應用音訊浮水印於對抗式攻擊與深度學習之數位護聲符系統

Applying Audio Watermarking to Adversarial Attacks and Deep Learning in a Digital Voice Guard System

Overview

護聲符是一個全面的系統, 旨在保護音訊檔案的安全。它包括浮水印與AI 的偵測功能, 用於防止未經授權的存取。該專案利用現代技術、框架和 AI 模型, 確保系統的強大安全性。

目錄

Project Root

project-root/

```
├── ai-detection-backend/              # 後端，包含用於檢測和保護的 AI 模型
│   ├── pycache/                       # Python 緩存文件（已在版本控制中忽略）
│   ├── assets/                        # 靜態資源，例如預處理數據集或相關資源
│   │   ├── core_scripts/              # 核心腳本，用於數據處理、模型操作和實用工具
│   ├── generate/                      # 生成的對抗性音頻文件或中間結果
│   ├── key/                           # 用於存放音頻相關的密鑰，例如驗證密鑰或嵌入水印的密鑰。
│   ├── input_data/                    # 測試用的輸入文件
│   ├── myenv/                         # 虛擬環境（已在版本控制中忽略）
│   ├── app.py                         # Flask API，用於暴露後端功能（AI 檢測和對抗功能）
│   ├── dev.txt                        # Python 開發依賴列表
│   ├── eval.py                        # 用於評估 AI 檢測模型性能的腳本
│   ├── main.py                        # 訓練 AI 檢測模型的主程序
│   ├── model.pth                      # 預訓練的 AI 檢測 PyTorch 模型，用於推理
│   ├── model.py                       # 定義 AI 檢測模型的架構和加載邏輯
│   ├── model_config_RawNet.yaml       # 模型參數的配置文件
│   ├── eddsa_signer.py                # 用於數據數字簽名的工具，確保數據的真實性和完整性。
│   ├── lsb_steganography.py           # 用於在音頻中嵌入隱藏信息（如水印）的工具，基於 LSB（最低有效位）
│   ├── models.py                      # 定義 AI 模型的架構和邏輯，包括 TSSDNet 的相關實現。
│   ├── TSSDNet_model.pth              # 預訓練模型文件，用於音頻分類或對抗性樣本檢測任務。
│   ├── Watermark_model.pth            # 預訓練模型文件，用於音頻水印嵌入或檢測。
│   ├── v4.py                          # 用於生成對抗性音頻攻擊的腳本或管道工具。
│   ├── test.txt                       # 測試數據集配置文件
│   └── train.txt                      # 訓練數據集配置文件
├── app/                               # 使用 JavaScript 編寫的前端應用
│   ├── components/                    # 可重用的 UI 元件
│   │   ├── AudioListItem.js           # 用於顯示音頻列表項的元件
│   ├── context/                       # 管理全局狀態的上下文提供程序
```

```
│   │   ├── AudioProvider.js          # 管理多個音頻文件和邏輯
│   │   └── SingleAudioProvider.js    # 管理單個音頻文件的狀態
│   ├── misc/                         # 雜項實用工具
│   │   └── color.js                  # 應用的集中顏色定義
│   ├── navigation/                   # 與導航相關的文件
│   │   └── AppNavigator.js           # 配置應用的導航棧
│   ├── screens/                      # 應用的各個功能頁面
│   │   ├── AIDetectionScreen.js      # 顯示 AI 檢測主頁
│   │   ├── AudioList.js              # 顯示音頻文件列表的頁面
│   │   ├── AudioPlaybackPage.js      # 播放音頻文件和檢測結果的頁面
│   │   ├── DetectionRecordPage.js    # 顯示錄音檢測結果的頁面
│   │   ├── DetectionResultPage.js    # 顯示上傳音頻的 AI 檢測結果
│   │   ├── Adversarial.js            # 顯示對抗攻擊的頁面
│   │   ├── VoiceRecordingPage.js     # 用於錄音和檢測的頁面
│   │   ├── Watermark.js              # 將水印功能頁面轉換為 JavaScript
│   │   └── WatermarkScreen.tsx       # 用於音頻文件水印的頁面
├── assets/                           # 靜態資源, 例如圖片或示例音頻文件
├── ffmpeg-7.1/                       # 用於音頻處理的 FFmpeg 二進制文件和配置
├── App.js                            # React Native 應用的主入口點
├── app.json                          # React Native 應用的配置文件
├── package.json                      # 前端的 Node.js 依賴項
├── tsconfig.json                     # TypeScript 配置文件
├── babel.config.js                   # 用於 JavaScript 編譯的 Babel 配置文件
├── .gitignore                        # 指定從版本控制中排除的文件和文件夾
├── README.md                         # 項目文檔
├── package-lock.json                 # npm 依賴項的鎖定文件
└── yarn.lock                         # Yarn 依賴項的鎖定文件
```

System Source Code

1. ai-detection-backend

後端負責基於 AI 的語音偵測和保護，包括模型推理、評估和訓練等任務。

- 主要組件：
    - app.py: 提供後端功能（例如音頻檢測和處理）的 Flask API。
    - model.pth: 用於推理的預訓練 PyTorch 模型。
    - model.py: 定義 AI 模型架構和邏輯的腳本。
    - main.py: 用於訓練 AI 模型的入口程序
    - core_scripts/: 包含核心功能。
    - model_config_RawNet.yaml: 定義模型參數的配置文件。
    - eval.py: 用於評估模型性能的腳本。
    - generate/: 用於存儲生成的中間結果或對抗性音頻文件。
    - key/ : 用於存放音頻相關的密鑰，例如驗證密鑰或嵌入水印的密鑰。
    - input_data/: 包含用於測試和訓練的數據集。
    - train.txt 和 test.txt: 用於訓練和測試的數據集配置文件。
    - eddsa_signer.py：透過數位簽章輸出數據，確保資料的真實性和完整性。
    - lsb_steganography.py：提供在音訊資料中嵌入隱藏資訊（如浮水印）的工具。
    - v4.py：用於產生對模型進行對抗性音訊攻擊的管道。
    - models.py : 定義 AI 模型的架構和邏輯，包括 TSSDNet 的相關實現。
    - TSSDNet_model.pth：用於特定任務（如對抗性防禦或音訊分類）的預訓練模型。
    - Watermark-model.pth：用於音訊浮水印或嵌入隱藏標識符的模型。

2. app

前端基於 React Native 構建，為系統提供與用戶交互的界面。

- 主要組件::
  - components/: 可重用的 UI 元件。
    - `AudioListItem.js`: 顯示音頻列表項目。
    - `Screen.js`: 用於一致的樣式和佈局的基礎頁面元件。
  - context/: 使用 React Context API 管理應用程序的全局狀態。
    - `AudioProvider.js`: 提供管理多個音頻文件的上下文。
    - `SingleAudioProvider.js`: 管理單個音頻文件的狀態。
  - misc/: 包含雜項工具。
    - `color.js`: 用於全局一致主題的集中顏色定義。
  - navigation/: 配置應用程序的導航。
    - `AppNavigator.js`: 設置應用程序的導航堆疊。
  - screens/: 用於不同功能的應用程序頁面。
    - `AIDetectionScreen.js`: 顯示 AI 檢測結果。
    - `AudioList.js`: 顯示上傳或處理過的音頻文件列表。
    - `AudioPlaybackPage.js`: 播放音頻文件並顯示檢測詳細信息。
    - `DetectionRecordPage.js`: 用於錄製音頻並進行檢測。
    - `DetectionResultPage.js`: 顯示 AI 檢測結果。
    - `Adversarial.js`: 顯示 Adversarial畫面。
    - `Watermark.js`: 管理浮水印功能換成javascript檔案的處理。
    - `WatermarkScreen.tsx` ：提供浮水印功能的界面。
  - assets/: 包含靜態文件, 例如圖標和圖片。

3. assets

此目錄用於存放項目中使用的靜態資源。

4. ffmpeg-7.1

系統包含 FFmpeg 庫（版本 7.1），用於音頻處理。

5. 配置和元數據

- .gitignore: 指定需從版本控制中排除的文件和文件夾。
- README.md: 提供項目文檔。
- package-lock.json 與 yarn.lock:用於管理前端和後端的依賴版本。

Third-Party-Libraries

本項目利用了多個第三方庫和工具：

- PyTorch：構建和訓練神經網絡的核心深度學習框架。
- Torchaudio：PyTorch 的音頻處理庫，用於處理音頻數據。
- Transformers (Hugging Face)：用於 Wav2Vec2 模型庫，用於語音識別任務。
- FFmpeg：多媒體框架，用於預處理音頻文件，例如修剪和標準化。
- Librosa：音頻分析庫，用於提取特徵（如頻譜圖和梅爾頻率係數）。
- Soundfile (PySoundFile)：讀寫音頻文件，支持 WAV 和 FLAC 格式。
- TQDM：用於在訓練或評估等長時間運行操作中創建進度條的庫。
- Multiprocessing：Python 的內建庫，用於任務並行化。
- YAML：解析配置文件，例如 model_config_RawNet.yaml。
- Flask：輕量級框架，用於創建 RESTful API。
- Flask-CORS：為 Flask API 啟用跨域資源共享（CORS）。
- NumPy：支持數值計算和矩陣操作的庫，用於預處理管道。
- AudioSeal：用於保護音訊資料的自訂工具或方法，可能涉及浮水印嵌入或完整性校驗。
- Eddsa_signer：用於對檔案進行加密簽署的工具，確保檔案的真實性和完整性。
- UUID：生成文件和任務處理的唯一標識符。
- Warnings：用於在運行時管理警告。

2. 前端應用

前端基於 React Native 和 Expo 構建，使用以下庫：

核心庫：

- React Native: 用於構建跨平台移動應用的框架。
- Expo: 用於開發 React Native 應用的框架和平台。
- TypeScript: 提供強類型支持，以提高可維護性並減少錯誤。

導航：

- React Navigation: 管理多個頁面之間的導航。
    - @react-navigation/native: 核心導航包。
    - @react-navigation/bottom-tabs: 實現底部標籤導航。
    - @react-navigation/stack: 提供基於堆疊的導航。

音訊處理：

- expo-av: 處理音頻播放和錄音。
- expo-file-system: 訪問設備的文件系統。
- expo-document-picker: 允許從設備選擇文件。

UI 和圖標：

- @expo/vector-icons: 提供 Material 和 FontAwesome 圖標，用於 UI 元件。
    - MaterialIcons：Material Design 圖標。
    - FontAwesome5：FontAwesome 5 圖標。

工具：

- React Context API: 管理應用的全局狀態。
- Mime: 識別文件的 MIME 類型。
- RecyclerListView: 高效渲染大型列表。

3. 共享工具

以下工具用於後端和前端的開發與測試。

- Git: 版本控制系統，用於跟蹤源代碼的變更。
- Postman: API 測試與調試工具。
- Babel: JavaScript 編譯器，用於轉譯現代 JavaScript 代碼。
- TypeScript:**在前端提供強類型支持，減少運行時錯誤。**

Installation

前置條件

在開始之前，請確保您的設備已安裝以下工具和依賴項：

1. Node.js: 從 Node.js official website 下載並安裝。
2. Python: 從 Python official website 下載並安裝。
3. Expo CLI: Install Expo CLI globally using npm:

   npm install -g expo-cli
4. Mobile Device:Install the Expo Go app on your mobile device:

   iOS App Store Google Play Store.

安裝步驟

步驟 **1**：安裝代碼

將整個項目代碼下載並放置到您的文件夾中。

步驟 **2**：安裝依賴項

安裝 React Native 前端所需的依賴：

npm install

步驟 **3**：設置 **AI** 後端

啟動 Flask 服務器來處理音頻檢測，在新的命令提示符中運行：

cd ai-detection-backend

python app.py

步驟 **4**:啟動開發服務器

啟動 Expo 開發服務器:

npx expo start

步驟 **5**:在 **Expo Go** 中運行應用

掃描命令提示符中提供的 QR 碼, 應用程序將自動加載並在您的移動設備上運行。

License

1. AI 檢測模型

本模型基於 Synthetic-Voice-Detection-Vocoder-Artifacts 倉庫，其遵循 MIT 授權條款。

MIT 授權條款摘要：

允許免費使用、複製、修改和分發該軟件。

所有副本或實質部分必須包含授權條款的副本以及原版版權聲明。

如需詳細信息，請參閱原始倉庫中的 LICENSE 文件。

本項目遵循與原始倉庫相同的授權條款。

2. Hugging Face Transformers 模型

本模型基於 **Hugging Face Transformers** 庫中提供的 Wav2Vec2 模型開發，並遵循 Apache 2.0 授權條款。

Apache 2.0 授權條款摘要：

- 允許免費使用、複製、修改、合併、發佈、分發及再授權。- 使用時需要保留原始版權聲明及本授權條款的副本。

- 禁止以任何方式暗示原始作者對衍生產品的認可。

- 本軟件按「現狀」提供，不提供任何明示或暗示的擔保。

如需詳細資訊，請參閱 Hugging Face 官方文件或 Transformers 倉庫中的 LICENSE 文件 https://github.com/huggingface/transformers

本項目遵循與 Hugging Face Transformers 相同的授權條款，請在遵守 Apache 2.0 授權條款的基礎上使用本模型。