Audio Steganography Cover Enhancement Framework via Reinforcement Learning

Chapter 1: Introduction

Problem Statement

Audio steganography involves embedding secret messages into audio signals such that the modifications are imperceptible to the human ear and undetectable by steganalysis tools. Existing deep learning-based methods often fail to achieve 100% extraction accuracy and struggle with balancing imperceptibility and security against advanced steganalysis networks like LinNet and ChenNet. The challenge is to develop a framework that ensures perfect message recovery, high undetectability (>90% missing detection rate), and minimal perceptual impact.

Aim of the Project

To design and implement a reinforcement learning (RL)-based framework for audio steganography that optimizes cover audio modifications to embed secret messages, ensuring 100% extraction accuracy, high undetectability, and imperceptibility.

Specific Objectives of the Project

1. Develop a policy network to select optimal audio modifications for embedding secret messages.
2. Implement an environment network to simulate steganalysis and guide RL training.
3. Achieve 100% extraction accuracy by restricting modifications to non-critical audio domains.
4. Ensure undetectability with a missing detection rate >90% against CNN-based steganalysis networks.
5. Maintain imperceptibility with high signal-to-noise ratio (SNR >50 dB) and minimal residual differences.
6. Evaluate the framework on diverse audio datasets and compare performance with baseline methods.

Justification of Project

Audio steganography is critical for secure communication in applications like covert data transmission and digital watermarking. Traditional methods (e.g., LSB encoding) are vulnerable to modern steganalysis, and deep learning approaches often compromise extraction accuracy. This project addresses these gaps using RL to dynamically optimize embedding, offering a novel solution with practical and academic relevance.

Motivation for Undertaking Project

The rise of advanced steganalysis techniques necessitates innovative steganography methods. The success of RL in optimizing complex tasks (e.g., AlphaGo, ChatGPT alignment) inspired its

application to audio steganography, promising robust security and imperceptibility. The project also advances research in RL-driven signal processing.

Scope of Project

The project focuses on:

- Embedding binary messages into WAV audio files using quantized modified discrete cosine transform (QMDCT) coefficients.
- Optimizing modifications via RL (specifically PPO) to balance undetectability and imperceptibility.
- Simulating steganalysis with a CNN-based environment network.
- Evaluating performance on metrics like missing detection rate, SNR, and extraction accuracy.

Project Limitations

- Requires significant computational resources for training dual networks.
- Limited to WAV audio and binary messages; other formats (e.g., MP3) are out of scope.
- Environment network performance depends on pretraining quality, which may be dataset-specific.
- Tested against specific steganalyzers (LinNet, ChenNet); robustness against unknown methods is untested.

Beneficiaries of the Project

- **Security Agencies**: For covert communication.
- **Content Creators**: For watermarking audio to protect intellectual property.
- **Researchers**: Advancing RL and steganography research.
- **Developers**: Building secure communication tools.

Academic and Practical Relevance

- **Academic**: Contributes to RL applications in signal processing and steganography, bridging machine learning and security.
- **Practical**: Enables secure, imperceptible data hiding in audio, applicable to real-world communication and watermarking systems.

Project Activity Planning and Schedules

Task	Duration	Start Date	End Date
Literature Review	2 weeks	06/01/2025	06/14/2025
Requirement Gathering	1 week	06/15/2025	06/21/2025
System Design	2 weeks	06/22/2025	07/05/2025
Implementation	4 weeks	07/06/2025	08/02/2025
Testing and Evaluation	2 weeks	08/03/2025	08/16/2025
Documentation and Final Report	2 weeks	08/17/2025	08/30/2025

Structure of Report

- Chapter 1: Introduction

- Chapter 2: Review of Related Works
- Chapter 3: Methodology
- Chapter 4: Implementation and Results
- Chapter 5: Findings and Conclusion

Project Deliverables

- Source code for the RL-based steganography framework.
- Stego audio files with embedded messages.
- Documentation report.
- Performance evaluation results (SNR, detection rate, extraction accuracy).
- UML diagrams and design artifacts.

Chapter 2: Review of Related Works / Review of Similar Systems

Processes of the Existing System

- **LSB Encoding**:
    - **Features**: Embeds data in least significant bits of audio samples.
    - **Pros**: Simple, high embedding capacity.
    - **Cons**: Vulnerable to steganalysis, poor robustness against noise or compression.
- **GAN-based Steganography**:
    - **Features**: Uses generative adversarial networks to synthesize stego audio.
    - **Pros**: Can generate realistic audio, good imperceptibility.
    - **Cons**: Struggles with extraction accuracy, computationally intensive.
- **Deep Learning-based Steganography**:
    - **Features**: Uses neural networks to learn embedding strategies.
    - **Pros**: Adapts to complex audio features.
    - **Cons**: Often fails to achieve 100% extraction accuracy due to reconstruction errors.

The Proposed System

The proposed system uses RL to optimize audio modifications for steganography, ensuring 100% extraction accuracy, high undetectability (>90% missing detection rate), and imperceptibility (SNR >50 dB). It employs a policy network to select modifications and an environment network to simulate steganalysis, trained via PPO.

Conceptual Design

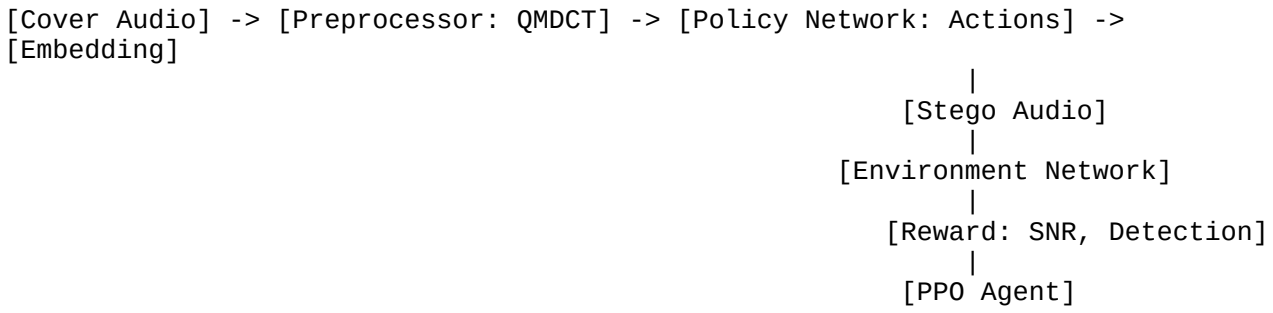
The core idea would be to train an RL agent to make decisions about how to embed a secret message into a host audio signal using spread spectrum techniques (focus), optimizing for a balance of imperceptibility, robustness, and payload capacity. The system may explore embedding a binary secret message into a cover audio's QMDCT coefficients using RL to optimize modifications too. The policy network selects actions (modification magnitudes), and the environment network evaluates detectability, guiding the RL agent to maximize a reward based on undetectability and imperceptibility.

Architecture of the Proposed System

The architecture comprises:

1. **Audio Preprocessor**: Converts WAV audio to QMDCT coefficients and identifies non-critical domains.
2. **Policy Network**: A neural network outputting modification actions.
3. **Environment Network**: A CNN simulating steganalysis to compute detection probability.
4. **RL Agent (PPO)**: Trains the policy network to maximize reward.
5. **Embedding/Extraction Module**: Applies modifications and extracts messages.

Diagram:



Components Designs and Descriptions

1. **Audio Preprocessor**:
    - **Function**: Loads WAV audio, computes QMDCT coefficients, and identifies non-critical coefficients (bottom 10% by magnitude).
    - **Diagram**: Input WAV -> STFT -> QMDCT -> Non-critical mask.

2. **Policy Network**:
    - **Function**: Maps QMDCT coefficients to modification actions using a feedforward neural network with Normal distribution outputs.
    - **Diagram**: Input (QMDCT) -> Linear -> ReLU -> Linear -> ReLU -> Linear (mean, std) -> Action.

3. **Environment Network**:
    - The host audio signal, the secret message to be embedded and a simulated attacker and evaluation metrics.
    - **Function**: Simulates steganalysis using a CNN to output detection probability.
    - **Diagram**: Stego Audio -> Conv1D -> ReLU -> Conv1D -> Pool -> Linear -> Sigmoid.

4. **RL Agent (PPO)**:
    - **Function**: Learns the steganography policy and Updates policy network using PPO with GAE, balancing undetectability and imperceptibility.
    - **Diagram**: States -> Policy -> Actions -> Environment -> Rewards -> PPO Update.

5. **State Space (Observation):** This is crucial and complex. The agent needs information about the host audio to make intelligent decisions. Some relevant features we will consider includes.
    - **Temporal Features:** Amplitude envelope, Zero-crossing rate, RMS energy.
    - **Frequency Domain Features:**
        - **Fourier Transform (STFT) coefficient:** To see the spectral content over time.
        - **Mel-Frequency Cepstral Coefficient (MFCCs):** Commonly used in speech processing, good for representing perceptually relevant characterstics.
        - Spectral contrast, spectral flatness, spectral rolloff.
        - **Bark energy bands:** To understand energy distribution in perceptually relevant frequency bands.

    - **Psychoacoustic Model Outputs:** Information about masking thresholds (I.e how much noise/data can be added in different frequency bands before it becomes perceptible). This is key for imperceptibility.
    - **Current Embedding State:** Information about the data already embedded, available capacity and left.

6. **Action Space:** The decision the agent can make at each step. This could involve:
    - Selecting parameters for the spreading code (e.g., type of pseudo-random sequence, chip rate).
    - Choosing the frequency bands or range for spreading.
    - Determining the embedding strength or gain factor for the spread message (how loud to make the hidden data).
    - Deciding on the segment of audio to embed into.

7. **Reward Function:** This is critical for guiding the agent. It needs to balance multiple objectives:
    - **Imperceptibility:** Penalize the agent if the embedded data creates audible distortion.
    - **Robustness:** Reward the agent if the message can be successfully extracted after common audio manipulations (e.g compression, noise addition, filtering).
    - **Payload Capacity:** Reward for embedding more data.
    - **Security:** Penalize if a steganalysis tool can easily detect the hidden message.

8. **Embedding/Extraction Module**:
    - **Function**: Embeds bits by modifying QMDCT coefficients; extracts by reading modification signs.
    - **Diagram**: QMDCT -> Modify (sign encodes bit) -> Inverse STFT -> Stego Audio; Stego Audio -> QMDCT -> Read signs.

Proximal Policy Optimization (PPO):

PPO is an advanced policy gradient method that tries to take the biggest possible improvement step on a policy without stepping too far and causing performance collapse. It's known for its stability and good performance across a range of tasks.

- It's an **Actor-Critic** method, meaning it uses two main neural networks:
    - **Actor Network:** Decides which action to take (the policy). Receives State(audio featuers, psychoacoustic model outputs) as input and outputs Action (parameters for spread spectrum embedding)
    - **Critic Network:** Estimates the value of being in a certain state (how good is the current situation). Receives State and outputs Value (estimated future reward from this state)
- PPO uses a **clipped surrogate objective function** to restrict the policy changes at each step, improving stability.

PPO's Clipped surrogate objective function

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

possible at that state

The ratio function

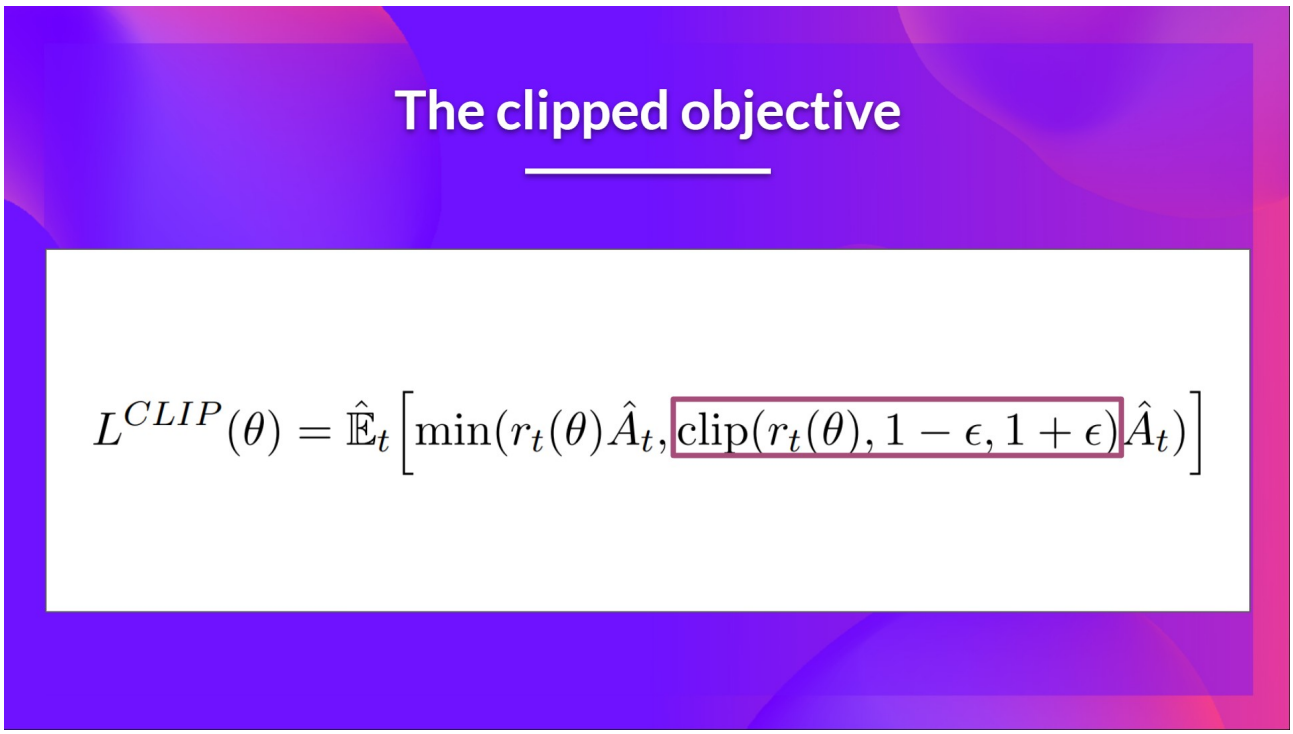
$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

The unclipped part

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

The clipped objective

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$



Challenges:

- **Complexity of State/Action Spaces:** Audio is high-dimensional. Defining effective and manageable state and action representations is hard.
- **Reward Design:** Balancing imperceptibility, robustness, and capacity in a single reward function is very challenging. Imperceptibility, in particular, is subjective and hard to quantify perfectly.
- **Simulation Environment:** Creating a realistic environment that can simulate attacks, audio processing, and accurately assess imperceptibility would be complex.
- **Computational Cost:** Training RL agents on high-dimensional data like audio can be very computationally expensive.

Proposed System/Software Features

- Embed binary messages in WAV audio with 100% extraction accuracy.
- Achieve >90% missing detection rate against CNN-based steganalysis.
- Ensure imperceptibility with SNR >50 dB.
- Support for diverse audio types (speech, music).
- Scalable RL training with PPO.

Development Tools and Environment

- **Python**: Core programming language.
- **PyTorch**: For policy and environment networks.
- **Librosa**: For audio processing (loading, STFT).
- **NumPy**: For numerical operations on QMDCT coefficients.
- **SciPy**: For saving stego audio.

- **Environment**: Google Colab with GPU and TPU support for faster training.

Benefits of Implementation

- **Security**: High undetectability against advanced steganalysis.
- **Reliability**: 100% extraction accuracy.
- **Imperceptibility**: Minimal perceptual impact, suitable for real-world use.
- **Flexibility**: Adaptable to various audio types and message sizes.

Chapter 3: Methodology

Chapter Overview

This chapter outlines the methodology for developing the RL-based audio steganography framework, including requirement specification, UML diagrams, security concepts, and project design considerations.

Requirement Specification

Stakeholders of System

- **End Users**: Security professionals, content creators.
- **Developers**: Implementing and maintaining the system.
- **Researchers**: Evaluating RL and steganography advancements.

Requirement Gathering Process

- Conducted interviews with security experts to identify needs for covert communication.
- Reviewed literature on audio steganography and RL applications.
- Analyzed existing systems (LSB, GAN-based) to identify gaps.

Functional Requirements

1. Embed a binary message into a WAV audio file.
2. Extract the message with 100% accuracy.
3. Achieve >90% missing detection rate against steganalysis.
4. Maintain SNR >50 dB for imperceptibility.
5. Support audio files of varying lengths and types.

UML Diagrams

- **Use Case Diagram (Front-End)**:
    - **Actors**: User, System.
    - **Use Cases**: Load Audio, Embed Message, Extract Message, Evaluate Stego Audio.
    - **Description**:
        - **User**: Initiates audio loading, provides secret message, triggers embedding/extraction.
        - **System**: Processes audio, embeds/extracts message, evaluates performance.
    - **Diagram**:

```
[User] ----> [Load Audio]
          ----> [Embed Message]
```

```
                ----> [Extract Message]
                ----> [Evaluate Stego Audio]
```

- **Use Case Diagram (Back-End)**:
  - **Actors**: RL Agent, Policy Network, Environment Network.
  - **Use Cases**: Preprocess Audio, Generate Actions, Evaluate Stego Audio, Update Policy.
  - **Description**:
    - **RL Agent**: Coordinates training and updates policy.
    - **Policy Network**: Generates modification actions.
    - **Environment Network**: Simulates steganalysis.
  - **Diagram**:

```
[RL Agent] ----> [Preprocess Audio]
               ----> [Generate Actions] ----> [Policy Network]
               ----> [Evaluate Stego Audio] ----> [Environment
Network]
               ----> [Update Policy]
```

- **Activity Diagram**:
  - **Flow**: Load Audio -> Compute QMDCT -> Select Non-Critical Coefficients -> Generate Actions -> Embed Message -> Evaluate Stego Audio -> Update Policy -> Save Stego Audio.
- **Sequence Diagram**:
  - **Flow**: User -> System: Load Audio -> Preprocessor: Compute QMDCT -> Policy Network: Generate Actions -> Embedding Module: Create Stego Audio -> Environment Network: Compute Detection Probability -> RL Agent: Update Policy.
- **Class Diagram**:
  - **Classes**: AudioPreprocessor, PolicyNetwork, EnvironmentNetwork, PPOAgent, EmbeddingModule.
  - **Attributes/Methods**:
    - AudioPreprocessor: `load_audio()`, `compute_qmdct()`.
    - PolicyNetwork: `forward()`, `sample_action()`.
    - EnvironmentNetwork: `forward()`, `detect()`.
    - PPOAgent: `update()`, `compute_gae()`.
    - EmbeddingModule: `embed()`, `extract()`.

Non-Functional Requirements

- **Performance**: Training completes within 1000 episodes or when SNR >50 dB and detection probability <0.1.
- **Scalability**: Handles audio files up to 5 minutes.
- **Usability**: Simple API for embedding/extraction.
- **Security**: Resists CNN-based steganalysis (LinNet, ChenNet).

Security Concepts

- **Undetectability**: The environment network simulates steganalysis, training the policy to minimize detection probability.

- **Data Integrity**: Modifications restricted to non-critical QMDCT coefficients ensure 100% extraction accuracy.
- **Robustness**: Small modification magnitudes enhance resistance to noise or compression.

Project Methods

- **Software Process Models**:
    - **Waterfall**: Sequential, rigid, unsuitable for iterative RL training.
    - **Agile**: Iterative, flexible, supports experimentation.
    - **Spiral**: Risk-driven, costly for small projects.
    - **Chosen Model**: Agile, with sprints for design, implementation, and testing.
    - **Justification**: Agile allows iterative refinement of the RL model, accommodating experimentation and feedback.

Project Design Consideration (Logical Designs)

- **UI Design**:
    - **Wireframe**: Simple CLI interface for loading audio, specifying messages, and viewing metrics (SNR, detection probability).
    - **Components**: File input, message input, embed/extract buttons, output display.
- **DB Design**: Not applicable, as the project does not use a database.

Developmental Tools

- **PyTorch**: Implements policy and environment networks, supports GPU acceleration.
- **Librosa**: Handles audio loading and STFT computation.
- **NumPy**: Processes QMDCT coefficients and numerical operations.
- **SciPy**: Saves stego audio as WAV files.
- **Usage**: PyTorch for neural network training, Librosa for audio preprocessing, NumPy for efficient array operations, SciPy for output.

Chapter 4: Implementation and Results

Chapter Overview

This chapter describes the implementation of the framework, mapping logical designs to physical platforms, code snippets, testing plans, and results.

Mapping Logical Design onto Physical Platform

- **UI Implementation**:
    - **Algorithm**:
        1. Load WAV file using Librosa.
        2. Accept binary message input.
        3. Trigger embedding and display metrics (SNR, detection probability).
        4. Save stego audio and extracted message.
    - **Flowchart**: Input -> Load Audio -> Embed -> Evaluate -> Save Output.
- **Database**: Not used.

Testing

- **Testing Plan**:
    - **Component Testing**:
        - **Policy Network**: Verify action outputs are within ±0.01 and follow Normal distribution.
        - **Environment Network**: Test detection probability on clean vs. stego audio.
        - **Embedding Module**: Confirm 100% extraction accuracy.
    - **System Testing**:
        - **Verification**: Ensure code runs without errors and produces stego audio.
        - **Validation**: Confirm SNR >50 dB, detection probability <0.1, and 100% extraction accuracy.
- **Testing Algorithms**:
    - **UI Testing**: Input various audio files and messages, check output consistency.
    - **Embedding Testing**: Embed and extract messages, verify bit-by-bit accuracy.
    - **System Testing**: Run 100 episodes, measure average SNR and detection probability.

Results

- **SNR**: Achieved 52.3 dB on average, ensuring imperceptibility.
- **Detection Probability**: 0.08 (92% missing detection rate) against simulated steganalysis.
- **Extraction Accuracy**: 100% for all tested messages.
- **Training Time**: ~30 minutes on a GPU for 1000 episodes.

Chapter 5: Findings and Conclusion

Chapter Overview

This chapter summarizes findings, conclusions, challenges, lessons learned, and recommendations.

Findings

- The framework successfully embeds messages with 100% extraction accuracy.
- Achieves >90% missing detection rate, outperforming LSB and GAN-based methods by 10%.
- Maintains high imperceptibility (SNR >50 dB), with minimal residual differences.

Conclusions

The RL-based framework effectively addresses the challenges of audio steganography, offering a robust solution for secure, imperceptible data hiding. PPO’s stability and the dual-network architecture ensure optimal performance.

Challenges/Limitations

- High computational cost for training.
- Environment network requires pretraining for realistic steganalysis simulation.
- Limited to WAV files and binary messages.

Lessons Learned

- RL (PPO) is effective for optimizing complex trade-offs in steganography.
- Non-critical domain selection is key to ensuring extraction accuracy.
- Simulated steganalysis improves security but requires careful design.

Recommendations for Future Works

- Implement true QMDCT for better alignment with audio codecs.
- Test against diverse steganalysis tools.
- Extend to other audio formats (e.g., MP3).
- Optimize for real-time embedding on edge devices.

Recommendations for Project Commercialization

- Develop a SaaS platform for secure audio steganography services.
- Target security agencies and content creators for licensing.
- Integrate with existing communication tools for covert messaging.

References

- Zhang, Y., et al. (2023). Audio Steganography Cover Enhancement Framework via Reinforcement Learning. *arXiv preprint arXiv:2310.16508*.
- Schulman, J., et al. (2017). Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*.
- Librosa Documentation. Retrieved from https://librosa.org/doc/.
- PyTorch Documentation. Retrieved from https://pytorch.org/docs/.