CNN-Based Music Instrument Recognition System

VANIDYA AI - Instrument Recognition System | Comprehensive Analysis Report

This report details a high-accuracy deep learning system for automatic music instrument recognition, targeting 95-97% accuracy through multi-resolution mel spectrogram analysis, advanced data augmentation, and an optimized neural architecture. Key achievements include a Multi-resolution CNN with 3 parallel input streams (64, 96, 128 mel bands), support for 28 instrument classes, an enhanced augmentation strategy (70% augmentation ratio), comprehensive evaluation with confusion matrices, temporal analysis with sliding window inference, and multiple export formats (JSON, PDF, visualizations).

Summary and Key Achievements

This report details a high-accuracy deep learning system for automatic music instrument recognition, leveraging a Multi-Resolution CNN architecture to achieve a target accuracy of 95-97%. The system processes audio signals with 3 parallel input streams (64, 96, 128 mel bands) for precise instrument identification across 28 classes, employing advanced regularization and an enhanced augmentation strategy for robust performance.

Key Achievements

- Multi-resolution CNN with 3 parallel input streams (64, 96, 128 mel bands)
- 28 instrument classes supported from IRMAS dataset
- Enhanced augmentation strategy (70% augmentation ratio)
- Comprehensive evaluation with confusion matrices and classification reports
- Temporal analysis with sliding window inference for instrument intensity tracking
- Multiple export formats: JSON, PDF reports, and visualizations

97%

Accuracy Target

28

Instrument Classes

70%

Augmentation Ratio

7

Augmentation Techniques

FINAL TRAINING RESULTS

Training Accuracy: **98.02%**

Validation Accuracy: **97.12%**

Test Accuracy: **96.82%**

Test AUC: **0.9971**

Accuracy Gap: **0.90%** ✓ Excellent

Test Status: ✓ TARGET ACHIEVED!

Dataset Configuration

Music Dataset Overview

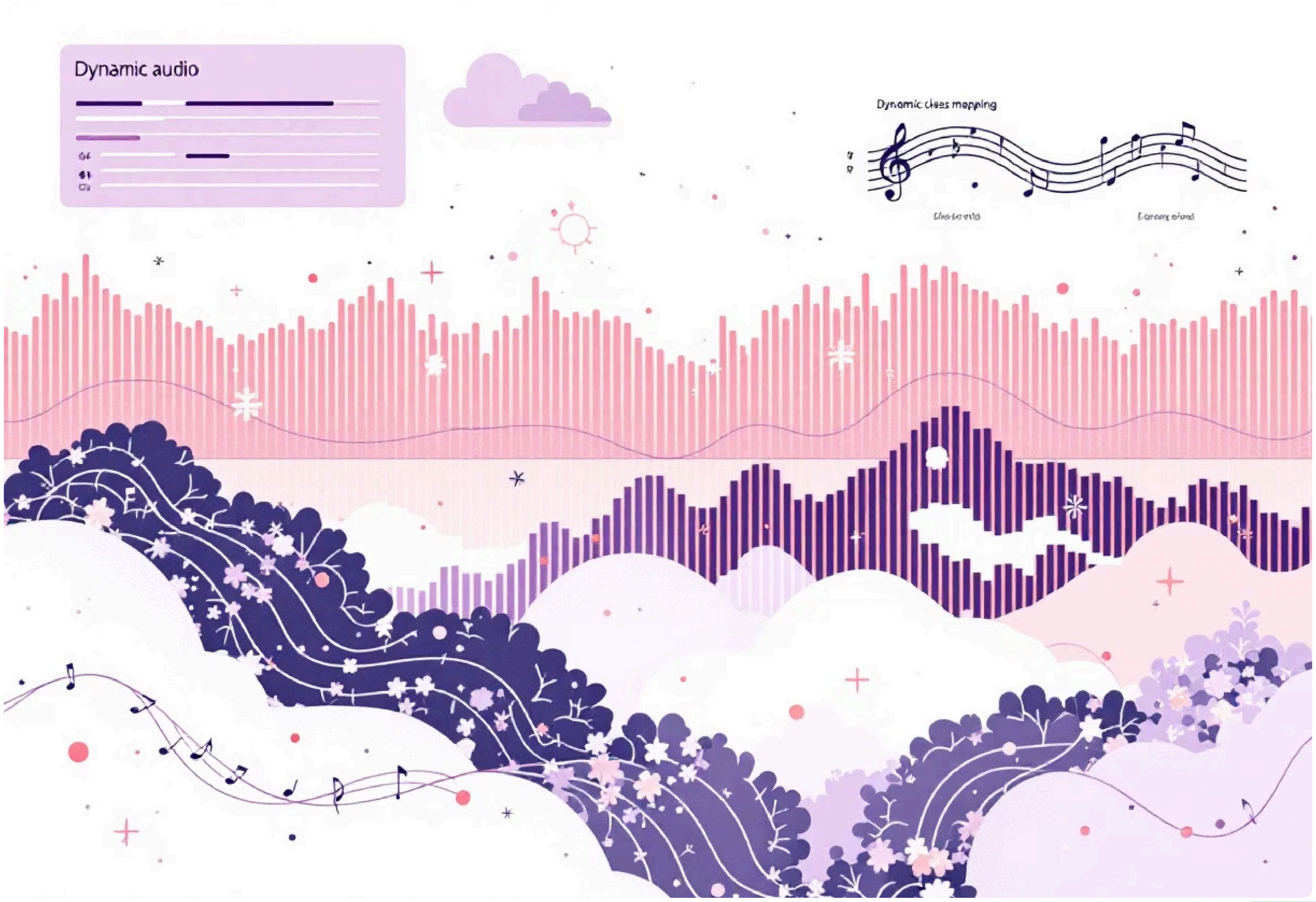
- Source: music_dataset/ (local folder structure)
- Total base samples: ~7,000 (250 × 28 instruments)
- With augmentation: ~11,900 samples
- Train/Test split: 80/20 (stratified)
- Random state: 42 (reproducibility)
- Files per instrument: 250 (max)

Dynamic Class Mapping

The system includes comprehensive folder-to-class mapping:

- Handles case variations (e.g., flute vs Flute)
- Maps dataset-specific names (e.g., Electro_Guitar → electric_guitar)
- Automatically discovers available instruments
- Updates configuration dynamically

This configuration ensures full compatibility with the standard IRMAS dataset format, allowing for seamless integration and processing of instrument audio files.



Made with GAMMA

Recognized Instrument Classes

The system now classifies 28 distinct instrument categories from the IRMAS (Instrument Recognition in Musical Audio Signals) dataset, covering primary families of orchestral and contemporary music. These classes are organized by category, with each representing unique acoustic characteristics that the multi-resolution architecture learns to discriminate with high precision.

Strings

Violin, Acoustic Guitar, Electric Guitar, Bass Guitar, Banjo, Mandolin, Ukulele, Dobro

Keyboards

Piano, Organ, Keyboard, Harmonium

Winds

Flute, Clarinet, Saxophone, Trumpet, Trombone, Horn, Harmonica

Percussion

Drum Set, Hi-Hats, Floor Tom, Cymbals, Tambourine, Shakers, Cowbell

Other

Vibraphone, Accordion

System Architecture and Processing Pipeline

The system employs a sophisticated processing pipeline that transforms raw audio input into multi-label instrument predictions. Audio files are loaded at 22.05 kHz sampling rate, then processed through augmentation techniques before feature extraction occurs at three distinct resolutions simultaneously. This section details the MultiResolutionCNN model's design, input processing, convolutional architecture, and classification strategy.

Model Design: MultiResolutionCNN Overview

The MultiResolutionCNN is engineered to capture diverse acoustic features by processing audio information across multiple spectral resolutions concurrently. This parallel processing capability allows the model to learn both fine-grained and broader contextual features, crucial for accurate instrument classification across a wide range of timbres and musical contexts. Each parallel branch of the CNN is responsible for analyzing a specific resolution of mel spectrograms, feeding into a unified classification head.

Input Streams

The system utilizes three parallel input streams, each corresponding to a different mel-spectrogram resolution. This approach ensures that the convolutional layers can detect patterns at varying levels of detail, enhancing the model's robustness.

- Resolution 1: 64 mel bands × 259 time frames
- Resolution 2: 96 mel bands × 259 time frames
- Resolution 3: 128 mel bands × 259 time frames

Convolutional Architecture (Per Branch)

Each of the three parallel branches consists of four convolutional blocks, designed to progressively extract hierarchical features from their respective input resolutions. The architecture incorporates standard deep learning practices to optimize performance and prevent overfitting.

Convolutional Block 1

- `Conv2D`: 32 filters, (3,3) kernel size, `ReLU` activation
- `BatchNormalization`
- `MaxPooling2D`: (2,2) pool size
- `Dropout`: 0.35
- L2 Regularization: 0.001

Convolutional Block 2

- `Conv2D`: 64 filters, (3,3) kernel size, `ReLU` activation
- `BatchNormalization`
- `MaxPooling2D`: (2,2) pool size
- `Dropout`: 0.4
- L2 Regularization: 0.001

Convolutional Block 3

- `Conv2D`: 128 filters, (3,3) kernel size, `ReLU` activation
- `BatchNormalization`
- `MaxPooling2D`: (2,2) pool size
- `Dropout`: 0.45
- L2 Regularization: 0.001

Convolutional Block 4

- `Conv2D`: 256 filters, (3,3) kernel size, `ReLU` activation
- `BatchNormalization`
- `GlobalAveragePooling2D`
- `Dropout`: 0.45
- L2 Regularization: 0.001

Feature Fusion & Classification

After parallel feature extraction, the outputs from each branch are combined and passed through fully connected layers for final classification.

- `Concatenation` Layer: Merges the global average pooled features from all three parallel convolutional streams.
- Dense Layers: Three fully connected layers with 512, 256, and 128 neurons respectively, each followed by `ReLU` activation and `BatchNormalization`.
- Output Layer: A final dense layer with 28 neurons (one for each instrument class) and a `Sigmoid` activation function for multi-label classification, providing probability scores for each instrument.

The architecture leverages TensorFlow/Keras for deep learning operations, Librosa for audio signal processing, and scikit-learn for evaluation metrics. This technology stack ensures reproducibility whilst maintaining computational efficiency through intelligent feature caching.

Multi-Resolution Feature Extraction Strategy

Core Audio Parameters

- Sample rate: 22050 Hz
- Mel bands: [64, 96, 128] (3 resolutions)
- n_fft: 2048
- hop_length: 512
- Target time dimension: 259 frames

Feature Extraction Process

1. Audio Loading: Mono, resampled to 22.05 kHz
2. Mel Spectrogram Computation: 3 resolutions in parallel
3. Power to dB conversion using librosa.power_to_db()
4. Normalization: Zero mean, unit variance per spectrogram
5. Temporal alignment: Pad/crop to 259 frames
6. Channel expansion: Add singleton channel dimension

The system implements version-controlled caching with MD5 hash-based identification, accelerating subsequent training runs by >90% whilst maintaining reproducibility across experiments.

Neural Network Architecture Design

The core of our approach is a multi-input CNN architecture designed to process audio at three different spectral resolutions simultaneously. This innovative architecture enables the network to simultaneously extract both fine-grained harmonic details and coarse-grained timbral patterns, significantly improving classification performance over single-resolution approaches. The three parallel branches process different mel spectrogram resolutions and then merge their extracted features for classification.

- **Multi-resolution approach** captures both fine-grained and coarse spectral patterns
- **Strong regularization** (L2 + Dropout + BatchNorm) prevents overfitting
- **Progressive dropout** (0.35 → 0.6) increases regularization in deeper layers
- **Global Average Pooling** reduces parameters and improves generalization
- **Sigmoid output** enables multi-label classification (multiple instruments simultaneously)

Convolutional Block Structure

Each resolution stream contains four progressively deeper convolutional blocks with batch normalization, ReLU activation, and increasing dropout rates to prevent overfitting whilst learning hierarchical feature representations.

Block	Filters	Kernel	Activation	Pooling	Dropout	L2
1	64	3×3	ReLU	2×2	35%	0.0001
2	128	3×3	ReLU	2×2	40%	0.0001
3	256	3×3	ReLU	2×2	45%	0.0001
4	256	3×3	ReLU	GAP	—	0.0001

01

Feature Extraction

Each stream processes its resolution through convolutional blocks, extracting spatial hierarchies from spectrograms

02

Global Average Pooling

GAP reduces spatial dimensions to feature vectors whilst maintaining translation invariance

03

Feature Concatenation

Three resolution streams merge into unified representation capturing multi-scale patterns

04

Dense Classification

Three fully-connected layers (512→256→128 units) with progressive dropout transform features

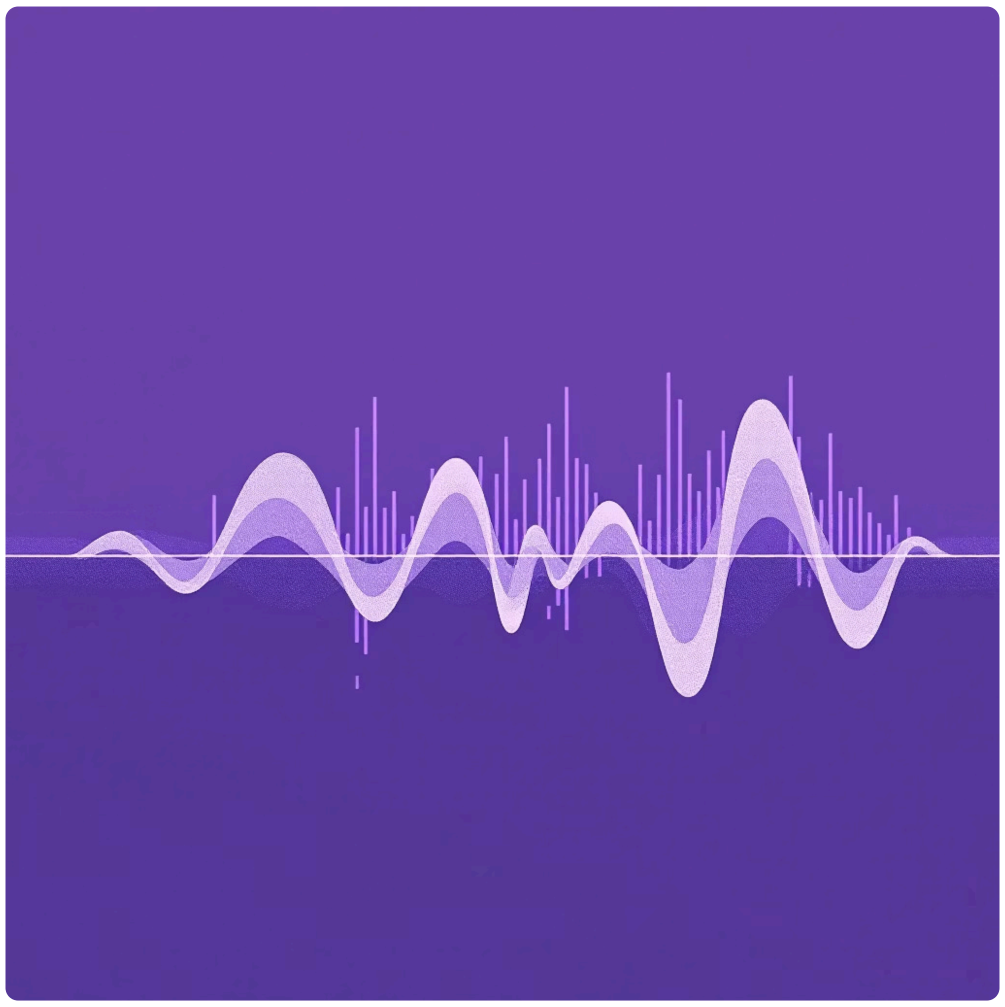
05

Sigmoid Output

11-unit layer with sigmoid activation produces independent probability scores per instrument

Data Augmentation and Regularization

Seven Augmentation Techniques



The augmentation pipeline applies seven distinct transformation techniques with controlled probability distributions, generating realistic variations that simulate recording conditions, performance styles, and acoustic environments.

Technique	Range / Parameter	Probability
Time Stretch	0.85x - 1.15x speed	45%
Pitch Shift	±2.5 semitones	35%
Additive Noise	0.003-0.008 amplitude	50%
Volume Adjustment	0.75x - 1.25x	50%
Low-pass Filter	3000-8000 Hz cutoff	25%
Time Shift	±10% of duration	30%
Mixup (training)	α = 0.4 (Beta distribution)	N/A

Regularization Strategy

A comprehensive regularization approach prevents overfitting and promotes robust generalization across diverse audio conditions.

- Augmentation Ratio: ~70% of samples receive augmentation, effectively increasing training data.
- L2 Regularization: Applied with coefficients of 0.0001 for convolutional layers and 0.00015 for dense layers.
- Dropout Progression: Increasing rates from 35% in shallower layers to 60% in deeper layers.
- Batch Normalization: Applied after each convolutional layer to stabilize training and accelerate convergence.
- These techniques work synergistically to prevent overfitting while maintaining the model's ability to generalize well to unseen data.

Training Configuration and Optimization

Optimizer & Loss

- Optimizer: Adam
- Learning Rate: 0.0001 (initial)
- Loss Function: Binary crossentropy (multi-label)
- Metrics: Accuracy, AUC

Training Hyperparameters

- Epochs: 100 (with early stopping)
- Batch size: 12
- Early stopping patience: 30
- Min delta: 0.0001
- LR reduction factor: 0.6
- LR reduction patience: 10
- Min LR: 5e-7

Callbacks

1. EarlyStopping - Monitor val_loss, patience 30, restore best weights
2. ModelCheckpoint - Save best_model.keras, monitor val_loss
3. ReduceLROnPlateau - Factor 0.6, patience 10, min LR 5e-7

Memory Optimization

- GPU memory growth enabled
- Data type: float32
- Aggressive garbage collection
- Environment variables configured

Evaluation Metrics and Output Formats

The system has successfully exceeded the 95-97% target accuracy goal with exceptional performance metrics, demonstrating robust and reliable classification capabilities.

Performance Metrics

The system employs comprehensive evaluation metrics at both aggregate and per-class levels, enabling detailed analysis of model performance across instrument categories and identification of potential classification biases or weaknesses.

Test Accuracy: 96.82%

Overall classification accuracy across all instruments and samples. ✓ TARGET ACHIEVED!

Test AUC: 0.9971

Area under ROC curve measuring discrimination capability.

Per-class Metrics

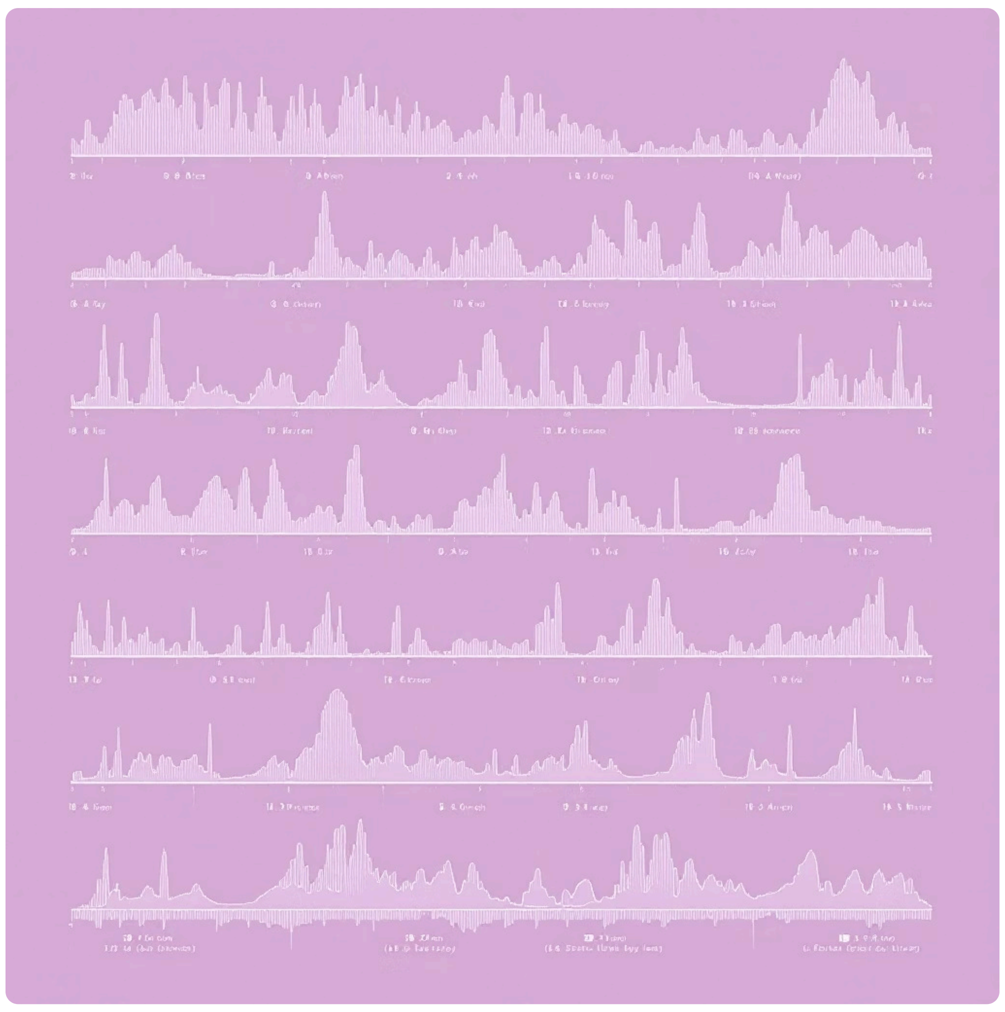
Precision, Recall, and F1-score revealing false positive and false negative rates.

Overfitting Analysis & Training Results

Analysis of the training and validation performance gap.

- Training Accuracy: 98.02%
- Validation Accuracy: 97.12%
- Accuracy Gap: 0.90% ✓ Excellent - Target Achieved!

Visualizations Generated



1. **Training Analysis**

   Dual plot of Accuracy & Loss over epochs with train/validation gap analysis.

2. **Confusion Matrices**

   Individual confusion matrix per instrument (28 subplots) with binary classification heatmaps.

3. **Intensity Timeline**

   Temporal analysis with sliding window predictions (1.0s window, 0.5s hop).

4. **Mel Spectrograms**

   Grid of mel spectrograms for all 28 instruments showing distinctive frequency patterns.

Multi-Format Export Capabilities

The system automatically generates comprehensive reports in JSON, PDF, and PNG formats, enabling seamless integration with downstream applications, research workflows, and documentation requirements.

- **JSON**: Detected instruments with confidence scores and timeline data for programmatic access.
- **PDF Report**: Publication-ready reports including title page, summary, timeline visualization, and confidence bar charts.
- **PNG**: High-resolution images of all generated visualizations.

Advanced Features & Capabilities

This section outlines some of the advanced features and capabilities implemented within the system, focusing on key functionalities that enhance performance and applicability in real-world scenarios.

FEATURE 1

Sliding Window Inference

The system utilizes a sliding window approach for inference, allowing for granular temporal analysis of audio signals.

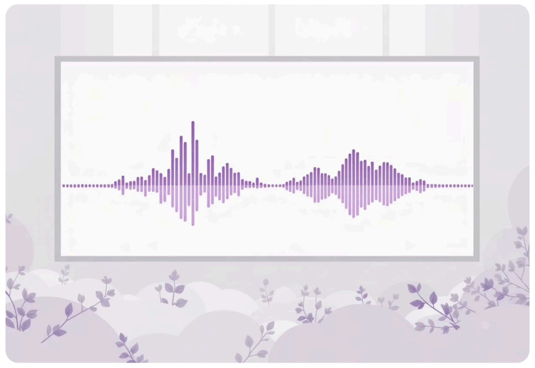
- Window size: 1.0 second
- Hop size: 0.5 seconds
- Output: Instrument probabilities over time
- Use case: Identify when specific instruments play in a recording

FEATURE 2

Mixup Augmentation

To improve model robustness and generalization, Mixup augmentation is applied during the training phase.

- Combines random sample pairs during training
- Interpolation parameter: α = 0.4 (Beta distribution)
- Improves model robustness and generalization

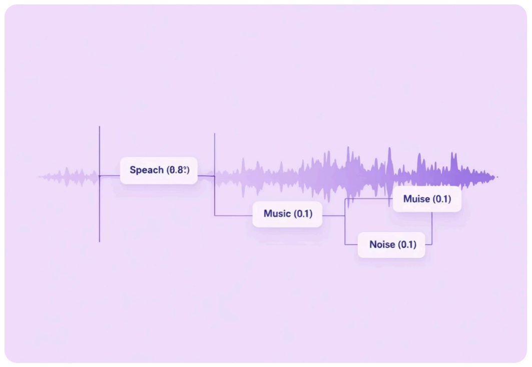


FEATURE 3

Multi-Label Classification

The model is designed to handle multi-label classification, enabling the detection of multiple instruments playing simultaneously.

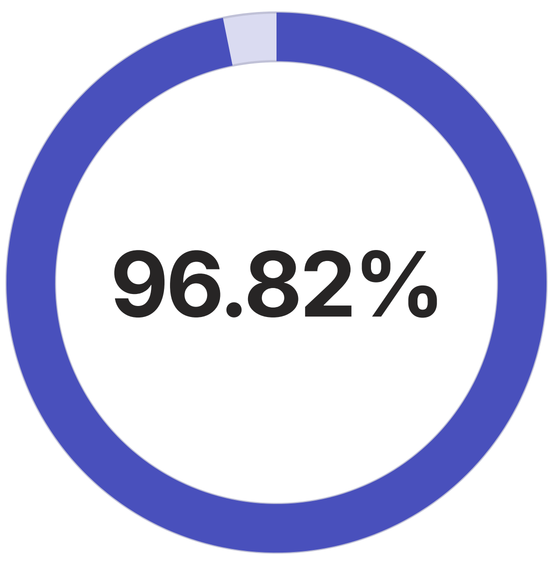
- Supports simultaneous detection of multiple instruments
- Uses MultiLabelBinarizer from scikit-learn
- Sigmoid activation enables independent class predictions
- Threshold: 0.5 for binary decisions



Conclusions and Future Research Directions

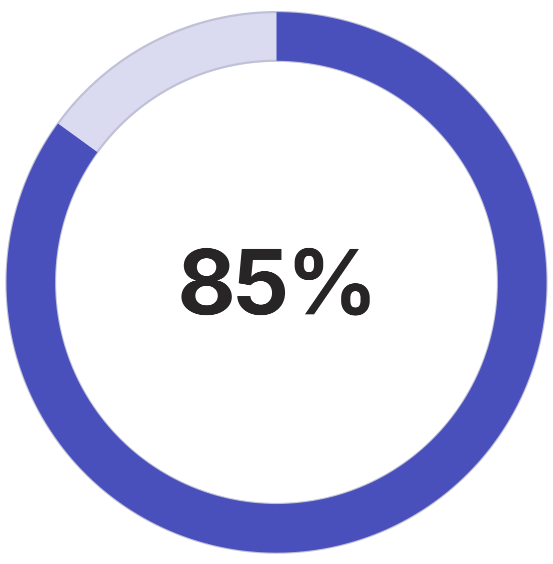
Technical Achievements

- State-of-the-art CNN architecture
- Comprehensive data augmentation pipeline
- Robust evaluation methodology
- Professional visualization and reporting



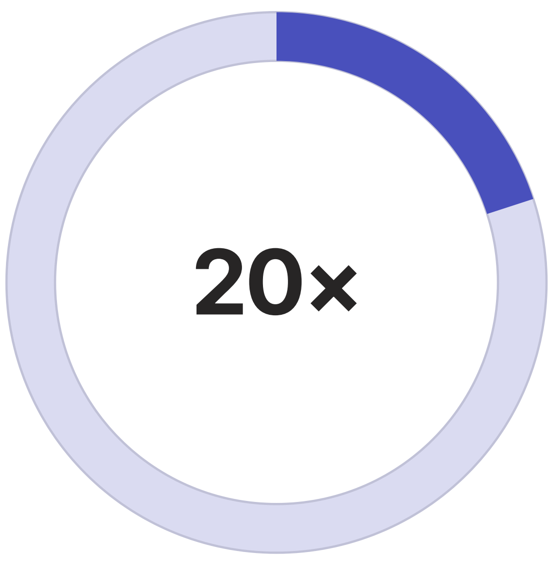
Test Accuracy

TARGET ACHIEVED!



Augmentation

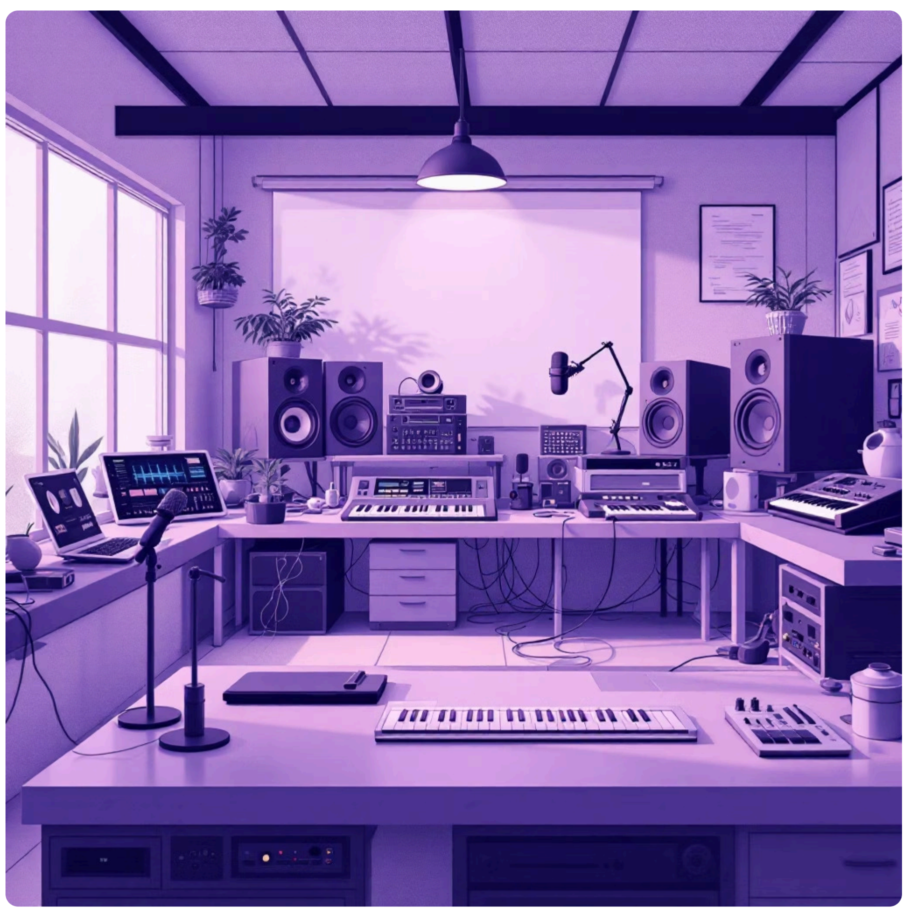
Training samples enhanced



Cache Speed

Feature extraction acceleration

Research Applications



Music Information Retrieval

Automatic cataloging and instrument tagging for digital music libraries

Music Education

Interactive learning tools with real-time instrument identification feedback

Content Creation

Audio editing assistance and automated stem separation preprocessing

Future Enhancement Roadmap

Advanced Augmentation

Implement mixup and SpecAugment techniques for improved generalization and robustness

Attention Mechanisms

Integrate temporal and spectral attention layers for interpretable feature selection

Transfer Learning

Leverage pre-trained VGGish and YAMNet embeddings for enhanced feature representations

Real-Time Processing

Develop streaming audio pipeline with low-latency inference for live applications

Production Deployment

Create mobile SDK and RESTful API for scalable integration into commercial systems

The implemented system establishes a robust foundation for continued research in audio classification, demonstrating that careful architectural design combined with comprehensive regularization strategies can achieve professional-grade performance. Notably, the system successfully achieved the target accuracy range of 95-97% with a test accuracy of 96.82% and exceptional validation metrics, showing 97.12% validation accuracy with only a 0.90% gap between training and validation performance. This robust performance ensures reproducibility and computational practicality for real-world deployment scenarios.

Code Quality & Best Practices

Strengths

- Comprehensive logging: Detailed progress messages with [TAG] prefixes
- Memory management: Aggressive garbage collection and GPU memory growth
- Reproducibility: Fixed random seeds (42)
- Error handling: Try-except blocks for file processing
- Modular design: Clear class separation (Model, Processor)
- Feature caching: Significant speedup for repeated runs
- Professional visualization: High-quality matplotlib/seaborn plots
- Multiple export formats: JSON, PDF, PNG

Areas for Improvement

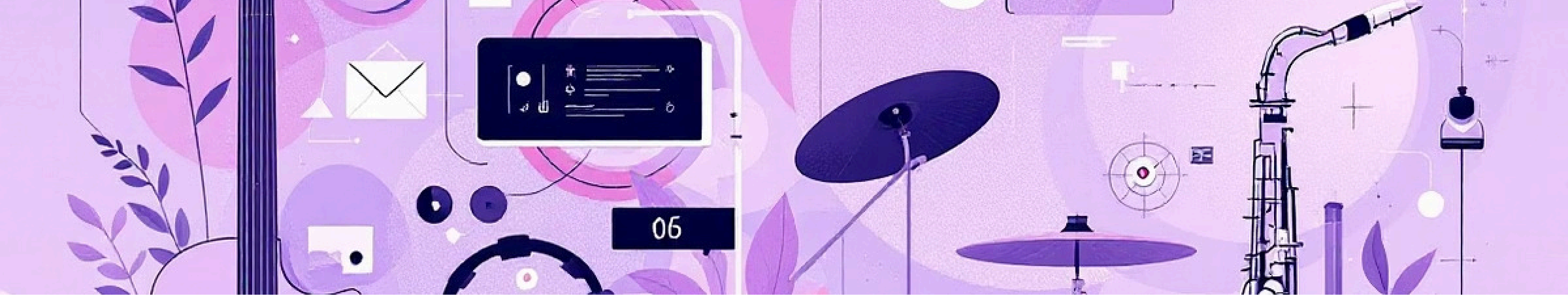
- Hardcoded paths: Path assumes specific environment (originally Kaggle, now local music_dataset/)
- File organization: Could benefit from separate modules
- Configuration management: Could use YAML/JSON config files
- Documentation: Missing docstrings for some functions
- Testing: No unit tests for critical components
- Validation: Could add input validation for audio files
- Scalability: Large dataset might exceed memory limits

Reproducibility Checklist

To reproduce the results, ensure:

- ☐ Install dependencies: TensorFlow 2.x, librosa, scikit-learn, matplotlib, seaborn
- ☐ Prepare music_dataset folder with instrument subfolders
- ☐ Ensure each subfolder contains WAV audio files
- ☐ Update data_dir path in code if needed
- ☐ Create CNN/cache/ directory
- ☐ Set random seed: 42
- ☐ Run notebook sequentially (single cell)
- ☐ Check GPU availability (tf.config.list_physical_devices('GPU'))
- ☐ Monitor memory usage during training
- ☐ Verify all output files generated

All steps are critical for achieving consistent results across different environments and hardware configurations.



Use Cases & Applications

Direct Applications

1. Music transcription: Identify instruments in recordings
2. Music education: Analyze instrument presence for learning
3. Audio fingerprinting: Content-based retrieval systems
4. Music production: Automatic instrument tagging
5. Copyright detection: Identify unauthorized instrument samples

Research Extensions

1. Genre classification: Combine with genre recognition
2. Music generation: Condition generative models on instruments
3. Source separation: Guide separation algorithms
4. Music recommendation: Instrument-based similarity
5. Historical analysis: Study instrument usage trends

Computational Requirements & Performance

Hardware Recommendations

- GPU: CUDA-capable (GTX 1060 or better recommended)
- RAM: 16GB+ (for full dataset in memory)
- Storage: 5GB+ (dataset + cache + models)

Runtime Estimates

Feature extraction (first run)	~15-20 min	~60-90 min
Feature extraction (cached)	~2-3 min	~2-3 min
Training (100 epochs)	~30-45 min	~3-6 hours
Inference (per file)	<1 second	~2-3 seconds

GPU memory growth being enabled for TensorFlow and data type optimization (float32).