CNN-Based Music Instrument Recognition System

1 Problem Statement

Identifying instruments in a music track is crucial for tasks such as **audio tagging, remixing, indexing, and recommendation systems**. Manual labeling of instruments is time-consuming and error-prone.

InstruNet AI aims to automatically detect instruments in a music track using **Convolutional Neural Networks (CNNs)** applied to audio spectrograms.

2 Objectives

- Convert audio tracks into spectrogram images (mel-spectrogram or STFT).
- Train a CNN to classify instruments.
- Detect multiple instruments in a single track.
- Provide a user-friendly output showing instruments present and intensity.
- Generate reports for track analysis in JSON/PDF format.

3 Expected Outcomes

- Automatic detection of instruments in music tracks.
- Multi-class or multi-label classification for instruments like piano, guitar, drums, violin, etc.
- Visualization of instrument presence and intensity over the track.
- Applications in music recommendation, audio tagging, and remixing.

4 Technology Stack

- **Programming Language:** Python 3
- **Audio Processing:** Librosa, PyDub
- **CNN Framework:** TensorFlow/Keras or PyTorch
- **Visualization:** Matplotlib, Seaborn
- **Dataset:** NSynth Dataset or custom labeled music dataset
- **Frontend (Optional):** Streamlit / Dash
- **Export:** JSON / PDF

5 System Architecture

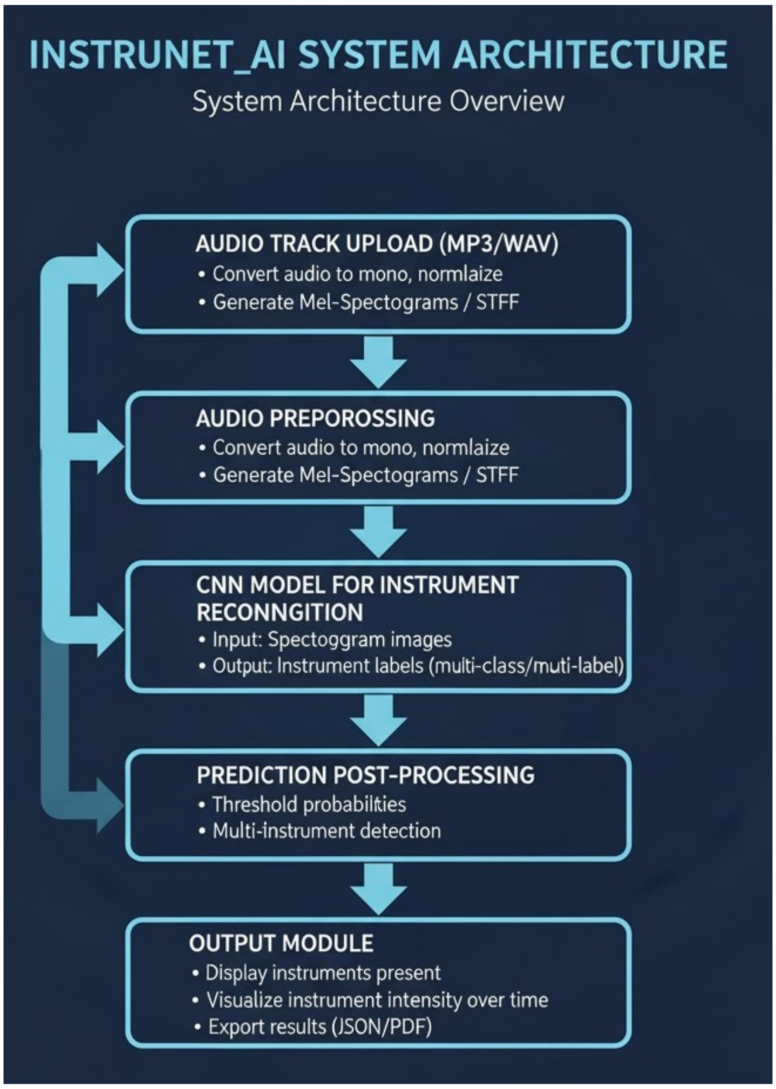
INSTRUNET_AI SYSTEM ARCHITECTURE
System Architecture Overview
AUDIO TRACK UPLOAD (MP3/WAV)
• Convert audio to mono, normlaize
• Generate Mel-Spectograms / STFF
AUDIO PREPOROSSING
• Convert audio to mono, normlaize
• Generate Mel-Spectograms / STFF
CNN MODEL FOR INSTRUMENT RECONNGITION
• Input: Spectoggram images
• Output: Instrument labels (multi-class/muti-label)
PREDICTION POST-PROCESSING
• Threshold probabilties
• Multi-instrument detection
OUTPUT MODULE
• Display instruments present
• Visualize instrument intensity over time
• Export results (JSON/PDF)

Figure 1: System Architecture Diagram for InstruNet AI

The system architecture of InstruNet AI consists of multiple components working together:

- **Audio Input Module:** Accepts uploaded or recorded audio files in WAV/MP3 format.
- **Preprocessing Unit:** Converts audio into mel-spectrograms or MFCC representations.
- **CNN Model:** Performs classification to detect instruments and their probabilities.
- **Post-processing Module:** Aggregates predictions across time segments for final labeling.
- **Visualization & Output Module:** Displays detected instruments, intensity graphs, and exports reports.

6 Example User Interface

The dashboard provides an intuitive and interactive experience where users can:

- Upload or record an audio track directly through the web interface.
- View a real-time waveform and spectrogram visualization of the uploaded track.

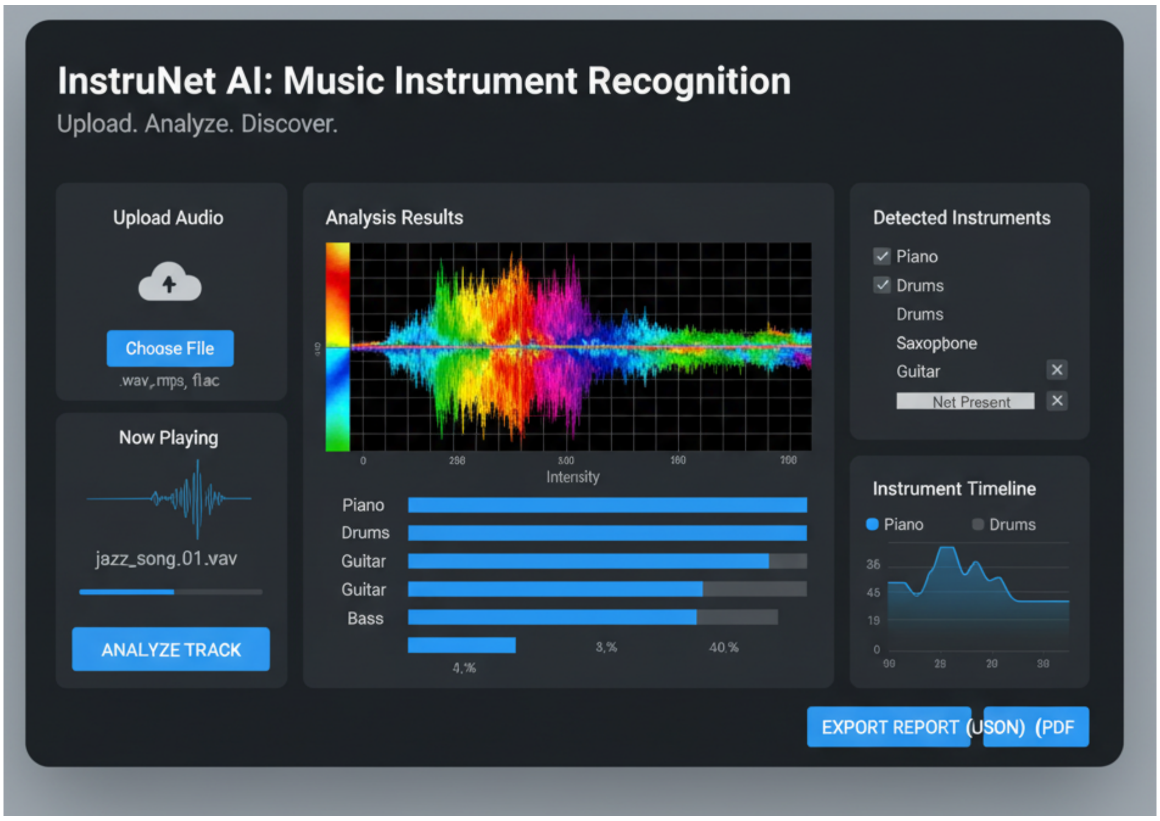

Figure 2: User Interface Mockup for InstruNet AI Dashboard

- Display the predicted instruments along with confidence percentages and visual bars.
- Explore a timeline graph showing which instruments are active during each time segment.
- Download detailed JSON or PDF reports summarizing the analysis.

This interface design aims to make the system accessible for both technical users (for research and music analytics) and general users (for music discovery or remixing).

7 Example Output

Listing 1: Sample Instrument Recognition Output from InstruNet AI

```
Audio Track: jazz_song_01.wav

Detected Instruments:
- Piano: Present
- Drums: Present
- Saxophone: Not Present
- Guitar: Present
- Bass: Present

Instrument Intensity Over Time:
Piano: |||||||||||||||||
Drums:  |||||||||||
Guitar: |||||||||
Bass:   ||||||||||||

Exported Report: jazz_song_01_instruments.json
```

8 Two-Month Milestone Plan

Week 1–2: Data Collection & Preprocessing

- Collect or download labeled audio dataset.
- Preprocess audio: normalize, convert to mono, generate spectrograms.
- **Milestone:** Dataset ready with spectrogram images and labels.

Week 3–4: CNN Model Development

- Build CNN architecture for instrument classification.
- Train model on preprocessed spectrogram dataset.
- **Milestone:** Initial CNN model achieves baseline accuracy.

Week 5–6: Model Evaluation & Tuning

- Evaluate on validation/test dataset.
- Tune hyperparameters (learning rate, filters, kernel size, etc.)
- Handle multi-instrument tracks if needed.
- **Milestone:** Optimized model with satisfactory accuracy.

Week 7–8: Deployment & Visualization

- Build output module for instrument detection visualization.
- Optional: Deploy via Streamlit web app.
- Export results as JSON/PDF for tracks.
- **Milestone:** End-to-end pipeline from audio input to instrument recognition and report generation.

9 Learning Outcomes

- Gain experience in applying CNNs to audio spectrograms.
- Learn audio preprocessing and feature extraction techniques.
- Understand multi-class and multi-label classification in deep learning.
- Implement end-to-end pipeline from audio input to instrument recognition and visualization.
- Explore real-world applications in music tech and streaming platforms.