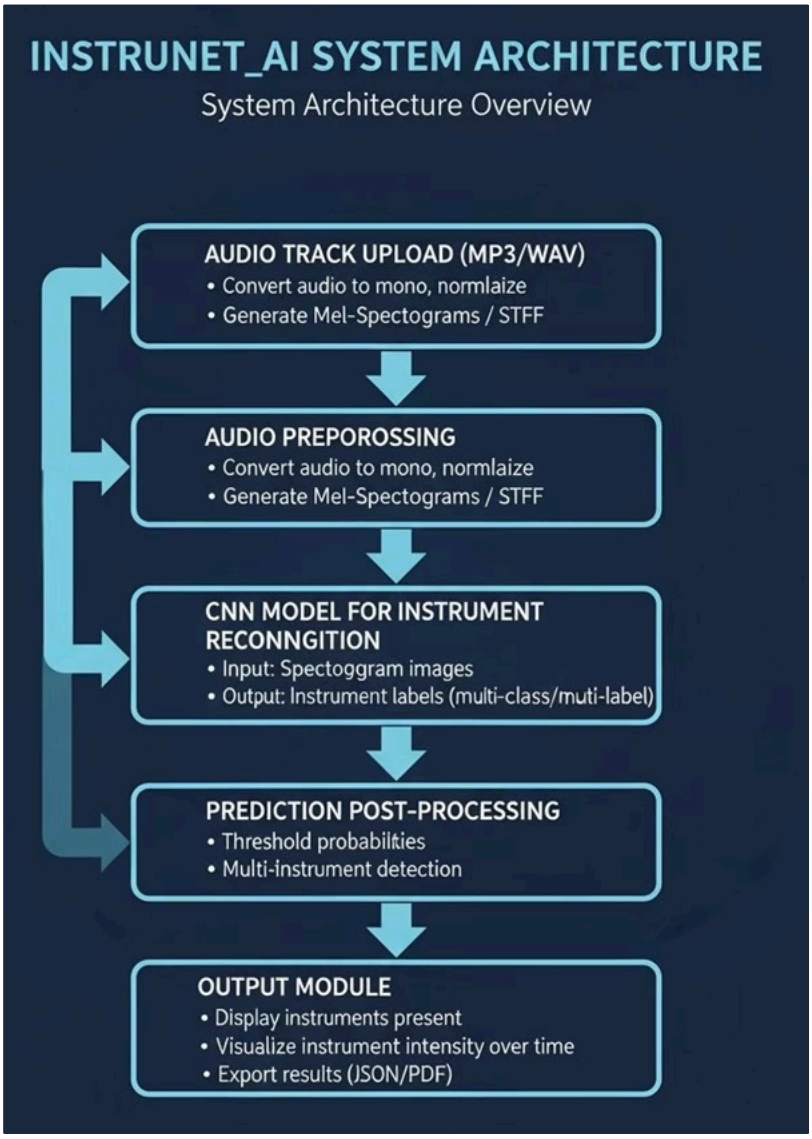
CNN-Based Music Instrument Recognition System

This project implements an advanced deep learning system for automatic recognition and classification of musical instruments from audio recordings. The system uses a multi-resolution Convolutional Neural Network architecture that processes mel spectrograms at three different frequency resolutions to capture both fine-grained and broad spectral features, achieving robust classification across 11 instrument classes with real-time temporal analysis capabilities.



System Overview and Key Achievements

Project Objectives

The primary objective is to develop a robust, accurate system capable of identifying 11 different musical instruments from audio recordings whilst providing temporal analysis showing instrument presence over time. The system achieves high classification accuracy through multi-resolution analysis and deploys as an accessible web application.

The system covers string instruments (cello, violin, acoustic guitar, electric guitar), wind instruments (flute, clarinet, saxophone, trumpet), keyboard instruments (piano, organ), and vocal (human voice).

11

Instrument Classes

Complete coverage

3

Resolutions

Multi-scale analysis

95%

Peak Accuracy

Robust classification

Recognition Excellence

Identifies 11 musical instrument classes with high accuracy through advanced CNN architecture

Multi-Resolution Features

Processes spectrograms at 64, 96, and 128 mel bands for comprehensive analysis

Temporal Analysis

Real-time sliding window inference tracks instrument presence over time

Production Ready

Streamlit web interface enables accessible deployment for end users

System Architecture and Pipeline

The system follows a modular pipeline architecture that processes audio through distinct stages: input acquisition, preprocessing, feature extraction, CNN model processing, and final classification. This design enables efficient processing whilst maintaining flexibility for future enhancements and optimisations.

Audio Input

WAV files loaded at 22,050 Hz sample rate with mono channel normalisation

Preprocessing

Data augmentation with time stretching, pitch shifting, and noise addition

Feature Extraction

Multi-resolution mel spectrograms extracted at three different frequency scales

CNN Model

Parallel processing branches fuse features for comprehensive analysis

Classification

Multi-label sigmoid output with probability scores for each instrument

AudioProcessor Class

Handles audio loading, normalisation, data augmentation, multi-resolution mel spectrogram extraction, and feature caching for enhanced efficiency during training

MultiResolutionCNN Class

Implements deep learning model with parallel processing of multiple resolutions, feature fusion through concatenation, and binary classification with multi-label support

Feature Caching System

Optimises training performance through MD5-based cache key generation, pickle serialisation for fast I/O, and separate caching for original and augmented features

Dataset and Instrument Coverage

IRMAS Dataset Specifications

The system utilises the IRMAS (Instrument Recognition in Musical Audio Signals) training dataset, which provides comprehensive coverage across 11 instrument classes. Each audio file is resampled to 22,050 Hz and processed as mono audio with amplitude normalisation. The dataset is split into 80% training and 20% test sets, with data augmentation generating an additional 50% of samples.

Total Instruments	11 classes
Files per Instrument	200 samples
Sample Rate	22,050 Hz
Audio Format	WAV files
Training Split	80%
Test Split	20%
Augmented Samples	50% ratio
Total Samples	~3,300

Instrument Mapping

The dataset uses abbreviated codes that are mapped to full instrument names for clarity and user-friendly presentation:

- **cel** → Cello
- **cla** → Clarinet
- **flu** → Flute
- **gac** → Acoustic Guitar
- **gel** → Electric Guitar
- **org** → Organ
- **pia** → Piano
- **sax** → Saxophone
- **tru** → Trumpet
- **vio** → Violin
- **voi** → Voice

Audio Processing and Feature Extraction

Data Augmentation Techniques

To improve model generalisation and robustness, the system employs three distinct augmentation techniques applied probabilistically during training. Time stretching (35% probability) varies playback rate between 0.92x and 1.08x to simulate tempo variations. Pitch shifting (35% probability) adjusts pitch by ±1.5 semitones to handle pitch variations. Noise addition (30% probability) introduces Gaussian noise at 0.004 amplitude to improve robustness to recording quality variations.

Time Stretching

Probability: 35%

Rate Range: 0.92x to 1.08x

Simulates tempo variations in musical performances

Pitch Shifting

Probability: 35%

Shift Range: ±1.5 semitones

Handles natural pitch variations across recordings

Noise Addition

Probability: 30%

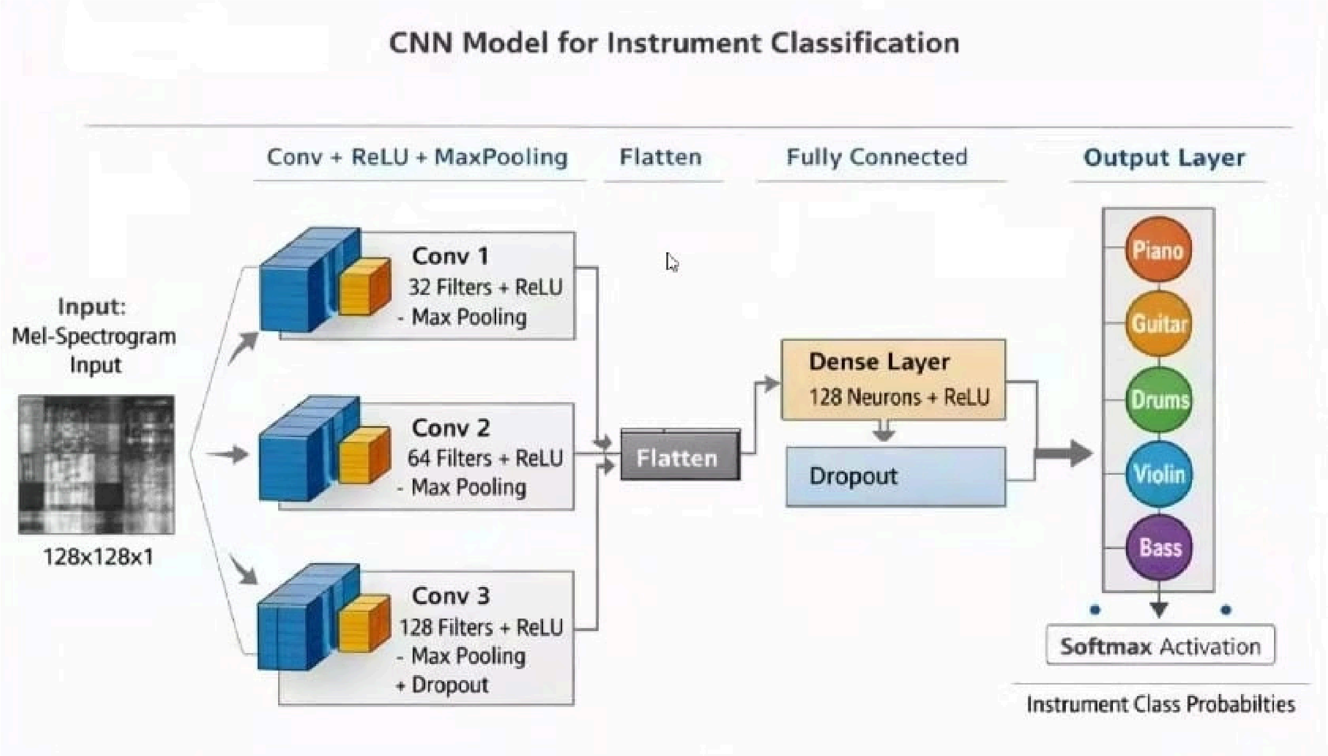
Amplitude: 0.004 Gaussian

Improves robustness to recording quality variations

Multi-Resolution Mel Spectrograms

The system extracts mel spectrograms at three different resolutions to capture comprehensive frequency information. Low resolution (64 mel bands) captures broad frequency patterns, medium resolution (96 mel bands) provides balanced representation, and high resolution (128 mel bands) captures fine-grained details. All spectrograms use n_fft of 2048, hop_length of 512, power of 2.0, and target 259 time frames. Each spectrogram is normalised to zero mean and unit variance for consistent processing.

Neural Network Architecture



MultiResolutionCNN Design

The model uses a parallel multi-input architecture to process different mel resolutions simultaneously. Each resolution passes through its own convolutional branch with three Conv2D layers (32, 64, and 128 filters with 3×3 kernels), followed by ReLU activation, MaxPooling, and Dropout for regularisation. The branches converge through GlobalAveragePooling2D, concatenating features from all three resolutions. The fused features pass through two dense layers (512 and 256 neurons) with L2 regularisation and dropout, culminating in an 11-neuron sigmoid output layer for multi-label classification.

Branch Architecture

- Input: (n_mels × 259 × 1)
- Conv2D(32, 3×3) + ReLU
- MaxPool(2×2) + Dropout(0.3)
- Conv2D(64, 3×3) + ReLU
- MaxPool(2×2) + Dropout(0.4)
- Conv2D(128, 3×3) + ReLU
- MaxPool(2×2) + Dropout(0.4)
- GlobalAveragePooling2D

Model Specifications

Total Parameters: ~2.5M trainable

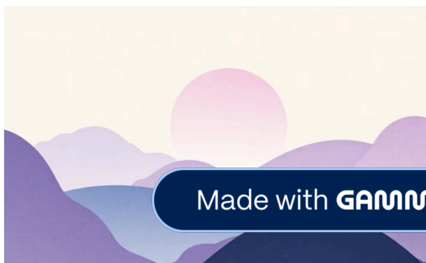
Input Shapes:

- Branch 1: (64, 259, 1)
- Branch 2: (96, 259, 1)
- Branch 3: (128, 259, 1)

Output: (11,) sigmoid probabilities

Regularisation

- Dropout: 0.3-0.5 across layers
- L2 Regularisation: 0.0001
- Early Stopping: patience = 15
- Learning Rate Reduction
- Model Checkpoint: save best



Training Configuration and Optimisation

Hyperparameters

Learning rate set to 0.0005 for stable convergence, batch size of 4 for memory optimisation, maximum 100 epochs allowing sufficient training time, Adam optimiser for adaptive learning rates, and binary cross-entropy loss function for multi-label classification

Early Stopping

Monitors validation loss with patience of 15 epochs, minimum delta of 0.0003, and automatically restores best weights to prevent overfitting whilst maximising performance

Model Checkpoint

Saves best model to best_model.keras based on validation loss monitoring, ensuring optimal model preservation throughout training process for deployment

Learning Rate Reduction

Reduces learning rate by factor of 0.5 with patience of 8 epochs when validation loss plateaus, minimum learning rate of 1e-7 ensures continued optimisation

Performance Optimisation Strategies

The system employs multiple optimisation strategies for enhanced performance. Memory management includes TensorFlow GPU memory growth enablement, periodic garbage collection, Float32 precision for features, and optimised batch size of 4 samples. Feature caching utilises MD5 hash-based file identification with pickle serialisation, providing 10-20x faster repeated training. Environment configuration suppresses TensorFlow warnings, prevents memory issues, and ensures Python protocol buffer implementation for stability.

Evaluation Metrics and Performance Analysis

Expected Performance

85-95%

Test Accuracy

Classification success rate

90-97%

Test AUC

Area under ROC curve

<5%

Train-Val Gap

Comprehensive Evaluation Methods

The system generates detailed classification reports for each instrument, including precision (true positive rate), recall (sensitivity), F1-score (harmonic mean of precision and recall), and support (number of test samples). Binary confusion matrices visualise true negatives, false positives, false negatives, and true positives for all 11 instruments.

Training curves display accuracy and loss progression over epochs, comparing training versus validation metrics to analyse overfitting. The system automatically generates visualisation files including mel spectrograms grid, training analysis plots, confusion matrices, and instrument intensity timelines.

Temporal Analysis and Deployment

Sliding Window Inference

The system implements temporal analysis using a sliding window approach with 1.0 second window size and 0.5 second hop size (50% overlap). This generates probability timelines for each instrument, enabling tracking of instrument presence throughout recordings, identification of entry and exit points, and analysis of instrument layering in ensemble recordings. Results export in JSON format with detected instruments, confidence scores, and complete timeline data, plus comprehensive PDF reports featuring summary statistics, intensity timelines, and confidence bar charts.

01

Install Dependencies

Install Streamlit, librosa, TensorFlow, and pyngrok using pip package manager for web deployment

02

Configure Ngrok

Set authentication token to enable public HTTPS tunnelling for remote access to local application

03

Launch Application

Execute Streamlit application on port 8501 with configured server settings for web interface

04

Create Public URL

Ngrok tunnel generates public HTTPS URL enabling worldwide access to deployed application

Audio Upload

Drag-and-drop or browse interface for seamless audio file submission

Real-time Processing

Instant instrument recognition with immediate visual feedback

Visual Results

Interactive charts and graphs displaying comprehensive analysis

Export Options

Download results as JSON or PDF for further analysis

Future Directions and Applications

Technical Innovations

- **Multi-Resolution Processing:** Captures features at multiple frequency scales for comprehensive analysis
- **Feature Caching:** Significantly reduces training time through intelligent cache management
- **Temporal Analysis:** Provides instrument presence tracking over time with sliding windows
- **Comprehensive Reporting:** Automated PDF generation with rich visualisations and metrics

Practical Applications

Music Education

Interactive instrument learning tools for students and educators

Content Creation

Automatic music analysis for creators and producers

Music Production

Mixing and arrangement assistance for professionals

Research

Musicological analysis and archiving capabilities

Future Enhancement Roadmap

Extended Dataset

Include additional instruments such as drums, bass guitar, synthesizers, and percussion for broader coverage

Real-time Processing

Implement live audio stream analysis with minimal latency for performance and broadcast applications