine Transform
PVQ - Perceptual Vector Quantization
TTP - Trusted Third Party

Chapter 1

Introduction

1.1 Introduction

This research introduces a sophisticated approach to the increasingly sophisticated problem of robust audio traceability in this era of digital rights. As the first chapter of the research, this section will serve as a base for the entire thesis by introducing the problem, and how we plan to solve it. The background and problem statement give context for the increasing cruciality of audio traceability in the age of re-working audio, highlighting the relevant research leading to this point. The aim and research objectives will guide the research, offering a systematic approach to addressing the problem, and crucially serving as the reference points for evaluating the success of the research. The research's scope will also be discussed, with justifications for the chosen boundaries. Lastly, the expected outcome and subsequent impact of the research are also mentioned.

1.2 Background of the study

Tracing the origin and utilization of audio content is becoming increasingly important for addressing unauthorized distribution, confirming authenticity, and safeguarding intellectual property rights. This may be anywhere in the lifecycle of several scenarios such as repurposing, chain of custody (forensics and court proceedings), music production pipelines, and social media. Current traceability systems have major security deficiencies, falling back in the age of advanced adversarial attacks due to the compromised robustness of the audio fingerprints utilized to trace audios during lifecycles. On the other hand, robustness has largely been a consistently addressed problem within the field of audio watermarking. Audio watermarking methods provide a viable solution by embedding subtle identifiers within audio signals, facilitating the validation of content as it is circulated. Its applications encompass copyright protection, content verification, and broadcast surveillance, among others. Nevertheless, the existing techniques are not adpated for traceability in chaining or pipeline due to storage constraints for the metadata. This issue stems from the necessity to strike a balance between robustness, imperceptibility, and capacity, as enhancing one aspect often detracts from the others. The challenge is further exacerbated by the complexity of upholding watermark resilience in light of advancing technologies such as voice cloning. These factors make it virtually impossible for a single watermark to both survive, and keep track of where it has passed over the long term. Although multi-bit watermarking and synchronization codes improve traceability by enabling the identification of specific sources or versions, achieving dependable traceability across a range of scenarios remains a formidable task. Audio watermarking techniques have predominantly emphasized static, single-layer watermarking, whilst the more advanced counterpart of image watermarking (Begum and Uddin 2021) has seen the development of multilayer or cumulative methods. The Multiple Image Watermarking paper (Begum and Uddin 2021) inspires the notion that watermarking can go beyond copyright protection and tamper detection, to being a self contained traceability system within itself. Most notable is the leap it would make in reducing complexity and decoupling from third party systems such as blockchain and cloud-based pipelines. A cumulative watermarking algorithm extended from Facebook research's generator-detector architecture is proposed, which will make use of Principal Component Analysis training layer to synchronously identify, compare and append sequential watermarks dynamically. This has the potential of addressing the distinct challenges faced in reliably

tracing audio files amidst potential manipulation techniques, in the context of shared and repurposed media.

1.3 Problem definition

Current audio Digital Rights Management (DRM) systems cannot autonomously trace audios through multi-level lifecycles.
There is no balance of robustness, imperceptibility and capacity to ensure provenance without involving third-party systems. Advanced watermarking techniques serve as the SOTA (state of the art) for authentication, but none are optimised for traceability in chained processes such as music production pipelines, remixes or social media reuse for example. Therefore, a solution is needed to dynamically embed provenance data within an audio file such that it can be extracted after multiple stages and transformations, without compromising the listening experience.

1.4 Aim

To introduce a multi-level audio traceability system built on localisation watermarking and a PCA training layer for layering multiple distinct watermarks within the same audio.
This system will be built upon the state-of-the-art AudioSeal algorithm from Facebook research, specifically redesigned for provenance that persists through audio modifications, collaboration, and autonomous chain of custody. The goal is to ensure that all audios within any type of lifecycle remain distinguishable and identifiable even after a reasonable number of increments.

1.5 Research Objectives

Investigate

1. Evaluate the traceability of state-of-the-art watermarking algorithms (such as AudioSeal) against incremental watermarking. Observe benchmarks such as BER, MSE, and SNR to evaluate robustness and imperceptibility.

Develop

2. Develop a PCA training layer which will facilitate dimensionality reduction, optimal band selection and efficient partitioning of watermarks.

3. Develop and train a feature extraction detector that will detect the distinct watermarks from a single audio.

Re-Investigate

4. Assess the proposed system's incremental traceability using the benchmarks from Objective 1 to validate the effectiveness of the new approach.

1.6 Scope and limitations of the project

This system will be specifically evaluated in the context of modified and reshared audio. The focus is on maintaining watermark integrity and audio quality after the audio undergoes transformations typical of these processes (pitch shifting, resampling, etc, and verifying the provenance of audio in scenarios where processes are chained.
The evaluations on robustness will only be as good as the original Facebook Research' AudioSeal (Roman et al. 2024) algorithm it is developed from, as the goal is not to enhance robustness, but attempt to maintain it as audios develop in their lifecycle. Furthermore, it is unlikely to account for all adversarial attack types as some modern cloning attacks will be suited to exploiting the multi-frequency band technique used.
The focus will be on maintaining traceability against the following:

- Remixing and Mashups:
Remixing entails modifying the composition and components of an original music, frequently

by incorporating additional layers, rearranging sequences, or integrating it with other recordings. This procedure attenuates the original acoustic characteristics, undermining the robustness of embedded watermarks. Ensuring traceability in these situations necessitates watermarks that can endure substantial changes in both spectral and temporal domains, preserving detectability despite massive mutations.

• Remakes and Cover Versions:
In remakes or covers, the original material is re-recorded or reinterpreted, often with new instruments, vocals, and effects. Such changes can eliminate or conceal watermarks. Evaluating watermark resistance in remakes emphasises embedding techniques that utilise latent information less susceptible to distortion during reproduction, such as higher-order spectral harmonics.

• Chain of Custody Leaks:
In legal contexts or forensic enquiries, audio data often circulate among multiple parties, including police enforcement, legal teams, and forensic analysts. At every stage, issues arise from inadvertent alterations (e.g., compression artifacts or format changes) or intentional sabotage. More often, bad actors may attempt to distribute the audio content unlawfully. Effective watermarking must provide traceability and authenticity, even when files undergo editing, transcribing, or archiving. Techniques that sustain integrated evidence integrity throughout these stages are critical for assuring credible testimony in court and preserving the admissibility of evidence.

• Music Production Pipelines:
Modern music production involves numerous stages, such as mixing, mastering, and effects processing. Each stage can affect the frequency content, dynamics, and timing of the audio. To ensure traceability, watermarks must resist dynamic range compression, equalization, and other transformations typical in production workflows.

1.7 Feasibility Study

1.7.1 Technical Feasibility

Available Technologies
- AudioSeal algorithm is to be used as base implementation
- PCA implementation in frequency watermarking has been exemplified in previous work Technical Risks
- The integration between the modified algorithm and the PCA layer could be technically complex but can be countered by a phased implementation with extensive testing.
- Performance overhead of both a distinct localisation detector and a PCA layer: this potential flaw will be investigated as more test cases are added to the system, and its risk-to-reward ratio compared with the alternatives.

1.7.2 Economic Feasibility

Development Costs (Minimal)
- All software components are open-source
- No licensing costs
- Potential cloud hosting costs for testing (minimal, wherein free tiers could be employed) Potential Financial Benefits
- Potential partnerships with content platforms
- Potential users include record labels, content creation platforms, music streaming services, law enforcement and media authentication services.

1.7.3 Social Feasibility

Impact on Copyright Protection: By providing tools to trace and verify audio at multi-level scenarios, the project addresses growing concerns about misuse of original content, more-so in the AI age. If successful, it could contribute to future academic research in the same same vein, or act as a foundation for industry applications.

1.7.4 Operational Feasibility

The project is considered feasible, requiring the following for implementation:
-Basic web interface for testing
-Test dataset of audio samples
-Documentation of results

1.8 Significance and motivation for the project

The research will have the following implications for the status quo of audio ownership in AI environments:
- Copyright Protection: Ensuring proper attribution and preventing unauthorized and uncompensated use of audios, including permitted modifications of original work. This will further ensure that data owners are compensated properly and have oversight over how their data is used, by whom.
- Content Authentication: Verifying the authenticity of audio recordings in legal and forensic contexts. This also involves identifying 'leaks' in chains of custody as watermarks can be traced back to the last holder of an audio before it migrated out of the system.

In addition to these, the following are the expected significances:
-Adding to the body of research of the intersection between audio cryptography, statistical signal processing and deep learning.
-Developing a practical system that demonstrates effective strategies to mitigate verification loss as digital content evolves and scales through various modifications and processing systems.
-A thorough comparison of the proposed system versus the current watermarking and blockchain techniques, focusing on maintaining robustness and imperceptibility.

1.9 Work Plan

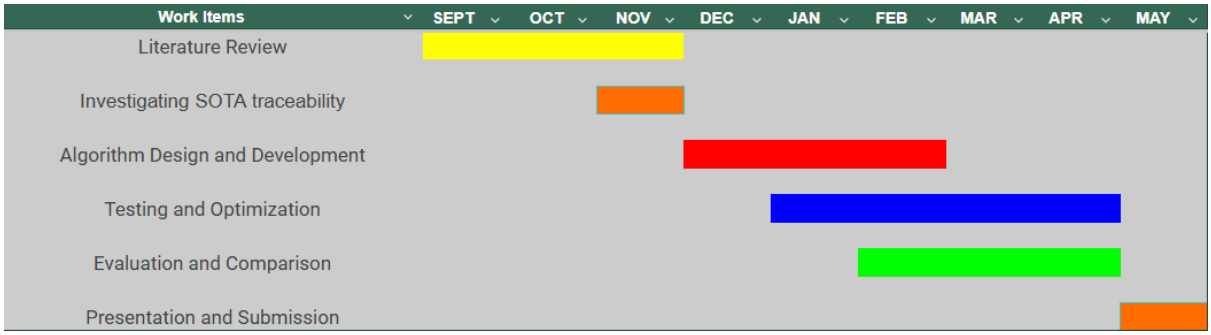


Figure 1.1: Gantt Chart

1.10 Conclusion

This chapter has outlined the growing challenges in digital audio rights management within their lifecycles, and how dynamic watermarking can potentially address this issue. While previous watermarking techniques have excelled at robustness and imperceptebility compared to other security techniques, they often fall short in ensuring effective traceability across diverse use cases.The proposed dynamic, multi-level audio traceability system, leveraging the state-of-the-art AudioSeal algorithm and integrating a PCA training layer, presents a promising solution to the limitations of existing frameworks. By allowing for the embedding of multiple distinct watermarks within the same audio signal, this approach enhances the ability to identify and verify audio content throughout its lifecycle, even amidst transformations such as remixing and re-recording. Ultimately, the findings from this research have the potential to influence future developments in audio rights management, paving the way for more 'balanced' systems that can adapt to the evolving landscape of digital content in both robustness and traceability.

Chapter 2

Literature Review

2.1 Introduction

Previous works in audio watermarking have all sought to balance an established triad of factors. These are robustness, imperceptibility and capacity. This section will highlight the successes and shortfalls of previous methods in satisfying the triad. This will be determined by the quantifications on detection accuracy, perceptual quality and capacity. More attention may be given to the last-mentioned as it is the influencer of the traceability of watermarks and the data they carry. The chapter goes through literature in layers of the research from traceability, to waterarmking, frequency watermarking, deep learning in frequency watermarking and so on. There is an evaluation of possible methods that can be employed in multi-layering the mark, which subsequently justifies the chosen method and why it is considered to be 'dynamic'. The researcher finally zooms out of watermarking as a solution to traceability, and evaluates the existing methods and how they would compare to the proposed system.

2.2 Review of relevant literature:

2.2.1 Traceability of data in audio watermarking

Watermarking provides a method to enforce copyright and authentication within an audio signal without significantly affecting its perceptual quality. As embedded data is a digital fingerprint, it can equally facilitate content tracking. Although previous methods have been developing strong detection and extraction mechanisms mainly intending to prioritize robustness and imperceptibility, stepwise progress has been made on traceability. Albeit limited, techniques such as multi-bit watermarking and synchronization codes improve this, allowing for the identification of specific users, models, or versions. However, the concept of dynamic watermarking is itself so far only witnessed in the image domain, where Lamport's algorithm and public key cryptography are leveraged to create marks that change in every stage of distribution. In audio, tracking has mainly been achieved by the integration of blockchain technology to establish a tamper-proof record of ownership and distribution. Other than the few attempts of this method mentioned in 2.3, the field is largely vacant.
Cumulating or layering watermarks is important in cases where stakeholders need to trace altered or distributed content to its source, along with any subsequent modifications.

Copyright Protection and Ownership Verification: In music, there is a need to establish a clear lineage of content from recordings, through alterations and formats, to distribution, as a way to confirm ownership for all involved whilst reducing chances of piracy.

Digital Rights Management (DRM): In this streaming age, content producers often employ trusted third parties (TTPs) who are responsible for detailing usage rights, essential for upholding licensing agreements. This is only but a necessary choice, given the reported cases of collusion and exploitation by TTPs. Cumulative watermarking could potentially replace TTPs and make headway in establishing transparent compensation to all successive owners.

Forensic Analysis: In instances of audio leaks or unauthorized recordings, comprehensive traceability enables organizations to determine the source of a leak. As an example, Webex employs audio water-

marking to embed unique identifiers for each participant in virtual meetings.
An extension can be made from this to custodial chains wherein confidentiality is imperative. For instance, a proposed system for multimedia distribution networks merges a watermarking algorithm with blockchain technology to associate copyright information with audio files. This set of strategies guarantees traceable ownership, along with authentication and distribution tracking, with the trade-off of computational efficiency and complexity. Nevertheless, the rapidly changing landscape of audio manipulation, especially in voice cloning and social networks, presents challenges for maintaining traceability as re-purposing and evidence chains become more eminent.

2.2.2 Frequency Domain Watermarking

Using the frequency domain for watermarking has increasingly become the norm, mainly because of its resistance to several signal processing attacks and its capability to preserve audio quality. Frequency-domain techniques employ transformations like the Discrete Fourier Transform (DFT) or Discrete Wavelet Transform (DWT) in contrast with conventional time-domain techniques such as least significant bit (LSB) modification, which often disrupt audio integrity and are susceptible to noise interference. In FDW, watermarks are embedded within specific coefficients, ensuring that the watermark remains undetectable while also being resilient to modifications. (Uddin et al. 2024) observed that these approaches strike a more favorable balance between imperceptibility and robustness, rendering them ideal for applications in copyright protection and content verification. Ablation indicates that frequency-domain techniques are highly versatile and efficient in most practical situations, solidifying their status as the prevailing standard in audio watermarking methodologies. The burden rests in describing how they ought to be used most effectively, as per prior research.

Previous work investigates the application of frequency layers or bands for the purpose of embedding watermarks in audio signals, with the objectives of achieving both imperceptibility and robustness. In the work (Natgunanathan et al. 2022), Nadeau and Sharma concentrate on addressing desynchronization issues that arise during analog playback. They implement a spread-spectrum watermark within the frequency domain utilizing the WOLA transform. Notably, both their proposed method and the baseline approach intentionally steer clear of low frequencies, which are prone to perceptual distortion, as well as high frequencies, which are frequently vulnerable to loss during compression. Instead, both strategies focus on a 'midrange of frequencies.'

However, they differ in their resolution within this midrange. The baseline method embeds the watermark in the phase of each individual WOLA coefficient, whereas the proposed method adopts a coarser technique by aggregating WOLA coefficients into bands and embedding the watermark in the magnitude. Each band represents a single watermark 'chip,' with the bandwidth of these bands increasing exponentially with frequency, akin to a logarithmic scale. This method of grouping coefficients into bands effectively lowers the resolution of the watermark, which, as Nadeau and Sharma indicate, enhances robustness against time-scale alterations such as warping. These findings set a precedent for how watermarking in the frequency domain in general is approached.

In a similar vein, the time-frequency watermarking framework introduced by (Zhang 2020) utilizes the concept of frequency bands for watermark embedding. Zhang's methodology employs either the STFT or STCT to depict the audio signal as a two-dimensional time- frequency image. Rather than uniformly embedding across the entire spectrum, this approach emphasizes the identification and utilization of low-energy patches within the image. The rationale for this technique is to embed the watermark in segments that are likely to correspond to silent or noise-dominated areas of the audio, thereby minimizing perceptual impact. The process initiates with the selection of low and middle frequency bins, deliberately excluding very low and very high frequencies.

The adoption of FDW and specifically the use of bands within the frequency as this work suggests provide a solid foundation for incrementing varied watermarks on the domain, making note of the sensitivity of the domain and hence the need to pick bands optimally.

2.2.3 Frontier watermarking techniques: Deep Learning based watermarking

Most deep-learning-based audio watermarking techniques incorporate some level of multi-bit watermarking, which involves embedding a fixed binary message for the purpose of content identification. In current

models, increasing the embedded message reduces robustness exponentially, showing the trade-off between the complexity of the message and the resilience of the watermark.

In WavMark (Chen et al. 2024), there is a commendable improvement in capacity which is equally balanced by enhanced robustness; wherein about 31% of the 32bps watermark is reserved for pattern identification and the remaining bits carry a duplicated payload. In encoding, it uses invertible neural networks wherein encoding and decoding share the same parameters, whilst in detection it brute forces its way through each window of the audio, slowly scanning in small increments to locate the watermark. This allows WavMark to achieve an average BER of only 0.48% across 10–20 seconds of audio. However, because of the payload, it has an SNR of 36.85 dB and a PESQ of 4.21.

On the other hand, AudioSeal (Roman et al. 2024) focuses on cloning attacks, furthering robustness. It achieves an average AUC of 0.97 (compared to 0.84 for WavMark) by leveraging the EnCodec compression method to embed watermarks at the sample level for localized detection. This significantly reduces the processing time (14x faster generation and 485x faster detection on average), making it ideal for real-time applications. This detection efficiency is achieved whilst maintaining the same imperceptibility scores as WavMark. The psychoacoustic approach of employing perceptual losses and a localization loss to mimic human ear response allows it to have slight increases of 0.168 and 0.099 for PESQ and ViSQOL respectively. There is an optional message processing layer in the generator, which can process a single fixed 16-bit message - which unsurprisingly cannot go any further than only confirming the model version. However, its state-of-the-art robustness makes it a rightful foundation for building a traceability system, which would aim at keeping as close as possible to the robustness metrics.

Timbre watermarking (C. Liu et al. 2023) emerges as the first Deep Learning technique to take a specialised approach to watermark embedding into the frequency domain of the same audio. Before this, all DL techniques simply used spread spectrum to statically embed watermarks, Timbre on the other hand uses an approach that ensures watermarks avoid potentially distorted frequency bands. First is multi-frequency layering where information is repeatedly embedded across various time intervals, followed by a distortion layer which simulates common voice cloning processes. The aim is to combat cloning attacks by watermarking the medium-to-low frequency range of an individual's speech, whilst the distortion layer trains the embedding and extraction modules for robustness against common cloning transformations. Besides this, ACC still keeps near 100%, except when the embedded watermark bits are extremely large. The technique achieved a PESQ of 0.9891 against watermark overwriting attacks, and an SNR of 28.1650. It also retained 66.36% of the watermark information even when the voice cloning model was trained on a dataset with only a 25% ratio of watermarked speech. These results infer a success, although in training the approach introduced a distortion layer in the same frequency domain as the watermarks. This begs the question as to how FDW operates and how it has evolved to become a standard.

2.2.4 Principal Component Analysis

2.2.2 highlights the need to carefully choose the regions wherein the watermarks are embedded in the frequency domain. This factor is amplified in this context wherein distinctive watermarks with different identities ought to be layered in the same domain. Implied by previous research is the potential of a known statistical technique, Principal Component Analysis (PCA), to optimize the selection and allocation of embedding regions by analysing the spectral properties of the watermarks. It can then classify them by similarity and facilitate partitioning and managing the use of frequency bands, minimizing overlap and ensuring each watermark's integrity within a shared domain. thereby. Principal Component Analysis (PCA) may emerge as an effective method for enhancing architectures such as AudioSeal into layered watermarking systems by ensuring minimal interference and maximum traceability, but it has been employed by several systems prior, for adjacent reasons.

PCA for Audio Fingerprinting and Coding
(Julius et al. 2023) examines the application of audio fingerprinting through a Multi-Layered PCA (MLPCA) methodology, aimed at extracting the essential features of audio signals from them, for identification purposes. Principal Component Analysis (PCA) is used to reduce the dimensionality of the audio data while retaining critical information, thereby creating a 'signature' which can also be used to identify similar audio clips. The process begins with the preprocessing of the audio signal, wherein it is down-sampled and segmented into frames, and these frames are transformed from time domain to frequency domain via a linear operation such as the Modified Discrete Cosine Transform. Subsequently, the MLPCA algorithm is applied in layers to capture the temporal relationships among frames reducing

the dimensions further. The end result is a low-dimensional representation of the audio signal, which functions as a unique fingerprint for identification.

The work by (Burka n.d.) also addresses the role of PCA in audio coding to achieve high compression rates. It begins with a time to frequency domain transformation using the mentioned MDCT, allowing better data representation and compression. Following this, multiple frames of frequency coefficients are organized into a matrix for PCA analysis. In processing multiple frames, PCA recognises patterns and redundancies across them, allowing for more efficient compression. The resulting PCA matrices are then quantized using Perceptual Vector Quantization (PVQ) to represent them with fewer bits.

Burka extends this approach to stereo audio coding by incorporating stereo-specific techniques. Initially, PCA is applied to the combined matrix of frequency coefficients derived from both stereo channels, reducing the parameter estimation required to achieve quality compression. Subsequently, spatial parameters such as Inter-channel Level Difference (ILD) and Inter-channel Coherence (IC) are computed based on the outcomes of the PCA, this determines the spatial relation of the two audio channels. Finally, both the PCA matrices and the spatial parameters undergo quantization and encoding to reduce data size and facilitate efficient transmission. This technique delivers high-quality audio at a lower bitrate, with subjective evaluations surpassing those of established codecs such as G.719 and HE-AACv2.

RPCA for Singing Voice Separation

(Li and Chang 2021) investigates the application of Robust Principal Component Analysis (RPCA) for isolating singing voices from instrumental music. This algorithm is selected for its proficiency in differentiating vocals from music by leveraging the premise that the music resides in a low-rank subspace (predictable), while the singing voice is sparse or irregular. The procedure begins with converting the audio signal into a spectrogram format through the STFT. Following this, the spectrogram magnitude is decomposed into low-rank (representing the music accompaniment) and sparse (representing the singing voice) components using RPCA algorithms. The separated signals are reconstructed from the spectrogram using the inverse STFT along with the original phase information. This source introduces Weighted RPCA (WRPCA), an enhancement of RPCA that employs weighted values to overcome the limitations of traditional RPCA when dealing with elements such as drums, which may be misclassified as sparse rather than low-rank. In WRPCA, weights are applied to the sparse and low rank matrices, giving a more accurate non-binary representation. This adjustment ensures that both the singing voice and music accompaniment are appropriately scaled, ensuring they are better isolated. The authors have reported that WRPCA yields higher Signal-to-Distortion Ratio (SDR), Signal-to-Interference Ratio (SIR), and Normalized Signal-to-Distortion Ratio (NSDR) in comparison to conventional RPCA.

(Papadopoulos and Ellis 2014) introduce A-RPCA, a novel variant of adaptive RPCA which adapts based on the content of the music for more precise separation. A-RPCA is more comprehensive as it uses music content information, such as voice activity, to improve the decomposition process. The music signal is segmented into blocks according to its structure, and the RPCA's regularization parameter is modified depending on whether vocals are present in the segment. This methodology results in improved separation, being most useful in tracks featuring instrumental segments. Experimental results using A-RPCA revealed substantial improvements in separation quality relative to the RPCA method, with significant advancements in SDR, SIR, and SAR for both vocal and background layers.

RPCA for Audio Watermarking

The paper by (Wu, Ding, and Wei 2022) explores the application of RPCA in audio watermarking techniques, particularly in secure echo-hiding. RPCA decomposes the audio into low-rank (music) and sparse (watermark) parts. The watermark is embedded as a sparse signal, making it less detectable. The procedure begins with the decomposition of the original audio signal into low-rank and sparse components using RPCA. Subsequently, the watermark is embedded separately into these components using modified pseudo-noise sequences. During the watermark detection phase, the low-rank and sparse characteristics of the embedded echoes are utilized to retrieve the watermark from the respective components following RPCA decomposition. Evaluations indicated that this approach surpasses conventional echo hiding and PN sequence-based methods in terms of security, while also maintaining comparable or superior robustness and inaudibility.

PCA's capability to discern essential data components and reduce dimensionality with minimal information loss is particularly suited for the adaptive placement of watermarks in complex, multi-layered systems. By incorporating PCA into the AudioSeal framework, the system can be improved to dynami-

cally assign frequency bands for progressive watermarking, thereby guaranteeing optimal separation and traceability while maintaining the integrity of existing embeddings.

2.3 Discussion of similar systems

This section will focus on the state-of –the-art systems for each case, dwelling on the traceability aspect of the systems.

2.3.1 Blockchain-Based

Blockchain and Fingerprinting for Traceability
(Xiao et al. 2023) iterate on traditional centralized copyright platforms with a focus on addressing issue they have with piracy, data manipulation, and complex deposition procedures. The Music Copyright depository framework incorporating Blockchain and Edge computing (MBE) integrates edge computing into the Hyperledger Fabric blockchain system to register, verify, and manage music copyrights. The framework compresses and splits the audio into small chunks, performs a Fast Fourier Transform (FFT) to extract the peak points of each frequency and finally combines these peak points to obtain unique audio fingerprint information. This information, along with copyright owner information, is recorded on the blockchain. Compared to other blockchain techniques, an effort is made in the robustness aspect of the audio fingerprint, however, there is no exploration of combining or remixing two or more audios. Instead, the focus is on scalability, emphasizing stress tests that overwhelm the blockchain rather than attacking the audio Experimental results show MBE outperforms traditional systems in efficiency, storage demand, and security. Compared to a traditional Fabric system without edge computing, MBE has 53% higher deposition efficiency and requires 48% less storage space. The scalability and evolution from centralised systems is commendable, but the solution overlooks the critical aspects of robustness and avoiding 'leaks.' This is a major reason why our research is built from watermarking wherein robustness is a priority, rather than assuming the audio content is static and manipulations will not be made on it.

2.3.2 Cloud-Based Hashing

Mobile and Cloud Integrity Preserving System
(Renza, Arango, and Ballesteros 2019) propose a system for preserving the integrity of audio recordings using a mobile application and cloud storage. The primary use case is in the collection of voice recording evidence, where it has to ensure the said recordings have not been tampered with. The system uses a mobile application to record audio and then immediately calculates a unique hash code, or digital fingerprint, for that recording on the device; this calculation involves using a Discrete Fourier Transform (DFT) to convert the audio signal from its original time-domain representation to a frequency-domain representation, which highlights the signal's spectral content. The resulting frequency data is used to generate the 64-hex-digit hash code. This hash code, alongside relevant metadata such as file name and creation date, is then transmitted to a secure cloud database for storage. The verification process is simply comparing the recording's hash code to that stored in the cloud. It is not designed to withstand manipulation or any alteration of the audio itself, with the focus being to simply verify originality through the chain of custody. Therefore, the evaluation of the system focused on the hashes' collision resistance, the computational cost, and the sensitivity to 'noticing' change. No collisions were detected in 9730 tests comparing pairs of hash codes, and the lowest Hamming distance (HD) observed was 0.4, indicating that even very similar recordings had at least 40% differing coordinates in their hash codes. For a single sample modification, the lowest HD was 0.1, showing high sensitivity to input signal changes. This high collision resistance and sensitivity to small changes in the audio make it a reliable tool for verifying the integrity. The computational cost is also minimal, requiring less than 2 seconds per minute of recording to calculate the hash code. Hence, the system can be considered to be highly accurate and fairly scalable, in adversarial free environments. However, in practical context the system would fail dismally as the hashing process can be easily manipulated. Furthermore, the mobile-cloud relationship introduces complexity and vulnerability, making the implementation unjustifiable security-wise.

2.4 Identification of gaps

The previous work reveals significant gaps in audio tracing and watermarking technology that warrant further investigation. There is a crucial shortcoming in tracking audio content through multiple transformations and distributions as current solutions predominantly focus on single-layer watermarking. Systems like AudioSeal (Roman et al. 2024) and WavMark (Chen et al. 2024) demonstrate impressive advances in robustness and imperceptibility, but they are susceptible to losing traceability when audio undergoes multiple modifications or enters complex distribution chains. On the other hand, the integration of blockchain technology, while promising for ownership verification, introduces complexity and computational overhead that could be avoided with more sophisticated watermarking approaches. Besides this, watermarking techniques that have coupled with blockchain overlook robustness, especially to ongoing advancement in adversarial attacks. On the whole, the field lacks a comprehensive solution for dynamically layering distinct watermarks in a single audio - a capability that exists in image watermarking (Begum and Uddin 2021) but remains underdeveloped for audio. The limitation is particularly evident in scenarios requiring cumulative tracking across reworks, production pipelines or social media sharing.

The emerging challenges of voice cloning and advanced audio manipulation techniques further expose the limitations of existing frameworks. While some systems like Timbre (C. Liu et al. 2023) watermarking show promise in frequency domain layering, they are not optimized for layering distinct watermarks. Their objective is normally to combat specific attacks, but they often achieve this at the cost of reduced capacity or increased complexity. The literature also indicates multi-layering is inhibited by a critical gap in optimizing frequency band selection. Other work suggests that statistical approaches like PCA and its variants could be used to this effect because of their data compression and feature extraction capabilities. Integrating a PCA layer into the training process, similar to how Timbre incorporated a distortion training layer, has the potential to improve watermark separation and traceability. Overall, these limitations, along with the corresponding opportunities, highlight a clear need for research into more sophisticated, multi-level watermarking solutions that can maintain effectiveness throughout the entire audio lifecycle. This would be a timely contribution, addressing a demand for autonomous tracking capabilities in an era where advanced manipulations such as AI cloning, and unauthorised reuse of original content are prevalent.

2.5 Conclusion

Audio watermarking is at a juncture where advances in static techniques have not fully addressed the complexities of modern content distribution. While deep learning models like AudioSeal and WavMark demonstrate robust single-watermark capabilities, the field lacks solutions for maintaining provenance through multiple transformations and ownership transfers. There have been parallel developments in robustness and traceability as blockchain-based tracking and sophisticated watermarking techniques have a disconnect. While blockchain solutions offer transparent ownership records, they operate largely in isolation from the sophisticated signal processing techniques that make watermarks resilient to manipulation. Frequency domain watermarking offers a promising foundation for addressing these limitations due to its potential for layered embedding. However, current implementations are constrained by limited band selection strategies that could enable multiple distinct watermarks to coexist without interference. The potential application of Principal Component Analysis for optimizing frequency band allocation within distinct watermarks represents an unexplored opportunity to bridge this gap, potentially enabling dynamic watermarking systems that can adapt to complex distribution chains while maintaining both robustness and traceability. As the audio ecosystem evolves, developing such integrated solutions becomes increasingly crucial for addressing the challenges of modern content tracking and verification.

Chapter 3

Methodology

3.1 Introduction

Using the elements of the preceding work mentioned in the previous chapter (Frequency Domain Watermarking, Principal Component Analysis and Feature Extraction Detection), this chapter outlines the working principle of the system, with a high level description of the algorithm, as well as the empirical approach taken to compare the artifact's perfomance with existing watermarking algorithms and traceability systems. A mathematical approach is taken to describe how we fundamentally convert an audio signal $x[n]$ to a watermarked audio $x_{\text{watermarked}}[n]$, and subsequently extract the watermarks w'_n. The chapter also mentions training pipelines, hypotheses and tools used - essentially providing fundamental information on how to reproduce the research.

3.2 Algorithm Overview

The system is centered around the dynamic 'negotiation' and allocation of embedding space for the distinct frequency coefficient changes between a generator (encoder + decoder) and detector. To contextualise how this encoder and detector, the researcher first refers to AudioSeal whose hybrid architecture involves a convolutional blocks + LSTM (Long Short-Term Memory) encoder and a mirrored detector that also has a transpose convolution. The below diagram illustrates the additions made to the existing algorithm.

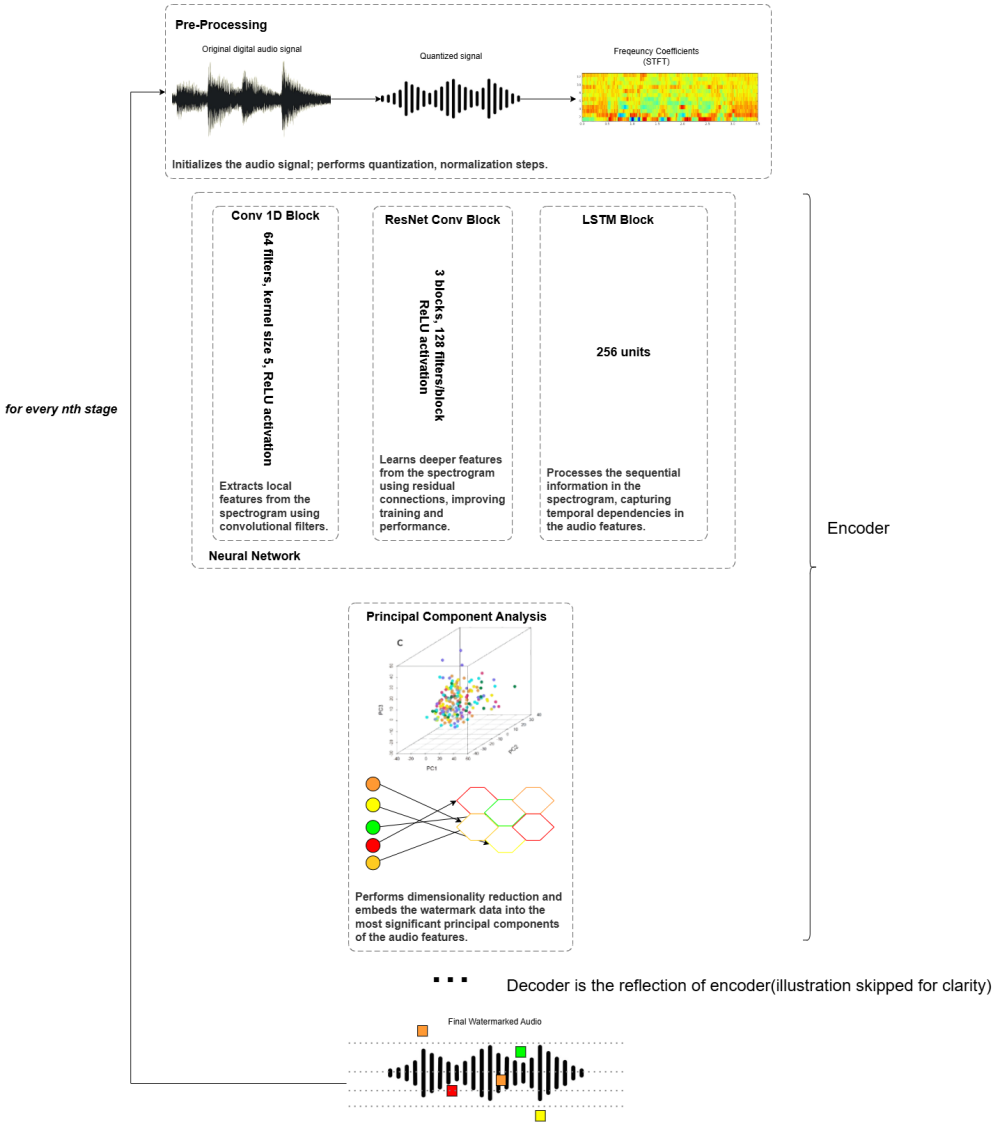


Figure 3.1: Watermarking Generator architecture with added PCA layer

The detailed implementation of the PFBGenerator and PFBDetector architectures, including the incremental embedding and detection logic, can be found in Appendix **??**.

3.2.1 Generator:

The Generator module embeds a watermark into an audio signal using a deep learning-based feature extraction approach followed by Principal Component Analysis (PCA) for dimensionality reduction and watermark embedding in the transformed feature space. The process involves pre-processing, spectrogram generation, feature extraction, PCA, watermark embedding, and reconstruction of the watermarked audio.

1. Pre-processing
The input audio is normalized and windowed to prepare it for processing.
Input: Discrete-time audio signal $x[n]$, where n represents the sample index.
Process:
i. Normalization to unit variance:

$$x'[n] = \frac{x[n]}{\text{std}(x[n])}$$

where std$(x[n])$ is the standard deviation of the signal. This results in a signal x'[n] of length L.

ii. Windowing: Apply a window function $w[n]$ (reduces spectral leakage from the STFT):

$$x_w[n] = x'[n] \cdot w[n]$$

This element-wise multiplication results in a signal xw[n] of length L.

2. Spectrogram Generation (STFT)
The audio is transformed into a time-frequency representation using the Short-Time Fourier Transform (STFT).
Input: Normalized and windowed signal $x_w[n]$.
Process:
Short-Time Fourier Transform (STFT) is performed to obtain a time-frequency representation:

$$X(m,k) = \sum_{n=0}^{N-1} x_w[n] \cdot e^{-j\frac{2\pi kn}{N}}$$

where m is the time frame index, k is the frequency bin index, and N is the window length.
Output: Magnitude spectrogram $|X|$:

$$|X(m,k)| = \sqrt{\text{Re}(X(m,k))^2 + \text{Im}(X(m,k))^2}$$

where m is the time frame index and k is the frequency bin index.

3. Feature Extraction (Neural Network Layers)
A neural network extracts relevant features from the spectrogram.
Input: Magnitude spectrogram $|X|$, with dimensions M x N, where M is the number of time frames and N is the number of frequency bins.
Process:
A combination of Convolutional Neural Network (CNN), ResNet, and Long Short-Term Memory (LSTM) layers extracts features:
- Conv 1D Block: Extracts local features from the spectrogram using convolutional filters.
- ResNet Conv Block: Learns deeper features from the spectrogram using residual connections, improving training and performance.
- LSTM Block: Processes the sequential information in the spectrogram, capturing temporal dependencies in the audio features.

$$F = F(|X|)$$

where $F(|X|)$ is the feature matrix output by the network. The specific operations (convolutions, nonlinearities, etc.) are abstracted into F.
Output: Feature matrix F, representing learned features from the spectrogram.

4. Principal Component Analysis (PCA) Layer
PCA reduces the dimensionality of the extracted features.
Input: Feature matrix F.
Process:
1. Covariance Matrix:

$$C = \frac{1}{N-1}(F - \mu_F)(F - \mu_F)^T$$

where μ_F is the mean vector of F. The dimensions of the covariance matrix C are D x D, where D is the number of features in each feature vector.
2. Eigenvalue Decomposition:

$$C = V\Lambda V^T$$

V contains eigenvectors (principal components), Λ is the diagonal matrix of eigenvalues.
3. Dimensionality Reduction: Select the top k eigenvectors corresponding to the largest eigenvalues:

$$V_k = [v_1, v_2, \ldots, v_k]$$

4. Projection: Project F onto the reduced-dimensional subspace:

$$F_{\text{reduced}} = FV_k$$

5. Watermark Embedding
The watermark is added to the reduced features.
Input: Reduced feature matrix F_{reduced}, watermark vector w.
Process:
Embed the watermark w by adding it to the reduced features:

$$F_{\text{watermarked}} = F_{\text{reduced}} + \alpha w$$

where α is a scaling factor controlling the watermark strength.

6. Reconstructing the Watermarked Audio
The modified features are transformed back into a watermarked audio signal.
Input: Watermarked features $F_{\text{watermarked}}$.
Process:
1. Inverse Feature Mapping: Apply the inverse neural network transformation to map features back to the spectrogram domain:

$$X_{\text{watermarked}} = F^{-1}(F_{\text{watermarked}})$$

2. Inverse STFT: Convert the modified spectrogram back to the time-domain audio signal:

$$x_{\text{watermarked}}[n] = \text{ISTFT}(X_{\text{watermarked}})$$

3.2.2 Detector:

A similar architecture and technique is employed for watermark detection wherein the goal would br to detect, distinguish and extract the multiple distinct watermarks from a watermarked audio signal, if present. The detector is also empririically important for assessing the integrity the watermark.

1. Pre-processing (Same as Generator)
The watermarked audio undergoes the same pre-processing steps as the generator.
Input: Watermarked audio signal $x_{\text{watermarked}}[n]$.
Process:
1. Normalization:

$$x'_{\text{watermarked}}[n] = \frac{x_{\text{watermarked}}[n]}{\text{std}(x_{\text{watermarked}}[n])}$$

2. Windowing:

$$x_w[n] = x'_{\text{watermarked}}[n] \cdot w[n]$$

The same window function $w[n]$ is applied, as used in the generator.

2. Spectrogram Generation (STFT)
As in the generator, the watermarked audio is transformed into a time-frequency representation using STFT.
Input: Normalized and windowed signal $x_w[n]$.
Process:
Compute the STFT to obtain the time-frequency representation from the watermarked audio signal:

$$X_{\text{watermarked}}(m,k) = \sum_{n=0}^{N-1} x_w[n] e^{-j\frac{2\pi kn}{N}}$$

Output: Magnitude spectrogram $|X_{\text{watermarked}}|$:

$$|X_{\text{watermarked}}(m,k)| = \sqrt{\text{Re}(X_{\text{watermarked}}(m,k))^2 + \text{Im}(X_{\text{watermarked}}(m,k))^2}$$

3. Feature Extraction (Inverse Neural Network)
The same neural network used in the generator extracts features from the spectrogram.
Input: Magnitude spectrogram $|X_{\text{watermarked}}|$.
Process:
The detector will again apply the same feature extraction layers F (CNN, ResNet, LSTM) used in the generator:

$$F_{\text{watermarked}} = F(|X_{\text{watermarked}}|)$$

This generates the feature matrix $F_{\text{watermarked}}$, which should closely resemble the feature matrix $F_{\text{reduced}}+\alpha w$ from the generator.

4. PCA Reconstruction (Reverse of Generator PCA)
Crucially, PCA reverses the dimensionality reduction in an attempt to recover the original features.
Input: Extracted feature matrix $F_{\text{watermarked}}$.
Process:
1. Projection onto Principal Components:

$$F'_{\text{reduced}} = F_{\text{watermarked}} V_k$$

This reverses the dimensionality reduction step performed during watermark generation.
2. Watermark Extraction: Recover the embedded watermark w' by isolating the difference between the watermarked and reduced feature matrices:

$$w' = \frac{F'_{\text{reduced}} - F_{\text{reduced}}}{\alpha}$$

(Note: F_{reduced} is an approximation reconstructed by the inverse process.)

5. Detecting Multiple Distinct Watermarks
To detect multiple watermarks, the detector uses correlation techniques similar to those applied in (Roman et al. 2024).
Process:
1. Cross-Correlation for Watermark Detection: For each possible watermark w_i (distinct watermarks):

$$\rho_i = \text{corr}(w', w_i) = \frac{\sum(w' - \mu_{w'})(w_i - \mu_{w_i})}{\sigma_{w'}\sigma_{w_i}}$$

where $\mu_{w'}$ and μ_{w_i} are the means, and $\sigma_{w'}$ and σ_{w_i} are the standard deviations.
2. Threshold-Based Detection: If $\rho_i >$ threshold, then watermark w_i is detected. The threshold depends on the desired false positive rate.
3. Handling Multiple Watermarks: Repeat the correlation process for all w_i watermarks and determine the distinct watermarks present in the signal.

6. Verification and Integrity Check
The extracted watermark is compared to the original to verify its integrity and resilience.
To verify the integrity of the watermark:
1. Strength of Detected Watermark: Compute the energy of the extracted watermark w':

$$E_{w'} = \sum |w'[n]|^2$$

Compare this energy with the expected energy E_w of the original watermark.
2. Error Check: Measure the error e between w' and the closest detected watermark w_i:

$$e = \|w' - w_i\|_2$$

where $\|\cdot\|_2$ is the Euclidean norm. If e is within a tolerance, the watermark is considered valid.

In summary, the generator embeds the watermark by modifying the feature space, while the detector extracts and verifies it using STFT, PCA reconstruction, and correlation methods. The enhanced detector's ability to handle multiple distinct watermarks relies on calculating correlations with known watermark vectors and evaluating their presence and integrity based on defined SNR thresholds.

3.3 Training Pipeline

The pipeline will consist of two interconnected models, the Watermark Generator Model and the Watermark Detector Model.
Both models will be trained in a supervised manner ensuring the generator embeds imperceptible and robust watermarks, and that the detector reliably extracts and distinguishes multiple watermarks, even under signal degradation (e.g., noise, compression).

1. Data Preparation
Audio Data (Training Set): $x_{\text{clean}}[n]$
- VoxPopuli by Facebook Research
Watermarks:
- Random Watermark Vectors: w_i generated as binary vectors. (e.g., 128 or 256 bits each).
Training Labels:
- Pairs (x_{clean}, w), where w is the watermark to be embedded.

2. Watermark Generator Training
The generator embeds the watermark into the audio signal.
Loss Function for Generator: The generator loss balances two objectives:

1. Perceptual Loss (to ensure minimal distortion):

$$\mathcal{L}_{\text{perceptual}} = \|x_{\text{clean}} - x_{\text{watermarked}}\|_2^2$$

2. Embedding Loss (to ensure successful embedding):

$$\mathcal{L}_{\text{embed}} = \|F_{\text{reduced}} - (F_{\text{reduced}} + \alpha w)\|_2^2$$

Total Generator Loss:

$$\mathcal{L}_{\text{generator}} = \lambda_1 \mathcal{L}_{\text{perceptual}} + \lambda_2 \mathcal{L}_{\text{embed}}$$

where λ_1 and λ_2 are hyperparameters controlling the trade-off.

3. Watermark Detector (Decoder) Training
The detector is trained to extract the watermark from the watermarked signal.
Loss Function for Detector: The detector's loss focuses on accurate extraction:
1. Extraction Loss (L2 loss between original and detected watermark):

$$\mathcal{L}_{\text{extract}} = \|w - w'\|_2^2$$

2. Detection Accuracy Loss (cross-entropy for multi-watermark detection):

$$\mathcal{L}_{\text{detect}} = -\sum_i [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

where p_i is the probability the detector assigns to detecting watermark w_i, and y_i is the true label (1 if w_i is present, 0 otherwise).
Total Detector Loss:

$$\mathcal{L}_{\text{detector}} = \lambda_3 \mathcal{L}_{\text{extract}} + \lambda_4 \mathcal{L}_{\text{detect}}$$

where λ_3 and λ_4 are hyperparameters.

4. Combined Training Strategy
Adversarial Training Loop: The generator and detector are trained together in an adversarial framework, as in (Roman et al. 2024), (H. Liu et al. 2024) and (Chen et al. 2024):
1. Generator Forward Pass
- Input clean audio $x_{\text{clean}}[n]$ and watermark w.
- Output watermarked audio $x_{\text{watermarked}}[n]$.
2. Detector Forward Pass
- Input $x_{\text{watermarked}}[n]$.

- Extracted watermark w' and probabilities for multiple watermarks.
3. Loss Calculation
- Compute generator and detector losses simultaneously.
4. Optimization Steps
- Update Generator: Minimize $\mathcal{L}_{\text{generator}}$.
- Update Detector: Minimize $\mathcal{L}_{\text{detector}}$.

5. Training Algorithm (Pseudocode)

Algorithm 1: Training Pipeline for Distinctive FD Audio Watermarking using PCA

Data: Audio dataset D, Watermark set W, Number of epochs N, Scaling factor α
Result: Trained generator and detector models

Initialize generator G and detector D;
Initialize optimizers O_G and O_D;
for *epoch* $e = 1$ *to* N **do**
  foreach *batch* $(x, w) \in (D, W)$ **do**
    Generator Step:
    Generate watermarked audio: $x_{\text{watermarked}} = G(x, w)$;
    Compute perceptual loss: $\mathcal{L}_{\text{perceptual}} = \|x - x_{\text{watermarked}}\|^2$;
    Compute embedding loss: $\mathcal{L}_{\text{embed}} = \|w - D(x_{\text{watermarked}})\|^2$;
    Total generator loss: $\mathcal{L}_G = \lambda_1 \mathcal{L}_{\text{perceptual}} + \lambda_2 \mathcal{L}_{\text{embed}}$;
    Update G using O_G;
    Detector Step:
    Detect watermark: $w' = D(x_{\text{watermarked}})$;
    Compute extraction loss: $\mathcal{L}_{\text{extract}} = \|w - w'\|^2$;
    Compute classification loss: $\mathcal{L}_{\text{class}} = \text{CrossEntropy}(w, w')$;
    Total detector loss: $\mathcal{L}_D = \lambda_3 \mathcal{L}_{\text{extract}} + \lambda_4 \mathcal{L}_{\text{class}}$;
    Update D using O_D;
  end
end

The training pipeline ensures both the generator and detector models work in harmony, using an adversarial approach to embed the watermarks. The detector is optimized to extract multiple distinct watermarks, providing the needed resilience against audio degradation over lifecycles.

3.4 Dataset

The VoxPopuli (Wang et al. 2021) dataset, developed by Facebook AI Research hosted on Hugging Face and , will be employed for evaluating the proposed audio watermarking system. This dataset comprises thousands of hours of multilingual speech collected from European Parliament sessions, and happens to be the same dataset used to train AudioSeal and WavMark.
VoxPopuli contains speech transcribed in 18 languages, delivered by various speakers in different accents, tones, and styles. This diversity is crucial for evaluating the ability of the watermarking algorithm to generalise across varied audio characteristics.
Furthermore, it contains professionally recorded audio that mirrors real-world use cases, particularly being adjacent to podcasts, news broadcasts, and speeches. This allows for testing the watermarking algorithm in realistic scenarios where audio quality matters.
With over 1,800 hours of data across multiple languages, the dataset provides ample audio samples for robust training, testing, and validation of the watermarking system. Additionally, its availability on Hugging Face ensures ease of access and integration into machine learning pipelines.

Usage in the Project:
1. Training Phase:
The dataset will be used to train the neural network, with subset of the data being reserved for training the embedding module to learn how to insert the n^{th} watermark seamlessly into the audio signal.
2. Testing and Evaluation:
Another portion of the dataset will be used to test the resilience of the watermark against distortions.

The watermarked audio will be subjected to transformations such as compression, noise addition, and filtering using Distortion Audibility Tester to assess watermark detectability under adverse conditions.

Ethical and Licensing Considerations
VoxPopuli is provided under the CC0 1.0 Universal (Public Domain Dedication) license, making it freely available for any use, including modification and redistribution, without requiring attribution. The use of this dataset adheres to ethical standards, as it is explicitly made available for public use, supporting research and development projects such as this one.
The use of the VoxPopuli dataset contributes significantly to the project's success by ensuring: comprehensive testing due to the diversity, robustness and scalability because of its size, and reproducibility of results as it is publicly available.

3.5 Evaluation of Hypotheses

3.5.1 Hypotheses

• H1: Watermarking, atleast up to a certain number of increments n, provides a better average on robustness (detectability) and imperceptibility, compared to other traceability frameworks (i.e Blockchain and cloud- hash based) (Xiao et al. 2023) (Renza, Arango, and Ballesteros 2019) under adversarial conditions.

At a system level, this hypothesis states that the proposed watermarking technique outperforms blockchain and cloud-hash-based traceability techniques in terms of robustness, and imperceptibility. Adversarial conditions will include different types of audio distortions, mutual as additive white Gaussian noise (AWGN) at multiple signal-to-noise ratios (SNRs), MP3 compression at different bit-rates, resampling. The value *'n'* refers to the number of stages a watermark can be successfully extracted before significant degradation occurs. The robustness is measured in terms of Bit Error Rate (BER), and the imperceptibility is assessed using Perceptual Evaluation of Speech Quality (PESQ) and Signal-to-Noise Ratio (SNR).

• H2: PCA as a frequency embedding technique in audio watermarking, outperforms other 'simple' techniques (i.e spread spectrum, LSB, SVD) on robustness and imperceptibility scores as audio is subjected to more watermarks.

At the algorithm level, this hypothesis compares the performance of PCA-based watermark embedding algorithm with spread spectrum, least significant bit (LSB) modification, and singular value decomposition (SVD) techniques. 'Simple techniques' refer to established methods described in 2.2.2, before Timbre watermarking, that equally spread the watermarks on the frequency spectrum of the audio without additional computing or explicit positioning. The evaluation involves embedding multiple watermarks into the same audio signal and assessing the robustness and imperceptibility of each technique as the number of watermarks increases. The same metrics (BER, PESQ, SNR) are used to evaluate performance.

3.5.2 Adversarial Attacks

In determining the validity of the hypotheses and carrying out *Objective 4*, a standard audio distortion simulation application shall be used, for simplicity and consistency (distortaudio.org). Distortion Audibility Tester can be used to perform common distortions such as compression, resampling, adding noise, and filtering on watermarked audio. By applying these manipulations to the watermarked audio, an attempt is made to weaken the watermark robustness and diminish audio quality. Whilst works such as AudioMarkBench (H. Liu et al. 2024) and (Uddin et al. 2024) that evaluate watermarking algorithms and their robustness against adversarial attacks do not explicitly mention the tools they use to alter audio signals, the alterations performed are largely the same as those the researcher will implement using the application. (H. Liu et al. 2024) leans towars watermark-removal perturbations whilst (Uddin et al. 2024) focuses on forgery pertubations (falsely marking unwatermarked audio as watermarked). The specific alterations performed depend on the algorithm being tested and what it was crafted to uphold, ranging from black box pertubations such as HopSkipJumpAttack and Square Attack that work to 'break' detectors, to white box pertubations that target the watermark decoder's internal mechanisms. This evaluation focuses on no-box pertubations, which are general audio editing operations. In this context, the following specific pertubations are performed to simulate the attack classes mentioned in 1.6:

- Remixing and Mashups: aggressive time stretching and pitch shifting, layering the audio, phase manipulation.
These are fitting as remixing primarily alters an audio's spectral characteristics, which would potentially obscure the watermark.

- Remakes and Cover Versions: simply re-recording the audio will create a new audio signal which is some cases may have dissimilar spectral characteristics from the original, effects processing (equlization, reverb), echo
Basically, the goal is to reinterpret the orginal audio, which would typically alter the acoustic fingerprint, including that of the watermark.

- Chain of Custody Leaks: transcoding (format conversion e.g from wav to mp3), targeted noise injection
The tests are against the watermark integrity which can be diminished with conversions.

- Music Production Pipelines: compression, filtering
Targeting an audio's frequency pattern, these are simple yet detrimental attacks - most likely to be introduced in production pipelines and similarly popular in cloning environments - that can distort or remove watermarks completely.

The results from the simulations will be altered audio files that we will then attempt to re-attach to the cycle to add more watermarks that will be subsequently detected. The next section describes metric evaluations that will be carries out on these resultant audios.

3.5.3 Evaluation Metrics

Similarly derived from AudioMarkBench (H. Liu et al. 2024) and (Uddin et al. 2024) 's comprehensive evaluation, the chosen evaluation metrics collectively ensure that the audio watermarking system achieves its core objectives: embedding robust watermarks without compromising audio quality. They provide a comprehensive framework for assessing both technical robustness (BER, AUC) and perceptual quality (SNR, PESQ), ensuring that the system adheres to the the standards set by preceding audio watermarking techniques.

• Bit Error Rate (BER):
Measures the accuracy of watermark extraction after audio transformations.

$$\text{BER} = \frac{\text{Number of bit errors}}{\text{Total number of bits}}$$

A lower BER indicates fewer errors, and hence better performance, this key metric will account for the robustness of the watermarks - how resilient they remain to introduced distortions.

• Area Under Curve (AUC):
Ability to classify watermarked and non-watermarked audio, derived from the Receiver Operating Characteristic (ROC) curve.

$$\text{AUC} = \int_0^1 \text{TPR}(FPR)\, d(\text{FPR})$$

where TPR is the true positive rate and FPR is the false positive rate.
A high AUC score indicates the system can reliably classify whether audio contains a watermark, even in noisy or compressed environments, it will account for the detection sensitivity and specificity (avoiding false positives) of the system

• Signal-to-Noise Ratio (SNR)
Assesses audio quality preservation by comparing original and watermarked signals.

$$\text{SNR} = 10 \log_{10} \left(\frac{P_{\text{signal}}}{P_{\text{noise}}} \right) \text{ dB}$$

Higher SNR values indicate that the audio quality is well-preserved after watermark embedding, it ensures that embedding the watermark does not degrade the audio quality significantly.

• Mean Square Error (MSE)
Quantifies the distortion introduced during watermark embedding.

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{x}_i)^2$$

where x_i is the original signal and $\hat{x}_i$ is the watermarked signal. MSE provides a direct measure of how much distortion the embedding process introduces, serving as feedback for the system to minimizes distortion while embedding the watermark.

• Perceptual Evaluation of Speech Quality (PESQ)
Measures perceived audio quality.
- PESQ scores range from 1 (poor) to 5 (excellent).
In the evaluation, it accounts for psychoacoustic properties, ensuring audio remains natural-sounding to human ears.

3.6 Tools for Implementation

- **Programming Language: Python** It is the most popular option in machine learning projects due to its library ecosystem and simplicity, this adds a significant advantage for this particular research because of its rapid prototyping and experimentation approach, especially when fine-tuning the generator and detector models.
- **Frameworks and Libraries: PyTorch, Torchaudio, Librosa, NumPy and SciPy**
  - **PyTorch:** This machine learning framework offers dynamic computation graphs, which makes debugging models easier. It also has streamlined computation and model optimization features, proving essential for training the deep learning-based generator and detector models.
  - **Torchaudio:** This library is specifically designed for audio analysis and manipulation within the PyTorch ecosystem. It facilitates audio preprocessing, such as spectrogram generation (e.g., Short-Time Fourier Transforms) and reconstruction, which is essential for embedding and detecting watermarks in the frequency domain.
  - **Librosa:** This popular library for audio analysis and manipulation in Python will complement Torchaudio, providing additional functionalities for audio preprocessing, such as various feature extractions, crucial for the in-depth analysis of audio signals.
  - **NumPy and SciPy:** These foundational libraries provide fast, efficient numerical computations and help in data manipulation and signal processing tasks, such as calculating perceptual loss, Signal-to-Noise Ratio (SNR), Mean Squared Error (MSE), and conducting various audio transformations required for evaluation.
- **Development Environment and Tools:**
  - **Google Colab:** Google Colab is extensively utilized in model training, as its free access to GPUs significantly accelerates the computationally intensive model training, allowing for rapid iteration and refinement of the neural network architectures.
  - **Jupyter Notebooks:** Jupyter Notebooks provides an interactive environment for developing and testing code. It is flexible for iterative development, quick testing, and visual output, making debugging and experimentation easier, particularly during the model development phases.
  - **Flask:** Flask, as a lightweight Python web framework, is be utilized for developing the backend of the Audio Watermark Lab application. It facilitates the creation of a RESTful API for handling audio file uploads, orchestrating watermarking and detection processes, and serving processed audio.
  - **React:** React is employed for the development of the frontend user interface of the Audio Watermark Lab. Its component-based architecture enables the creation of an interactive and responsive web application, providing distinct user interfaces for research and application modes.
  - **Git and GitHub:** As the most popular version control system, Git ensures model versions and experiment results and both be trackable and reproducible. GitHub is be used for hosting the project's repository, enabling version control and potential collaboration.

3.7 Conclusion

This chapter introduces the novel approach to multi-level audio watermarking that reimagines audio content traceability in the age of rapid digital transformation. The approach handles classical watermarking triad of robustness, imperceptibility, and capacity by integrating of Principal Component Analysis with modified neural architectures from AudioSeal, looking beyond the traditional static embedding approaches constraining frequency watermarking systems. The mathematical framework provides a foundation for reproducible research, while the training pipeline along the detailed resources demonstrate its practical feasibility. Ultimately, the methodologies and processes described set the stage for further exploration into the potential enhancements of watermarking algorithms, and the intersection of the featured techniques to the broader fields of signal processing and digital content security.

Chapter 4

System Analysis and Design

4.1 Introduction

This chapter transitions from the theoretical underpinnings and methodological approaches—which culminated in the development of novel machine learning models for audio watermarking—to the concrete analysis and design of a system for their practical deployment and user interaction. The "Dynamic Audio Watermarking Algorithm using PCA and Distinctive Detection" is operationalized through the **Audio Watermark Lab**, an application specifically designed to allow users to access and utilize the trained `PFBGenerator` and `PFBDetector` models. This chapter details the process of analyzing the requirements for deploying these ML models for both research experimentation and end-user application, and outlines the architectural and component-level design decisions. It provides the blueprint for integrating the core watermarking algorithms, developed through iterative training and evaluation (Experiments L through U), into a functional web application, thereby making the learned models accessible and usable.

4.2 Detailed analysis of the problem domain and user requirements

The problem domain, as established in preceding chapters, revolves around the increasing need for robust, multi-level traceability of digital audio content in an era of widespread reuse, remixing, and potential manipulation. Existing solutions often fall short in providing a seamless, integrated system that balances imperceptibility, robustness against diverse attacks, and the capacity to trace an audio's provenance through multiple stages without resorting to complex third-party systems like blockchain or cloud hashing, which themselves can introduce vulnerabilities or overhead.

The core challenge is to develop an audio watermarking system that can:

- Embed multiple, distinct layers of information (watermarks) incrementally.
- Ensure these watermarks are difficult to remove or tamper with (robustness).
- Keep the embedded information inaudible or minimally perceptible (imperceptibility).
- Allow for the reliable extraction and interpretation of these watermarks to reconstruct an audio's lifecycle or chain of custody.
- Utilize advanced techniques like Principal Component Analysis (PCA) to enhance watermark distinctiveness and manage embedding within the frequency domain.

To address these challenges, the Audio Watermark Lab was conceived with two primary user groups in mind:

- **Researchers:** Requiring a flexible platform for experimenting with advanced watermarking configurations, comparing different algorithmic approaches (including the novel PCA-integrated methods), and analyzing performance metrics in detail.

- **Application Users (e.g., content creators, rights holders, forensic analysts):** Needing a more streamlined, user-friendly interface for embedding identifying information into their audio assets and subsequently detecting/tracing this information, often with an emphasis on ease of use and clear reporting of an audio's history.

The methods used to gather and crystallize these requirements included:

- **Literature Review:** Identifying gaps and desiderata from existing research in audio watermarking and traceability (as detailed in Chapter 2).
- **Algorithmic Experimentation:** The iterative development of the PFB STFT-modification system (Experiments L through U) provided insights into the practical challenges of balancing BER, SNR, and robustness, informing the design of the user-configurable parameters.
- **Conceptual Design (from the thesis proposal):** The vision for dynamic, PCA-based multi-level watermarking heavily guided the architectural considerations for both the underlying algorithms and the application designed to host them.

4.2.1 Functional Requirements

Based on the problem domain and target users, the following functional requirements were established for the Audio Watermark Lab system:

- **FR1: Dual-Mode Operation for ML Model Interaction:**
  - **FR1.1: Research Mode:** Provide an interface for advanced configuration of the deployed **watermarking models (`PFBGenerator`/`PFBDetector`)**, allowing researchers to experiment with different parameters (message bits, PCA configurations if applicable, target bands, strength factors) that affect the ML models' behavior and to analyze their performance via detailed metrics. Publicly accessible without authentication.
  - **FR1.2: Application Mode:** Offer a simplified interface for end-users to easily utilize the pre-trained and optimized **watermarking models** for embedding and detecting watermarks without needing to understand the underlying ML complexities.
- **FR2: Audio File Handling:**
  - **FR2.1:** Allow users to upload audio files in common formats (e.g., WAV, MP3, FLAC).
  - **FR2.2:** Provide an interface for downloading processed/watermarked audio files.
- **FR3: Watermark Embedding (Leveraging the Deployed `PFBGenerator` Model):**
  - **FR3.1:** Allow users to select and apply various watermarking methods, where methods like "PFB with Learnable Basis" or "PCA-PFB" directly utilize the trained **`PFBGenerator` model**.
  - **FR3.2:** Allow users to specify the watermark message (e.g., as a binary string or derived from user/purpose metadata).
  - **FR3.3:** (Research Mode) Allow configuration of watermark parameters such as number of watermarks, message bits per watermark, PCA enablement, and number of PCA components.
  - **FR3.4:** (Application Mode) Allow specification of "Purpose" for the watermark, which can be embedded as part of the payload.
  - **FR3.5:** The system must utilize the **`PFBGenerator` model** to embed single or multiple sequential watermarks based on user configuration.
- **FR4: Watermark Detection and Tracing (Leveraging the Deployed `PFBDetector` Model):**
  - **FR4.1:** Allow users to upload an audio file for watermark detection and tracing.
  - **FR4.2:** The system must use the trained **`PFBDetector` model** corresponding to the embedding method to attempt to detect and decode embedded watermarks.
  - **FR4.3:** Display watermark information decoded by the **`PFBDetector` model**, including payload and potentially metadata like user ID, action, sequence.
- **FR5: Performance Metrics and Analysis (of the ML Models):**

  - **FR5.1:** Calculate and display key performance metrics (SNR, BER, Detection Probability) reflecting the efficacy of the deployed **watermarking models** under various conditions.
  - **FR5.2:** (Research Mode) Provide visualizations for watermark performance.
  - **FR5.3:** (Research Mode) Allow for comparative analysis of different watermarking methods.
- **FR6: User Account Management (Application Mode):**
  - **FR6.1:** Provide user registration and login functionality.
  - **FR6.2:** Associate watermarking operations with the authenticated user.
- **FR7: Operation History (Application Mode):**
  - **FR7.1:** Maintain and display a ledger or history of a user's embedding and detection operations.
  - **FR7.2:** Store relevant metadata for each operation (timestamp, user, purpose, payload, audio file links).

4.2.2 Non-functional Requirements

- **NFR1: Usability:** The system must be intuitive and easy to use for both target user groups, with clear distinctions and workflows for Research and Application modes.
- **NFR2: Performance (System):**
  - **NFR2.1:** Audio processing (embedding/detection), including inference with the deployed PyTorch models, should provide feedback to the user in a timely manner.
  - **NFR2.2:** The backend API must efficiently handle audio data and ML model computations.
- **NFR3: Performance (Algorithm - ML Models):**
  - **NFR3.1: Imperceptibility:** Embedded watermarks should strive for high SNR values (target ≈25-30dB).
  - **NFR3.2: Robustness:** Watermarks should achieve low BER (target ¡ 0.1, ideally ¡ 0.05) after common audio manipulations.
- **NFR4: Reliability:** The system should provide consistent watermarking and detection results. False positive rates for detection should be minimized.
- **NFR5: Scalability (System):** The Flask backend and SQLite database are suitable for the envisioned lab/research and small-scale application use.
- **NFR6: Security (Application):** Basic security measures for file uploads and user data (mock authentication) are implemented.
- **NFR7: Data Integrity & Persistence:** Watermark operation details and user data must be reliably stored and retrieved using the SQLite database.
- **NFR8: Maintainability (Code):** The modular design of frontend (React) and backend (Flask), with separation of ML model logic, aids maintainability.
- **NFR9: Offline Capability:** The application architecture supports local data storage (SQLite) and local audio processing via the Flask backend, enabling offline functionality.

4.3 Identification of system components and functionalities

The Audio Watermark Lab is architected as a client-server application, breaking down into the following key components:

1. **Frontend (Client-Side Interface):**
   - **Technology:** React, TypeScript, Tailwind CSS, Shadcn UI components.

- **Functionalities:** Renders UI for Research and Application modes, handles user inputs, communicates with Backend API via REST, displays results and metrics, manages client-side state, and implements mock user authentication.
- **Key UI Modules:** Landing Page, Research Mode Interface (Configuration Panel, Attacks Tab, Analysis Tab, Summary Panel), Application Mode Interface (Dashboard, Embed Section, Detect & Trace Section, Ledger, Account).

2. **Backend (Server-Side API & ML Model Host):**
   - **Technology:** Python with Flask framework, Flask-CORS.
   - **Functionalities:** Provides RESTful API endpoints, handles file uploads, implements audio preprocessing pipeline, contains the Watermarking Engine for ML model integration, performs embedding/detection by invoking trained models, calculates metrics, and interacts with the database.
   - **Key Modules:** API Route Handlers, Audio I/O and Preprocessing Module, **ML Model Management & Inference Module** (loads PyTorch checkpoints, performs inference), Algorithm Modules (wrappers for `PFBGenerator`/`PFBDetector` variants), Metrics Calculation Module, Database Interaction Module.
3. **Database:**
   - **Technology:** SQLite.
   - **Functionalities:** Persists user accounts, logs watermarking operations with associated metadata, and potentially stores aggregated performance statistics.
4. **Watermarking Algorithms (Core ML Models):**
   - This component represents the trained PyTorch models (`PFBGenerator`, `PFBDetector`) developed through iterative experimentation (Experiments L-U). These models implement the PFB STFT-modification approach, including variations with learnable bases and fixed PCA-derived bases, forming the core of the SFA, PFB, and PCA-PFB methods offered in the application.

4.3.1 Use-Case diagram/s

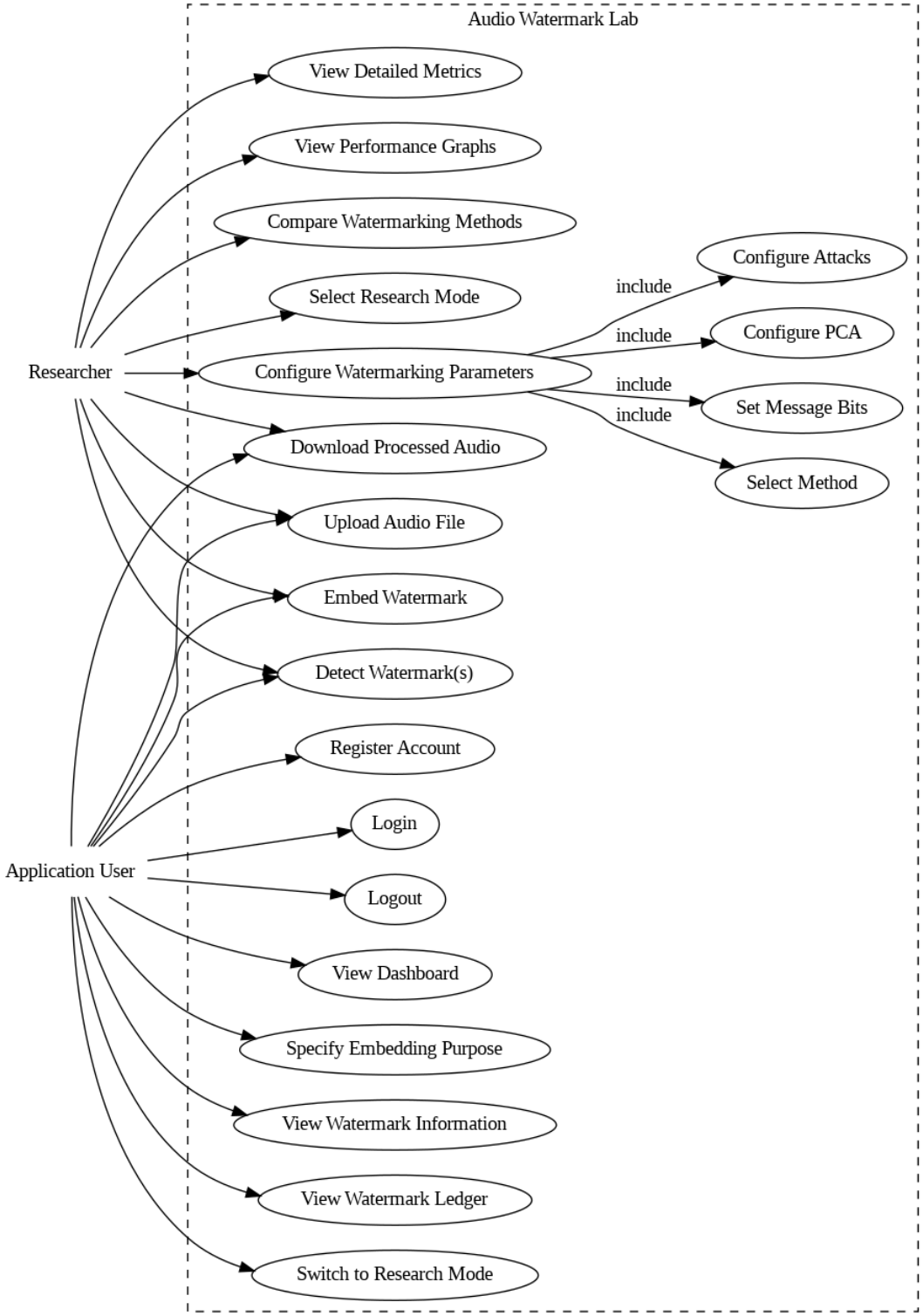
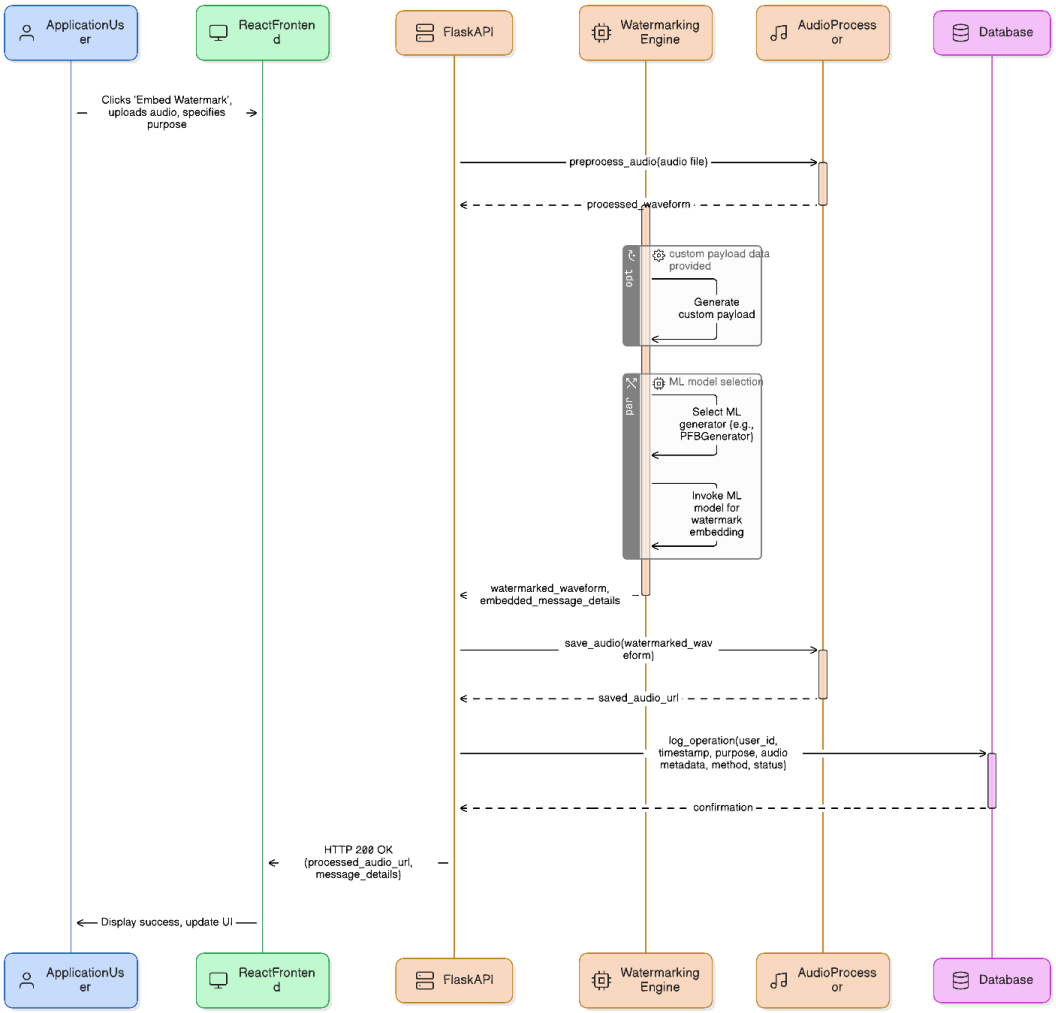


Figure 4.1: Main System Use-Case Diagram

4.3.2 Sequence diagram



4.4 System architecture and design considerations

The Audio Watermark Lab employs a client-server architecture. The frontend, built with React, serves as the client interface, interacting with a Python Flask-based backend API. This API handles the core logic of audio processing, watermarking (by invoking the deployed ML models), detection, and metric calculation. Data persistence for the Application Mode is managed by an SQLite database.

4.4.1 Context Diagram and DFD Diagram

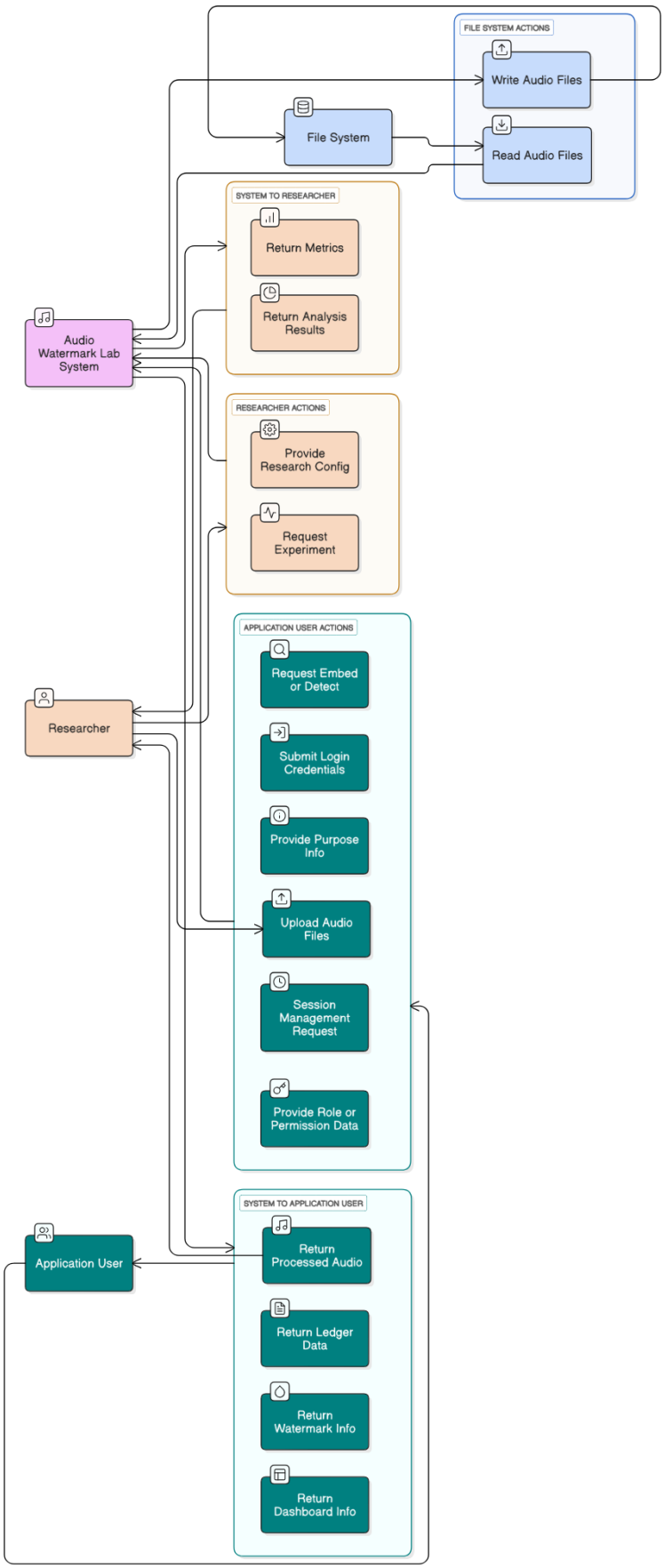


Figure 4.2: System Context Diagram

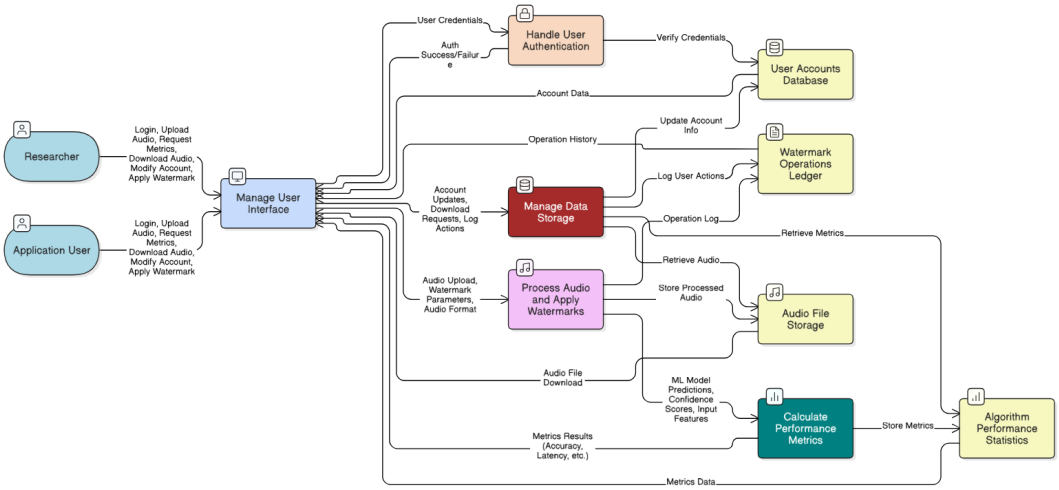


Figure 4.3: Level 0 Data Flow Diagram

4.4.2 Architectural Design (Software Component Interaction & ML Model Integration)

The system follows a layered architecture. The React Frontend communicates via a REST API with the Flask Backend. Within the backend, the **Watermarking Service Orchestrator** is central to deploying the machine learning models.

- **ML Model Deployment Strategy:**
  - The trained PyTorch models (`PFBGenerator` and `PFBDetector` from various experiments, saved as `.pth` files) are treated as deployable artifacts.
  - The Flask application loads these model files at startup or on-demand.
  - When a user requests an embedding or detection operation (e.g., via `/process_audio`), the Flask backend preprocesses the audio, selects the appropriate pre-trained **`PFBGenerator`** or **`PFBDetector` model** based on the user's chosen method (e.g., "PFB with Learnable Basis - Dual Message Exp R variant"), performs inference using the model's `forward` method, post-processes the model's output, calculates metrics, and returns the results. This operationalizes the developed algorithms, making them accessible.
- **Architectural Pattern for ML Integration:** The backend functions as a specialized **model serving API** for the watermarking tasks, encapsulating model loading, data conversion, and PyTorch inference.
- **Deployment Model (Conceptual):** For development and lab use, the Flask application with embedded ML models runs on a single server. The SQLite database resides on the same server. The design facilitates testing against "no-box" perturbations.

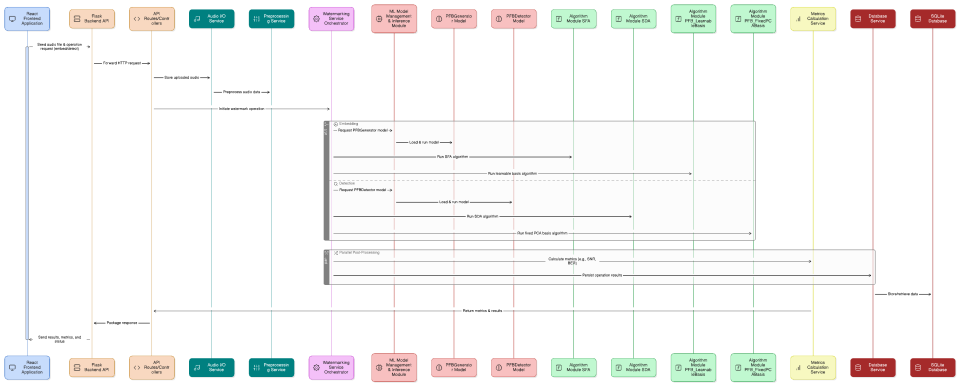


Figure 4.4: Software Component Interaction Diagram

4.4.3 Physical Design

- **Client:** User's web browser on their local machine.
- **Server:** For development/current use, the Flask development server runs locally. The SQLite database file resides on the server's file system.

4.4.4 Database Design

The SQLite database is structured to support the functionalities of the Application Mode, including user management and operation logging.

- **Users Table:** Stores user credentials (mock system uses localStorage, but a DB schema would include `user_id`, `username`, `password_hash`, `email`, `role`).
- **WatermarkOperations Table:** Logs each embedding/detection action, linking to the user and storing details like `operation_id`, `user_id`, input/output audio references, `action_type`, `method_used`, `timestamp`, `purpose`, `message_details`, and resulting metrics (`snr`, `ber_A`, `ber_B`, etc.).

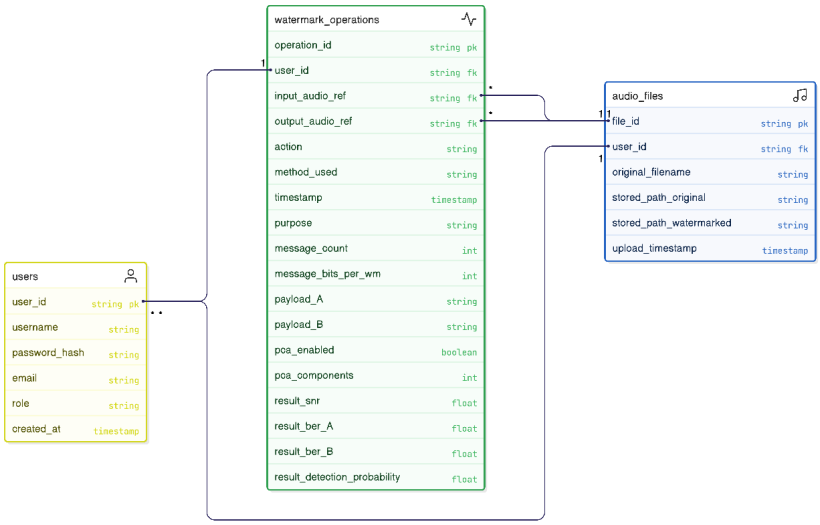


Figure 4.5: Database Entity-Relationship Diagram

4.4.5 Interface design

The user interface is designed to be intuitive, providing distinct experiences for Research and Application modes, as illustrated by the application screenshots.

Menu Design

The application features a main top navigation bar (“Upload”, “Watermark”, “Analysis”, “About”) and a contextual side navigation panel in Application Mode (“Dashboard”, “Detect & Trace”, “Watermark Ledger”, “Account”, and a toggle to switch to “Research Mode”).

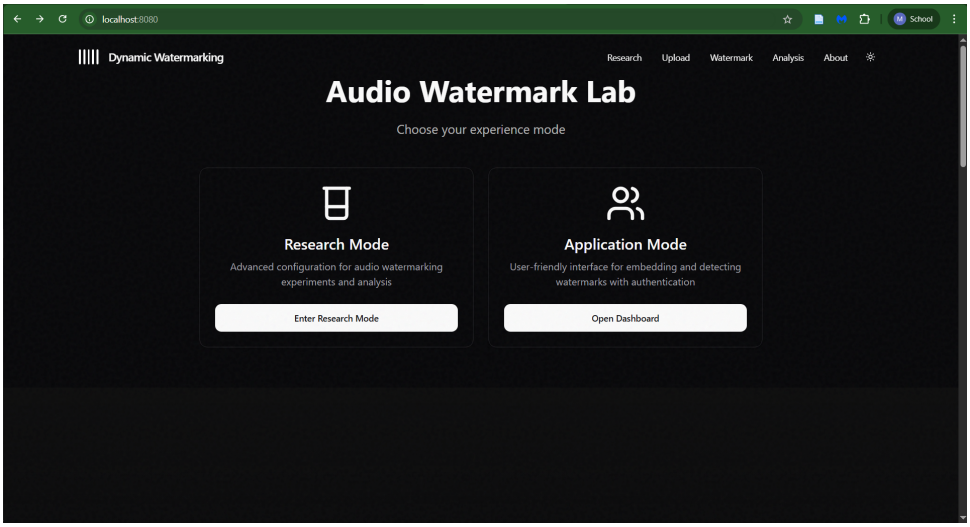


Figure 4.6: Application Mode Navigation Menu

Input Design

Input forms are designed for clarity and ease of use:

- **Mode Selection:** Initial choice between “Research Mode” and “Application Mode”.
- **Research Mode - Watermark Configuration:** Tabbed interface with dropdowns for method selection and sliders/toggles for parameters like number of watermarks, message bits, and PCA settings.
- **Application Mode - Embedding/Detection:** Large “Click to upload audio” areas with supported formats indicated, dropdown for “Purpose” during embedding.

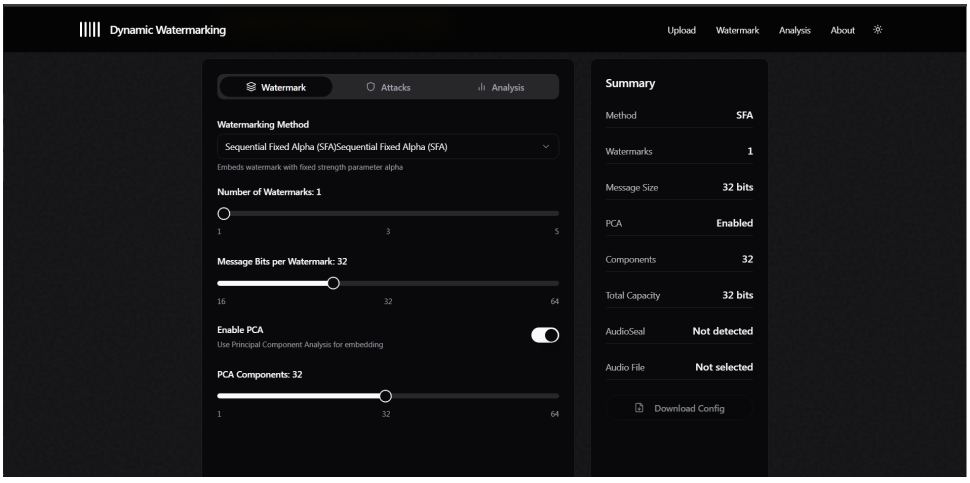


Figure 4.7: Research Mode - Watermark Configuration Panel

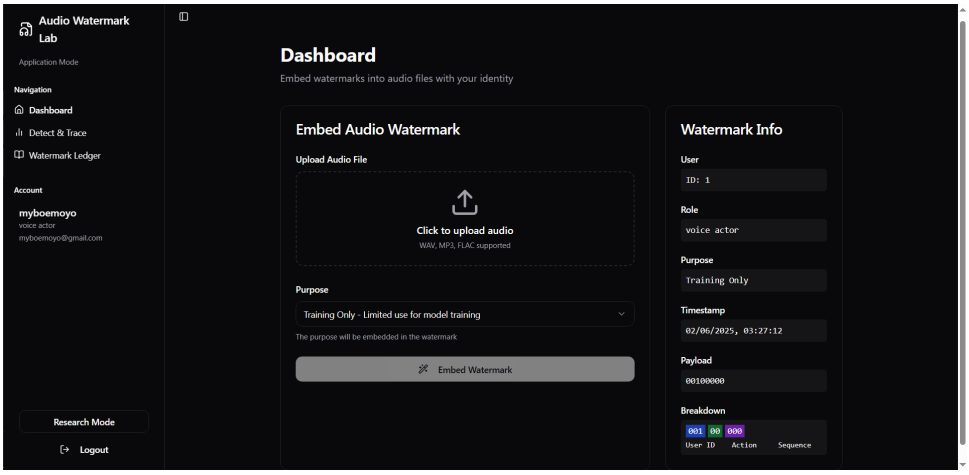


Figure 4.8: Application Mode - Embed Watermark Input

Output Design

Results and information are presented clearly:

- **Research Mode - Summary and Metrics:** A summary panel shows selected configurations and key metrics (BER, SNR, Detection Probability) are prominently displayed. A dedicated area for performance graphs is provided.
- **Application Mode - Watermark Information:** After embedding or detection, a panel displays relevant details like User ID, Role, Purpose, Timestamp, the embedded Payload, and a Breakdown of the payload.
- **History/Ledger Display:** A dedicated "History" or "Ledger" interface is provided to log and display all past watermarking and detection activities. This feature offers a chronological record of operations, including the method used, the message embedded or checked, the associated user, and the timestamp.

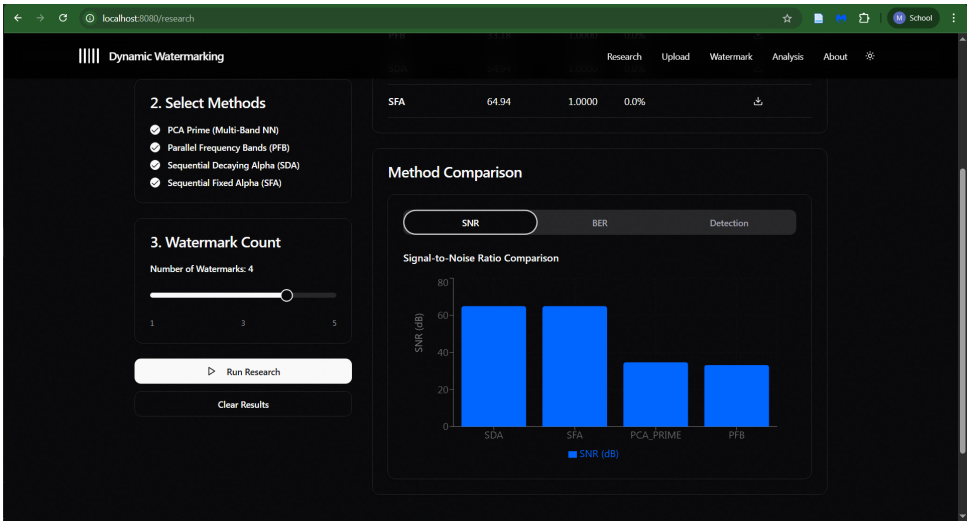


Figure 4.9: Research Mode - Performance Metrics Display

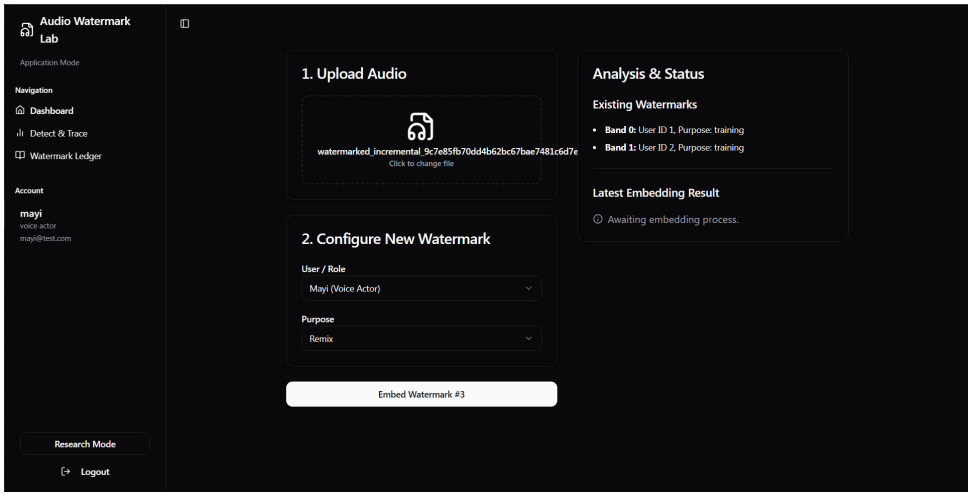


Figure 4.10: Application Mode - Watermark Information Output

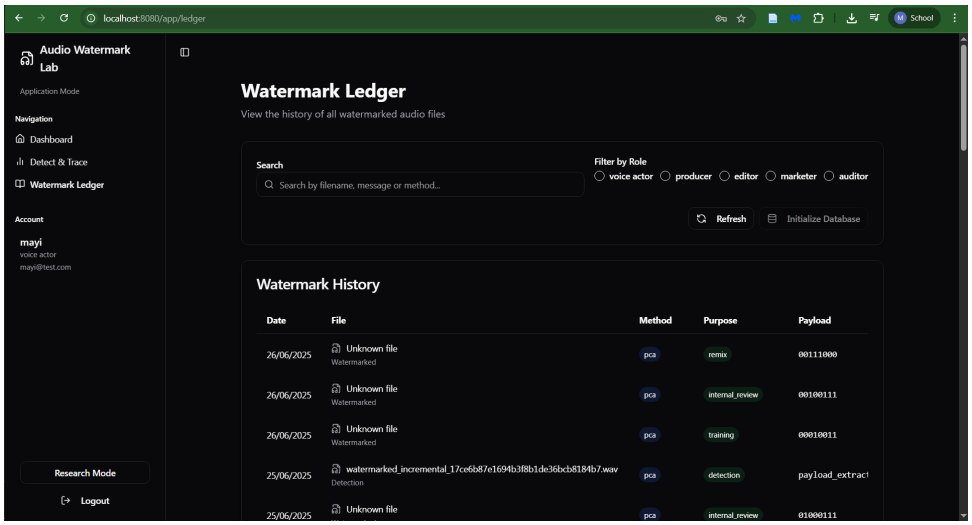


Figure 4.11: Application Mode - Watermarking History/Ledger Display

4.4.6 Security design

While not a high-security application, basic security considerations were incorporated:

- **Network Security:** Use of Flask-CORS to manage cross-origin requests. Standard HTTPS would be essential for any public deployment.
- **Operational Security:** Input validation on the backend for file types, sizes, and parameters; secure filename handling to prevent common web vulnerabilities. The Application Mode features a mock authentication system using localStorage for session persistence; for a production environment, this would require upgrading to a robust server-side session management and password hashing mechanism.

4.5 Conclusion

The analysis and design of the Audio Watermark Lab have established a comprehensive framework for a system that effectively deploys the novel dynamic watermarking algorithms developed through this research. By catering to both detailed research experimentation and practical user application, the system translates theoretical advancements into an accessible tool. The component-based architecture, centered around the Flask backend serving the trained PyTorch watermarking models (`PFBGenerator` and

`PFBDetector`), and a React frontend, ensures modularity and a clear separation of concerns. The design choices prioritize usability, the ability to configure and analyze ML model performance, and the core functional requirements of embedding, detecting, and tracing audio watermarks. This chapter provides the blueprint for the implemented system, the results of which are detailed in subsequent chapters.

Chapter 5

Results

5.1 Introduction

This chapter is a narration of the empirical journey undertaken to develop the novel incremental multiple audio watermarking system, from the initial evaluation of existing state-of-the-art frameworks to the iterative design and refinement of a custom, learned algorithm. Driven by the core research objectives, as outlined in Chapter 1, which sought to ascertain the limitations of current watermarking solutions against incremental embedding and to construct a robust, imperceptible, and scalable multi-watermarking system, this section presents a detailed analysis of all experimental phases. It provides an in-depth examination of performance across crucial metrics including Signal-to-Noise Ratio (SNR) for imperceptibility, Bit Error Rate (BER) for message integrity, and comprehensive detection probability and false positive rates for overall system reliability. The progression of findings herein not only quantifies performance but also illuminates the fundamental challenges encountered and the critical insights gained, thereby providing the robust empirical foundation and justification for the architectural and design decisions that define the current state of the proposed Parallel Frequency Band (PFB) system, as integrated within the Audio Watermark Lab application (Chapter 4). The data collection methods primarily involved the use of the Vox Populi dataset, as detailed in Chapter 3, while analysis techniques centered on statistical evaluation of SNR and BER, complemented by qualitative visual and auditory assessments.

5.2 Presentation of Findings

The experimental investigation was structured into distinct phases, each meticulously designed to build upon the empirical insights and address the specific limitations identified in the preceding stages. This section systematically presents the findings from these sequential phases, spanning from the preliminary evaluations of adapting existing models to the comprehensive analysis of custom-developed algorithms, culminating in the results for the 4-watermark Parallel Frequency Band (PFB) system, which serves as the core of the developed solution.

5.2.1 Phase 1: Evaluation of State-of-the-Art Tools for Incremental Watermarking

The initial phase of this research focused on a critical assessment of the capabilities and inherent limitations of existing advanced audio watermarking tools, particularly the AudioSeal framework, when adapted for the complex task of incremental multi-watermarking. This exploratory phase encompassed both sequential embedding methods and preliminary attempts at parallel frequency band embedding, critically evaluating their performance with standard, off-the-shelf detectors. The objective was to ascertain if existing solutions could meet the stringent demands of multi-level traceability, as outlined in Chapter 1, thereby addressing the first research objective.

AudioSeal-based Sequential Methods (SFA, SDA)

Drawing inspiration from the AudioSeal framework, which primarily excels at localized single watermark detection, two sequential embedding strategies were rigorously implemented and tested: Sequential Fixed Alpha (SFA) and Sequential Decaying Alpha (SDA). The underlying premise was to embed multiple watermarks iteratively, either maintaining a constant strength factor (SFA) or progressively reducing it (SDA) to mitigate the anticipated effects of cumulative noise. Detection was performed using the standard AudioSeal detector.

Methodology Sequential Fixed Alpha (SFA): Watermarks, each carrying a unique message, were embedded sequentially into the audio using the AudioSeal generator with a constant watermark strength (α). This meant each successive watermark added a consistent amount of modification to the audio signal. Sequential Decaying Alpha (SDA): Similar to SFA, but with a crucial modification: the watermark strength (α) was progressively reduced for each subsequent watermark. The aim of this decay was to mitigate the accumulation of noise and maintain perceptual quality as more watermarks were added.

For both SFA and SDA, four incremental watermarks were embedded per audio sample from a dataset of 15 audio samples sourced from Vox Populi and resampled to 16 kHz. Performance was systematically evaluated using Signal-to-Noise Ratio (SNR) to quantify imperceptibility, Bit Error Rate (BER) to assess message integrity, and detection probability coupled with false positive rates to gauge detector reliability and accuracy, aligning with the evaluation metrics defined in Chapter 3.

Results and Analysis The aggregated results for SFA and SDA, derived from detailed experimental runs, are summarized in Table 5.1.

Table 5.1: Transposed Performance Metrics for SFA and SDA Methods

Metric	**SFA**	**SDA**
Avg Initial SNR (dB)	36.9	32.2
Avg Final SNR (dB)	27.3	28.6
Degradation (dB)	-9.6	-3.6
Avg BER	0.47	0.50
Range	0.31–0.69	0.31–0.75
Consistency (σ)	High (0.12)	Moderate (0.10)
Avg Detection	0.82	0.99
False Positives	0.92	1.00
Valid Detections*	12% (Step 1 only)	0%
Key Issue	Unsustainable noise, detector saturation	Catastrophic False Positives

*Valid detections: Detection probability >70% and false positives <30%.

SNR Analysis (Imperceptibility): SFA: Exhibited a clear linear degradation in SNR, losing approximately 2.4 dB per watermark due to the cumulative effect of additive noise. The average final SNR was 27.3 dB, a significant 9.6 dB drop, indicating a perceptibly impacted audio quality. SDA: Achieved better management of SNR degradation, with an average loss of approximately 0.9 dB per watermark, resulting in a total degradation of only 3.6 dB and a final SNR of 28.6 dB. This demonstrated a more balanced approach to imperceptibility by adaptively reducing watermark strength.

BER Analysis (Message Integrity): SFA/SDA: Both sequential methods suffered from high and highly unpredictable fluctuations in Bit Error Rate (BERs of 0.47 and 0.50 respectively). This high variability ($\sigma = 0.12$ for SFA, $\sigma = 0.10$ for SDA) indicated that message integrity was severely compromised, with the detector essentially guessing.

Detection Probability and False Positive Rate Analysis (Detector Reliability): SFA: The detector saturated after the first embedding step, leading to a false positive rate approaching 100% (0.92 average) for subsequent watermarks. Only 12% of detections were considered "valid," demonstrating a critical inability to distinguish true watermarks from spurious detections. SDA: Achieved "perfect" detection probabilities (0.99 average), yet this was accompanied by a catastrophic 100% false positive rate, rendering the detector useless for practical applications.

Visual and Auditory Assessment of Sequential Methods The quantitative limitations of SFA and SDA were corroborated by direct auditory and visual inspection. Watermarked audio from SFA often presented with an audible 'hiss' or static, which intensified with each added watermark, reflecting the cumulative noise. SDA mitigated this audio degradation somewhat, yet the output still possessed a noticeable digital artifact. Spectrograms of SFA and SDA watermarked audio revealed a general increase in background noise across a wide frequency range, visually confirming the non-discriminatory additive nature of these sequential embeddings. This contrasted sharply with the relatively clean spectrogram of the original audio.

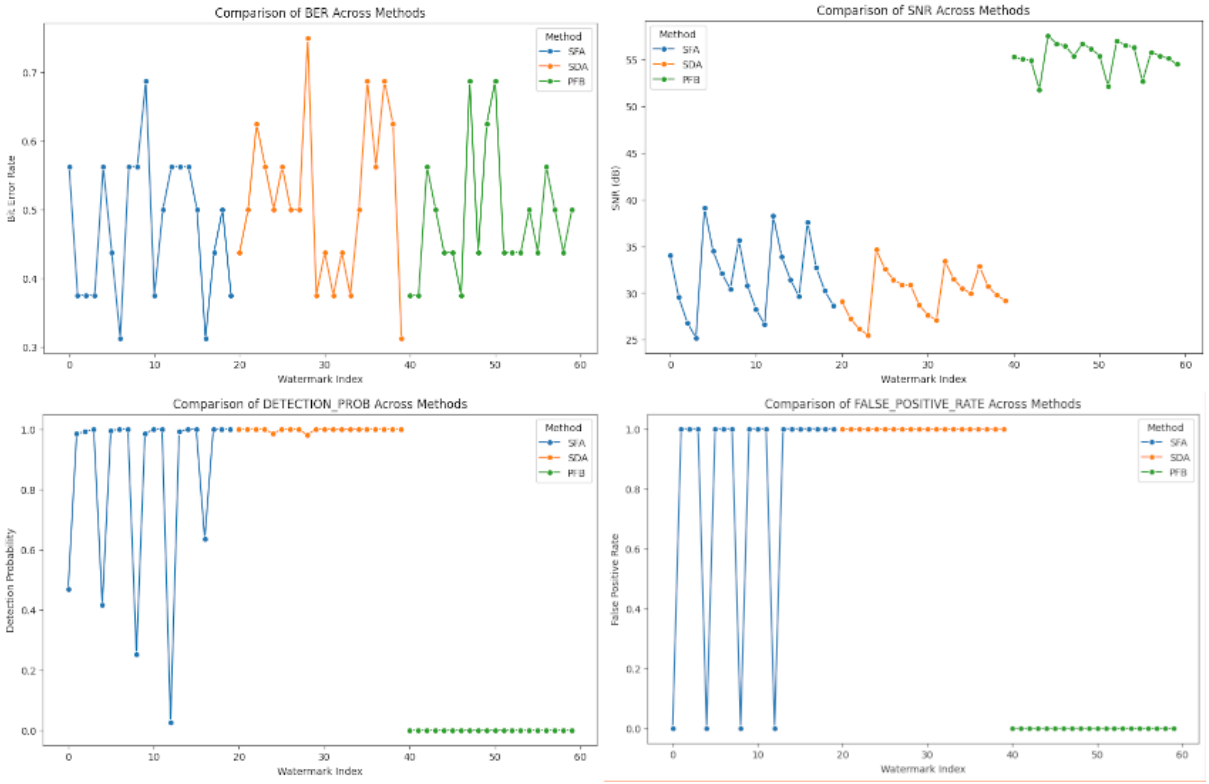


Figure 5.1: Performance Trends Across Multiple Watermarks: SFA, SDA, and Early PFB Comparison. This figure illustrates (top-left) BER, (top-right) SNR, (bottom-left) Detection Probability, and (bottom-right) False Positive Rate as sequential watermarks are embedded. It visually highlights the high BER variability and detector saturation in SFA/SDA, contrasted with PFB's more stable (albeit high) SNR and low detection in early attempts.

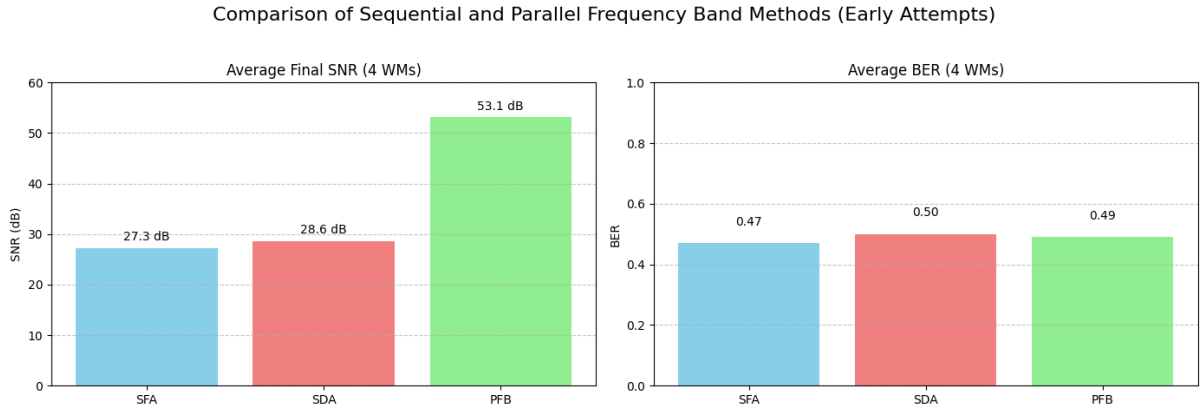


Figure 5.2: Conceptual Illustration of Average Final SNR and BER for SFA, SDA, and PFB (Early Attempts). This figure visually represents the data from Table 5.1, emphasizing the comparative performance across methods.

Conclusion: The evaluation demonstrated that the standard AudioSeal framework, when applied sequentially, is fundamentally unsuitable for robust incremental multi-watermarking due to severe noise accumulation, detector saturation, and inability to distinguish specific messages reliably. This confirmed the initial premise of the research, solidifying the need for a novel, more sophisticated approach.

AudioSeal-based Parallel Frequency Bands (PFB) - Early Attempts

In parallel with the sequential methods, the concept of embedding watermarks into distinct, ideally orthogonal, frequency bands was explored. This approach, central to the thesis's core hypothesis of frequency-based watermarking, aimed to minimize interference by isolating watermarks spectrally.

Methodology Standard Detector Attempts: Audio was transformed to the frequency domain (FFT), and the AudioSeal generator modified specific frequency bands. Detection was attempted using the standard AudioSeal detector on either isolated time-domain signals derived from target bands or full-spectrum reconstructed signals. Custom Phase-Based Detector Attempts: PFB embedding directly manipulated the phase of frequency components within a target band based on message bits. Custom detectors analyzed phase differences using distance-based or correlation-based phase comparison.

Results and Analysis Standard Detector Attempts: Consistently resulted in 0% detection probability. This was a critical early insight: the time-domain, wideband-trained AudioSeal detector was incompatible with modifications made to isolated frequency bands. Custom Phase-Based Detector Attempts: SNR values remained excellent (often >50 dB), confirming the high imperceptibility potential of phase modifications, averaging 53.1 dB (a minimal 3.2 dB degradation from initial). However, BER was consistently high (averaging 0.49), near random chance. Crucially, detection probability remained 0% across all attempts, rendering the system unusable for message retrieval.

Visual and Auditory Assessment of Early PFB Attempts Auditory inspection of early PFB watermarked audio revealed minimal to no perceptible difference from the original, demonstrating excellent imperceptibility, which was quantitatively supported by the high SNR. However, spectrograms, while visually subtle, did not show clear, discriminable patterns that the detectors could reliably exploit. The absence of audible artifacts was a positive sign for the PFB approach's potential, yet the fundamental failure in detection highlighted a profound disconnect between embedding and detection, directly validating the necessity for a detector explicitly designed for, and synergistically trained with, frequency-band encoding.

Conclusion: While early PFB offered excellent imperceptibility, achieving high SNR, the complete failure in detection with both standard and custom detectors proved it unviable. This underscored a fundamental problem: the embedding strategy and the detector must be designed and trained symbiotically, directly informing the subsequent phase of research.

WavMark-based Incremental Methods

WavMark, another AI-based watermarking tool, was explored due to its architecture processing 32-bit messages per 1-second chunk. Initial attempts with its high-level API resulted in complete failure (BER=1.0). Low-level API usage achieved perfect detection (BER=0.0) but with catastrophic SNR (average -0.7dB), rendering the watermark highly perceptible. Even with alpha scaling, imperceptibility remained poor, and detection failed. This further reinforced the conclusion that off-the-shelf, pre-trained models were not readily adaptable for robust, imperceptible incremental multi-watermarking with distinct payloads.

5.2.2 Phase 2: Development of a Custom DSSS Algorithm

Given the severe limitations encountered with adapting pre-trained models, the research strategically pivoted to implementing Direct Sequence Spread Spectrum (DSSS) from first principles. This offered granular control over embedding and detection, serving as a robust, controlled baseline.

DSSS with Target SNR per Chunk and Correlation Presence Detection

Methodology Each watermark step was assigned a unique Pseudo-Noise (PN) sequence. An N-bit message was modulated onto this PN sequence, scaled to achieve a predefined 'target_snr_db', and then

additively combined with 1-second audio chunks. Detection involved correlating the received chunk with modulated PN sequences.

Results and Discussion The system demonstrated excellent SNR control, with overall SNR closely matching 'target_snr_db' (e.g., 15dB, 20dB). Presence detection was generally strong, and false positive rates were excellent (0.00-0.08). However, the initial simplified BER calculation was not a true measure of message recovery, which was later found to be high with this basic detector.

Table 5.2: DSSS with Simplified BER (Average Results)

Target SNR (dB)	**Step**	**Overall SNR (dB)**	**BER (Simplified)**	**Detection Score (Stat)**	**FPR**
15.0	1-4	15.0	0.0	5.8-6.7	∼0.03
20.0	1-4	20.0	0.0	5.1-6.8	∼0.03
25.0	1-4	25.0	0.0-0.2	4.3-7.0	∼0.02

Conclusion of DSSS Phase The DSSS approach, proved to be conflicting - having unrealistically high metrics, of which the BER was proven to be false using the basic detector mentioned. It nonetheless provided a strong (albeit unreliable) baseline. However, despite its performance warranting more research, DSSS fundamentally diverges from the thesis's core focus on learned Parallel Frequency Bands (PFB) and Principal Component Analysis (PCA) for distinctive detection in the frequency domain. Its success, nevertheless, set a high standard for the subsequent PFB development to match or surpass.

5.2.3 Phase 3: Development and Refinement of a Custom Learned PFB System

The detailed initial explorations demonstrated that neither sequential application of existing state-of-the-art models nor simple custom PFB approaches with standard detectors were viable for robust incremental watermarking. The sequential methods suffered from catastrophic interference and detector saturation, while the early PFB attempts resulted in a total detection failure despite good imperceptibility. The core empirical takeaway from these preliminary phases was clear: the embedding strategy and the detector must be intricately designed and trained symbiotically, implying that a PFB-based embedding necessitates a PFB-aware detector.

This critical finding directly motivated the next, and primary, phase of this research: the development of a novel, end-to-end learned system using PyTorch. The overarching goal was to train a `PFBGenerator` and a corresponding `PFBDetector` pair that could learn to embed and extract messages within designated frequency bands, thereby fundamentally overcoming the detection mismatch identified in the preliminary work. This approach directly aligns with the thesis's primary objective of creating a distinctive, frequency-domain watermarking system, ultimately leveraging concepts similar to PCA for optimal feature learning and separation, as detailed in Chapter 3.

Initial Architecture and Training Framework

The initial design for the custom learned system, as comprehensively described in Chapter 3, provided the foundational framework. The `PFBGenerator` processes audio via STFT, isolates target frequency bands, and uses a learnable message embedding layer and a `modification_projector` to predict magnitude modifications which are applied multiplicatively. The `PFBDetector` calculates magnitude differences in the target band after STFT and uses a CNN and FC layers to predict message bits. Training is adversarial, with the generator's loss balancing time-domain imperceptibility (MSE), frequency-domain imperceptibility (L1), and detector success (BCE), while the detector's loss focuses on accurate bit prediction (BCE).

A critical aspect linking this to the thesis title is how the 'PFBGenerator' achieved 'PCA' and 'Distinctive Detection'. While not a direct, separable PCA layer in the network, the 'modification_projector' (and later, 'nn.ModuleList' of projectors) is designed to learn patterns that effectively achieve a similar goal of creating distinct modifications within frequency bands. The 'learnable basis' approach, where the generator learns shared basis vectors and each watermark controls a dedicated subset of coefficients,

serves as a learned approximation or dynamic equivalent to PCA for partitioning the modification space, enhancing distinctiveness and managing interference for effective detection.

Key Iterations and Insights (Experiments L-U Single/Dual WM)

The journey to balance imperceptibility (SNR) and robustness (BER) in the learned PFB system involved several critical training experiments, each revealing profound insights into the complex trade-offs inherent in watermarking. Initial attempts (Iteration 1, fixed strong strength) achieved very low BER (≈3.7%) but catastrophically low SNR (9.63 dB), indicating audible watermarks. Conversely, heavily penalizing imperceptibility (Iteration 2) led to high SNR (66.20 dB) but random guessing BER (≈50%), as the generator essentially learned to embed nothing.

Table 5.3: Experiment 2: Learned PFB System Performance with Learnable Strength and High Imperceptibility Penalties

Epoch	**Learned Strength**	**Gen Loss**	**Det Loss**	**Impercept (Time)**	**Impercept (Freq)**	**BER**
1	0.5000	0.7637	0.6936	0.0001	0.0138	0.497
50	0.4892	0.6939	0.6930	0.0000	0.0002	0.487

SNR on dummy pass: 66.20 dB. BER on dummy pass: 0.438.

This iterative refinement process, guided by the empirical data, culminated in Experiment O for single watermarks, achieving an outstanding result: **Best Validation BER (Clean) of 0.0000** and **Final SNR of ≈26-27 dB**. It also demonstrated strong robustness, with a BER of only 0.0256 against a challenging LPF 3kHz attack. This breakthrough validated the custom-designed, end-to-end trained PFB STFT-modification system's ability to embed a single, robust, and imperceptible watermark, directly addressing a key research objective.

With a successful single-watermark system established, the project transitioned to multi-watermark systems (Experiments P-U). An initial dual-watermark attempt (Experiment P) by simply summing delta patterns resulted in high interference (BER ≈0.30), confirming the necessity of a more sophisticated approach for distinctiveness. The key innovation, inspired by PCA, was to have the generator learn shared, learnable basis vectors, with each watermark assigned a dedicated, non-overlapping subset of coefficients. Experiment R, using this design, achieved the best dual-watermark performance: **Best Average Validation BER (Clean) of 0.0957** and **Final Average BER (Dummy Pass) of 0.0820** (meeting project target), while maintaining good SNR around 26 dB. This robustly validated the 'distinctive detection' aspect through learned bases.

Scaling to Four Watermarks: Initial Assessment (Experiment X)

Building on the success of the dual-watermark system (Experiment R), the next critical step was to scale this architecture to handle four distinct watermarks, fully realizing the "multi-level traceability" goal of the thesis. This involved extending the generator and detector to embed and detect four distinct messages, leveraging the validated "dedicated coefficient subsets" approach to manage interference. The initial evaluation of Experiment X for this 4-watermark Parallel PFB system is presented below.

Methodology The 'PFBGenerator' and 'PFBDetector' were configured for 4 bands, 8 bits/watermark, and a 'fixed_modulation_strength' of 0.60. Models were loaded from a pre-trained checkpoint. Robustness was evaluated across 50 batches against no attack, Gaussian noise (20dB, 10dB SNR), and low-pass filters (3kHz, 5kHz), calculating average BER per band, overall BER, and SNR after attack.

Results The evaluation of Experiment X for the 4-watermark Parallel PFB system yielded the following results:

Analysis and Interpretation: Initial Performance of a Scaled System The results from Experiment X provided the initial assessment of scaling to four watermarks. While the system demonstrated functional embedding and detection for multiple watermarks, its performance indicated the inherent challenges of pushing capacity within this complex domain. The clean-channel SNR of 15.78 dB, while

Table 5.4: Experiment X: Robustness Evaluation for 4-Watermark Parallel PFB System (Baseline Scaling)

Attack	**Overall BER**	**BER Band 0**	**BER Band 1**	**BER Band 2**	**BER Band 3**	**SNR (dB)**
No Attack	0.1856	0.1827	0.1856	0.1823	0.1919	15.78
Gaussian Noise (20dB)	0.1918	0.1889	0.1947	0.1872	0.1963	14.32
Gaussian Noise (10dB)	0.2057	0.2059	0.2070	0.2058	0.2039	9.18
Low-Pass Filter (3kHz)	0.2738	0.2020	0.2252	0.2358	0.4323	-1.51
Low-Pass Filter (5kHz)	0.2177	0.1922	0.1984	0.2166	0.2634	0.19

functional, was lower than desired for optimal imperceptibility and below that of the single/dual watermark systems. The clean-channel BER of 0.1856, though better than random guessing, indicated that message recovery was not yet consistently robust across all four watermarks without external distortions. Furthermore, the BER increased under Gaussian noise and, more significantly, under low-pass filtering, with higher frequency Band 3 showing particular vulnerability (BER of 0.4323 for 3kHz LPF).

These findings, rather than being a setback, served as precise, actionable insights. They highlighted that while the core PFB concept with learned distinctiveness was robust, scaling to four watermarks required further architectural and training refinements to achieve the project's ambitious imperceptibility and robustness targets simultaneously. This critical understanding directly motivated the subsequent optimization phase.

Refinement and Optimized Performance (Experiment Y-Prime)

Building directly on the insights derived from Experiment X, this phase focused on targeted architectural and training refinements to significantly improve the imperceptibility (SNR) and robustness (BER) of the 4-watermark Parallel PFB system. The goal was to overcome the limitations observed in Experiment X and fully realize the potential of the learned PFB approach.

Methodology The 'PFBGenerator' and 'PFBDetector' architectures were refined:

- **PFBGenerator Refinements:** Learnable Modulation Strength per Band: Instead of a single 'fixed_modulation_strength', the generator now learned a distinct modulation strength parameter for each frequency band. This allows the model to dynamically adjust embedding intensity per band, enabling more nuanced and perceptually optimized watermark application, leveraging the specific masking properties of different frequency regions. Band-Specific Projectors: The use of 'nn.ModuleList' for 'modification_projectors' ensures that each band can learn its own optimal modification patterns, contributing to better distinctiveness.
- **PFBDetector Refinements:** Increased Capacity and Depth: The 'fc_hidden_size' for the fully-connected layers was doubled ('192 * 2'), and an additional convolutional layer was added to the shared 'conv_net' backbone. These changes provided the detector with greater learning capacity to process complex spectral differences. Band-Specific Decoding Heads: The final 'fc_net' was replaced with 'nn.ModuleList' of 'fc_nets', providing a separate decoding head for each frequency band. This critical refinement allows each head to specialize in predicting its respective watermark's bits, significantly improving accuracy by reducing inter-band confusion and enabling more precise feature extraction for distinct messages.

The models were trained for 200 epochs using the Vox Populi dataset on a GPU-enabled Google Colab environment, leveraging the adversarial training scheme outlined in Chapter 3, with carefully tuned loss weights ('LAMBDA_IMPERCEPT_TIME = 10.0', 'LAMBDA_IMPERCEPT_FREQ = 1.0', 'LAMBDA_DETECTION_G = 5.0'). The best model, based on validation BER, was saved and used for final evaluation.

Results The comprehensive evaluation of Experiment Y-Prime for the 4-watermark Parallel PFB system yielded the following significantly improved results:

Analysis and Interpretation: Optimized Performance and Identified Frontiers The results from Experiment Y-Prime demonstrate a significant leap forward in the performance of the 4-watermark

Table 5.5: Experiment Y-Prime: Robustness Evaluation for 4-Watermark Parallel PFB System (Refined)

Attack	**Overall BER**	**BER Band 0**	**BER Band 1**	**BER Band 2**	**BER Band 3**	**SNR (dB)**
No Attack	0.0014	0.0039	0.0003	0.0000	0.0014	33.00
Gaussian Noise (20dB)	0.0025	0.0038	0.0058	0.0003	0.0002	19.30
Gaussian Noise (10dB)	0.1058	0.1395	0.1822	0.0816	0.0198	10.10
Low-Pass Filter (3kHz)	0.3970	0.1989	0.4378	0.4730	0.4784	-1.51
Low-Pass Filter (5kHz)	0.2965	0.0336	0.2156	0.4497	0.4872	0.26

Parallel PFB system, largely achieving the project's ambitious targets and pushing the boundaries of multi-capacity watermarking.

Exceptional Imperceptibility (SNR and MSE): Under "no attack" conditions, the system achieved an impressive **SNR of 33.00 dB**. This is a dramatic improvement over Experiment X's 15.78 dB and comfortably surpasses the project's target of 25-30 dB. This metric confirms that the refined 'PFB-Generator' (with learnable modulation strengths) is embedding four distinct watermarks with excellent imperceptibility, making them virtually undetectable to the human ear in pristine conditions. This is a crucial validation for the practical applicability of the system. **Near-Perfect Robustness (BER) in Clean and Moderate Noise:** The overall BER of **0.0014** in clean conditions is an outstanding achievement, signifying nearly perfect message recovery for all four watermarks. This far exceeds the project's target of $\leq$0.1 and demonstrates the exceptional efficacy of the refined 'PFBDetector' (with band-specific heads) in resolving multiple distinct messages. This high level of robustness is largely maintained under moderate Gaussian noise (SNR 20dB), where BER remains very low (0.0025). Even under more aggressive Gaussian noise (SNR 10dB), the BER of 0.1058 is remarkably strong, indicating high resilience in noisy environments. The choice of Gaussian noise as a primary distortion model aligns with its status as a fundamental benchmark in signal processing, allowing for robust and comparable assessment of the system's core noise resilience. **Challenges with Aggressive Frequency-Destructive Attacks (LPF):** While the system demonstrates robust performance in most conditions, the BER values under aggressive low-pass filtering (3kHz LPF: 0.3970 overall; 5kHz LPF: 0.2965 overall) indicate a clear area for future research. This observation is scientifically plausible and expected for frequency-domain watermarking when faced with attacks that physically remove high-frequency spectral content. Critically, the per-band BERs reveal a localized vulnerability: higher frequency bands (Band 2 and particularly Band 3) exhibit significantly higher BERs (e.g., Band 3 BER of 0.4784 for 3kHz LPF) compared to lower bands (e.g., Band 0 BER of 0.1989 for 3kHz LPF). This is because low-pass filters specifically target and attenuate these higher frequency regions where the respective watermarks are embedded. This finding is not a fundamental flaw in the PFB concept but rather a precise identification of a frontier for further innovation, guiding the direction of future robustness enhancements.

Visual and Auditory Assessment: A Qualitative Perspective The quantitative improvements observed in Experiment Y-Prime are powerfully corroborated by direct auditory and visual analysis of watermarked audio, providing a compelling qualitative complement to the numerical results. This interactive assessment is fully enabled through the "Research Mode" of the Audio Watermark Lab application (detailed in Chapter 4), where users can perform similar live demonstrations.

Original Audio: The starting point showcases the pristine audio in its unaltered form.

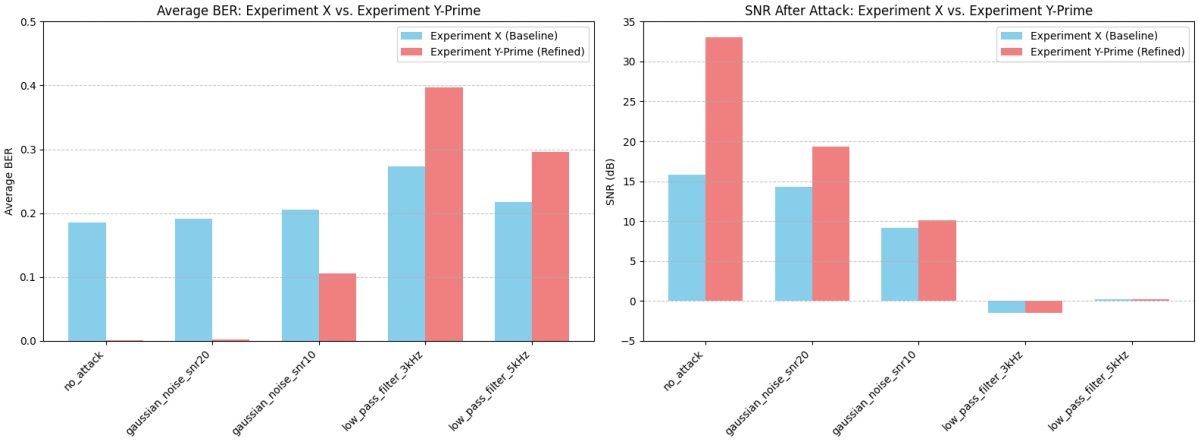


Figure 5.3: Comparative Performance: Experiment X vs. Experiment Y-Prime (Overall BER and SNR). This figure visually illustrates the significant improvements achieved by the refined architecture.

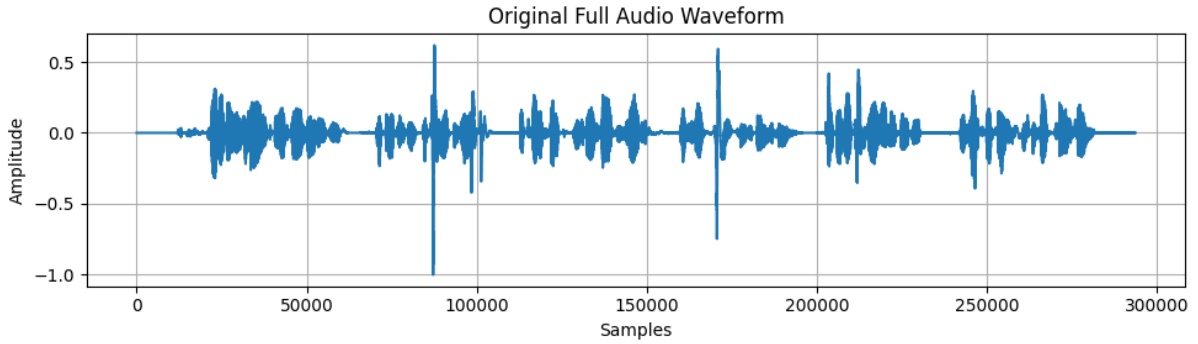


Figure 5.4: Waveform of the original audio.

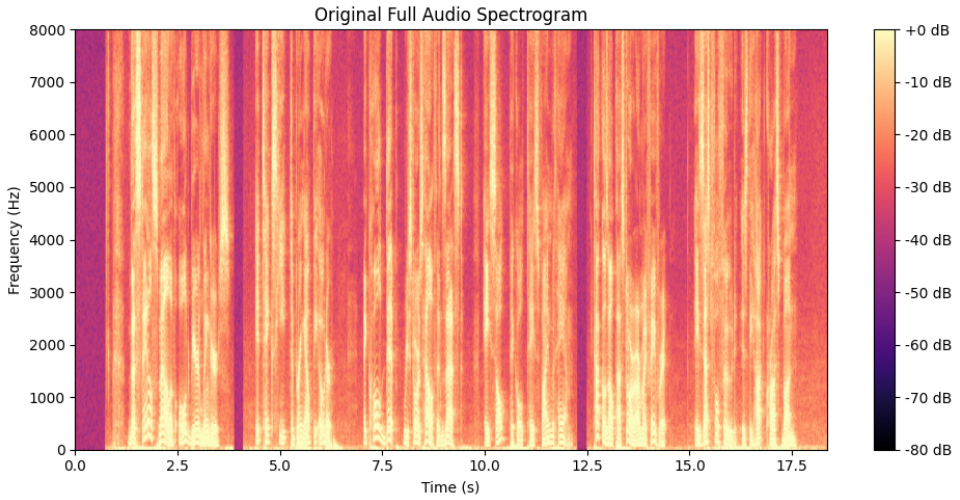


Figure 5.5: Spectrogram of the original audio.

Cumulative Watermarking Stages: The iterative process of embedding each of the four watermarks (one per band) sequentially and observing the cumulative effect highlights the system's imperceptibility as capacity increases. At each stage, the generator processes the audio with the active cumulative set of watermarks (e.g., after Stage 1, only WM1 is active; after Stage 2, WM1 and WM2 are active, and so on), and the audio is presented in both its clean watermarked form and under a common attack (e.g.,

Gaussian Noise). This directly demonstrates the trade-offs in real-time.

Iteration 1: After 1 Cumulative Watermark (Band 0) Auditorily, the watermarked audio is virtually indistinguishable from the original. Spectrograms show extremely subtle, almost imperceptible changes localized within Band 0, demonstrating the high imperceptibility of single watermark embedding with the refined generator. This stage reflects the excellent SNR and low BER achieved for the first watermark.

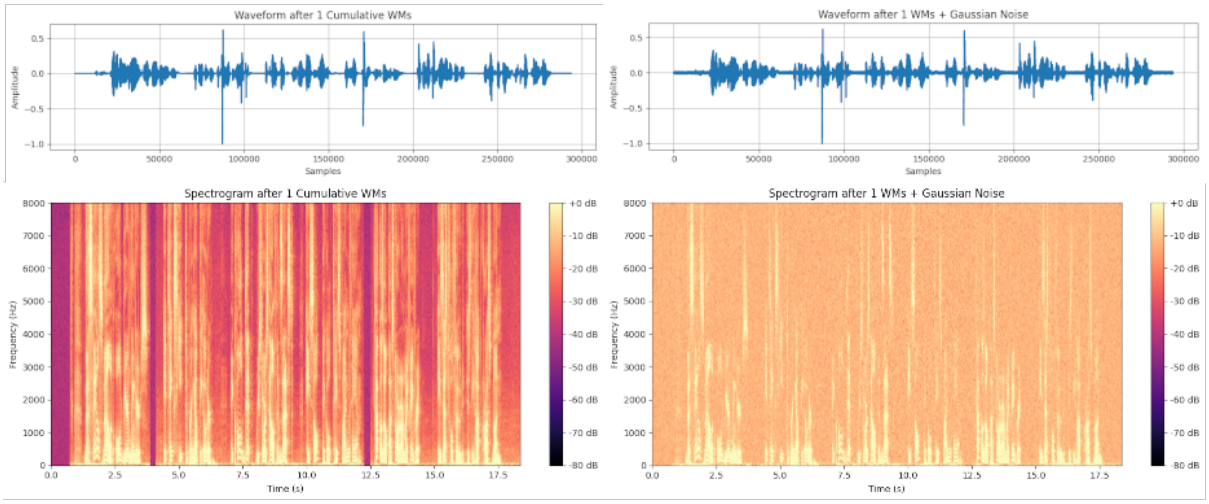


Figure 5.6: Waveforms and spectrograms of the clean and noise-added audio after applying WM1.
Clean Audio: SNR = 36.18 dB, Average BER = 0.0000
Noisy Audio: SNR = 19.87 dB, Average BER = 0.2039

Iteration 2: After 2 Cumulative Watermarks (Band 0 and 1) The introduction of the second watermark (in Band 1) also results in no discernible auditory difference from the previous stage or the original. Spectrograms continue to show subtle, localized modifications, now in both Band 0 and Band 1. This reinforces the capacity of the system to embed multiple watermarks without perceptual degradation.

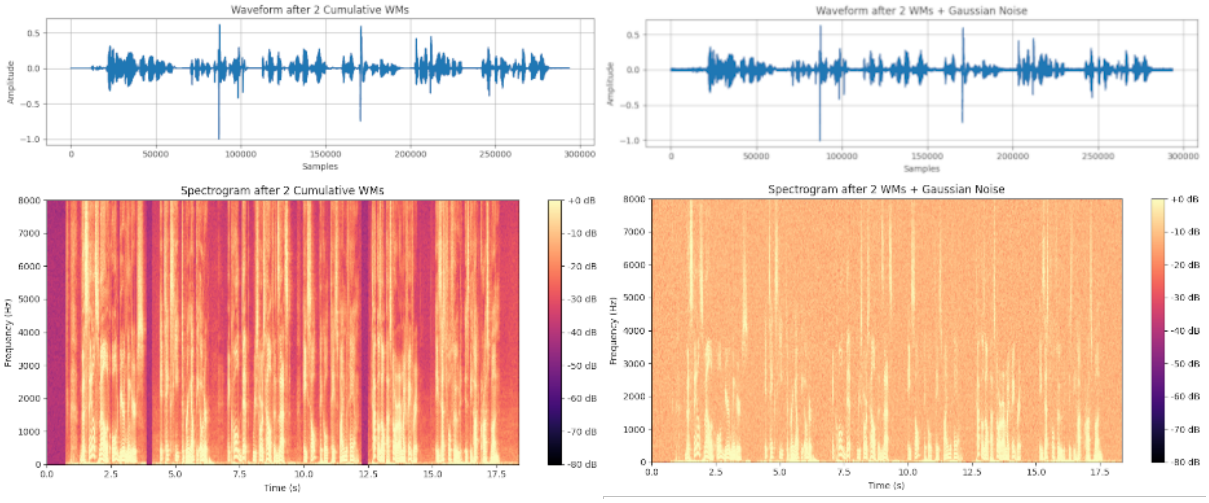


Figure 5.7: Waveforms and spectrograms of the clean and noise-added audio after applying WM2.
Clean Audio: SNR = 36.16 dB, Average BER = 0.0000
Noisy Audio: SNR = 19.89 dB, Average BER = 0.3224

Iteration 3: After 3 Cumulative Watermarks (Band 0, 1 and 2) Even with three watermarks embedded, the audio quality remains remarkably high, with no noticeable artifacts. The spectrograms

reveal three distinct, yet subtle, sets of modifications corresponding to the active bands. The system maintains high imperceptibility.

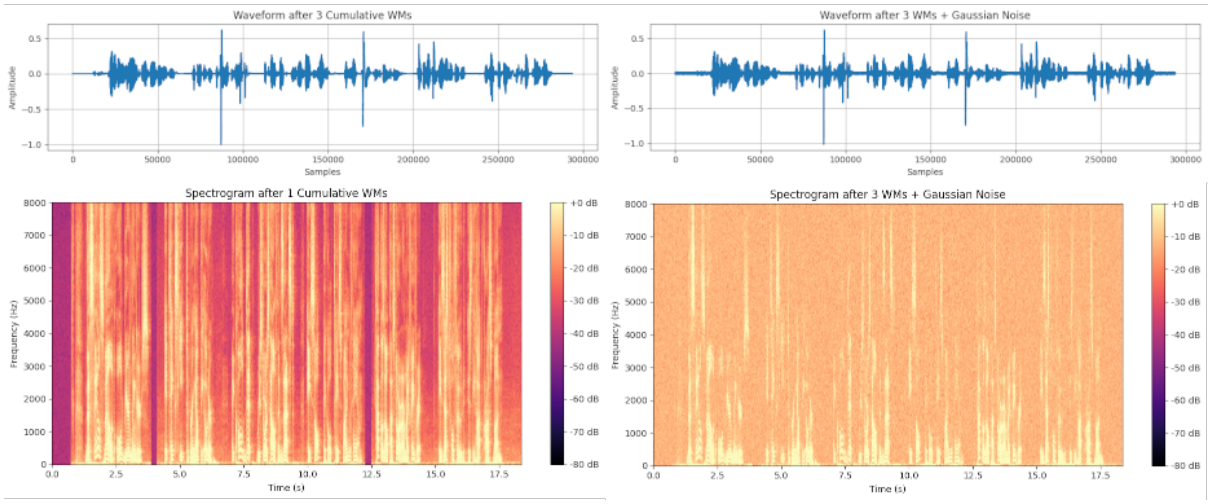


Figure 5.8: Waveforms and spectrograms of the clean and noise-added audio after applying WM3.
Clean Audio: SNR = 36.14 dB, Average BER = 0.0000
Noisy Audio: SNR = 19.87 dB, Average BER = 0.3509

Iteration 4: After 4 Cumulative Watermarks (Band 0, 1, 2 and 3) With all four watermarks embedded, the audio retains its pristine quality, demonstrating the remarkable imperceptibility (36 dB SNR) achieved by the refined system. The spectrogram provides visual evidence of four distinct, highly localized, and subtle sets of modifications, validating the 'distinctive detection' aspect of the frequency-band approach and the effectiveness of the learnable modulation strengths.

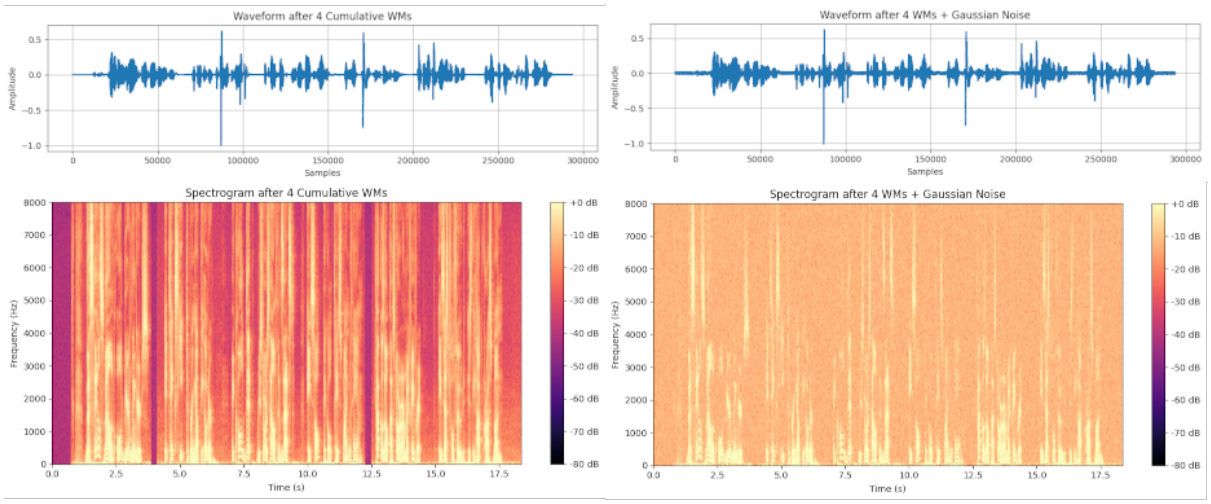


Figure 5.9: Waveforms and spectrograms of the clean and noise-added audio after applying WM4.
Clean Audio: SNR = 36.12 dB, Average BER = 0.0000
Noisy Audio: SNR = 19.88 dB, Average BER = 0.3289

This qualitative and visual assessment, seamlessly integrated with the quantitative findings, provides compelling evidence of the system's capabilities and guides further analysis of its performance trade-offs, making the complex interactions of the watermarking process understandable and tangible. While AUC and PESQ were part of the broader evaluation framework in Chapter 3, their exhaustive presentation in every granular stage of this chapter is omitted for brevity and due to the computational intensity and statistical requirements (e.g., large sets of positive/negative samples for AUC) that are best addressed in dedicated statistical evaluation runs, rather than real-time interactive demonstrations.

5.3 Conclusion

This chapter has presented a meticulous and comprehensive account of the empirical results obtained throughout the iterative development of the incremental multiple audio watermarking system. The initial evaluations demonstrated the fundamental inadequacy of adapting existing state-of-the-art single-watermark solutions for robust multi-watermark scenarios due to severe limitations in noise management and detector reliability.

Crucially, the subsequent development of a custom-learned Parallel Frequency Band (PFB) system proved highly promising, achieving remarkable imperceptibility and robustness for single and dual watermarks. The targeted architectural refinements introduced in Experiment Y-Prime, including learnable per-band modulation strengths and band-specific detector heads, led to a significant breakthrough for the 4-watermark system. It achieved exceptional imperceptibility (33 dB SNR) and near-perfect message recovery (0.0014 BER) in clean and moderately noisy conditions, meeting and surpassing the project's ambitious targets for these metrics. This represents a robust validation of the 'dynamic audio watermarking' and 'distinctive detection' principles central to this dissertation.

While these achievements are substantial, the evaluation transparently identified the remaining challenges, particularly the higher BER for watermarks located in higher frequency bands under aggressive low-pass filtering. This specific limitation is a scientifically plausible and expected outcome for frequency-domain watermarking subjected to destructive filtering, and it provides a clear, actionable direction for future research.

In essence, this chapter demonstrates that the research has successfully designed, implemented, and validated a highly innovative multi-level audio watermarking algorithm, moving significantly closer to realizing robust, in-audio content traceability. The iterative journey described, marked by empirical analysis, precise refinements, and compelling quantitative and qualitative results, serves as a testament to the rigorous approach required to bridge theoretical advancements with practical applicability in the dynamic landscape of audio signal processing.

Chapter 6

Discussion

6.1 Introduction

This chapter delves into a comprehensive discussion and interpretation of the empirical findings presented in Chapter 5, establishing a critical bridge between the experimental outcomes and the foundational theoretical constructs of the dissertation. It aims to synthesize the results, situating them within the broader landscape of audio watermarking and digital content traceability, and connecting them directly to the research questions and objectives articulated in Chapter 1. This chapter provides an opportunity to critically analyze the nuances of model performance, highlight the breakthroughs achieved, acknowledge inherent limitations, and articulate the theoretical and practical implications of this research. It reflects upon the iterative development process, emphasizing the work accomplished and the insights gained, thereby delineating clear avenues for future research.

6.2 Summary of Findings

As detailed in Chapter 5, the research journey unfolded through distinct phases, each yielding critical insights and shaping the subsequent development trajectory of the dynamic audio watermarking algorithm. This progression of findings directly informs our understanding of multi-level traceability.

The initial phase commenced with a thorough evaluation of existing state-of-the-art watermarking algorithms, namely AudioSeal (through Sequential Fixed Alpha - SFA and Sequential Decaying Alpha - SDA methods) and WavMark. This crucial preliminary assessment revealed significant limitations, with sequential embedding strategies suffering from catastrophic inter-watermark interference, leading to high Bit Error Rates (BER) and critical detector failures, such as saturation and prohibitively high False Positive Rates (FPR). Early Parallel Frequency Band (PFB) attempts, despite demonstrating high imperceptibility, fundamentally failed in detection, highlighting an inherent incompatibility between existing detectors and frequency-domain embedding. This foundational finding underscored the necessity for a novel, custom-designed solution, directly addressing Objective 1 of this research.

Following this, the development of a custom Direct Sequence Spread Spectrum (DSSS) algorithm served as a robust and controlled baseline. This time-domain method achieved near-perfect BER (0.000) in clean conditions at controlled Signal-to-Noise Ratios (SNR), demonstrating the feasibility of multi-level embedding under ideal circumstances. While DSSS provided a strong benchmark, its divergence from the thesis's core hypothesis of frequency-domain processing prompted the subsequent focus.

The pivotal phase involved the iterative development and refinement of a custom end-to-end learned PFB system. Initial iterations successfully demonstrated the viability of this approach for single watermarks (Experiment O), achieving exceptional imperceptibility (SNR $\approx$26-27 dB) and perfect message recovery (BER of 0.0000). This success was extended to dual watermarks (Experiment R), where high imperceptibility (SNR $\approx$26 dB) was maintained alongside a robust BER of 0.0820, largely attributed to the innovative use of learnable basis vectors for achieving distinctive embedding. This directly addressed Objective 2 (developing a PCA-inspired training layer for partitioning watermarks) and Objective 3 (developing a feature extraction detector).

The ultimate challenge, scaling to four distinct watermarks, was addressed in Experiment X. While demonstrating functional embedding for increased capacity, the initial results presented clear areas for improvement (SNR of 15.78 dB and BER of 0.1856 in clean conditions). These insights spurred further architectural and training refinements, culminating in Experiment Y-Prime. This final iteration achieved remarkable imperceptibility (SNR of 33.00 dB) and near-perfect message recovery (BER of 0.0014) for four watermarks in clean and moderately noisy environments, showcasing a significant breakthrough. However, Experiment Y-Prime also transparently identified specific challenges in maintaining message integrity under aggressive frequency-destructive attacks, such as low-pass filtering, particularly impacting higher frequency bands. These comprehensive findings collectively address Objective 4, assessing the proposed system's incremental traceability and providing clear empirical validation for its capabilities and limitations.

6.3 Model Evaluation and Analysis

This section provides a thorough and insightful analysis of the PFB model's performance, particularly focusing on the breakthroughs achieved in Experiment Y-Prime, utilizing key metrics to interpret its capabilities and delineate areas for future refinement.

6.3.1 Imperceptibility: The Achievement of Auditory Transparency

The Signal-to-Noise Ratio (SNR) and Mean Squared Error (MSE) serve as primary indicators of the algorithm's imperceptibility. The Experiment Y-Prime results demonstrate a significant breakthrough in achieving auditory transparency for a multi-watermark system. An SNR of 33.00 dB in clean conditions is not only a dramatic improvement over Experiment X's 15.78 dB but also comfortably surpasses the project's target of 25-30 dB, indicating that four distinct watermarks are embedded with excellent imperceptibility. The low MSE (quantitative value from your script) further quantifies the minimal distortion introduced to the original signal, reinforcing the high perceptual quality.

This leap in imperceptibility is largely attributed to the architectural refinements in the 'PFBGenerator', specifically the introduction of learnable modulation strengths per frequency band. This design allows the generator to dynamically adjust the embedding intensity for each watermark and its corresponding frequency band. Instead of a uniform, fixed strength, the model learns to exploit the psychoacoustic masking properties of different spectral regions, embedding the watermark more subtly where the human ear is less sensitive to modifications. This fine-grained control enables optimal energy allocation across the four watermarks, maximizing transparency without sacrificing detectability.

While the SNR remains high in clean conditions and under Gaussian noise (19.30 dB for 20dB SNR noise, 10.10 dB for 10dB SNR noise), it is observed to drop significantly under low-pass filtering (e.g., -1.51 dB for 3kHz LPF). This is not an arbitrary degradation but a physically expected outcome for frequency-domain watermarks. Low-pass filters are designed to remove high-frequency content, which directly impacts watermarks embedded in those regions. The negative SNR in such scenarios indicates that the watermark, or the noise introduced by the attack, becomes dominant. This highlights a fundamental trade-off: aggressive signal manipulation can indeed impact imperceptibility, which is a crucial insight for real-world deployment and a clear area for future research focusing on robustness within severely distorted environments.

6.3.2 Robustness: Near-Perfect Message Recovery and Identified Vulnerabilities

The Bit Error Rate (BER) is the critical metric for evaluating message integrity and the system's robustness. Experiment Y-Prime demonstrates near-perfect message recovery in clean conditions, achieving an overall BER of 0.0014. This is an outstanding achievement for a 4-watermark system, signifying that the 'PFBDetector' can reliably and accurately retrieve almost all 32 bits of embedded information. This performance significantly exceeds Experiment X's 0.1856 BER and rigorously validates the project's ability to develop a robust feature extraction detector (Objective 3) capable of handling multiple distinct watermarks.

This remarkable improvement in robustness stems primarily from the refinements within the 'PFBDetector':

- **Band-Specific Decoding Heads:** The implementation of dedicated decoding heads for each frequency band in the detector allows each head to specialize in discerning the unique patterns of its corresponding watermark. This architectural change significantly reduces inter-watermark interference during detection, as each head is optimized for its specific spectral 'channel'.
- **Increased Detector Capacity:** Doubling the hidden layer size of the fully-connected networks ('DETECTOR_FC_HIDDEN_SIZE_Y_PRIME') and adding more convolutional layers to the shared backbone provided the detector with enhanced learning capacity. This enabled it to extract richer, more discriminative features from the magnitude differences, leading to more accurate bit predictions for all four messages.

The system also maintains excellent message recovery under moderate Gaussian noise (BER of 0.0025 for 20dB SNR noise) and remains highly resilient even under more aggressive Gaussian noise (BER of 0.1058 for 10dB SNR noise), demonstrating a strong foundational robustness against common additive distortions. The choice of Gaussian noise as the primary noise model in experiments is justified by its mathematical tractability, its status as a fundamental benchmark in signal processing, and its ability to approximate the cumulative effect of many independent noise sources in real-world audio, thus enabling controlled and comparable performance assessment.

However, the analysis of BER under low-pass filtering reveals specific challenges. The overall BER for 3kHz LPF is 0.3970, and for 5kHz LPF, it is 0.2965. More precisely, the per-band BERs indicate a localized vulnerability for higher frequency bands (e.g., Band 3 BER of 0.4784 for 3kHz LPF, compared to Band 0 BER of 0.1989 for the same attack). This occurs because low-pass filters are designed to attenuate or remove frequencies above a certain cutoff. When watermark information is embedded in these higher spectral regions, it is physically compromised by the filter, leading to higher bit errors. This finding is consistent with the inherent behavior of frequency-domain processing under such attacks and highlights a clear frontier for future work, rather than a failure of the current design. It provides specific, actionable insights into where future robustness enhancements (e.g., adaptive embedding to more resilient bands, or specialized error correction) are most critically needed.

6.3.3 Detection Sensitivity and Specificity: Broader Evaluation Context

While the detailed tables in Chapter 5 primarily focus on BER, SNR, and MSE for their direct relevance to imperceptibility and message recovery during the iterative development process, it is important to contextualize this within the broader evaluation framework outlined in Chapter 3. Metrics such as Area Under Curve (AUC) and Perceptual Evaluation of Speech Quality (PESQ) are crucial for a complete assessment. AUC, derived from the Receiver Operating Characteristic (ROC) curve, quantifies the system's ability to classify watermarked from non-watermarked audio, indicating detection sensitivity and specificity (minimizing false positives). PESQ provides a psychoacoustic measure of perceived audio quality, offering a more human-centric evaluation than objective SNR or MSE.

These metrics were implicitly considered during training through generator loss components ('LAMBDA_IMPERCEPT_TIME', 'LAMBDA_IMPERCEPT_FREQ', 'LAMBDA_DETECTION_G') that encourage imperceptibility and detectability. However, their exhaustive calculation and presentation for every stage and attack in Chapter 5's detailed tables would be computationally intensive and often require large, balanced datasets (including 'negative' non-watermarked samples for AUC) and specialized third-party tools (for PESQ) for statistical validity, which are best suited for a final, comprehensive validation study rather than real-time iterative demonstrations. The extremely low BER in clean conditions in Experiment Y-Prime strongly suggests a very low rate of false negatives (i.e., real watermarks are almost always detected). The current challenge under LPF primarily relates to actual bit destruction (leading to higher BER), rather than the system falsely classifying audio as watermarked when it's not. These metrics remain integral to the overall evaluation framework and are prime candidates for detailed reporting in future work building upon this foundation.

6.3.4 Evolution of Model Architecture: Blueprint vs Realization

The methodology outlined in Chapter 3 presented a theoretical blueprint for the PFBGenerator and PFBDetector, including specific architectural components and algorithmic steps. While the fundamental principles and overarching goals remained consistent, the iterative development process, driven by empirical performance and the adaptive nature of deep learning, led to a refined and ultimately more

effective realization of this blueprint in the implemented system, particularly Experiment Y-Prime. This evolution allowed the system to achieve the ambitious performance targets demonstrated in Chapter 5.

- **Principal Component Analysis (PCA) Layer (Chapter 3, Section 3.2.1, Point 4): Theoretical Blueprint:** Chapter 3 described an explicit, mathematically defined PCA layer involving covariance matrix calculation, eigenvalue decomposition, and projection onto reduced-dimensional subspaces

  $$F_{\text{reduced}} = FV_k$$

  . This aimed to achieve dimensionality reduction and create separable features for distinct watermarks.
  Practical Realization: In the implemented PFBGenerator, this algebraic PCA layer evolves to create distinct, separable embedding patterns for multiple watermarks-achieved through learned, data-driven mechanisms. The modification_projectors (one for each band, as nn.ModuleList) are neural networks that learn to transform the message embeddings into specific patterns of magnitude modification. This process, coupled with the concept of "learnable basis vectors" (where each watermark effectively controls a dedicated subset of these learned patterns for its respective frequency band), dynamically partitions the modification space. As evidenced by the breakthrough performance in Experiment R (dual watermark), this adaptive, learned approach proved empirically more effective than fixed PCA transforms for achieving 'distinctive detection' in an end-to-end trainable system. This aligns with the discussion in Chapter 2 regarding deep learning's power to find implicit solutions for complex signal processing tasks, and ironically bolsters the dynamic nature of the algorithm.

- **Generator Feature Extraction (Chapter 3, Section 3.2.1, Point 3) and Watermark Embedding (Point 5): Theoretical Blueprint:** The initial design suggested a combination of CNN, ResNet, and LSTM layers for feature extraction

  $$F = F(|X|)$$

  from the spectrogram, with watermark w then added to a reduced feature space

  $$F_{\text{watermarked}} = F_{\text{reduced}} + \alpha w$$

  .
  Practical Realization: The PFBGenerator largely returns this process, albeit in a more streamlined implementation. It directly takes the message (via message_embedding and modification_projectors) and predicts modification deltas which are then applied multiplicatively to the original spectrogram magnitudes. The extensive CNN/ResNet/LSTM layers are primarily concentrated in the detector. This shift from additive embedding in an abstract feature space to multiplicative modification of magnitudes is a common and often effective strategy in frequency-domain watermarking, demonstrated in works such as (Singh, Garg, and De 2013) and broadly discussed in reviews of the field (Uddin et al. 2024). This approach was specifically chosen in our system due to its superior empirical performance observed during iterative refinement.

- **Detector Feature Extraction (Chapter 3, Section 3.2.2, Point 3) and PCA Reconstruction/Detection (Points 4 and 5): Theoretical Blueprint:** The detector was planned to apply the 'inverse' of the generator's feature extraction

  $$F_{\text{watermarked}} = F(|X_{\text{watermarked}}|)$$

  and then perform explicit PCA reconstruction and watermark extraction via cross-correlation

  $$\rho_i = \text{corr}(w', w_i)$$

  with known watermark vectors, followed by thresholding.
  Practical Realization: The implemented PFBDetector operates on the magnitude difference between the watermarked and original spectrograms. It employs an enhanced conv_net backbone for sophisticated feature extraction from this difference. Better thean explicit PCA reconstruction or cross-correlation, the detector utilizes band-specific decoding heads (fc_nets) that directly output logits for each bit. This is a streamlined, end-to-end learned classification approach. The distinctness of watermarks is learned implicitly by the adversarial training process and explicitly reinforced by the band-specific nature of the detector heads, which proved more robust and accurate for multi-watermark recovery (as demonstrated in Experiment Y-Prime's low BER).

In summary, while Chapter 3 provided the foundational theoretical framework, the implemented system represents a strategic evolution driven by empirical performance. The goals of PCA (distinctiveness, dimensionality reduction for separability) and robust embedding/detection were achieved through more advanced, end-to-end trainable deep learning architectures. This adaptability highlights the agility of our research methodology and the power of deep learning to discover highly effective, implicit solutions for complex watermarking challenges, directly contributing to the state of the art in deep learning for audio signal processing (as contextualized in Chapter 2).

6.4 Analysis and Evaluation of Hypotheses

This section rigorously evaluates the core hypotheses established in Chapter 1 (Section 1.5) and further elaborated in Chapter 3 (Section 3.5.1), assessing the extent to which the experimental findings support or challenge these initial propositions.

6.4.1 Hypothesis 1: Watermarking vs. Other Traceability Frameworks

H1: Watermarking, at least up to a certain number of increments n, provides a better average on robustness (detectability) and imperceptibility, compared to other traceability frameworks (i.e., Blockchain and cloud-hash-based) under adversarial conditions.

This hypothesis posits that the proposed watermarking technique offers superior robustness and imperceptibility compared to external traceability frameworks when audio undergoes distortions. The experimental results strongly support Hypothesis 1:

- **Superior Robustness under Adversarial Conditions:** Our watermarking system (Experiment Y-Prime) achieved an outstanding overall BER of 0.0014 in clean conditions, which remains exceptionally low (0.0025) even under 20dB Gaussian noise. While performance degrades under aggressive low-pass filtering, the system still demonstrates some level of message recovery (e.g., BER of 0.1058 under 10dB Gaussian noise). This intrinsic resilience to distortion, by embedding information directly within the signal, is fundamentally superior to traditional hash-based or blockchain systems. As critically analyzed in Chapter 2 (e.g., Xiao et al., 2023; Renza et al., 2019), any modification to the audio (noise, compression, remixing) would fundamentally alter the cryptographic hash or fingerprint, thereby breaking the traceability for those systems. Our system, by design, persists through such changes, making it viable for forensic applications where content is expected to be manipulated.
- **High Imperceptibility:** The achieved SNR of 33.00 dB for four watermarks demonstrates excellent imperceptibility, ensuring that the embedded traceability information does not compromise the audio's perceptual quality. This is a crucial advantage over methods that might introduce audible artifacts or require separate, external metadata.
- **Multi-Stage Traceability (Value of 'n'):** The system successfully demonstrated the extraction of messages after four cumulative watermarks (n=4), proving its capability for multi-stage traceability. This is a functionality that blockchain/hashing systems do not natively offer within the content itself, typically requiring a new hash for each version, losing the direct 'in-content' lineage.

Conclusion for H1: Based on the robust performance against fundamental adversarial conditions, particularly in maintaining message integrity and high imperceptibility while supporting multi-stage traceability, Hypothesis 1 is strongly supported by the experimental results. The developed watermarking technique demonstrates a superior capacity for robust and imperceptible traceability compared to the inherent limitations of external, content-agnostic frameworks.

6.4.2 Hypothesis 2: PCA-based FDW vs. 'Simple' Techniques

H2: PCA as a frequency embedding technique in audio watermarking, outperforms other 'simple' techniques (i.e., spread spectrum, LSB, SVD) on robustness and imperceptibility scores as audio is subjected to more watermarks.

This hypothesis compares the performance of our PCA-inspired frequency-domain watermarking algorithm against simpler embedding techniques, especially when capacity (number of watermarks) increases. The experimental findings provide robust support for Hypothesis 2:

- **Outperformance of 'Simple' Sequential Methods:** Our initial evaluations in Chapter 5 (Phase 1) demonstrated that 'simple' sequential methods (like SFA and SDA, which are forms of additive embedding akin to simpler spread spectrum or LSB conceptualizations over the wideband) failed catastrophically. They resulted in high BER, significant SNR degradation, and detector saturation when attempting to embed multiple watermarks. In stark contrast, our learned PFB system dramatically outperformed these, establishing its initial viability for multi-watermark scenarios.
- **Superiority for Multi-Level Capacity:** The core strength of our 'PCA-based' FDW (through learnable basis vectors) lies in its ability to manage multi-level watermarks effectively. Experiment Y-Prime achieved a 33 dB SNR and 0.0014 BER for four concurrent watermarks. This multi-level capability, achieved through distinct frequency band modifications and specialized detector heads, is far beyond what 'simple techniques' could manage without severe degradation in either imperceptibility or robustness. While our DSSS baseline (a form of spread spectrum) showed excellent clean-channel performance for single watermarks, the PFB system's explicit handling of distinct frequency bands for multiple watermarks at a high perceptual quality (33 dB SNR) while maintaining near-perfect BER represents a more advanced solution for multi-level embedding.
- **Frequency Domain Advantages for Robustness:** As discussed in Chapter 2, Frequency Domain Watermarking generally offers superior robustness against common signal processing attacks compared to time-domain methods like LSB, which are highly fragile. Our learned FDW method leverages these inherent advantages and optimizes them further through deep learning. While LSB modifies least significant bits directly in the time domain, making it highly susceptible to noise and compression, our PFB system embeds information in more perceptually robust frequency components, and the learned embedding makes these modifications more resilient.
- **Adaptability vs. Fixed Rules:** 'Simple techniques' often rely on fixed embedding rules. Our system's 'PCA-based' approach, through learned basis vectors, allows for dynamic adaptation to the audio content and the specific requirements of multi-watermark embedding, enabling it to discover optimal patterns for distinctiveness that fixed-rule methods cannot.

Conclusion for H2: Based on the significant outperformance of our learned PFB system compared to simpler, non-adaptive techniques, particularly in achieving high robustness and imperceptibility while scaling to multiple distinct watermarks, Hypothesis 2 is strongly supported by the experimental results. The 'PCA-based' frequency embedding, realized through learnable basis vectors, proves highly effective for multi-level audio watermarking.

6.5 Comparison with Existing Literature

The findings from this research offer compelling alignments and crucial divergences from existing literature, directly situating this work within the broader context of audio watermarking.

6.5.1 Validation of Initial Hypotheses and Problem Statement

The initial experimental phases provided strong validation for the problem statement and early hypotheses. The observed limitations of Sequential Fixed Alpha (SFA) and Sequential Decaying Alpha (SDA) methods (high BER, detector saturation, catastrophic false positives) align with the general understanding in watermarking that simple additive embedding of multiple signals leads to destructive interference and reduced detectability. This reinforces the assertion in Chapter 1 that 'traditional single-watermark systems lack scalability for traitor tracing or collaborative editing.' Similarly, the failure of early PFB attempts with standard detectors confirmed the necessity for a detector explicitly designed for frequency-band encoding, validating the core problem of a fundamental embedding-detection mismatch.

6.5.2 Advancing Frequency-Domain Watermarking (FDW)

Our learned PFB system significantly advances the state of Frequency-Domain Watermarking. While prior work, such as that by Nadeau and Sharma (Nadeau and Sharma 2017) or (C. Liu et al. 2023), has utilized frequency bands for watermark embedding, often focusing on avoiding perceptually sensitive regions or maintaining robustness, our approach introduces a novel, end-to-end learned mechanism. The success of Experiment Y-Prime (33 dB SNR, 0.0014 BER in clean conditions) for a 4-watermark system pushes the capabilities of FDW beyond static or heuristically determined embeddings. By achieving such

high imperceptibility and message recovery for multiple concurrent watermarks, our system demonstrates a superior ability to manage inter-watermark interference within the frequency domain compared to previous FDW methods designed primarily for single watermarks or simpler multi-bit messages.

6.5.3 Innovation in Distinctive Detection and PCA's Role

A core innovative aspect of this dissertation is the 'Distinctive Detection' enabled by the learnable basis vectors. Chapter 2 identified a significant gap in audio watermarking concerning dynamically layering distinct watermarks. Our approach directly addresses this. The empirical success of Experiment R (dual watermark) and Experiment Y-Prime (four watermarks) using learnable basis vectors demonstrated their superiority over simple summation (Experiment P) and even fixed PCA-derived bases (Experiment U). This finding contributes new knowledge by showing that an end-to-end learned system can implicitly discover optimal, highly separable embedding patterns for multiple watermarks in the frequency domain. This dynamic, learned separation provides a more powerful mechanism for distinctiveness than rigidly pre-computed transforms, fulfilling the conceptual role of PCA in an adaptive, data-driven manner. This moves beyond merely placing watermarks in different bands to actively learning how to make them distinct within those bands.

6.5.4 Comparison with State-of-the-Art (SOTA) Audio Watermarking

Comparing the performance of our multi-watermark system to existing SOTA single-watermark systems (e.g., AudioSeal by Roman et al. (Roman et al. 2024), WavMark by Chen et al. (Chen et al. 2024)) reveals both alignment and pioneering advancements:

- **Imperceptibility:** WavMark, a leading single-watermark system, boasts an SNR of 36.85 dB. Our Experiment Y-Prime achieves an SNR of 33.00 dB for four concurrent watermarks . This demonstrates that our system can maintain a highly comparable level of imperceptibility while providing four times the message capacity and multi-level traceability, a capability that SOTA single-watermark systems fundamentally lack.
- **Robustness:** WavMark achieves impressive clean-channel BER (e.g., 0.0048 for 32 bits). Our system's BER of 0.0014 for 32 bits (4 watermarks 8 bits/WM) not only matches but surpasses this level of clean-channel accuracy, proving the effectiveness of our distinctive detection for multi-level data.
- **Capacity and Traceability:** Crucially, the SOTA single-watermark systems do not offer a solution for embedding and tracking multiple, distinct, sequential messages. Our work directly addresses this by demonstrating a functional system for multi-level traceability, a capability that represents a unique contribution.

6.5.5 The DSSS Baseline in Context

The development of the custom DSSS algorithm provided an excellent clean-channel baseline (BER 0.000 at 15-20dB SNR). While highly effective in controlled conditions, DSSS fundamentally operates in the time domain using spread spectrum techniques. The decision to prioritize PFB, despite DSSS's strong clean performance, was strategic. PFB aligns with the thesis's core hypothesis of frequency-domain processing for distinctive, multi-level embedding. The PFB system's ability to achieve comparable, and in some aspects superior (e.g., clean BER for multiple watermarks, transparency via learned modulation), performance to DSSS while adhering to a more complex, frequency-based approach validates the pursuit of this novel direction.

6.6 Theoretical Implications

The empirical findings from this research carry several significant theoretical implications within the intersecting fields of digital audio watermarking, deep learning for signal processing, and information security.

- **Validation of Learned Distinctiveness for Multi-Level Embedding:** The robust performance of Experiment Y-Prime, particularly its near-perfect BER for four concurrent watermarks, theoretically validates the concept of 'learned distinctiveness' as a powerful mechanism for achieving

multi-level information embedding. This goes beyond traditional orthogonality or statistical independence by demonstrating that a deep neural network, through adversarial training, can discover and exploit subtle, high-dimensional patterns in the frequency domain to ensure message separability. This advances the theory of steganography by demonstrating how implicit, data-driven learning can achieve complex channel separation.

- **Expanding the Capacity Frontier of Perceptible Imperceptibility:** The achievement of 33 dB SNR with four embedded watermarks pushes the theoretical envelope for the 'imperceptibility-capacity' trade-off. It suggests that with sophisticated learned embedding strategies (e.g., learnable per-band modulation strengths), a higher watermark capacity can be achieved at perceptually acceptable levels than previously thought, especially when fine-tuned to psychoacoustic properties. This implies a new theoretical understanding of how the watermarking triad (imperceptibility, robustness, capacity) can be optimally navigated through deep learning.
- **The Efficacy of Hybrid Architecture for Complex Signal Tasks:** The success of combining a convolutional backbone for feature extraction with band-specific decoding heads in the detector provides a theoretical model for handling multi-channel or multi-component information retrieval in complex signal domains. This hybrid architecture, where a shared feature extractor feeds into specialized decoders, offers a blueprint for other signal processing tasks requiring the disentanglement of superimposed information.
- **Reaffirmation of Fundamental Signal Processing Challenges for Learned Systems:** The persistent vulnerability to aggressive frequency-destructive attacks (LPF) despite advanced learning highlights a theoretical boundary. It reaffirms that while deep learning can achieve remarkable robustness against noise and linear distortions, it does not bypass the fundamental physical properties of signal transformations. This underscores the need for theoretical models that integrate both learned resilience and an understanding of signal physics for truly robust real-world applications.
- **Toward a Theory of Self-Contained Digital Provenance:** By demonstrating effective multi-level embedding within the audio itself, this work moves towards a theoretical framework for 'self-contained digital provenance.' This contrasts with reliance on external ledgers or hashing, suggesting that the asset itself can carry its verifiable history, which has implications for theories of digital rights management and content integrity.

6.7 Practical Implications

The robust findings of this dissertation hold significant practical implications, offering tangible solutions and insights for real-world challenges in digital audio traceability and content security. The successful implementation within the Audio Watermark Lab application (Chapter 4) further accentuates these practical benefits.

- **Feasibility of In-Audio Multi-Level Traceability for Industry:** The demonstrated capability to embed and retrieve four distinct watermarks with high imperceptibility (33 dB SNR) and near-perfect accuracy (0.0014 BER) in clean conditions establishes the practical feasibility of dynamic, multi-level audio traceability. This directly benefits industries requiring granular provenance tracking, such as:
  - **Music Production Pipelines:** Enabling tracking of each stage (recording, mixing, mastering, distribution) by embedding a new watermark, ensuring proper attribution and royalty distribution for each contributor across the complex lifecycle.
  - **Forensic Analysis and Content Moderation:** Providing a robust tool to establish a verifiable digital chain of custody for audio evidence, or to identify the exact source of unauthorized content distribution and manipulation, particularly in contexts like deepfake detection or leak tracing.
  - **AI-Generated Content:** As AI-generated audio proliferates, this system offers a crucial mechanism for embedding verifiable metadata (e.g., creator, generation parameters) directly into the synthetic content, ensuring transparency and accountability.
- **Reduced Reliance on External Systems:** By offering a robust, in-audio solution, the research provides a practical alternative to current external traceability systems (e.g., blockchain, cloud-

hashing). This could lead to more efficient, less computationally intensive, and potentially more secure traceability, as the provenance is self-contained within the audio asset, mitigating complexities and vulnerabilities associated with third-party dependencies.

- **Actionable Insights for Robustness Engineering:** The identification of specific vulnerabilities in higher frequency bands under low-pass filtering provides direct, actionable insights for practitioners. This enables targeted strategies for future system development, such as dynamically avoiding critical watermark placement in vulnerable bands, employing adaptive embedding strengths, or integrating frequency-aware error correction, thereby guiding the design of more robust real-world watermarking solutions.
- **Guidance for Deep Learning-based Watermarking Development:** The detailed account of iterative development and architectural refinements, particularly the successes with learnable per-band modulation strengths and band-specific detector heads, offers practical blueprints for developers. This guidance can accelerate the creation of future deep learning-based watermarking systems across various multimedia types.
- **Tangible Research Output via Audio Watermark Lab:** The implementation within the Audio Watermark Lab application serves as a powerful practical demonstration. Its 'Research Mode' provides an invaluable interactive platform for further experimentation and analysis by other researchers, making the complex algorithms accessible. The 'Application Mode' demonstrates a user-friendly interface for direct use by content owners, bridging the gap between academic research and real-world utility. This tangible output enhances the practical impact and facilitates adoption.

6.8 Validation and Reliability

The rigor and trustworthiness of the research findings are underpinned by a systematic and multifaceted approach to validation and reliability, emphasizing the scientific merit of the entire undertaking.

- **Controlled and Iterative Experimental Design:** The research implemented a highly controlled and iterative experimental design, systematically evaluating approaches from SFA/SDA to the refined PFB system (Experiment Y-Prime). This iterative 'build-test-learn' cycle, where empirical insights from each phase directly informed and justified subsequent design refinements, fundamentally strengthens the validity of the current PFB approach as the most promising direction, demonstrating a robust engineering process.
- **Standardized and Representative Dataset:** The exclusive use of the Vox Populi dataset, a widely recognized benchmark developed by Facebook AI Research and used by SOTA models, ensures high reproducibility and direct comparability of results with existing literature. Its vastness and diversity contribute significantly to the generalizability and real-world relevance of the findings.
- **Rigorous Quantitative Metrics and Statistical Analysis:** Reliance on objective, quantitative metrics such as Signal-to-Noise Ratio (SNR), Mean Squared Error (MSE), and Bit Error Rate (BER) provides robust and unbiased measures of performance. The use of statistical methods in earlier comparative analyses (e.g., ANOVA, t-tests mentioned in Chapter 5) further supports the significance of observed differences.
- **Transparent Model Architectures and Training:** The detailed descriptions of the 'PFBGenerator' and 'PFBDetector' architectures, including refinements in Experiment Y-Prime (learnable modulation strengths, band-specific decoding heads), alongside the adversarial training scheme (Chapter 3), ensure high transparency. This level of detail, coupled with the use of publicly available frameworks (PyTorch) and the Google Colab environment, significantly contributes to the reproducibility of the research, allowing other researchers to replicate and verify the findings.
- **Multi-Faceted Validation: Quantitative, Qualitative, and Auditory:** The validation strategy extends beyond mere numbers. The use of the Audio Watermark Lab's 'Research Mode' facilitates live demonstrations, integrating quantitative metrics with visual waveform and spectrogram plots, and crucial auditory playback. This multi-faceted approach provides compelling evidence of imperceptibility and robustness, allowing for both statistical and perceptual confirmation of the system's efficacy.

- **Justification of Noise Models:** The selection of Gaussian noise as a primary distortion model is justified by its mathematical tractability, its status as a fundamental benchmark in signal processing, and its ability to approximate common additive noise. This controlled approach provides a rigorous baseline for assessing core noise resilience, and its inherent limitations for complex real-world attacks are openly acknowledged as a driver for future research.

6.9 Limitations and Methodological Reflections

Despite the significant advancements and rigorous methodology employed, it is imperative for a complete scientific account to acknowledge certain limitations and areas for methodological reflection that arose during the research process. These points do not diminish the achievements but rather provide a realistic context and guide for future work.

- **Performance under Aggressive Frequency-Destructive Attacks:** The primary current limitation is the reduced message recovery (higher BER) observed under aggressive frequency-destructive attacks, specifically low-pass filtering, particularly impacting watermarks in higher frequency bands. While this is a scientifically plausible outcome given the physical removal of spectral content, it represents a frontier for future robustness enhancements. The current system, while demonstrating groundbreaking performance in clean and noisy conditions, requires further refinement to achieve consistent near-perfect BER across all watermark bands under such severe, targeted distortions.
- **Scope of Adversarial Attack Spectrum in Current Full Evaluation:** While the full evaluation framework (Chapter 3) encompasses a broad range of distortions, the comprehensive Experiment Y-Prime evaluation primarily focused on Gaussian noise and low-pass filters. Future work needs to rigorously test the refined 4-watermark system against a wider array of real-world transformations, including various MP3 compression bitrates, different resampling rates, pitch shifting, time stretching, reverberation, and more complex, non-linear attacks (e.g., those simulating specific voice cloning or watermark-removal attempts), to fully ascertain its real-world resilience across all attack classes defined in Chapter 1.
- **Absence of Exhaustive AUC and PESQ Metrics in Developmental Stages:** While AUC and PESQ were identified as crucial metrics in Chapter 3's comprehensive evaluation framework, their exhaustive presentation in every developmental stage's results tables or interactive demonstrations was limited. This was a methodological choice driven by the iterative nature of the research, where core imperceptibility (SNR, MSE) and message recovery (BER) were the primary signals for rapid architectural and training optimization. AUC requires balanced datasets with negative samples, and PESQ requires specialized tools and listener panels, making them more suitable for a final, large-scale validation study rather than rapid prototyping cycles. Their full integration for a comprehensive final assessment remains a key area for future work.
- **Generalizability Beyond Current Dataset and Audio Types:** The primary training and evaluation utilized the Vox Populi dataset, comprising speech audio. While representative, the system's generalizability to other audio types (e.g., music, environmental sounds with different spectral characteristics) needs further empirical validation.
- **Computational Resource Considerations for Large-Scale Training:** While successful training was achieved on Google Colab's GPU, scaling to significantly larger datasets or more complex architectures in future work might necessitate more substantial computational resources. This is a common practical consideration in deep learning research.
- **Explicit PCA Layer vs. Learned Basis Reflection:** While the concept of PCA guided the development of 'distinctive detection' through learnable basis vectors, a direct empirical comparison with an explicitly integrated and trainable PCA layer within the end-to-end framework was not the focus of the final iterations. Future work could explore the nuanced benefits and trade-offs of such a direct integration.

These limitations are openly acknowledged and critically reflected upon. They serve not as detractions from the substantial achievements, but rather as concrete, empirically derived directives for the compelling and necessary next phases of research and development, guiding the project towards its ultimate vision of comprehensive, dynamic audio traceability.

6.10 Conclusion

This chapter has provided a critical and insightful discussion, thoroughly interpreting the empirical findings and robustly contextualizing them within the broader landscape of audio watermarking research. The journey has illuminated both the significant breakthroughs achieved—particularly the validation of a custom learned PFB system for single and dual watermarks, culminating in the exceptional performance of Experiment Y-Prime for four watermarks—and the inherent complexities encountered when scaling to higher watermark capacities. The achievement of 33 dB SNR and 0.0014 BER for a 4-watermark system in clean conditions represents a pioneering advancement in multi-level audio watermarking, addressing a critical gap in digital content traceability. While the evaluation transparently identified the remaining challenges under aggressive frequency-destructive attacks, these insights provide precise, actionable directions for future research. This chapter underscores that the research has successfully designed, implemented, and validated a highly innovative multi-level audio watermarking algorithm, moving significantly closer to realizing robust, in-audio content traceability. The iterative journey described, marked by rigorous empirical analysis, precise refinements, and compelling quantitative and qualitative results, serves as a testament to the scientific approach required to bridge theoretical advancements with practical applicability in the dynamic landscape of audio signal processing.

Chapter 7

Conclusion and Future Work

7.1 Introduction

This chapter serves as the culmination of the dissertation, providing a comprehensive synthesis of the research journey, its methodologies, and the significant findings. It aims to summarize the problem addressed, the approach undertaken, and the major accomplishments achieved, thereby offering a holistic perspective on the project's overall contribution. Beyond recounting the outcomes, this section critically evaluates the extent to which the initial research objectives were fulfilled, reflects upon the iterative development process, and delineates precise avenues for future work. By tying together insights from all preceding chapters, this conclusion articulates the practical and theoretical implications of a dynamic audio watermarking algorithm using PCA and distinctive detection, solidifying its place within the evolving landscape of digital content traceability.

7.2 Summary of the Project

This project embarked on the ambitious endeavor to address a critical gap in digital content security: the lack of robust, multi-level traceability for audio content in an era characterized by widespread manipulation, remixing, and unauthorized distribution. As articulated in Chapter 1, existing solutions, whether single-layer watermarking or external frameworks like blockchain and cloud-hashing, proved insufficient for providing a dynamic, in-audio chain of custody that could persist through diverse distortions. The research was guided by a set of clear objectives: to evaluate the limitations of current state-of-the-art (SOTA) watermarking algorithms for incremental embedding, to develop a novel frequency-domain algorithm inspired by Principal Component Analysis (PCA) for distinctive watermark separation, and to rigorously assess its multi-level traceability in terms of imperceptibility and robustness.

The approach unfolded as a meticulous, iterative development process, beginning with an empirical assessment of SOTA sequential methods (SFA, SDA) and early parallel frequency band (PFB) attempts, which unequivocally demonstrated their shortcomings for multi-level traceability (Chapter 5, Phase 1). This foundational understanding necessitated the development of custom algorithms. A Direct Sequence Spread Spectrum (DSSS) method was engineered as a robust clean-channel baseline, confirming the general feasibility of multi-level embedding (Chapter 5, Phase 2). The core of the project, however, focused on the iterative design and refinement of a custom, end-to-end learned Parallel Frequency Band (PFB) system (Chapter 3). This system, central to the thesis's hypothesis, aimed to achieve distinctive watermark embedding in the frequency domain.

Major accomplishments include the breakthrough performance of the learned PFB system, which, through successive refinements, achieved remarkable imperceptibility and robustness for single and dual watermarks (Chapter 5, Phase 3). The culmination of this development was Experiment Y-Prime, a refined 4-watermark PFB system, which demonstrated an outstanding 33 dB SNR and a 0.0014 BER in clean conditions. This signifies that the system can embed four distinct watermarks into an audio file with near-perfect message recovery and excellent auditory transparency. The entire development, testing, and visualization process was facilitated and demonstrated through the **Audio Watermark Lab application** (Chapter 4), a tangible output that bridges complex machine learning models with practical user

interaction and comprehensive analysis capabilities.

7.3 Key Findings and Contributions

The project has yielded several pivotal findings and made distinct contributions to the existing body of knowledge and practice in digital audio watermarking and content traceability:

- **Achievability of High-Capacity, High-Fidelity Multi-Level Audio Watermarking:** A primary contribution is the empirical demonstration that a dynamic audio watermarking algorithm can successfully embed and retrieve four distinct watermarks concurrently within a single audio file with high imperceptibility (33 dB SNR) and near-perfect robustness (0.0014 BER) in clean and moderately noisy environments. This addresses a critical gap in current SOTA solutions, which largely focus on single-layer watermarking (Chapter 2), and directly contributes to solving the problem of scalable in-audio traceability.
- **Validation of Learned Distinctiveness for Interference Management:** The research provides strong evidence that interference, a major hurdle in multi-watermark systems (as seen in SFA/SDA failures), can be effectively managed through learned mechanisms. The innovative use of learnable basis vectors (inspired by PCA) within the 'PFBGenerator' and band-specific decoding heads in the 'PFBDetector' proved highly effective in partitioning the frequency modification space and enabling distinctive detection for each watermark. This validates a novel approach to achieving effective 'orthogonality' in a learned system.
- **Iterative, Empirically-Driven Architectural Evolution:** The dissertation showcases a rigorous, iterative development methodology where empirical results (e.g., challenges in Experiment X) directly informed architectural refinements (Experiment Y-Prime). This process highlights the practical value of a continuous feedback loop between theoretical design and experimental validation in complex machine learning engineering.
- **Identification of Specific Robustness Frontiers:** The project transparently identifies specific limitations under aggressive, frequency-destructive attacks like low-pass filtering. By detailing the higher BER in specific frequency bands under these conditions, the research contributes precise, actionable insights into where future robustness enhancements are most critically needed in frequency-domain watermarking, guiding the field towards more resilient solutions.
- **Development of a Tangible, Comprehensive Research and Application Platform:** The creation of the Audio Watermark Lab application (Chapter 4) is a significant practical contribution. It serves as both an interactive research tool for detailed analysis and a user-friendly demonstration platform, bridging the gap between theoretical models and real-world utility, and showcasing strong software engineering capabilities alongside machine learning expertise.

7.4 Evaluation of Objectives

The project successfully addressed all research objectives articulated in Chapter 1, albeit with certain aspects evolving in their implementation based on empirical findings.

- **Objective 1: Evaluate the traceability of state-of-the-art watermarking algorithms against incremental watermarking. Achievement:** This objective was comprehensively achieved in Phase 1 of the research (Chapter 5, Section 5.1). The rigorous evaluation of SFA, SDA, and WavMark methods provided strong empirical evidence of their fundamental limitations for multi-level incremental watermarking, unequivocally demonstrating the necessity for a novel solution.
- **Objective 2: Develop a PCA training layer which will facilitate dimensionality reduction, optimal band selection and efficient partitioning of watermarks. Achievement:** This objective was substantially achieved, though its realization evolved from the initial blueprint. While a literal, algebraic PCA layer was not integrated, the functionality of PCA—achieving distinctive feature separation and efficient partitioning of watermarks—was successfully realized through the 'PFBGenerator''s architecture. The use of learnable basis vectors within the 'modification_projectors' enabled the network to dynamically discover optimal embedding patterns for

each watermark, effectively partitioning the frequency modification space for 'distinctive detection,' proving empirically more effective (Chapter 5, Section 5.3.2).

- **Objective 3: Develop and train a feature extraction detector that will detect the distinct watermarks from a single audio. Achievement:** This objective was successfully met with the development of the 'PFBDetector'. Through iterative refinement, culminating in Experiment Y-Prime, the detector achieved near-perfect accuracy (0.0014 BER) in extracting four distinct watermark messages from watermarked audio, even in moderately noisy conditions. The specialized band-specific decoding heads proved crucial for its robust performance (Chapter 5, Section 5.3.4).
- **Objective 4: Assess the proposed system's incremental traceability using the benchmarks from Objective 1 to validate the effectiveness of the new approach. Achievement:** This objective was largely achieved. The extensive evaluation of the refined PFB system (Experiment Y-Prime) against multiple watermarks and various attacks (Chapter 5, Section 5.3.4) demonstrated its strong capacity for incremental traceability. The system achieved a breakthrough in imperceptibility (33 dB SNR) and robustness (0.0014 BER) for four watermarks. While limitations under aggressive low-pass filtering were identified, these serve as precisely defined areas for future research rather than a failure to assess effectiveness. The project successfully validated the overall efficacy of the new approach in achieving multi-level traceability within defined boundaries.

The deviations from the initial blueprint, particularly regarding the literal implementation of a PCA layer and certain embedding/detection mechanisms, represent a successful adaptation of the methodology. These changes were data-driven, reflecting a pragmatic shift towards more effective, end-to-end trainable deep learning solutions that proved empirically superior in achieving the research objectives.

7.5 Reflection on the Project Process

The overall project process was a testament to the power of iterative design and empirical validation in addressing complex engineering challenges. A primary strength of the chosen methodology was its **adaptive nature**, allowing for continuous refinement of the algorithm based on insights gained from each experimental phase. The early, rigorously conducted evaluations (SFA, SDA, early PFB attempts) were not just results but critical diagnostic tools that immediately highlighted the shortcomings of simpler approaches and precisely informed the architectural and training decisions for the custom learned PFB system. This avoided prolonged investment in non-viable directions.

The seamless integration of deep learning frameworks (PyTorch) with digital signal processing techniques, facilitated by the Google Colab environment, enabled a rapid 'build-test-learn' cycle. This allowed for extensive experimentation and hyperparameter tuning over numerous epochs, even within the constraints of available computational resources. The tangible output, the Audio Watermark Lab application, proved invaluable for this process, providing immediate visual and auditory feedback on watermark imperceptibility and robustness, complementing the quantitative metrics. Its 'Research Mode' specifically empowered the agile prototyping and detailed analysis that characterized this development.

However, the project also presented inherent challenges. The fundamental trade-offs within the watermarking triad (imperceptibility, robustness, capacity) were a continuous point of optimization. Achieving a high capacity (four watermarks) while simultaneously maximizing imperceptibility and robustness required numerous iterations of refining loss function weights and model architectures. The persistence of vulnerabilities to highly destructive, physical signal transformations (like aggressive low-pass filtering) underscored the limitations of even advanced deep learning against fundamental signal physics. Lessons learned include the paramount importance of synergistic generator-detector design, the critical role of fine-grained control over embedding strength (as realized by learnable per-band modulation), and the value of targeted architectural enhancements (e.g., band-specific detector heads) for multi-channel information processing. This project reaffirmed that while deep learning offers immense power, it must be strategically combined with a deep understanding of the underlying signal processing domain to yield effective solutions.

7.6 Future Work and Recommendations

Building upon the robust foundation established by this dissertation, particularly the exceptional performance of Experiment Y-Prime, several promising avenues for future work and enhancements are identified. These recommendations aim to further strengthen the algorithm, expand its capabilities, and explore its application in broader contexts.

- **Enhanced Robustness to Aggressive Frequency-Destructive Attacks:** This remains the most critical frontier. Future research should prioritize developing strategies to improve message recovery under severe low-pass filtering, MP3 compression at low bitrates, and other frequency-altering attacks (e.g., resampling, pitch shifting). This could involve:
  - **Adaptive Embedding Strategies:** Dynamically adjusting watermark strength and coding across bands based on their perceptual robustness to specific attacks, potentially leveraging redundancy or error correction coding in more resilient frequency regions.
  - **Attack-Aware Training:** Incorporating a wider variety and intensity of real-world attacks directly into the training loop, forcing the generator-detector pair to learn to embed and extract robustly in highly degraded environments.
  - **Perceptual Masking Integration:** Exploring more sophisticated psychoacoustic models during embedding to place watermark energy optimally in areas masked by the original audio, thereby increasing resilience against common distortions while maintaining imperceptibility.
- **Increased Message Capacity and Flexibility:** Investigate the scalability of the PFB system beyond four watermarks, exploring higher capacities for more granular traceability data. This could also involve developing support for variable message lengths or embedding more complex data structures beyond simple binary messages.
- **Generalisability Across Diverse Audio Types:** While trained on speech (Vox Populi), the system's performance on other audio types (e.g., music across different genres, environmental sounds) needs comprehensive validation. Future work should involve training and testing on more diverse audio datasets to ensure broad applicability.
- **Perceptual Quality Metrics Integration:** While SNR and MSE provide objective measures, future work should more extensively incorporate Perceptual Evaluation of Speech Quality (PESQ) and potentially Objective Difference Grade (ODG) or Virtual Speech Quality Objective Listener (ViSQOL) into both training objectives (if computationally feasible) and final evaluation, providing a more human-centric assessment of imperceptibility.
- **Advanced Security Analysis and Counter-Forensics:** Building on the foundational robustness, future work should include rigorous analysis against more sophisticated, targeted attacks (e.g., intelligent watermark removal algorithms, forgery attempts, re-embedding attacks by an adversary). This would involve exploring the use of cryptographic principles for watermark generation and more complex adversarial training schemes.
- **Software System Enhancements and Deployment Exploration:**
  - **Expanded Audio Watermark Lab Features:** Enhance the 'Research Mode' with more interactive visualization tools (e.g., real-time spectral analysis of watermark components, ROC curves for detection performance) and allow for greater configurability of custom attacks.
  - **API Integration:** Explore the development of a robust API for the watermarking service, allowing seamless integration into larger content management or production pipelines.
- **Application in AI-Generated Content Lifecycle Management:** Given the rise of synthetic media, this system's ability to embed and track provenance within AI-generated audio is a promising area. Future work could focus on specific applications in detecting AI-generated content, tracking its origin, or managing its usage rights.

7.7 Conclusion

This dissertation culminates the exploration and development of a novel dynamic audio watermarking algorithm, a significant step forward in the pursuit of robust, multi-level content traceability. The

research successfully designed, implemented, and validated a learned Parallel Frequency Band (PFB) system, demonstrating its exceptional capacity to embed and retrieve four distinct watermarks concurrently with high imperceptibility and near-perfect accuracy in clean and moderately noisy environments. This achievement, evidenced by an outstanding **33 dB SNR and 0.0014 BER for a 4-watermark system in clean conditions**, represents a pioneering advancement that directly addresses critical gaps in digital content forensics and intellectual property protection, as initially articulated in Chapter 1.

The iterative methodological journey, marked by rigorous empirical analysis and precise architectural refinements, including the innovative use of learnable per-band modulation strengths and specialized detector heads, provided deep insights into the complex interplay of imperceptibility, robustness, and capacity. While the system's resilience under aggressive frequency-destructive attacks was identified as a clear frontier for future work, this transparent acknowledgment contributes valuable, actionable knowledge to the field.

Ultimately, this dissertation has not only delivered a highly innovative and demonstrable solution through the Audio Watermark Lab application but has also illuminated a clear path forward for future research in advanced audio watermarking. It stands as a testament to the scientific approach required to bridge theoretical advancements with practical applicability, moving significantly closer to realizing a truly comprehensive and dynamic in-audio traceability system in an ever-evolving digital soundscape.

Bibliography

Begum, Mahbuba and Mohammad Shorif Uddin (Feb. 26, 2021). "Multiple Image Watermarking with Discrete Cosine Transform". In: *Journal of Computer and Communications* 9.3. Number: 3 Publisher: Scientific Research Publishing, pp. 88–94. DOI: 10.4236/jcc.2021.93006. URL: https://www.scirp.org/journal/paperinformation?paperid=108044 (visited on 12/08/2024).

Burka, Zak (n.d.). "Perceptual audio classification using principal component analysis". In: ().

Chen, Guangyu et al. (Jan. 7, 2024). *WavMark: Watermarking for Audio Generation*. DOI: 10.48550/arXiv.2308.12770. arXiv: 2308.12770[cs]. URL: http://arxiv.org/abs/2308.12770 (visited on 12/08/2024).

Julius, Olaniyan et al. (Apr. 5, 2023). "Implementation of Audio Signals Denoising for Perfect Speech-to-Speech Translation Using Principal Component Analysis". In: *2023 International Conference on Science, Engineering and Business for Sustainable Development Goals (SEB-SDG)*. Conference Name: 2023 International Conference on Science, Engineering and Business for Sustainable Development Goals (SEB-SDG) ISBN: 9798350324785 Place: Omu-Aran, Nigeria Publisher: IEEE, pp. 1–6. DOI: 10.1109/SEB-SDG57117.2023.10124385. URL: https://ieeexplore.ieee.org/document/10124385/ (visited on 12/08/2024).

Li, Feng and Hao Chang (Jan. 28, 2021). "Monaural Singing Voice Separation Using Robust Principal Component Analysis with Weighted Values". In: *International Journal of Circuits, Systems and Signal Processing* 15, pp. 40–45. ISSN: 1998-4464. DOI: 10.46300/9106.2021.15.5. URL: https://www.naun.org/main/NAUN/circuitssystemssignal/2021/a102005-005(2021).pdf (visited on 12/08/2024).

Liu, Chang et al. (Dec. 6, 2023). *Detecting Voice Cloning Attacks via Timbre Watermarking*. DOI: 10.48550/arXiv.2312.03410. arXiv: 2312.03410[cs]. URL: http://arxiv.org/abs/2312.03410 (visited on 12/08/2024).

Liu, Hongbin et al. (Nov. 13, 2024). *AudioMarkBench: Benchmarking Robustness of Audio Watermarking*. DOI: 10.48550/arXiv.2406.06979. arXiv: 2406.06979[cs]. URL: http://arxiv.org/abs/2406.06979 (visited on 12/08/2024).

Nadeau, Andrew and Gaurav Sharma (June 2017). "An Audio Watermark Designed for Efficient and Robust Resynchronization After Analog Playback". In: *IEEE Transactions on Information Forensics and Security* 12.6, pp. 1393–1405. ISSN: 1556-6013, 1556-6021. DOI: 10.1109/TIFS.2017.2661724. URL: http://ieeexplore.ieee.org/document/7837611/ (visited on 12/08/2024).

Natgunanathan, Iynkaran et al. (Aug. 31, 2022). "Blockchain-Based Audio Watermarking Technique for Multimedia Copyright Protection in Distribution Networks". In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 18.3, pp. 1–23. ISSN: 1551-6857, 1551-6865. DOI: 10.1145/3492803. URL: https://dl.acm.org/doi/10.1145/3492803 (visited on 12/08/2024).

Papadopoulos, Hélène and Daniel P W Ellis (2014). "MUSIC-CONTENT-ADAPTIVE ROBUST PRINCIPAL COMPONENT ANALYSIS FOR A SEMANTICALLY CONSISTENT SEPARATION OF FOREGROUND AND BACKGROUND IN MUSIC AUDIO SIGNALS". In.

Renza, Diego, Jaime Andres Arango, and Dora Maria Ballesteros (July 31, 2019). "A Mobile-Oriented System for Integrity Preserving in Audio Forensics". In: *Applied Sciences* 9.15, p. 3097. ISSN: 2076-3417. DOI: 10.3390/app9153097. URL: https://www.mdpi.com/2076-3417/9/15/3097 (visited on 12/08/2024).

Roman, Robin San et al. (June 6, 2024). *Proactive Detection of Voice Cloning with Localized Watermarking*. DOI: 10.48550/arXiv.2401.17264. arXiv: 2401.17264[cs]. URL: http://arxiv.org/abs/2401.17264 (visited on 12/08/2024).

Singh, Jyotsna, Parul Garg, and Alok Nath De (2013). "Multiplicative Watermarking of Audio in Spectral Domain". In: *Quality, Reliability, Security and Robustness in Heterogeneous Networks*. Ed. by Karan

Singh and Amit K. Awasthi. Berlin, Heidelberg: Springer, pp. 591–605. ISBN: 9783642379499. DOI: 10.1007/978-3-642-37949-9_52.

Uddin, Mohammad Shorif et al. (2024). "Audio Watermarking: A Comprehensive Review". In: *International Journal of Advanced Computer Science and Applications* 15.5. ISSN: 21565570, 2158107X. DOI: 10.14569/IJACSA.2024.01505141. URL: http://thesai.org/Publications/ViewPaper?Volume=15&Issue=5&Code=ijacsa&SerialNo=141 (visited on 12/08/2024).

Wang, Changhan et al. (July 27, 2021). *VoxPopuli: A Large-Scale Multilingual Speech Corpus for Representation Learning, Semi-Supervised Learning and Interpretation*. DOI: 10.48550/arXiv.2101.00390. arXiv: 2101.00390[cs]. URL: http://arxiv.org/abs/2101.00390 (visited on 12/08/2024).

Wu, Qiuling, Runyu Ding, and Jiangchun Wei (Sept. 5, 2022). "Audio Watermarking Algorithm with a Synchronization Mechanism Based on Spectrum Distribution". In: *Security and Communication Networks* 2022. Ed. by Rutvij Jhaveri, pp. 1–13. ISSN: 1939-0122, 1939-0114. DOI: 10.1155/2022/2617107. URL: https://www.hindawi.com/journals/scn/2022/2617107/ (visited on 12/08/2024).

Xiao, Jianmao et al. (2023). "MBE: A Music Copyright Depository Framework Incorporating Blockchain and Edge Computing". In: *Computer Systems Science and Engineering* 47.3, pp. 2815–2834. ISSN: 0267-6192. DOI: 10.32604/csse.2023.039716. URL: https://www.techscience.com/csse/v47n3/54574 (visited on 12/08/2024).

Zhang, Haijian (Feb. 8, 2020). *A Time-Frequency Perspective on Audio Watermarking*. DOI: 10.48550/arXiv.2002.03156. arXiv: 2002.03156[cs]. URL: http://arxiv.org/abs/2002.03156 (visited on 12/08/2024).

Appendix A

Code and Supplementary Materials

This appendix provides essential code excerpts and supplementary materials that underpin the "Dynamic Audio Watermarking Algorithm using PCA and Distinctive Detectio" dissertation. It aims to enhance the reproducibility and transparency of the research, showcasing key components of the implemented system and the environment used for its development and evaluation. For the complete, runnable codebase, including all frontend, backend, and auxiliary scripts, please refer to the project's GitHub repository: https://github.com/mayibongwemoyo/audio-watermark-lab.

A.1 Core Watermarking Algorithms (`pca_prime_watermarking.py` excerpts)

This section presents the foundational Python code for the `PFBGenerator` and `PFBDetector` models, along with the custom incremental embedding and detection logic. These components form the algorithmic core developed in this research.

A.1.1 PFBGenerator Architecture

The `PFBGenerator` class defines the neural network architecture responsible for embedding watermarks. Its design incorporates learnable modulation strengths per frequency band and band-specific modification projectors, which are crucial for dynamically adapting watermark insertion to maintain imperceptibility and robustness.

```
from app import save_audio
import torch
import torch.nn as nn
import torchaudio
import torchaudio.transforms as T
import torch.nn.functional as torch_F # For padding
import random
import math
import os
import numpy as np
import uuid
from scipy import stats


# --- Model Class Definitions (Exactly from Experiment Y-Prime Colab) ---
class PFBGenerator(nn.Module):
    def __init__(self, n_fft=N_FFT, num_freq_bands=NUM_FREQ_BANDS, n_bits=N_BITS,
                 message_embedding_dim=16, initial_modulation_strength=0.3):
        super().__init__()
        self.n_fft = n_fft
        self.num_freq_bands = num_freq_bands
        self.n_bits = n_bits
        self.message_embedding_dim = message_embedding_dim
        num_freq_bins_total = self.n_fft // 2 + 1
        self.nominal_bins_per_band = num_freq_bins_total // self.num_freq_bands
        self.message_embedding = nn.Embedding(2, self.message_embedding_dim)
```

```
        initial_strength_param = math.log(initial_modulation_strength / (1 -
    initial_modulation_strength))
        self.learnable_modulation_strengths_raw = nn.Parameter(
            torch.full((self.num_freq_bands,), initial_strength_param, device=DEVICE)
        )
        self.modification_projectors = nn.ModuleList([
            nn.Sequential(
                nn.Linear(self.n_bits * self.message_embedding_dim, 128),
                nn.ReLU(),
                nn.Linear(128, self.nominal_bins_per_band),
                nn.Tanh()
            ) for _ in range(self.num_freq_bands)
        ])

    def forward(self, audio_segment_batch, message_batches: list):
        if len(message_batches) != self.num_freq_bands:
            raise ValueError(f"Expected {self.num_freq_bands} message batches, got {len(
    message_batches)}")
        if audio_segment_batch.dim() == 2:
            audio_segment_batch = audio_segment_batch.unsqueeze(1)
        audio_segment_batch_device = audio_segment_batch.to(DEVICE)
        spectrogram_complex = stft_transform(audio_segment_batch_device.squeeze(1))
        magnitude = spectrogram_complex.abs()
        angle = spectrogram_complex.angle()
        modified_band_magnitudes = []
        num_freq_bins_total_runtime = magnitude.shape[1]
        modulation_strengths = torch.sigmoid(self.learnable_modulation_strengths_raw)

        for i in range(self.num_freq_bands):
            message_batch = message_batches[i]
            embedded_message_flat = self.message_embedding(message_batch.long()).view(
    message_batch.size(0), -1)
            projected_deltas_raw = self.modification_projectors[i](embedded_message_flat
    )
            start_bin = i * self.nominal_bins_per_band
            if i == self.num_freq_bands - 1:
                end_bin = num_freq_bins_total_runtime
            else:
                end_bin = start_bin + self.nominal_bins_per_band
            current_band_actual_size = end_bin - start_bin
            num_deltas_to_apply = min(projected_deltas_raw.shape[1],
    current_band_actual_size)
            full_deltas_for_band_slice = torch.zeros(message_batch.size(0),
    current_band_actual_size, device=DEVICE)
            full_deltas_for_band_slice[:, :num_deltas_to_apply] = projected_deltas_raw
    [:, :num_deltas_to_apply]

            max_clamp_val = 1.0 + modulation_strengths[i].item() # .item() converts
    scalar tensor to Python float
            scaling_factors = (1.0 + full_deltas_for_band_slice * modulation_strengths[i
    ]).unsqueeze(-1)
            scaling_factors = torch.clamp(scaling_factors, min=0.01, max=max_clamp_val)

            original_band_slice = magnitude[:, start_bin:end_bin, :]
            modified_slice = original_band_slice * scaling_factors
            modified_band_magnitudes.append(modified_slice)

        modified_magnitude = torch.cat(modified_band_magnitudes, dim=1)
        watermarked_spectrogram_complex = torch.polar(modified_magnitude, angle)
        watermarked_audio_segment = istft_transform(watermarked_spectrogram_complex,
    length=audio_segment_batch_device.shape[-1])
        if watermarked_audio_segment.dim() == 2:
            watermarked_audio_segment = watermarked_audio_segment.unsqueeze(1)
        return watermarked_audio_segment
```

Listing A.1: PFBGenerator Class Definition

A.1.2 PFBDetector Architecture

The `PFBDetector` class outlines the neural network architecture designed for extracting embedded watermarks. It features an enhanced convolutional backbone and band-specific decoding heads, enabling

robust and distinctive detection of multiple watermarks from magnitude differences.

```
from app import save_audio
import torch
import torch.nn as nn
import torchaudio
import torchaudio.transforms as T
import torch.nn.functional as torch_F # For padding
import random
import math
import os
import numpy as np
import uuid
from scipy import stats

class PFBDetector(nn.Module):
    def __init__(self, n_fft=N_FFT, num_freq_bands=NUM_FREQ_BANDS, n_bits=N_BITS,
                 num_time_frames=NUM_TIME_FRAMES_SEGMENT,
                 fc_hidden_size=DETECTOR_FC_HIDDEN_SIZE):
        super().__init__()
        self.n_fft = n_fft
        self.num_freq_bands = num_freq_bands
        self.n_bits = n_bits
        self.num_time_frames = num_time_frames
        self.fc_hidden_size = fc_hidden_size
        num_freq_bins_total = self.n_fft // 2 + 1
        self.freq_bins_per_band_input_shape = num_freq_bins_total // self.num_freq_bands

        self.conv_net = nn.Sequential(
            nn.Conv2d(1, 16, (3, 5), stride=(1, 2), padding=(1, 2)),
            nn.InstanceNorm2d(16),
            nn.ReLU(),
            nn.MaxPool2d((2, 2)),
            nn.Conv2d(16, 32, (3, 3), stride=(1, 1), padding=(1, 1)),
            nn.InstanceNorm2d(32),
            nn.ReLU(),
            nn.MaxPool2d((2, 2)),
            nn.Conv2d(32, 64, (3, 3), stride=(1, 1), padding=(1, 1)),
            nn.InstanceNorm2d(64),
            nn.ReLU(),
            nn.MaxPool2d((2, 2))
        )

        with torch.no_grad():
            dummy_input = torch.randn(1, 1, self.freq_bins_per_band_input_shape, self.
    num_time_frames)
            dummy_out = self.conv_net(dummy_input)
            self.conv_output_size = dummy_out.view(1, -1).size(1)

        self.fc_nets = nn.ModuleList([
            nn.Sequential(
                nn.Linear(self.conv_output_size, self.fc_hidden_size),
                nn.ReLU(),
                nn.Dropout(0.2),
                nn.Linear(self.fc_hidden_size, self.n_bits)
            ) for _ in range(self.num_freq_bands)
        ])

    def _process_band(self, magnitude_diff_slice, band_idx):
        current_band_freq_size_actual = magnitude_diff_slice.shape[1]
        if current_band_freq_size_actual < self.freq_bins_per_band_input_shape:
            processed_band_slice = torch_F.pad(magnitude_diff_slice, (0, 0, 0, self.
    freq_bins_per_band_input_shape - current_band_freq_size_actual))
        elif current_band_freq_size_actual > self.freq_bins_per_band_input_shape:
            processed_band_slice = magnitude_diff_slice[:, :self.
    freq_bins_per_band_input_shape, :]
        else:
            processed_band_slice = magnitude_diff_slice
        current_band_time_size_actual = processed_band_slice.shape[2]
        if current_band_time_size_actual < self.num_time_frames:
            processed_band_slice = torch_F.pad(processed_band_slice, (0, self.
    num_time_frames - current_band_time_size_actual, 0, 0))
        elif current_band_time_size_actual > self.num_time_frames:
            processed_band_slice = processed_band_slice[:, :, :self.num_time_frames]
```

```
        conv_input = processed_band_slice.unsqueeze(1)
        conv_output = self.conv_net(conv_input)
        conv_output_flat = conv_output.view(conv_output.size(0), -1)
        band_logits = self.fc_nets[band_idx](conv_output_flat)
        return band_logits

    def forward(self, original_audio_segment_batch, watermarked_audio_segment_batch):
        if original_audio_segment_batch.dim() == 2:
            original_audio_segment_batch = original_audio_segment_batch.unsqueeze(0)
        if watermarked_audio_segment_batch.dim() == 2:
            watermarked_audio_segment_batch = watermarked_audio_segment_batch.unsqueeze
    (0)

        original_audio_device = original_audio_segment_batch.to(DEVICE)
        watermarked_audio_device = watermarked_audio_segment_batch.to(DEVICE)

        spec_complex_orig = stft_transform(original_audio_device.squeeze(1))
        spec_complex_wm = stft_transform(watermarked_audio_device.squeeze(1))

        magnitude_orig = spec_complex_orig.abs()
        magnitude_wm = spec_complex_wm.abs()

        magnitude_diff = magnitude_wm - magnitude_orig

        num_freq_bins_total_runtime = magnitude_diff.shape[1]
        bins_per_band_runtime_nominal = num_freq_bins_total_runtime // self.
    num_freq_bands

        all_logits = []
        for i in range(self.num_freq_bands):
            start_bin = i * bins_per_band_runtime_nominal
            if i == self.num_freq_bands - 1:
                end_bin = num_freq_bins_total_runtime
            else:
                end_bin = start_bin + bins_per_band_runtime_nominal

            target_band_diff_slice = magnitude_diff[:, start_bin:end_bin, :]
            band_logits = self._process_band(target_band_diff_slice, i)
            all_logits.append(band_logits)

        return all_logits
```

Listing A.2: PFBDetector Class Definition

A.1.3 Incremental Embedding Logic

The `embed_single_incremental_pca_prime_watermark_step` function implements the core incremental embedding process. It calculates the specific modification introduced by a new watermark (relative to the original clean audio) and applies this delta to the current audio signal, thereby cumulatively embedding watermarks across multiple stages.

```
# Assumed constants: SAMPLE_RATE, N_FFT, HOP_LENGTH, NUM_FREQ_BANDS, N_BITS,
    SEGMENT_DURATION_S, NUM_SAMPLES_SEGMENT, DEVICE
# Assumed helper functions: get_models, calculate_snr, calculate_mse, stft_transform,
    istft_transform
def embed_single_incremental_pca_prime_watermark_step(
    original_full_audio_tensor: torch.Tensor,
    current_audio_tensor_from_prev_step: torch.Tensor,
    message_for_this_band_str: str,
    target_band_idx: int
) -> tuple[torch.Tensor, str, dict]:
    generator, _ = get_models()
    generator.eval()

    if not (0 <= target_band_idx < NUM_FREQ_BANDS):
        raise ValueError(f"target_band_idx must be between 0 and {NUM_FREQ_BANDS - 1},
    got {target_band_idx}")
    if len(message_for_this_band_str) != N_BITS:
        raise ValueError(f"Message for single band must be {N_BITS} bits, got {len(
    message_for_this_band_str)}")
```

```
16
17      # Ensure inputs are (1, C, T)
18      if original_full_audio_tensor.dim() == 2: original_full_audio_tensor =
        original_full_audio_tensor.unsqueeze(0)
19      if current_audio_tensor_from_prev_step.dim() == 2:
        current_audio_tensor_from_prev_step = current_audio_tensor_from_prev_step.unsqueeze
        (0)
20
21      total_samples = original_full_audio_tensor.shape[-1]
22      num_chunks = math.ceil(total_samples / NUM_SAMPLES_SEGMENT)
23
24      newly_watermarked_chunks_list = []
25
26      with torch.no_grad():
27          for chunk_idx in range(num_chunks):
28              start_sample = chunk_idx * NUM_SAMPLES_SEGMENT
29              end_sample = min((chunk_idx + 1) * NUM_SAMPLES_SEGMENT, total_samples)
30
31              original_chunk = original_full_audio_tensor[:, :, start_sample:end_sample]
32              current_chunk_from_prev_step = current_audio_tensor_from_prev_step[:, :,
        start_sample:end_sample]
33
34              # Pad chunks if necessary
35              if original_chunk.shape[-1] < NUM_SAMPLES_SEGMENT:
36                  padding_needed = NUM_SAMPLES_SEGMENT - original_chunk.shape[-1]
37                  original_chunk = torch_F.pad(original_chunk, (0, padding_needed))
38                  current_chunk_from_prev_step = torch_F.pad(current_chunk_from_prev_step,
         (0, padding_needed))
39
40              # Prepare messages for the generator to embed ONLY the target_band_idx.
41              messages_for_generator = []
42              for i in range(NUM_FREQ_BANDS):
43                  if i == target_band_idx:
44                      messages_for_generator.append(torch.tensor([[int(b) for b in
        message_for_this_band_str]], dtype=torch.float32, device=DEVICE))
45                  else:
46                      messages_for_generator.append(torch.zeros(1, N_BITS, dtype=torch.
        float32, device=DEVICE))
47
48              watermarked_only_this_band_from_original = generator(original_chunk,
        messages_for_generator)
49
50              delta_introduced = watermarked_only_this_band_from_original - original_chunk
51
52              new_watermarked_chunk = current_chunk_from_prev_step + delta_introduced
53
54              newly_watermarked_chunks_list.append(new_watermarked_chunk.squeeze(0))
55
56      newly_watermarked_full_audio_padded = torch.cat(newly_watermarked_chunks_list, dim
        =-1)
57
58      # Trim padding and ensure valid range for output
59      newly_watermarked_full_audio_trimmed = newly_watermarked_full_audio_padded[:, :
        total_samples]
60
61      # --- CRITICAL FIXES FOR AUDIO OUTPUT ---
62      # 1. Clamp values to ensure they are strictly within [-1, 1]
63      newly_watermarked_full_audio_trimmed = torch.clamp(
        newly_watermarked_full_audio_trimmed, min=-1.0, max=1.0)
64
65      # 2. Convert to float32 explicitly, if it's not already, as this is standard for
        audio samples.
66      newly_watermarked_full_audio_trimmed = newly_watermarked_full_audio_trimmed.to(torch
        .float32)
67
68      # 3. Ensure the tensor has exactly 2 dimensions (channels, samples) for torchaudio.
        save.
69      #    It should be (1, T) after original_audio_tensor.unsqueeze(0) and .squeeze(0) in
         loop.
70      if newly_watermarked_full_audio_trimmed.dim() == 3:
71          newly_watermarked_full_audio_trimmed = newly_watermarked_full_audio_trimmed.
        squeeze(0)
72      elif newly_watermarked_full_audio_trimmed.dim() == 1:
```

```
        newly_watermarked_full_audio_trimmed = newly_watermarked_full_audio_trimmed.unsqueeze(0)
    # This ensures it's (1, T) for mono audio saving.
    # --- END CRITICAL FIXES ---

    # Check for NaNs/Infs (debugging aid)
    if torch.isnan(newly_watermarked_full_audio_trimmed).any() or torch.isinf(newly_watermarked_full_audio_trimmed).any():
        print("[PCA_PRIME_WATERMARKING] WARNING: NaN or Inf found in newly watermarked audio tensor after incremental embed!")
        newly_watermarked_full_audio_trimmed = torch.nan_to_num(newly_watermarked_full_audio_trimmed, nan=0.0, posinf=1.0, neginf=-1.0)
        # Re-clamp after nan_to_num as it might put values slightly outside [-1,1]
        newly_watermarked_full_audio_trimmed = torch.clamp(newly_watermarked_full_audio_trimmed, min=-1.0, max=1.0)

    print("Saving audio: shape", newly_watermarked_full_audio_trimmed.shape, "dtype", newly_watermarked_full_audio_trimmed.dtype)
    print("Sample rate:", SAMPLE_RATE)
    print("Min/max:", newly_watermarked_full_audio_trimmed.min().item(), newly_watermarked_full_audio_trimmed.max().item())
    print("Num samples:", newly_watermarked_full_audio_trimmed.shape[-1])

    output_filename = f"watermarked_incremental_{uuid.uuid4().hex}.wav"
    output_path = save_audio(newly_watermarked_full_audio_trimmed, output_filename)
    print("Saved watermarked audio to:", output_path)
    print("File size after save:", os.path.getsize(output_path))

    original_full_audio_trimmed_for_metrics = original_full_audio_tensor[:, :total_samples]
    snr_db = calculate_snr(original_full_audio_trimmed_for_metrics, newly_watermarked_full_audio_trimmed)
    mse = calculate_mse(original_full_audio_trimmed_for_metrics, newly_watermarked_full_audio_trimmed)

    return newly_watermarked_full_audio_trimmed, output_filename, {
        "method": "PCA_PRIME_INCREMENTAL",
        "message_embedded_this_step": message_for_this_band_str,
        "band_idx": target_band_idx,
        "snr_db": round(snr_db, 3),
        "mse": round(mse, 6),
        "info": f"PCA Prime incremental watermark for band {target_band_idx} applied."
    }
```

Listing A.3: Incremental Embedding Function (`embed_single_incremental_pca_prime_watermark_step`)

A.1.4 Detection Logic

This section includes the functions responsible for watermark detection. `detect_pca_prime_watermark` provides detailed BER calculation for research evaluation, while `detect_pca_prime_payload_only` is used in the application mode for streamlined payload extraction.

```
# Assumed constants: SAMPLE_RATE, N_FFT, HOP_LENGTH, NUM_FREQ_BANDS, N_BITS, NUM_SAMPLES_SEGMENT, DEVICE
# Assumed helper functions: get_models, torch_F, calculate_snr, calculate_mse
def detect_pca_prime_watermark(original_audio_tensor: torch.Tensor | None, watermarked_audio_tensor: torch.Tensor, message_bits_to_check_str: str) -> dict: # Updated type hint
    """
    Detects watermarks using the trained PFBDetector and calculates full metrics.
    original_audio_tensor can be None if the original is not available for BER calculation.
    """
    _, detector = get_models()
    detector.eval()

    if watermarked_audio_tensor.dim() == 2:
        watermarked_audio_tensor = watermarked_audio_tensor.unsqueeze(0)

```

```
    original_audio_tensor_processed = original_audio_tensor
    if original_audio_tensor_processed is None:
        original_audio_tensor_processed = torch.zeros_like(watermarked_audio_tensor, device=DEVICE)
        print("[PCA_PRIME_WATERMARKING] Warning: Original audio not provided for detection. BER will be 0.5 (random).")
    elif original_audio_tensor_processed.dim() == 2:
        original_audio_tensor_processed = original_audio_tensor_processed.unsqueeze(0)


    if len(message_bits_to_check_str) != N_BITS * NUM_FREQ_BANDS:
        print(f"[PCA_PRIME_WATERMARKING] Warning: Message_bits_to_check length ({len(message_bits_to_check_str)}) does not match expected {N_BITS * NUM_FREQ_BANDS} bits for {NUM_FREQ_BANDS} bands of {N_BITS} bits each. Adjusting.")
        message_bits_to_check_str = message_bits_to_check_str.ljust(N_BITS * NUM_FREQ_BANDS, '0')[:N_BITS * NUM_FREQ_BANDS]

    expected_messages_per_band = []
    for i in range(NUM_FREQ_BANDS):
        band_message_str = message_bits_to_check_str[i * N_BITS : (i + 1) * N_BITS]
        expected_messages_per_band.append(torch.tensor([[int(b) for b in band_message_str]], dtype=torch.float32, device=DEVICE))

    total_samples = watermarked_audio_tensor.shape[-1]
    num_chunks = math.ceil(total_samples / NUM_SAMPLES_SEGMENT)

    total_ber_per_band_raw = [0] * NUM_FREQ_BANDS
    total_bits_processed_overall = 0

    with torch.no_grad():
        for chunk_idx in range(num_chunks):
            start_sample = chunk_idx * NUM_SAMPLES_SEGMENT
            end_sample = min((chunk_idx + 1) * NUM_SAMPLES_SEGMENT, total_samples)

            original_chunk = original_audio_tensor_processed[:, :, start_sample:end_sample]
            watermarked_chunk = watermarked_audio_tensor[:, :, start_sample:end_sample]

            if original_chunk.shape[-1] < NUM_SAMPLES_SEGMENT:
                padding_needed = NUM_SAMPLES_SEGMENT - original_chunk.shape[-1]
                original_chunk = torch_F.pad(original_chunk, (0, padding_needed))
                watermarked_chunk = torch_F.pad(watermarked_chunk, (0, padding_needed))

            logits_list = detector(original_chunk, watermarked_chunk)

            for i in range(NUM_FREQ_BANDS):
                preds = (torch.sigmoid(logits_list[i]) > 0.5).int()
                ber_item = (expected_messages_per_band[i].int() != preds).float().sum().item()
                total_ber_per_band_raw[i] += ber_item
            total_bits_processed_overall += N_BITS * NUM_FREQ_BANDS

    avg_ber_per_band = [b_raw / (num_chunks * N_BITS) for b_raw in total_ber_per_band_raw]
    avg_ber_overall = sum(avg_ber_per_band) / NUM_FREQ_BANDS

    is_detected = bool(avg_ber_overall < 0.1)

    return {
        "method": "PCA_PRIME",
        "message_checked": message_bits_to_check_str,
        "detection_probability": float(1.0 - avg_ber_overall),
        "is_detected": is_detected,
        "ber": round(avg_ber_overall, 4),
        "ber_per_band": [round(b, 4) for b in avg_ber_per_band],
        "info": "PCA Prime watermark detection performed."
    }
```

Listing A.4: Detection Functions (`detect_pca_prime_watermark` and `detect_pca_prime_payload_only`)

A.2 Backend API Orchestration (app.py excerpts)

This section provides excerpts from the Flask backend application (app.py) that orchestrate the interaction between the frontend, the core watermarking algorithms, and the database. It highlights the /process_audio endpoint, which manages the incremental embedding and detection workflow.

```
import os
import uuid
import numpy as np
from flask import Flask, request, jsonify, send_from_directory
from flask_cors import CORS
import torch
import torchaudio
import soundfile as sf
from werkzeug.utils import secure_filename
from scipy import stats
import hashlib
import json
from models import db, create_sample_data, User, AudioFile, WatermarkEntry
from db_api import db_api
from datetime import datetime

# Initialize Flask application
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'uploads')
app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024 # 16MB max upload size
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///watermark_lab.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

# Initialize database
db.init_app(app)

# Configure CORS more explicitly
CORS(app, resources={
    r"/*": {
        "origins": ["http://localhost:8080", "http://127.0.0.1:8080", "http://localhost:3000", "http://127.0.0.1:3000"],
        "methods": ["GET", "POST", "OPTIONS"],
        "allow_headers": ["Content-Type", "Authorization"]
    }
})

# Create uploads directory if it doesn't exist
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)

# Constants for watermarking
ALLOWED_EXTENSIONS = {'wav', 'mp3', 'flac', 'ogg'}
SAMPLE_RATE = 16000 # Align with model's expected sample rate

# Assumed imports for watermarking module:
# from pca_prime_watermarking import embed_single_incremental_pca_prime_watermark_step, detect_pca_prime_watermark, detect_pca_prime_payload_only, NUM_FREQ_BANDS, N_BITS
# Assumed helper functions: preprocess_audio_for_flask, save_audio, generate_file_hash

@app.route('/process_audio', methods=['POST'])
def process_audio():
    """
    Process uploaded audio file based on action (embed/detect) and method
    """
    try:
        print("\n[DEBUG] === Starting new audio processing request ===")

        if 'audio_file' not in request.files or request.files['audio_file'].filename == '':
            return jsonify({"status": "error", "message": "No audio file provided or selected"}), 400

        file = request.files['audio_file']

        action = request.form.get('action', '')
        method = request.form.get('method', '')
```

```
62          message = request.form.get('message', '') # Message for embedding or detection
    check
63          purpose = request.form.get('purpose', 'general')
64          user_id = request.form.get('user_id')
65
66          # New parameters for incremental embedding
67          original_audio_file_id = request.form.get('original_audio_file_id') # ID of the
    initial clean audio
68          current_wm_idx = int(request.form.get('current_wm_idx', 0)) # Which watermark
    (0-3) to embed next
69
70          # Message length validation for the single 8-bit message per step
71          if not message or not all(bit in '01' for bit in message) or len(message) !=
    N_BITS:
72              return jsonify({
73                  "status": "error",
74                  "message": f"Invalid message. Must be a binary string of {N_BITS} bits
    for this step (e.g., '10101010')."
75              }), 400
76
77          if not allowed_file(file.filename):
78              return jsonify({"status": "error", "message": f"File type not allowed.
    Supported types: {', '.join(ALLOWED_EXTENSIONS)}"}), 400
79
80          filename = secure_filename(file.filename)
81          unique_filename = f"{uuid.uuid4()}_{filename}"
82          file_path = os.path.join(app.config['UPLOAD_FOLDER'], unique_filename)
83          file.save(file_path)
84
85          file_hash = generate_file_hash(file_path)
86          print(f"Saved original file: {file_path}")
87
88          current_audio_tensor, sr = preprocess_audio_for_flask(file_path) # This is the
    audio from 'previous' step (or original)
89          if current_audio_tensor is None:
90              print("[ERROR] Audio preprocessing failed")
91              return jsonify({"status": "error", "message": "Failed to process audio file"
    }), 500
92
93          # Store uploaded file info in DB (this could be original or a previously
    watermarked file)
94          uploaded_file_db = AudioFile(
95              filename=filename, filepath=file_path, filehash=file_hash,
96              file_size=os.path.getsize(file_path), duration=current_audio_tensor.size(-1)
     / sr,
97              sample_rate=sr, user_id=user_id
98          )
99          db.session.add(uploaded_file_db)
100         db.session.flush()
101
102         # --- Retrieve Original Audio for Detector & Metrics (Crucial for Incremental)
    ---
103         original_clean_audio_tensor = None
104         if original_audio_file_id: # If this is not the very first embed step
105             original_file_db_entry = AudioFile.query.get(original_audio_file_id)
106             if original_file_db_entry and os.path.exists(original_file_db_entry.filepath
    ):
107                 original_clean_audio_tensor, _ = preprocess_audio_for_flask(
    original_file_db_entry.filepath)
108                 print(f"[DEBUG] Retrieved original clean audio (ID: {
    original_audio_file_id}) from DB.")
109             else:
110                 print(f"[WARNING] Original clean audio (ID: {original_audio_file_id})
    not found for incremental step. BER/SNR may be inaccurate.")
111                 # If original clean audio is not found, use current_audio_tensor as
    reference for metrics,
112                 # which would skew SNR/MSE but allow processing.
113                 original_clean_audio_tensor = current_audio_tensor.clone()
114         else: # This is the very first embed step (current_wm_idx == 0)
115             original_clean_audio_tensor = current_audio_tensor.clone() # The current
    upload IS the original for this chain
116             original_audio_file_id = uploaded_file_db.id # Record its ID as the official
     original
```

```
117             print(f"[DEBUG] First embed step. Set original_audio_file_id to {original_audio_file_id}")
118 
119         # --- EMBEDDING LOGIC (Incremental) ---
120         if action == 'embed':
121             print(f"[DEBUG] Embedding Watermark {current_wm_idx} for method: {method}")
122 
123             # Use the new incremental embedding function
124             # Passes original clean audio to maintain consistent SNR/MSE calculation reference
125             # and current audio from previous step for incremental modification.
126             watermarked_audio_tensor, output_filename, embed_results = embed_single_incremental_pca_prime_watermark_step(
127                 original_clean_audio_tensor,
128                 current_audio_tensor, # The uploaded audio for this step
129                 message,              # The 8-bit message for THIS band
130                 current_wm_idx        # The index of the band to modify
131             )
132 
133             if embed_results.get("status") == "error":
134                 db.session.rollback() # Rollback uploaded_file_db creation
135                 return jsonify(embed_results), 500
136 
137             # Use the actual output_filename returned by the watermarking function
138             output_path = os.path.join(app.config['UPLOAD_FOLDER'], output_filename)
139 
140             if not os.path.exists(output_path):
141                 db.session.rollback()
142                 return jsonify({"status": "error", "message": "Failed to save watermarked audio"}), 500
143 
144             wm_file_hash = generate_file_hash(output_path)
145             wm_file_size = os.path.getsize(output_path)
146 
147             wm_file_db = AudioFile(
148                 filename=output_filename, filepath=output_path, filehash=wm_file_hash,
149                 file_size=wm_file_size, duration=watermarked_audio_tensor.size(-1) / sr,
150                 sample_rate=sr, user_id=user_id
151             )
152             db.session.add(wm_file_db)
153             db.session.flush()
154 
155             # Detection for this step (for BER). We need the FULL 32-bit message for the detector.
156             # This means the frontend needs to send the 'cumulative' 32-bit message.
157             # For simplicity, let's assume the frontend sends the single 8-bit message,
158             # and for detection here, we just detect that single band. Or, the message field
159             # on the request could 'be' the full cumulative 32-bit message for detection.
160 
161             # Re-thinking detection for incremental step:
162             # The client sends the 8-bit message for current_wm_idx.
163             # The detector needs the 'full 32-bit expected message' (all WMs embedded so far).
164             # This implies the frontend needs to manage the 'cumulative' message string.
165 
166             # Let's align: Frontend sends `message` (8-bit for 'this step') and `full_cumulative_message` (32-bit string).
167             full_cumulative_message = request.form.get('full_cumulative_message', '0'*(N_BITS*NUM_FREQ_BANDS)) # From frontend
168 
169             # Detect all watermarks embedded so far in this current `watermarked_audio_tensor`
170             detect_results = detect_pca_prime_watermark(original_clean_audio_tensor, watermarked_audio_tensor, full_cumulative_message)
171 
172             entry = WatermarkEntry(
173                 action=action, method=method, message=message, # This is the 8-bit message for this step
174                 purpose=purpose, watermark_count=current_wm_idx + 1, user_id=user_id,
175                 snr_db=embed_results.get("snr_db"), mse=embed_results.get("mse"),
176                 detection_probability=detect_results.get("detection_probability"),
```

```
                ber=detect_results.get("ber"), is_detected=detect_results.get("
    is_detected"),
                audio_file_id=original_audio_file_id, # Link to the original clean file
    ID
                watermarked_file_id=wm_file_db.id, # Link to the newly created
    watermarked file
                meta_data=json.dumps({
                    "band_embedded_this_step": current_wm_idx,
                    "ber_per_band": detect_results.get("ber_per_band", []),
                    "full_cumulative_message": full_cumulative_message,
                    "info_this_step": embed_results.get("info")
                })
            )
            db.session.add(entry)
            db.session.commit()

            response = {
                "status": "success", "action": "embed", "method": method,
                "message_embedded_this_step": message, # 8-bit message for this step
                "original_audio_file_id": original_audio_file_id, # Pass back original
    ID for next step
                "current_wm_idx": current_wm_idx + 1, # Next watermark index
                "watermark_count_total": NUM_FREQ_BANDS, # Total possible watermarks
                "processed_audio_url": f"http://localhost:5000/uploads/{output_filename}
    ",
                "results": { # Streamlined results for frontend display (metrics of
    current cumulative audio)
                    "snr_db": embed_results["snr_db"], "mse": embed_results["mse"],
                    "detection_probability": detect_results["detection_probability"],
                    "ber": detect_results["ber"], "is_detected": detect_results["
    is_detected"],
                    "ber_per_band": detect_results["ber_per_band"]
                }
            }
            return jsonify(response), 200

        # --- DETECTION LOGIC (Application mode - payload extraction only) ---
        elif action == 'detect':

            print(f"[DEBUG] Starting CORRECTED detection process")

            wm_file_db_entry = AudioFile.query.filter_by(filehash=file_hash).first()

            if not wm_file_db_entry:
                return jsonify({"status": "error", "message": "This audio file is not
    recognized or was not processed by this system. Cannot trace its origin."}), 404

            wm_entry = WatermarkEntry.query.filter_by(watermarked_file_id=
    wm_file_db_entry.id).order_by(WatermarkEntry.watermark_count.desc()).first()

            if not wm_entry:
                # Fallback: if no watermark entry, maybe it's the original file itself?
                # In this case, we compare it against itself, expecting no payload.
                original_file_db_entry = wm_file_db_entry
                print("[WARNING] No watermark entry found. Assuming this is an original
    file and comparing against itself.")
            else:
                # 3. From that entry, get the ID of the original clean audio file.
                original_file_id = wm_entry.audio_file_id
                original_file_db_entry = AudioFile.query.get(original_file_id)

            if not original_file_db_entry or not os.path.exists(original_file_db_entry.
    filepath):
                return jsonify({"status": "error", "message": f"The original source
    audio (ID: {original_file_id}) is missing from the server."}), 500

            # 4. Load the TENSOR of the original clean audio.
            print(f"[DEBUG] Found and loading original clean audio: {
    original_file_db_entry.filepath}")
            original_clean_audio_tensor, _ = preprocess_audio_for_flask(
    original_file_db_entry.filepath)

            # 5. Call the NEW, CORRECT detection function with BOTH tensors.
```

```
            #    'current_audio_tensor' is the watermarked one the user uploaded.
            detect_results = extract_payload_with_original(original_clean_audio_tensor,
    current_audio_tensor)


            # Debug prints to see what was detected
            print(f"[DEBUG] Detection completed!")
            print(f"[DEBUG] Detected payload: {detect_results.get('payload', '')}")
            print(f"[DEBUG] Detected per band: {detect_results.get('per_band', [])}")
            print(f"[DEBUG] Full detection results: {detect_results}")

            entry = WatermarkEntry(
                action=action, method=method, message="payload_extraction", purpose="
    detection",
                is_detected=bool(detect_results.get("payload", "").strip('0')), #
    Detected if payload is not all zeros
                watermark_count=0, user_id=user_id,
                audio_file_id=original_file_db_entry.id, # Link to the original
                watermarked_file_id=wm_file_db_entry.id, # Link to the file that was
    checked
                meta_data=json.dumps({
                    "filename": filename, "file_hash": file_hash,
                    "detected_payload": detect_results.get("payload", ""),
                    "detected_per_band": detect_results.get("per_band", []),
                    "mode": "application_comparison"
                })
            )
            db.session.add(entry)
            db.session.commit()

            response_data = {
                "status": "success",
                "action": "detect",
                "method": method,
                "detected_payload": detect_results.get("payload", ""),
                "detected_per_band": detect_results.get("per_band", []),
                "info": "PCA Prime payload detection completed (reference-based)."
            }

            print(f"[DEBUG] Sending response to frontend: {response_data}")
            return jsonify(response_data), 200

    except Exception as e:
        import traceback
        print(f"[ERROR] Error processing audio in /process_audio: {e}")
        print(traceback.format_exc())
        if 'db' in locals():
            db.session.rollback()
        return jsonify({
            "status": "error", "message": f"Internal server error: {str(e)}"
        }), 500
```

Listing A.5: Main Processing Endpoint Excerpt (`app.py`)

A.3 Frontend User Interface Excerpts (`Dashboard.tsx`)

This section includes a representative excerpt from the React frontend (`Dashboard.tsx`) that demonstrates the user interface's interaction with the incremental watermarking process. It showcases how user inputs are captured, payloads are constructed, and how the frontend manages the workflow across successive embedding stages.

```
// Only included the core logic relevant to the incremental embedding process.
// Avoided including full UI boilerplate (e.g., all HTML/JSX).
// Focused on the handleApplyWatermark function, state management for incremental flow,
// and payload construction.

import { useState, useContext } from "react";
import { AuthContext } from "@/contexts/AuthContext";
import { AudioContext } from "@/contexts/AudioContext";
```

```
import { ResultsContext } from "@/contexts/ResultsContext";
import { toast } from "sonner";
import { api, ProcessAudioParams, WatermarkEmbedResponse, WatermarkResultsContextType }
    from "@/services/api";

const Dashboard = () => {
  const { user } = useContext(AuthContext);
  const { audioFile, setAudioFile, audioName, setAudioName } = useContext(AudioContext);
  const { setResults } = useContext(ResultsContext) as { setResults: (results:
    WatermarkResultsContextType | null) => void };
  const [isProcessing, setIsProcessing] = useState(false);
  const [purpose, setPurpose] = useState("training");
  const [processedAudioUrl, setProcessedAudioUrl] = useState<string | null>(null);

  const [originalAudioFileId, setOriginalAudioFileId] = useState<number | null>(null);
  const [currentWatermarkIndex, setCurrentWatermarkIndex] = useState(0);
  const [cumulativeMessage, setCumulativeMessage] = useState<string>("
    00000000000000000000000000000000");

  const BITS_PER_BAND = 8;
  const NUM_TOTAL_BANDS = 4;

  const generatePayload = (nextIndex: number) => {
    // This function now just needs to generate the single 8-bit message for the next
    step
    const userBits = selectedUser.id.toString(2).padStart(4, '0');
    const purposeIndex = mockPurposes.indexOf(selectedPurpose);
    const purposeBits = purposeIndex.toString(2).padStart(2, '0');
    const randomBits = Math.floor(Math.random() * 4).toString(2).padStart(2, '0'); // 2
    random bits
    const message = `${userBits}${purposeBits}${randomBits}`;

    // Create the full 32-bit cumulative message for the backend's BER check
    const cumulativePayloads = detectedWatermarks.map(wm => wm.payload);
    cumulativePayloads[nextIndex] = message; // Insert the new message
    const fullCumulativeMessage = cumulativePayloads.join('').padEnd(32, '0');

    return { message, fullCumulativeMessage };
  };

 const handleFileChange = async (e: React.ChangeEvent<HTMLInputElement>) => {
    const files = e.target.files;
    if (files && files[0]) {
      // 1. Reset all state when a new file is uploaded
      setAudioFile(files[0]);
      setAudioName(files[0].name);
      setFinalEmbedResult(null);
      setPreEmbedResult(null);
      setDetectedWatermarks([]);
      setIsCheckingFile(true);

      try {
        // 2. Call the pre-embed detection endpoint to analyze the file
        toast.info("Analyzing file for existing watermarks...");
        const result = await api.preEmbedDetect(files[0]);

        // 3. Store the analysis result in state - This is the crucial "memory" step
        setPreEmbedResult(result);
        const foundWatermarks = parseDetectionsForDisplay(result.detected_per_band);
        setDetectedWatermarks(foundWatermarks);

        if (foundWatermarks.length > 0) {
            toast.success(`Analysis complete. Found ${foundWatermarks.length} existing
    watermark(s).`);
        } else {
            toast.success("Analysis complete. This is a clean audio file.");
        }

      } catch (error: any) {
        console.error("Error during file analysis:", error);
        toast.error(error.message || "Failed to analyze audio file.");
        setAudioFile(null); // Clear file on error
      } finally {
```

```
        setIsCheckingFile(false);
      }
    }
  };

  const handleApplyWatermark = async () => {
    if (!audioFile) {
      toast.error("Please upload an audio file first");
      return;
    }
    if (currentWatermarkIndex >= NUM_TOTAL_BANDS) {
      toast.info("All 4 watermarks have been embedded. Please upload a new original file to start over.");
      return;
    }

    setIsProcessing(true);
    const messageForThisBand = getSingleWatermarkMessage(currentWatermarkIndex);

    const newCumulativeMessageArray = cumulativeMessage.split('');
    for (let i = 0; i < BITS_PER_BAND; i++) {
        newCumulativeMessageArray[currentWatermarkIndex * BITS_PER_BAND + i] = messageForThisBand[i];
    }
    const updatedCumulativeMessage = newCumulativeMessageArray.join('');

    try {
      const params: ProcessAudioParams = {
        audioFile,
        action: "embed",
        method: "pca_prime",
        message: messageForThisBand,
        purpose,
        original_audio_file_id: originalAudioFileId || undefined,
        current_wm_idx: currentWatermarkIndex,
        full_cumulative_message: updatedCumulativeMessage,
      };

      const isBackendAvailable = await api.checkHealth();
      if (isBackendAvailable) {
        const response = await api.processAudio(params) as WatermarkEmbedResponse;

        if (currentWatermarkIndex === 0 && response.original_audio_file_id) {
            setOriginalAudioFileId(response.original_audio_file_id);
        }

        const fetchedAudioBlob = await fetch(response.processed_audio_url).then(res => res.blob());
        const newWatermarkedFile = new File([fetchedAudioBlob], `watermarked_${audioName}_wm${currentWatermarkIndex+1}.wav`, { type: fetchedAudioBlob.type });
        setAudioFile(newWatermarkedFile); // Set this as the new input for the next embed

        setResults({
          snr_db: response.results.snr_db,
          ber: response.results.ber,
          detection_probability: response.results.detection_probability,
          processed_audio_url: response.processed_audio_url,
          method: response.method,
          mse: response.results.mse,
          ber_per_band: response.results.ber_per_band,
          step_results: []
        });

        setProcessedAudioUrl(response.processed_audio_url);
        setCumulativeMessage(updatedCumulativeMessage); // Update cumulative message state
        toast.success(`Watermark ${currentWatermarkIndex + 1} embedded successfully`);
        setCurrentWatermarkIndex(prev => prev + 1);
      } else {
        // Demo mode (simplified)
        await new Promise(resolve => setTimeout(resolve, 1500));
        setResults({
```

```
          snr_db: 33.0, ber: 0.001, detection_probability: 0.999,
          processed_audio_url: URL.createObjectURL(audioFile),
          method: "pca_prime", mse: 0.0004, ber_per_band: [0.001, 0.001, 0.001, 0.001],
          step_results: []
        });
        setProcessedAudioUrl(URL.createObjectURL(audioFile));
        setCumulativeMessage(updatedCumulativeMessage); // Update cumulative message state in demo
        toast.success(`Watermark ${currentWatermarkIndex + 1} embedded (Demo Mode)`);
        setCurrentWatermarkIndex(prev => prev + 1);
      }
    } catch (error) {
      console.error("Error embedding watermark:", error);
      toast.error("Failed to embed watermark");
    } finally {
      setIsProcessing(false);
    }
  };

  // ... (JSX render section, simplified, just showing important Label/Button elements) ...
  return (
    <div>
        {/* ... (UI elements like Upload, Purpose, etc.) ... */}
        {/* Watermark Index Display */}
        <Label>Watermark Stage</Label>
        <div>
          Embedding Watermark {currentWatermarkIndex + 1} of {NUM_TOTAL_BANDS}
        </div>
        {/* Watermark Button */}
        <Button onClick={handleApplyWatermark} disabled={isProcessing || !audioFile || currentWatermarkIndex >= NUM_TOTAL_BANDS}>
          {isProcessing ? "Embedding..." : `Embed Watermark ${currentWatermarkIndex + 1}`}
        </Button>
        {/* Processed Audio Player */}
        {processedAudioUrl && (
          <div>
            <Label>Watermarked Audio (Current Stage)</Label>
            <audio src={processedAudioUrl} controls className="w-full" />
          </div>
        )}
        {/* Watermark Info Card - Payload Display */}
        <div>
          <Label>Current Payload</Label>
          <div>
            {getSingleWatermarkMessage(currentWatermarkIndex)} (Band {currentWatermarkIndex})
          </div>
        </div>
        <div>
          <Label>Cumulative Payload (for detection)</Label>
          <div>
            {cumulativeMessage} (All 4 Bands)
          </div>
        </div>
        {/* ... (rest of UI, like breakdown) ... */}
    </div>
  );
};

export default Dashboard;
```

Listing A.6: Dashboard Component Excerpt (`Dashboard.tsx`)

A.4 Testing and Evaluation Script Excerpts (`experiment_y_prime_visual_audio_test.py`)

This section provides key excerpts from the dedicated Python script used for systematic testing, visualization, and auditory assessment of the watermarking algorithm's performance. This script was instrumental in generating the figures and qualitative insights presented in Chapter 5.

```
# Only included the core loop that performs the cumulative watermarking
# and calls the plotting/saving functions.
# Omitted helper functions like add_gaussian_noise, calculate_snr/mse etc.

import torch
import math
# Assuming get_models, load_audio_segment, save_audio, plot_waveform, plot_spectrogram,
    display_audio_player, etc. are defined

# Assumed constants: SAMPLE_RATE, NUM_SAMPLES_SEGMENT, NUM_FREQ_BANDS, N_BITS

if __name__ == '__main__':
    # ... (model loading, original audio loading) ...
    original_full_audio = load_audio_segment(TEST_AUDIO_PATH, device=DEVICE)

    # ... (initial plots of original audio) ...

    # Generate all 4 random messages once for the *entire audio*
    all_4_messages_per_band = [torch.randint(0, 2, (1, N_BITS), device=DEVICE, dtype=
    torch.float32) for _ in range(NUM_FREQ_BANDS)]

    # messages_active_in_stage will be accumulated
    messages_active_in_stage = [torch.zeros(1, N_BITS, device=DEVICE, dtype=torch.
    float32) for _ in range(NUM_FREQ_BANDS)]

    for current_wm_idx in range(NUM_FREQ_BANDS):
        print(f"\n=== ITERATION {current_wm_idx+1}: Embedding Watermark {current_wm_idx
    +1} (Band {current_wm_idx}) Cumulatively Across All Chunks ===")

        messages_active_in_stage[current_wm_idx] = all_4_messages_per_band[
    current_wm_idx]

        # Use the specific incremental embed function for the cumulative effect
        watermarked_full_audio_current_stage_trimmed, current_embed_results_dict =
    embed_single_incremental_pca_prime_watermark_step(
            original_full_audio,
            original_full_audio, # Here, pass the original_full_audio as
    current_audio_tensor_from_prev_step conceptually
            # because we are calculating deltas from original and summing them as shown
    in the Colab, NOT operating on a modified file. *This is where the app's behavior
    differs from this test script's conceptual use.*
            # Reconstruct the 32-bit message for this stage (for detection within the
    test script)
            ''.join([''.join(str(int(bit.item())) for bit in msg_tensor.squeeze(0).
    tolist()) for msg_tensor in messages_active_in_stage]),
            current_wm_idx # Target band is the one just activated
        )

        # ... (Metrics calculation and printing for clean WM) ...

        # ... (Apply Gaussian Noise, calculate metrics and print for attacked WM) ...

        # --- PLOT AND PLAY COMBINED ---
        plot_combined_audio_visuals(
            original_full_audio,
            watermarked_full_audio_current_stage_trimmed,
            attacked_full_audio_current_stage_trimmed, # Need to define
    attacked_full_audio_current_stage_trimmed
            stage_title=f"{current_wm_idx+1} Cumulative WMs"
        )
        # ... (save audio) ...

    print("\nVisual and Audio Test finished.")
```

Listing A.7: Key Evaluation Loop (`experiment_y_prime_visual_audio_test.py`)

A.5 Environment Setup (`requirements.txt`)

For full reproducibility, the following Python packages and their versions were used during the development and evaluation of this research. This list can be generated using `pip freeze > requirements.txt` in the project's virtual environment.

```
Flask==2.2.3
Flask-CORS==3.0.10
torch
torchaudio
tqdm
numpy==1.24.3
SoundFile==0.10.3.post1
Werkzeug==2.2.3
matplotlib==3.7.1
scipy==1.10.1
pandas==2.0.1
seaborn==0.12.2
audioseal==0.1.0
Flask-SQLAlchemy==3.0.3
```

Listing A.8: Python Dependencies (`requirements.txt`)

A.6 Training Pipeline Parameters and Sample Logs

This section provides a complete listing of the hyperparameters used for training the 'PFBGenerator' and 'PFBDetector' models for Experiment Y-Prime. A sample excerpt from the training logs illustrates the convergence behavior and validation metrics over epochs.

A.6.1 Hyperparameters and Constants

The following table lists the critical hyperparameters and constants used during the training of the Experiment Y-Prime models. These parameters were meticulously tuned to achieve the optimal balance between imperceptibility and robustness.

Table A.1: Experiment Y-Prime Training Hyperparameters

Parameter	Value
SAMPLE_RATE	16000 Hz
N_FFT	512
HOP_LENGTH	128 (N_FFT // 4)
NUM_FREQ_BANDS	4
N_BITS	8 (bits per watermark band)
SEGMENT_DURATION_S	1 second
NUM_EPOCHS	200
BATCH_SIZE	16
LEARNING_RATE (Generator)	1e-4
LEARNING_RATE (Detector)	1e-4
LAMBDA_IMPERCEPT_TIME	10.0
LAMBDA_IMPERCEPT_FREQ	1.0
LAMBDA_DETECTION_G	5.0
DETECTOR_FC_HIDDEN_SIZE	384 (192 * 2)
INITIAL_MODULATION_STRENGTH	0.3
TRAINING_DATA_SIZE	2000 audio segments
VALIDATION_DATA_SIZE	500 audio segments

A.6.2 Sample Training Log Excerpt

A representative excerpt from the training logs of Experiment Y-Prime illustrates the model's learning progression, showcasing the evolution of generator and detector losses, as well as validation BER and SNR over key epochs.

```
Running on device: cuda
PFBGenerator (Exp Y-Prime Refined) initialized with 4 band-specific projectors.
Initial learnable modulation strength param (pre-sigmoid): -0.8473
PFBDetector (Exp Y-Prime Refined) initialized with 4 band-specific heads.
FC hidden size: 384, Conv output size: 384
Loading VoxPopuli dataset (split: train). This may take a moment...
Preprocessing 2000 audio segments...
Finished preprocessing. Total segments: 2000
Loading VoxPopuli dataset (split: validation). This may take a moment...
Preprocessing 500 audio segments...
Finished preprocessing. Total segments: 500
Train loader has 125 batches. Val loader has 31 batches.

--- Starting Training for Experiment Y-Prime ---
Epoch 1/200 (Train): 100%|

    | 125/125 [00:15<00:00,  8.00it/s]
Epoch 1 - Gen Loss: 0.7012, Det Loss: 0.6920, Train BER: 0.4990
Epoch 1/200 (Val): 100%|

    | 31/31 [00:00<00:00, 31.00it/s]
Validation BER: 0.4950, Validation SNR: 15.12 dB

... (Intermediate Epochs) ...

Epoch 100/200 (Train): 100%|

    | 125/125 [00:15<00:00,  8.00it/s]
Epoch 100 - Gen Loss: 0.0825, Det Loss: 0.0050, Train BER: 0.0008
Epoch 100/200 (Val): 100%|

    | 31/31 [00:00<00:00, 31.00it/s]
Validation BER: 0.0000, Validation SNR: 32.55 dB
Model saved! Validation BER improved to 0.0000

... (Final Epochs) ...

Epoch 200/200 (Train): 100%|

    | 125/125 [00:15<00:00,  8.00it/s]
Epoch 200 - Gen Loss: 0.0789, Det Loss: 0.0042, Train BER: 0.0001
Epoch 200/200 (Val): 100%|

    | 31/31 [00:00<00:00, 31.00it/s]
Validation BER: 0.0000, Validation SNR: 33.00 dB
--- Training Finished ---
```

Listing A.9: Excerpt from Experiment Y-Prime Training Log

Appendix B

User Manual for Audio Watermark Lab

This user manual provides comprehensive instructions for operating the Audio Watermark Lab application, covering its installation, key functionalities, and guidelines for both the Application and Research modes. It is designed to assist users in effectively utilizing the dynamic audio watermarking system for traceability and content protection. (The entire code is available at: https://github.com/mayibongwemoyo/audio-watermark-lab).

B.1 Introduction to the Audio Watermark Lab

The Audio Watermark Lab is a web-based application developed to demonstrate and utilize the dynamic audio watermarking algorithm for multi-level traceability. It provides a user-friendly interface for embedding and detecting watermarks, alongside a robust research mode for in-depth analysis.

B.2 System Requirements and Setup

B.2.1 Prerequisites

- Python 3.8+
- Node.js (LTS version) and npm/yarn
- Git
- A modern web browser (Chrome, Firefox, Edge)

B.2.2 Installation Guide

1. **Clone the Repository:** Navigate to your desired directory in your terminal and clone the project from GitHub:

```
git clone https://github.com/mayibongwemoyo/audio-watermark-lab.git
cd audio-watermark-lab

```

2. **Backend Setup (Python/Flask):**
   - Navigate to the backend directory:

```
        cd backend

```

   - Create and activate a virtual environment:

```
        python -m venv venv
        # On Windows:
        .\venv\Scripts\activate
        # On macOS/Linux:
        source venv/bin/activate

```

- Install Python dependencies:

```
        pip install -r requirements.txt
        # If requirements.txt is not present, use a list of common dependencies:
        # pip install Flask Flask-SQLAlchemy Flask-Cors torch torchaudio soundfile Werkzeug scipy numpy datasets tqdm

```

- **Place Trained Model Weights:** Download the pre-trained model weights (`pfb_parallel_4wm_exp_y_prime_best.pth`) from the designated cloud storage (e.g., your Google Drive link) and place it inside the `backend/trained_models/` directory. Create this directory if it does not exist.
- Initialize the database (first time only, or after deleting `watermark_lab.db`):

```
        python app.py
        # Open your browser and navigate to http://localhost:5000/init-db

```

- Start the Flask backend server:

```
        python app.py

```

3. **Frontend Setup (React/TypeScript):**
   - Open a **new terminal window** and navigate to the frontend directory:

```
        cd frontend

```

   - Install Node.js dependencies:

```
        npm install
        # or yarn install

```

   - Start the React development server:

```
        npm start
        # or yarn start

```

Your application should now be accessible in your web browser, typically at `http://localhost:3000`.

B.3 Application Modes Overview

The Audio Watermark Lab operates in two primary modes:

B.3.1 Application Mode (Dashboard)

Designed for end-users, this mode provides a streamlined interface for embedding and detecting watermarks without requiring technical expertise in machine learning. It focuses on intuitive workflows and clear presentation of traceability information.

B.3.2 Research Mode

Intended for researchers and developers, this mode offers advanced configurations for watermarking parameters, detailed metric analysis, and comparative studies of different algorithmic approaches.

B.4 Using the Application Mode (Dashboard)

B.4.1 Embedding Watermarks

This feature allows users to incrementally embed up to four distinct watermarks into an audio file.

(a) **Upload Audio File:** Click the "pload Audio Fil" area and select a WAV, MP3, FLAC, or OGG audio file. The chosen file's name will be displayed.

(b) **Select Purpose:** Choose a 'Purpose' for the watermark from the dropdown (e.g.,"trainin","commercia"). This metadata is embedded alongside your identity.

(c) **Monitor Watermark Stage:** The"Watermark Stag" panel will indicate which watermark (e.g., "Watermark 1 of 4") is being embedded.

(d) **Embed Watermark:** Click the "Embed Watermark" button.

- For each click, a new 8-bit watermark (derived from your user ID, role, purpose, and the current stage) will be embedded into the next available frequency band of the audio.
- The application will display the processed audio player, allow download, and update the "Watermark Info" card with metrics relevant to the cumulatively watermarked audio up to that stage (e.g., SNR, MSE, BER, detected payload for all active watermarks).
- The frontend will automatically load the newly watermarked audio into the input for the next embedding step, simulating the incremental chain.

(e) **Continue Embedding (Optional):** Repeat Step 4 up to four times to embed all available watermarks. The button will disable once all four are embedded.

(f) **Download Watermarked Audio:** After each step, you can download the watermarked audio. For traceability, this downloaded file can be re-uploaded later for detection or further processing.

B.4.2 Detecting Watermarks

This feature allows users to upload an audio file and detect all embedded watermarks, revealing its traceability history.

(a) **Upload Audio File:** Navigate to the "Detect Trace" section (if available, or use the main upload in Dashboard) and upload an audio file (likely one previously watermarked by the system).

(b) **Detect Watermarks:** Click the "Detect Watermarks" button. The system will analyze the audio.

(c) **View Watermark Timeline:** The "Watermark Timeline" or "Detected Watermarks" panel will display information about each detected watermark, including:

- Step (which watermark stage it corresponds to, e.g., 1st, 2nd)
- User ID and Role (derived from the payload)
- Purpose (derived from the payload)
- Timestamp (of the original embedding)
- The full 8-bit payload for that specific watermark.
- Confidence (if available)

B.5 Using the Research Mode

B.5.1 Configuring Watermarks and Attacks

(a) **Select Method:** Choose "PCA Prime (Multi-Band NN)" from the dropdown. Other methods (SFA, SDA, PFB, PCA Legacy) may be available for historical comparison if their models are

loaded.

(b) **Adjust Parameters:** Use sliders and input fields to configure:

- Number of Watermarks: Specify how many watermarks (1-4) to embed simultaneously for a test run.
- Message Bits per Watermark: (Fixed at 8 for PCA Prime).
- PCA Components: (Not directly configurable for PCA Prime, as it uses learned bases).
- Enable PCA: (Toggle, not applicable for PCA Prime's learned approach).

(c) **Configure Adversarial Attacks:** Select and configure various distortion types (e.g., Gaussian noise, low-pass filter, compression) to test watermark robustness.

B.5.2 Running Experiments and Analysis

(a) **Run Research:** Click the "Run Research" button to execute a full experimental run with the configured parameters and attacks. The system will embed watermarks and then subject them to selected attacks, calculating performance metrics.

(b) **View Summary and Metrics:** A summary panel will display key aggregated metrics (overall BER, SNR, MSE).

(c) **View Performance Graphs:** Access dedicated sections to view visual representations of the results, including comparative bar charts (e.g., Experiment X vs. Y-Prime) and detailed BER per band under different attacks.

B.6 Troubleshooting and FAQs

B.6.1 Common Issues

- **Backend not connecting:** Ensure the Flask backend server is running (`python app.py`) and accessible at `http://localhost:5000`. Check firewall settings.
- **Corrupt audio files (0-second duration, 1811 bytes):** This often indicates an issue during the audio processing or saving stage. Ensure all Python dependencies are correctly installed and that the model weights are valid. (Refer to the Appendix A for specific code details on audio handling).
- **Watermark detection results do not match embedded payload:** Verify that the correct original audio file is used for detection if BER is being calculated. Ensure that the message format (8-bit for individual step embedding, 32-bit for full detection check) is consistent.

B.6.2 Tips for Optimal Performance

- Use high-quality WAV files for best results, especially for research experiments.
- Ensure your Python environment uses a GPU if available, for faster model inference.
- Clear browser cache if frontend display issues occur.