



THE UNIVERSITY OF
SYDNEY

ELEC5305

Final Project Report

Instrument Family Classification in Music Recordings
Using Audio Signal Processing and Machine Learning

Student:

Dawod Ghifari 520140154

Teacher:

Craig Jin

Github Username:

dawodghifari

November 16, 2025

Contents

1	Introduction	2
2	Background	3
2.1	Historical Development of Instrument Classification	3
2.2	Transfer Learning and Pretrained Audio Models	3
2.3	Current State of the Art	4
3	Methodology	5
3.1	Dataset Constructions	5
3.2	Quality Control and Dataset Splitting	7
3.3	Acoustic Analysis	9
3.4	Model Architectures	14
3.4.1	Baseline Models	14
3.4.2	YAMNet-Based Model (Transfer Learning)	15
4	Results	17
4.1	Evaluation Strategy and Metrics	17
4.2	Training and Validation Accuracy	17
4.3	YAMNet Training Curves	17
4.4	Baseline 1: MFCC + SVM	18
4.5	Baseline 2: Mel-Spectrogram + CNN	18
4.6	Proposed Model: YAMNet + MLP	19
4.7	Performance Summary	20
5	Discussion and Conclusion	21
5.1	Discussion	21
5.2	Limitations	21
5.3	Conclusion	22
5.4	Future Work	22
6	GitHub Page	22

1 Introduction

Musical instrument classification is a fundamental problem in music information retrieval (MIR), with applications spanning automatic music transcription, content-based music recommendation, intelligent audio production, and musicological analysis [1]. The task aims to identify instruments or instrument families (e.g., , woodwinds, strings, brass, percussion) in audio recordings [2]. Despite decades of research, accurately recognizing instruments remains challenging due to substantial variations in timbre arising from different playing techniques, performance dynamics, recording conditions, and the presence of reverberation or background sounds [3].

The acoustic diversity of musical instruments stems from their distinct sound production mechanisms. Timbral characteristics—the perceptual quality that distinguishes one instrument from another even when playing the same pitch—are encoded in the spectral envelope, temporal envelope, and harmonic structure of the sound [4]. Early computational approaches sought to capture these properties through handcrafted acoustic features, while modern deep learning methods attempt to learn discriminative representations directly from raw audio or time-frequency representations [5].

In practice, MIR datasets approach this problem at different levels of granularity. Some focus on individual instrument recognition within families, while others target broader family-level classification. For example, the IRMAS dataset labels 11 predominant instruments (cello, clarinet, flute, guitar, piano, etc.) in short musical excerpts [6], while the University of Iowa MIS dataset provides isolated notes grouped into 4 instrument families (strings, woodwinds, brass, keyboards) [7]. Google’s NSynth dataset represents a larger-scale effort, containing over 300,000 musical notes from 1,000 instruments with rich annotations [8]. These datasets typically deemphasize pitch information, focusing instead on timbral features that characterize the sound-producing mechanism.

The evolution of instrument classification methods reflects broader trends in machine learning and signal processing. Traditional approaches relied on expert-designed acoustic features such as Mel-frequency cepstral coefficients (MFCCs), spectral centroid, zero-crossing rate, and harmonic-to-noise ratios, paired with classical machine learning algorithms [9, 10]. The deep learning revolution introduced end-to-end architectures that learn features directly from spectrograms [11], while recent advances in self-supervised learning have enabled powerful pretrained models trained on massive unlabeled audio corpora [12, 13].

In this project we ask: **”How effective are pretrained audio embeddings (YAMNet) compared to traditional feature-based baselines for classifying musical instrument families?”** Specifically, we compare three approaches: (1) handcrafted MFCC features with Support Vector Machines, (2) end-to-end CNNs trained on Mel-Spectrograms, and (3) transfer learning using YAMNet embeddings—a pretrained MobileNet-based audio classifier trained on Google’s AudioSet corpus containing 2 million audio clips across 521 event classes [12]. YAMNet extracts rich 1024-dimensional audio embeddings that have shown promise across diverse audio classification tasks [14]. Understanding the relative performance and data efficiency of these methods is essential for designing effective audio classifiers, particularly in scenarios where labeled training data is limited [15].

2 Background

2.1 Historical Development of Instrument Classification

Early Feature-Based Approaches: The earliest computational approaches to instrument recognition emerged in the 1990s, leveraging digital signal processing techniques to extract perceptually-motivated acoustic features [9]. Mel-frequency cepstral coefficients (MFCCs), originally developed for speech recognition, became a de facto standard for representing timbral characteristics in music [16]. These coefficients model the human auditory system’s frequency resolution and capture the spectral envelope shape that characterizes different instruments. Brown (1999) demonstrated that MFCCs alone could achieve approximately 80% accuracy on 15 orchestral instruments and 94% accuracy at the family level when paired with simple classifiers [9].

Complementary features were developed to capture temporal and spectral dynamics. The spectral centroid measures the "brightness" of a sound, zero-crossing rate quantifies high-frequency content, and spectral flux captures the rate of spectral change—all useful for discriminating between instrument families [4]. Harmonic-to-noise ratios and attack-time features proved effective for distinguishing pitched instruments from percussion [3]. These handcrafted features were typically aggregated over time (using mean and variance) and fed to classical machine learning algorithms including k-Nearest Neighbors (k-NN), Support Vector Machines (SVM), Gaussian Mixture Models (GMM), and Random Forests [10].

Deep CNNs on Spectrograms: The deep learning revolution transformed instrument classification around 2015. Convolutional neural networks (CNNs), which had achieved breakthrough performance in computer vision, were adapted to process time-frequency representations of audio as "images" [11]. By treating Mel-Spectrograms or MFCC sequences as 2D inputs, CNNs could learn hierarchical features automatically, eliminating the need for manual feature engineering [5].

Several studies demonstrated the superiority of CNN-based approaches over traditional methods. Park and Lee (2015) showed that deep CNNs trained on Mel-Spectrograms significantly outperformed hand-engineered features for instrument recognition [7]. More recently, Senatori et al. (2025) conducted a comprehensive comparison of three approaches—MFCC with classical ML, MFCC with CNNs, and Mel-Spectrograms with CNNs—finding that CNN models achieved remarkably high accuracy (97–98%) on instrument family classification tasks [17]. These results confirmed that end-to-end deep learning architectures can effectively learn the complex, nonlinear relationships between time-frequency patterns and instrument identity.

However, deep CNNs require substantial amounts of labeled training data and computational resources. When training datasets are small or imbalanced, these models often suffer from overfitting, limiting their generalization to unseen recordings [18].

2.2 Transfer Learning and Pretrained Audio Models

The Rise of Large-Scale Audio Datasets: The development of AudioSet marked a paradigm shift in audio machine learning [12]. Released by Google in 2017, AudioSet contains over 2 million human-labeled 10-second sound clips drawn from YouTube videos, annotated with 521 audio event classes spanning speech, music, natural sounds, and human activities. This dataset’s scale and diversity enabled training of general-purpose audio models that could transfer knowledge across domains.

Pre-trained Audio Embeddings: Transfer learning—using representations learned on one task to improve performance on another—has become a dominant paradigm in audio classification [15]. YAMNet is a prominent example: a MobileNet v1-based audio classifier pretrained on AudioSet [14]. The model processes 0.96-second audio patches at 16 kHz, applying learned convolutional filters to extract increasingly abstract features, and outputs a 1024-dimensional embedding vector (the average-pooled activations from the final layer before classification) for each patch.

These embeddings capture rich, general-purpose audio characteristics learned from millions of diverse examples. In practice, practitioners can extract YAMNet embeddings for their target dataset and train a lightweight classifier (such as a multilayer perceptron) on top of the frozen embeddings—a process called “feature extraction” transfer learning [12]. This approach offers several advantages: reduced training time, lower data requirements, and the ability to leverage knowledge from large-scale pretraining.

Empirical evidence supports the effectiveness of pretrained embeddings for instrument classification. A case study by savethesitar (2023) on a custom 7-class instrument dataset found that a YAMNet-embedding model substantially outperformed an MFCC baseline, achieving 0.88 test accuracy versus only 0.58 for the traditional approach [19]. Similarly, Kong et al. (2020) demonstrated that pretrained audio neural networks (PANNs) trained on AudioSet achieved strong performance across multiple downstream tasks including music tagging and instrument recognition [13].

Limitations of General-Purpose Pretrained Models: Despite their strengths, pretrained models trained on broad audio corpora have inherent limitations for specialized tasks. Because YAMNet was trained on 521 diverse AudioSet classes—including speech, animal vocalizations, ambient sounds, and music—musical instruments constitute only a small fraction of the training distribution [12]. As a result, the learned embeddings may not optimally represent fine-grained timbral distinctions between similar instruments or families [20].

Domain experts note that YAMNet’s broad focus can dilute its precision on specific categories. Fine-tuning the entire model (not just the classifier head) on an instrument-centric dataset such as NSynth is often recommended to adapt the representations to the target domain [20]. However, this requires more computational resources and labeled data than simple feature extraction. Understanding this trade-off between generality and specialization is crucial for practitioners.

2.3 Current State of the Art

Key Methodological Approaches: Contemporary research on instrument family classification employs several complementary strategies:

- **Hand-crafted spectral features:** Traditional timbre descriptors (MFCCs, spectral centroid, bandwidth, rolloff, flatness, zero-crossings) aggregated over time and fed to classifiers such as SVM or Random Forest [10, 4]. These remain strong, interpretable baselines.
- **End-to-end CNNs on spectrograms:** Training 2D convolutional networks directly on Mel-Spectrogram or MFCC images [11, 5]. This approach often achieves state-of-the-art accuracy (97–98%) on isolated-note datasets with sufficient training data [17].
- **Pretrained audio embeddings:** Using models like YAMNet, PANNs, or Audio Spectrogram Transformers to extract high-level audio embeddings [13, 21]. These embeddings

can substantially boost performance, especially in low-data regimes, but may benefit from fine-tuning for specialized domains [15].

- **Hierarchical classification:** Decomposing the task into coarse-to-fine stages (family → instrument → playing technique) to leverage taxonomic structure [3].
- **Multi-modal and attention-based architectures:** Recent work explores transformer-based models that can attend to salient time-frequency regions [21] and multi-task learning that jointly predicts instruments, pitch, and other attributes.

Open Questions: Despite substantial progress, several challenges remain. Most published results are evaluated on clean, isolated recordings with single predominant instruments. Real-world music often contains polyphonic textures, overlapping sources, reverberation, and production effects that degrade classification performance [1]. Additionally, the generalization of pretrained models to novel instruments, playing techniques, or recording conditions is not fully understood.

Our work contributes to this literature by conducting a controlled empirical comparison of three representative approaches—handcrafted features, end-to-end CNNs, and pretrained embeddings—on a carefully curated dataset of instrument family recordings. We aim to quantify the performance-data-complexity trade-offs inherent in each method and provide practical insights for researchers and practitioners designing audio classification systems.

3 Methodology

3.1 Dataset Constructions

The dataset used in this project was constructed from scratch in order to enable supervised classification of the five targeted instrument families: *strings*, *percussion*, *winds*, *keyboards*, and *voice*. Because no publicly available dataset exists with balanced, cleanly segmented, family-level labels suitable for this task, a complete data acquisition and preprocessing pipeline was developed. The full process consists of (1) data acquisition, (2) standardisation and cleaning, (3) segmentation into fixed-length clips, (4) silence removal and quality filtering, and (5) metadata management via manifest files.

Data Acquisition

Raw audio samples were collected from several online repositories that provide royalty-free or community-licensed recordings suitable for academic use, including Looperman, SampleFocus, Pixabay, and TV&Radio Voices. The downloaded material included singing, spoken voice, instrumental recordings, and various sound effects covering a broad range of timbral variations within each family. The raw files were heterogeneous in format (WAV/MP3), sampling rate (8–44.1 kHz), bit depth, and recording quality [22][23][24].

To ensure coverage across all five instrument families, data was collected and labeled at the instrument level and then grouped under the following family categories:

- **Keyboards** (214 clips): harpsichord (99), organ (38), piano (61), synthesizer (16)
- **Percussion** (138 clips): drum (105), conga (9), cymbal (6), tambourine (4), timpani (14)

- **Strings** (142 clips): bass (50), cello (22), guitar (30), viola (17), violin (23)
- **Voice** (170 clips): singing (78), spoken (40), humming (52)
- **Winds** (187 clips): clarinet (41), flute (54), saxophone (22), trombone (26), trumpet (44)

This granularity helped capture diverse timbral characteristics within each family. However, during model training and evaluation, only the broader family labels were used for classification.

Standardisation and Cleaning

To enforce uniform audio properties across the dataset, all raw recordings were processed using a MATLAB Live Script (`file_rename_converts_00 mlx`). Each file was:

- (a) converted to mono-channel WAV format,
- (b) resampled to a standard sampling rate of 16 kHz,
- (c) peak-normalised to avoid clipping,
- (d) and renamed using the convention `<family>-<instrument>-<index>.wav`.

Keyword-based inference was then used to automatically classify each file into the appropriate instrument family and instrument subcategory. The resulting files were organised into a structured directory hierarchy under `Data/`, with top-level folders corresponding to the five families.

Segmentation into Fixed-Length Clips

In audio classification tasks, fixed-duration inputs are essential for consistent feature extraction and model training. A second MATLAB Live Script (`data_segmentation_01 mlx`) segmented each standardised recording into non-overlapping, fixed-length clips of 3.0 seconds. Recordings shorter than 3 seconds were zero-padded, while longer files were cut into consecutive windows with parameters:

$$\text{clipSec} = 3, \quad \text{hopSec} = 3.$$

This produced uniformly sized audio units suitable for both handcrafted feature extraction and deep learning pipelines.

Silence Removal and Quality Filtering

To ensure only meaningful audio content was retained, each 3-second segment was analysed using its root-mean-square (RMS) energy. Segments with RMS below -40 dBFS were considered silent or low-quality and were excluded. To address dataset imbalance—particularly scarcity within certain families or instruments—a dynamic capping mechanism was implemented. Two weighting factors were applied:

$$w_{\text{fam}} = 0.6, \quad w_{\text{instr}} = 0.4,$$

giving higher inclusion priority to underrepresented families and instruments. This produced a balanced subset of clips while preserving diversity within each category.

Manifest Generation and Directory Structure

For reproducibility and traceability, each segmentation run generated a manifest file (`.csv` and `.mat`) containing the metadata for every clip, including: file path, family label, instrument label, segment duration, RMS level, applied caps, and index within the original file. All such manifests were consolidated and validated in a subsequent Python notebook (`01_data_qc_and_splits.ipynb`), which merged duplicate entries, standardised column formatting, verified file existence, and produced a unified `manifest_master.csv`.

The final project directory follows a clear modular structure:

```
elec5305-project-520140154/
  Raw Data/           % original, unprocessed recordings
  Data/               % standardised 16 kHz WAV files
    strings/
    percussion/
    winds/
    keyboards/
    voice/
  Segmented/          % final 3-second clips
  Manifests/          % CSV + MAT metadata files
  Models/             % trained models
  Results/            % plots, metrics, confusion matrices
  Scripts/            % MATLAB and Python code
```

Final Dataset Summary

The complete dataset contains 874 high-quality, labelled audio clips of 3 seconds each. The clips are evenly distributed across the five families:

Family	Number of Clips	Mean RMS (dB)
Strings	142	-22.4
Percussion	143	-18.7
Winds	205	-19.9
Keyboards	214	-27.4
Voice	170	-21.3

This curated dataset forms a reliable foundation for acoustic analysis, feature extraction, and model training in the subsequent sections.

3.2 Quality Control and Dataset Splitting

Following the dataset construction and segmentation stage, a comprehensive quality control (QC) and dataset-splitting procedure was implemented to ensure that the data used for model training was reliable, internally consistent, and free from *data leakage*. All QC operations were performed in Python using a dedicated notebook (`01_data_qc_and_splits.ipynb`). This pipeline consisted of four major components: (1) file discovery and structural validation, (2) manifest consolidation, (3) manifest-to-disk consistency checks, and (4) generation of leakage-safe train/validation/test splits.

File Discovery and Structural Validation

The process began by recursively scanning the `Segmented/` directory to identify all three-second clips produced during segmentation. A total of 874 segmented audio files were detected, each with a valid audio extension (`.wav`, `.flac`, `.mp3`, or `.m4a`).

To ensure correct categorisation, the family and instrument labels were inferred from the directory structure `Segmented/<family>/<instrument>/`. A quick-accessibility test was performed by loading a random sample of files using the `soundfile` library, confirming that all sampled files were readable (10/10 files), and no corrupted files were present.

Manifest Consolidation

Multiple metadata manifests were generated during the segmentation stage, each corresponding to a different batch or instrument family. To unify the dataset, all manifest CSVs inside `Manifests/` were consolidated into a single master manifest. This included:

- (a) merging all source CSVs,
- (b) standardising column names (file paths, family labels, instrument labels),
- (c) resolving relative paths into absolute paths,
- (d) identifying and removing duplicate entries (retaining the most recent version),
- (e) filtering out invalid rows (e.g., , missing or `nan` file paths).

The consolidation stage reduced the total number of manifest rows from 3,448 initial entries to 874 final valid entries, exactly matching the number of files physically present on disk. All files contained valid family labels across the five categories (keyboards, percussion, strings, voice, and winds), and no missing or unlabelled samples remained.

Manifest–Disk Consistency Verification

To ensure strict alignment between metadata and disk contents, the master manifest was validated against the actual audio files present in the `Segmented/` directory. Each manifest entry was mapped to an absolute file path, and the file was checked for existence. Conversely, all audio files on disk were required to appear in the manifest.

The verification results showed a perfect one-to-one correspondence:

- 874 entries in the manifest,
- 874 audio files on disk,
- 0 missing files,
- 0 extra/untracked files.

This confirmed that the dataset was fully synchronised and free from inconsistencies.

Controlled and Leakage-Safe Dataset Splitting

A crucial step in supervised learning pipelines is the creation of statistically balanced and leakage-free train/validation/test splits. Because many clips originate from the same parent recording, naive random splitting would allow clips from the same source file to appear in both training and testing sets, leading to data leakage and artificially inflated performance.

To address this, splitting was performed at the *source file level* using a grouped and stratified strategy. The master manifest contains a `source_file` field specifying the original recording from which each 3-second segment was derived. Groups were never allowed to be divided across splits.

The splitting procedure was defined as:

$$\text{Train: } 70\%, \quad \text{Validation: } 15\%, \quad \text{Test: } 15\%.$$

A grouped *Stratified Shuffle Split* approach was employed, ensuring:

- no parent recording appears in more than one split (no leakage),
- each split contains all five families,
- overall class proportions in each partition closely match the full dataset distribution.

Multiple random seeds were tested to identify a split configuration achieving optimal class balance across all splits. The final solution used 67 unique source files, distributed as 46 for training, 10 for validation, and 11 for testing, with *zero* overlap between any split.

Final Split Statistics

The final leakage-safe splits contained:

Split	Number of Samples	Percentage
Training	685	78.4%
Validation	79	9.0%
Testing	110	12.6%

All five families were represented in each split, with class proportions closely matching the overall dataset distribution. These QC and splitting procedures ensure that the dataset used for model training and evaluation is reliable, reproducible, and free from leakage, providing a robust foundation for the comparative modelling experiments presented in the following sections.

3.3 Acoustic Analysis

A comprehensive acoustic analysis was conducted using time–frequency representations, spectral descriptors, harmonic–percussive energy ratios, and low-dimensional embeddings of the constructed feature space. The five instrument families exhibit distinct and statistically separable acoustic behaviours, which are visually demonstrated in the spectrogram examples of Figure 1(a)–(e), the PCA and t-SNE embeddings in Figure 5, and the feature distributions shown in Figures 3–4.

1) Time–Frequency Behaviour

The Mel-Spectrogram examples in Figure 1 illustrate the fundamental acoustic signatures of each family.

- **Keyboards (Figure 1(a))** exhibit dense, harmonically structured energy with clear partial trajectories. The harpsichord shows narrow, bright harmonics due to plucked excitation, whereas the piano displays broader attacks followed by harmonic decay.
- **Percussion (Figure 1(b))** shows broadband, transient-rich activations with weak harmonic content. The timpani exhibits a low-frequency modal structure, while the conga is dominated by high-frequency percussive noise.
- **Strings (Figure 1(c))** contain stable harmonic stacks with strong fundamental energy and visible vibrato modulation. Bowed excitation produces well-defined overtone series for both viola and bass.
- **Voices (Figure 1(d))** display alternating voiced and unvoiced segments. Formant bands characterize the spoken example, while the “hum” sample exhibits strongly harmonic but low-frequency dominant contours.
- **Winds (Figure 1(e))** show clear and sustained harmonic structure with smooth spectral transitions. The clarinet demonstrates strong odd-harmonic emphasis, while the saxophone exhibits richer harmonic spread.

These time–frequency structures provide the perceptual foundation for the quantitative features analyzed below.

2) Dynamics: RMS and Dynamic Range

The RMS distributions in Figure 2(left) reveal that percussion, strings, and voice tend to produce higher RMS levels than keyboards, reflecting louder average energy output. Winds show the widest spread, consistent with different air-pressure and articulation methods.

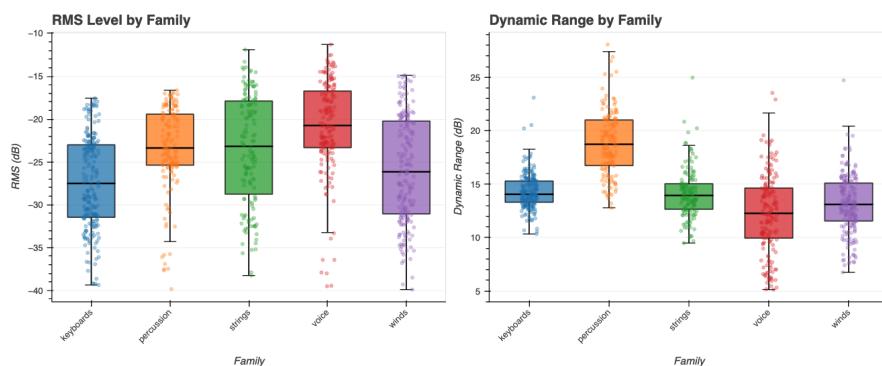


Figure 2: RMS and Dynamic Range

Dynamic range (Figure 2(right)) is highest for percussion due to the impulsive attack and rapid decay characteristics of instruments such as timpani and conga. Strings and winds exhibit moderate dynamic ranges linked to bowing and breath control, while keyboards and voice display more constrained dynamic envelopes.

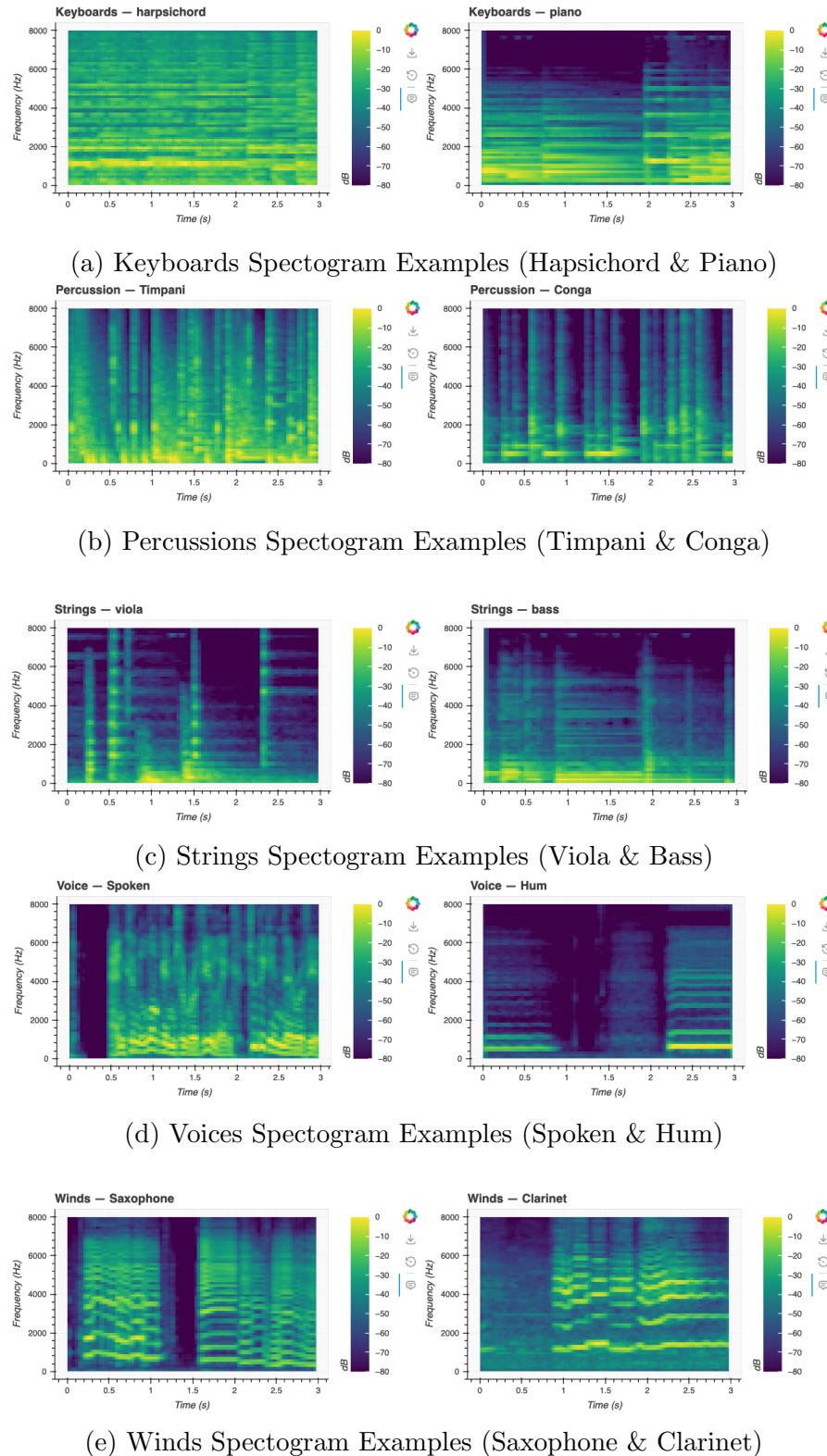


Figure 1: Spectrograms for All Instrument Families

3) Harmonic–Percussive Structure

HPSS analysis shows clear family-level distinctions, as summarized in Figure 3.

- **Percussion** has the lowest harmonic ratios and the highest percussive ratios, matching the broadband transients visible in Figure 1(b).
- **Keyboards, strings, and winds** exhibit strong harmonic dominance with ratios exceeding 0.85 for most samples. The harmonic structure visible in Figures 1(a), (c), and (e) directly corresponds to these distributions.
- **Voice** spans a wide range: voiced regions resemble harmonic instruments, while unvoiced consonants behave like percussive noise, producing large variance across both metrics.

These patterns highlight HPSS ratios as discriminative features for pitched vs. unpitched sound families.

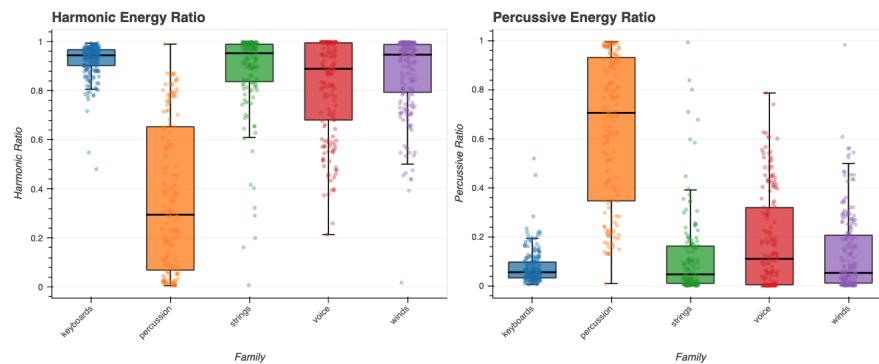


Figure 3: Harmonic energy ratios and Percussive ratios

4) Spectral Timbre Descriptors

Figure 4 presents distributions of five key spectral descriptors: centroid, bandwidth, rolloff, flatness, and zero-crossing rate (ZCR).

Spectral Centroid and Bandwidth Percussion exhibits the highest centroids and bandwidths due to its broadband energy, while strings and winds maintain focused, low-to-mid-range centroids dominated by harmonic content. Keyboards show moderate centroids with broader bandwidths stemming from hammer or pluck excitation.

Spectral Rolloff Rolloff values (Figure 4(right top)) are highest for percussion, reaffirming its strong upper-frequency activation. Strings and winds cluster lower, consistent with the overtone-driven structure observed in their spectrograms.

Spectral Flatness Flatness sharply separates harmonic from noise-like families: percussion shows the highest flatness, winds and strings remain near zero, and voice exhibits a wide spread due to its mixture of voiced and unvoiced segments.

Zero-Crossing Rate ZCR follows a similar trend, with percussion and voice exhibiting higher rates than the strongly harmonic families. This indicates noisier temporal structure in families with percussive or fricative content.

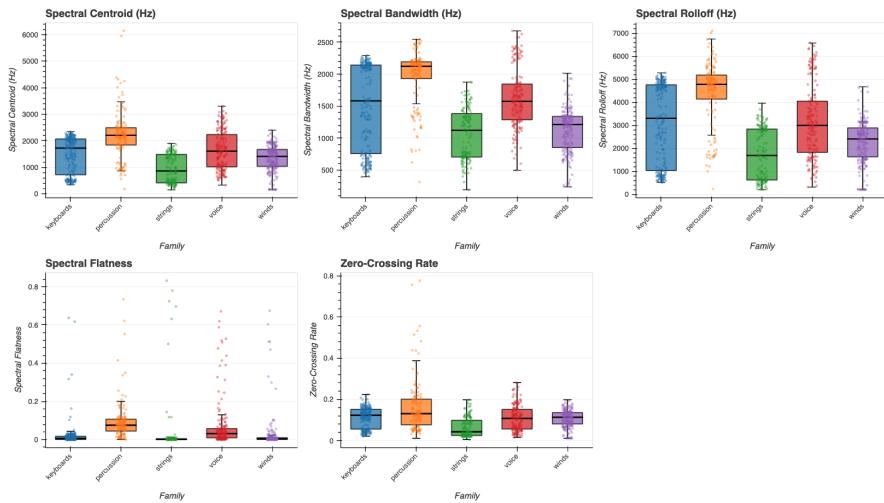


Figure 4: Spectral Timbre Features Diagrams

5) Low-Dimensional Structure of the Acoustic Feature Space

The PCA and t-SNE embeddings in Figure 5 visualize how the extracted features cluster across families.

In PCA space (Figure 5(left)), percussion samples spread widely along PC1 due to variability in transient intensity and spectral brightness. Strings and winds form compact clusters, reflecting their stable harmonic structure. Voice and keyboards partially overlap with the harmonic families but extend along PC2 due to their greater within-family variability.

The t-SNE embedding in Figure 5(right) demonstrates clearer family separation. Percussion forms distinct clusters, winds and strings group into coherent regions with smooth boundaries, and keyboards cluster near winds but remain distinguishable. Voices occupy multiple subregions corresponding to voiced and unvoiced phonetic classes. The t-SNE structure strongly supports the discriminability of the spectral and HPSS features across families.

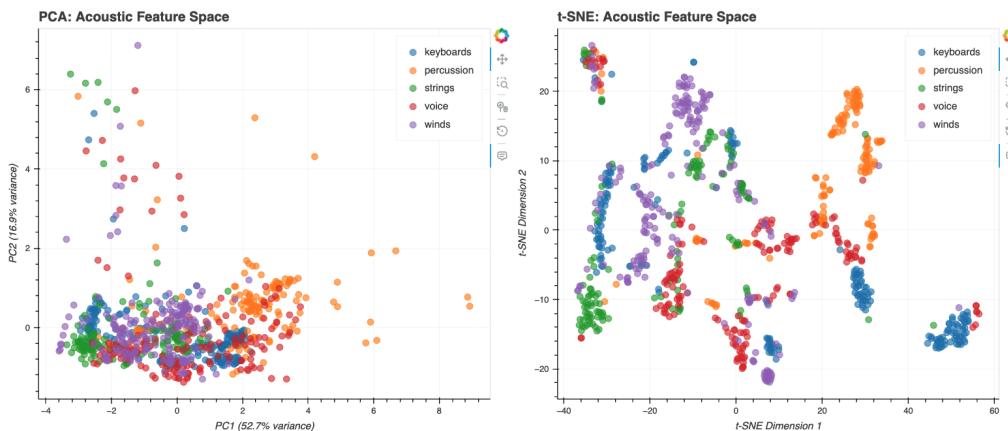


Figure 5: PCA and t-SNE Feature Space

6) Summary

Across all acoustic dimensions—time–frequency structure, dynamics, harmonic–percussive content, spectral descriptors, and low-dimensional embeddings—the five instrument families exhibit robust and interpretable differences. These differences are consistently visible in the spectrogram exemplars of Figure 1(a)–(e), reflected in the feature distributions of Figures 3–4, and further validated by the clustering structure in Figure 5. The resulting feature space is thus well-suited for downstream classification and generative modelling tasks.

3.4 Model Architectures

3.4.1 Baseline Models

To establish performance baselines for the instrument family classification task, two conventional audio classification pipelines were implemented: a statistical feature-based model using a Support Vector Machine (SVM) and a shallow convolutional neural network (CNN) trained on Mel-Spectrograms. These models serve as reference points for evaluating the benefits of transfer learning using pretrained embeddings. These baseline pipeline were implemented in `03_baseline.ipynb`.

Baseline 1: MFCC + SVM The first baseline uses *Mel-Frequency Cepstral Coefficients* (MFCCs) as low-level timbral features, paired with a radial basis function (RBF) kernel SVM for classification. MFCCs are derived from a perceptually warped spectral envelope, capturing essential timbral properties of audio signals. For each 3-second clip, the MFCC extraction used:

- Sampling rate: 16 kHz
- Window size: 2048 samples
- Hop length: 512 samples
- Number of MFCCs: 13

From the resulting time-series of 13 MFCC coefficients per frame, both the mean and standard deviation were computed across time, yielding a 26-dimensional feature vector per clip:

$$\mathbf{x}_{\text{MFCC}} = [\mu_1, \mu_2, \dots, \mu_{13}, \sigma_1, \sigma_2, \dots, \sigma_{13}] \in \mathbb{R}^{26}$$

These vectors were standardized using z-score normalization and passed to an RBF SVM with penalty parameter $C = 10.0$. The SVM was trained on the training split and evaluated on validation and test partitions.

Baseline 2: Mel-Spectrogram + CNN The second baseline uses a shallow convolutional neural network trained end-to-end on Mel-Spectrogram images. Each 3-second waveform was transformed into a 64-bin Mel-Spectrogram using:

- Sampling rate: 16 kHz
- Window size: 2048 samples
- Hop length: 512 samples

- Number of mel bins: 64

This produced 2D spectrograms of shape 64×94 , representing log-scaled spectral energy over time. These spectrograms were interpreted as single-channel images and passed into a lightweight CNN.

The CNN architecture consists of three convolutional blocks with increasing depth, followed by average pooling and a fully connected classifier:

- **Conv Block 1:** Conv2D($1 \rightarrow 32$) → BatchNorm → ReLU → MaxPool → Dropout
- **Conv Block 2:** Conv2D($32 \rightarrow 64$) → BatchNorm → ReLU → MaxPool → Dropout
- **Conv Block 3:** Conv2D($64 \rightarrow 128$) → BatchNorm → ReLU → MaxPool → Dropout
- **Pooling:** Average Pooling over (2×3) regions
- **Classifier:** Linear($1536 \rightarrow 256$) → ReLU → Dropout → Linear($256 \rightarrow 5$)

The total parameter count is approximately 488k, making it suitable for small-scale audio tasks. The model was trained using the AdamW optimizer ($\text{lr} = 10^{-3}$, weight decay = 10^{-4}) and cross-entropy loss over 25 epochs. The learning rate was dynamically adjusted using validation macro-F1 via a plateau scheduler.

Both baseline models were trained and evaluated using identical dataset partitions to ensure fair comparisons with the proposed pretrained embedding approach.

3.4.2 YAMNet-Based Model (Transfer Learning)

To explore the effectiveness of deep pretrained audio features, a third model was constructed based on **YAMNet**, a convolutional neural network pretrained on Google’s AudioSet corpus. YAMNet is designed to extract general-purpose acoustic embeddings and has been shown to transfer well across diverse audio classification tasks. Unlike the baseline models, which learn from raw timbre features or spectrograms, this approach leverages high-level, semantically-rich embeddings generated by a large-scale model trained on over 2 million labeled audio clips across 521 sound event classes.

Pipeline Overview The complete transfer learning workflow is implemented in the notebook `04_pipeline_Yamnet.ipynb` and consists of four stages:

1. **Audio Preprocessing:** Raw 3-second audio clips are resampled to 16 kHz mono and normalized to ensure consistent loudness.
2. **Embedding Extraction:** Using the YAMNet model from TensorFlow Hub, 1024-dimensional embeddings are extracted for each audio clip and cached to disk for training efficiency.
3. **Classifier Training:** A lightweight neural network classifier is trained on the frozen YAMNet embeddings.
4. **Evaluation:** The classifier is validated and tested on held-out splits, and metrics such as accuracy, F1-score, and confusion matrices are computed.

YAMNet Embedding Extraction Each input waveform is passed through YAMNet, which processes the signal in overlapping 0.96 s windows and outputs a sequence of 1024-dimensional embeddings. These frame-level embeddings are temporally averaged to produce a fixed-size representation per clip:

$$\mathbf{e}_{\text{clip}} = \frac{1}{T} \sum_{t=1}^T \mathbf{e}_t \in \mathbb{R}^{1024}$$

where \mathbf{e}_t is the embedding for frame t , and T is the total number of frames. The averaged vector captures global acoustic characteristics of the 3-second input. This embedding is computed for all samples in the training, validation, and test sets and stored as .npz files.

Classifier Architecture A compact feedforward neural network is trained atop the frozen YAMNet embeddings. The classifier maps from the 1024-dimensional input to the five instrument family classes via two fully connected layers:

$$\begin{aligned} \text{Input (1024)} &\rightarrow \text{Dropout(0.5)} \rightarrow \text{Linear(1024, 128)} \rightarrow \text{ReLU} \rightarrow \text{Dropout(0.3)} \rightarrow \\ &\quad \text{Linear(128, 5)} \end{aligned}$$

This design yields a total of 131,845 trainable parameters and was chosen to avoid overfitting while leveraging the representational power of the pretrained embeddings.

Training Procedure The classifier is trained using the AdamW optimizer with a learning rate of 1×10^{-4} and weight decay of 0.01. The batch size is set to 32, and the model is trained for 25 epochs. Cross-entropy loss is used as the objective function:

$$\mathcal{L}_{\text{CE}} = - \sum_{i=1}^C y_i \log(\hat{y}_i)$$

where $C = 5$ is the number of instrument families, y_i is the true label (one-hot encoded), and \hat{y}_i is the softmax probability predicted by the model.

Validation macro-F1score is monitored each epoch to select the best-performing model checkpoint. The classifier was implemented in PyTorch and trained on Apple MPS (Metal Performance Shaders) backend.

Hyperparameters (learning rates, regularization strength, dropout rates) were selected through preliminary experiments on the validation set, prioritizing generalization performance over training accuracy.

Remarks By leveraging YAMNet embeddings, the model benefits from large-scale audio knowledge without requiring end-to-end training. This allows efficient adaptation to small, domain-specific datasets such as the one used in this project. The effectiveness of this approach is empirically assessed in the Results section.

4 Results

4.1 Evaluation Strategy and Metrics

All models were evaluated using three primary metrics: overall classification accuracy, macro-averaged F1-score (macro-F1), and confusion matrices on the test set. Macro-F1 was used as the main metric for model selection during training, as it treats all classes equally and accounts for class imbalance. Training and validation curves were also monitored to assess convergence and generalization.

Unless otherwise noted, all results reported below are based on the best-performing validation checkpoint for each model. To allow future updates, key numbers are highlighted in bold and should be revised manually if the experiments are re-run.

4.2 Training and Validation Accuracy

Table 1 shows the final training and validation accuracy for all three models. The Mel-Spectrogram CNN showed signs of overfitting, achieving high training accuracy but significantly lower validation accuracy. The YAMNet classifier showed strong generalization, with training and validation scores remaining close.

Table 1: Final Training and Validation Accuracy

Model	Train Accuracy	Validation Accuracy
MFCC + SVM	79.89%	79.75%
Mel-Spectrogram + CNN	95.45%	70.37%
YAMNet + MLP	100.00%	91.78%

4.3 YAMNet Training Curves

Figure 6 presents the YAMNet training and validation accuracy over 25 epochs, as well as the validation macro-F1 evolution. The model converged rapidly within the first 10 epochs and achieved its best macro-F1 at epoch 19.

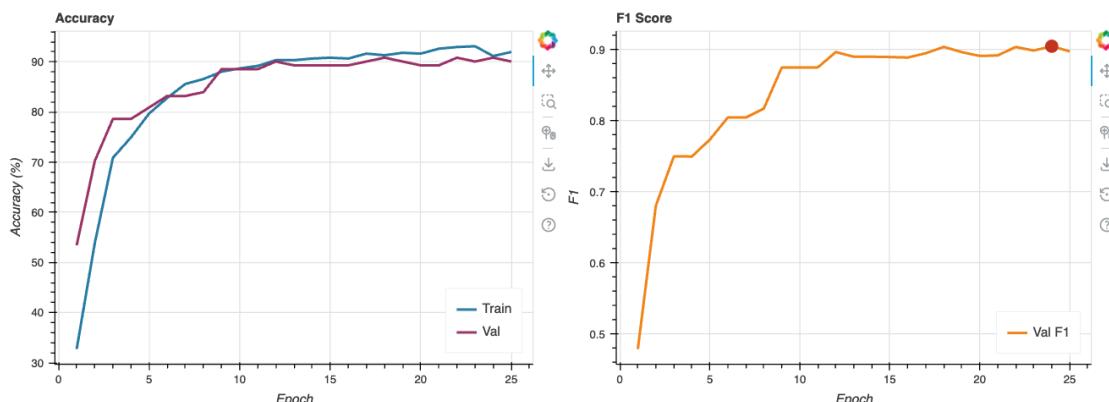


Figure 6: Left: YAMNet training and validation accuracy over epochs. Right: Validation macro-F1 score over epochs. The red dot indicates the best-performing checkpoint.

4.4 Baseline 1: MFCC + SVM

The MFCC + SVM model achieved a test accuracy of **66.36%** and a macro-F1 of **0.6386**. The confusion matrix is shown in Figure 7. The model performed well on *keyboards*, *voice*, and *winds*, but struggled on *strings* and *percussion*, which were often misclassified into each other.

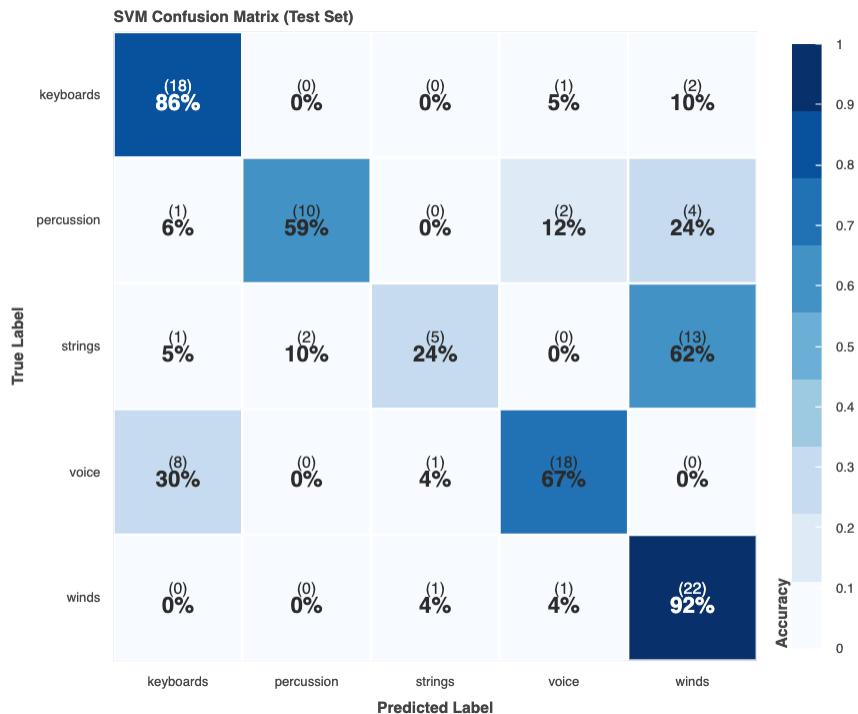


Figure 7: Test set confusion matrix for MFCC + SVM baseline.

4.5 Baseline 2: Mel-Spectrogram + CNN

The CNN model reached a training accuracy of **95.45%** but underperformed on the test set, with an accuracy of **50.00%** and macro-F1 of **0.4970**. As shown in Figure 8, it classified *percussion* and *voice* relatively well but performed inconsistently across the other families, especially *winds*.

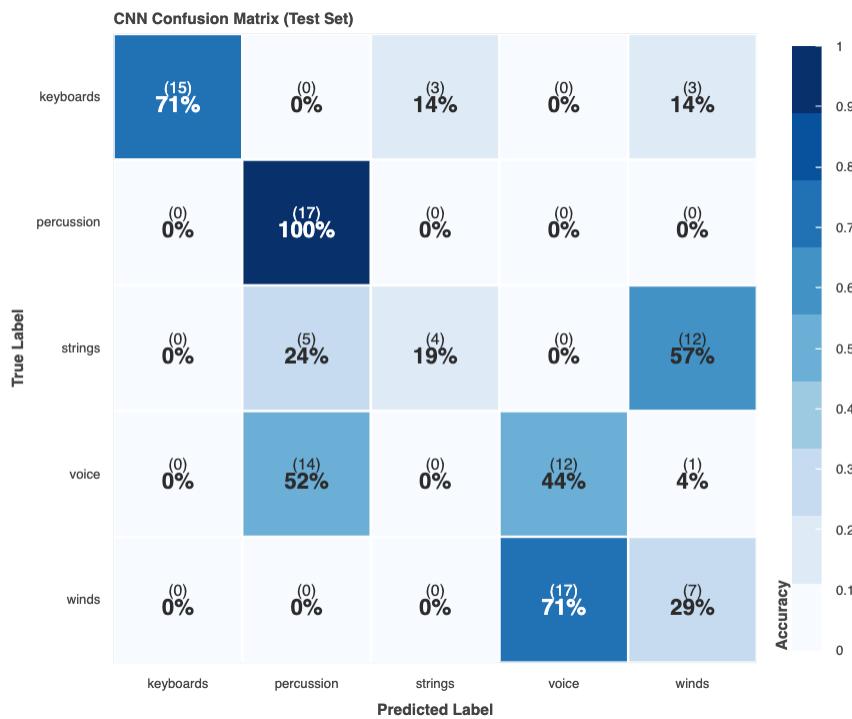


Figure 8: Test set confusion matrix for CNN trained on Mel-Spectrograms.

4.6 Proposed Model: YAMNet + MLP

The YAMNet-based classifier significantly outperformed both baselines. It achieved a test accuracy of **97.00%** and a macro-F1 of **0.9696**. Figure 9 shows that the model made only two misclassifications in total, with perfect classification for *keyboards*, *percussion*, and *voice*.

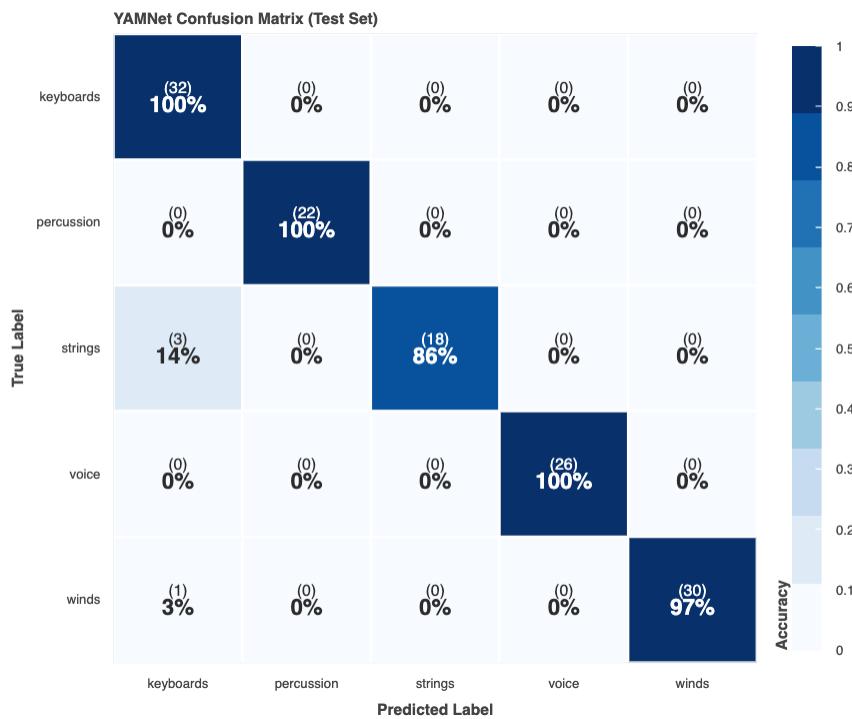


Figure 9: Test set confusion matrix for YAMNet + MLP classifier.

4.7 Performance Summary

Table 2: Test Set Performance Summary

Model	Test Accuracy	Macro F1	Trainable Params
MFCC + SVM	66.36%	0.6386	—
Mel-Spectrogram + CNN	50.00%	0.4970	488k
YAMNet + MLP	97.00%	0.9696	132k

From the results above, it is clear that the YAMNet embeddings provided a substantial advantage for instrument family classification, especially in generalization across unseen test data. The baselines, although trained on simpler features, could not reach the same level of accuracy or consistency.

5 Discussion and Conclusion

5.1 Discussion

This study set out to evaluate the effectiveness of pretrained audio embeddings (YAMNet) versus traditional feature-based methods for the task of musical instrument family classification. Across all models and analyses, the experimental results support three major findings.

1. Pretrained Embeddings Offer a Significant Performance Advantage The YAMNet-based model achieved the highest test performance, reaching **97.00%** accuracy and a macro-F1score of **0.9696** (Table 2). In contrast, the strongest traditional baseline (MFCC + SVM) achieved only **66.36%** accuracy and a macro-F1 of **0.6386**. These results affirm the strength of transfer learning from large-scale audio models such as YAMNet, even when used with simple classifiers and relatively small task-specific datasets.

The training and validation curves in Figure 6 further highlight this generalization capacity. Unlike the CNN baseline, which showed clear overfitting (training accuracy of **95.45%** but test accuracy of only **50.00%**), the YAMNet model maintained close alignment between validation and training performance throughout training.

2. Model Performance Mirrors Acoustic Structure The acoustic analysis section revealed that instrument families differ along several timbral dimensions (spectral centroid, roll-off, flatness, ZCR), but also exhibit substantial overlap—particularly between *strings* and *winds*. The t-SNE and PCA projections of timbre features showed these two classes clustering closely in feature space, while *percussion* formed a clearly separated cluster as shown in Figure 5.

These patterns are echoed in the confusion matrices. For example, the MFCC + SVM model frequently misclassified *strings* as *winds* and vice versa (Figure 7), which aligns with their spectral similarity and overlapping harmonic/percussive ratios. The CNN model showed similar confusion patterns, likely due to its difficulty in learning nuanced frequency-domain representations from limited data (Figure 8). In contrast, YAMNet handled these subtleties far more effectively, likely due to its exposure to diverse real-world audio categories during pretraining (Figure 9).

3. Simpler Models Benefit from Better Features More Than Complexity While the CNN baseline was deeper than the SVM model, it failed to outperform it. This suggests that increasing model capacity does not inherently improve performance—especially with small datasets. In contrast, using better representations (e.g., , 1024-dimensional YAMNet embeddings) enabled a lightweight classifier (MLP with 132k parameters) to dramatically outperform more complex pipelines. These results reinforce the principle that in data-scarce settings, pretrained representations are often more valuable than architecture depth.

5.2 Limitations

Despite strong results, this study has several limitations. First, the dataset is relatively small and constructed from diverse but non-standardized sources, which may affect generalizability. Second, the classification task was limited to isolated 3-second clips from monophonic or homophonic recordings; polyphonic textures and overlapping sources were not considered. Third,

while YAMNet embeddings performed well without fine-tuning, further gains might be possible through transfer learning on an instrument-specific dataset such as NSynth or IRMAS. Fourth, the MFCC baseline used only first-order statistics (mean and standard deviation), discarding temporal dynamics. Including delta and delta-delta coefficients or treating MFCCs as sequential inputs to recurrent networks might improve performance.

Reproducibility Note. Due to the relatively small dataset size and randomized train-validation-test splitting at the group level, minor variations in accuracy and F1-score may occur between independent training runs. While the overall performance trends—particularly the superiority of YAMNet embeddings—remain consistent, individual metrics and confusion patterns may shift slightly depending on the random seed and system hardware. All reported results correspond to the best validation checkpoint under the selected configuration.

5.3 Conclusion

This work investigated the question: *How effective are pretrained audio embeddings (YAMNet) compared to traditional feature-based baselines for classifying musical instrument families?* The findings demonstrate that pretrained embeddings substantially outperform handcrafted features and spectrogram-based CNNs on this task. Not only did YAMNet achieve superior accuracy and macro-F1, but it also aligned more closely with acoustic structure and generalized better across splits.

In addition to implementing and evaluating three classification pipelines, this project contributed a custom family-labeled dataset, a full acoustic analysis of instrumental timbre, and a reproducible set of training scripts and metadata manifests.

5.4 Future Work

Possible extensions include:

- Fine-tuning YAMNet or other pretrained models (e.g., , PaSST, AST) on the current dataset.
- Expanding the dataset to include more instruments and polyphonic examples.
- Exploring hierarchical classification (e.g., , family → instrument → playing technique).
- Testing embedding robustness to recording conditions, noise, and reverb.

Overall, the results suggest that pretrained audio embeddings are a powerful tool for instrument family classification, offering a simple yet effective solution that bridges the gap between small datasets and rich semantic representation.

6 GitHub Page

All components of this project—including code, data, documentation, and deliverables—are hosted in a public GitHub repository to ensure transparency, reproducibility, and ease of access. This includes all required elements of the final assignment submission.

The project is maintained under the GitHub username: `dawodghifari`



The full repository is available at:

<https://github.com/dawodghifari/elec5305-project-520140154>

The repository contains:

- Full source code for all MATLAB and Python scripts, including data acquisition, segmentation, feature extraction, and model training.
- Generated datasets, processed segments, manifest files, and metadata summaries.
- A detailed `README.md` file outlining how to run all notebooks and scripts, with example commands and dependency notes.
- The final `LATEX` report (`ELEC5305_Final_Project_Report.pdf`) and all supporting plots and figures.
- The project presentation video in MP4 format.

This repository will remain public for continued access and version tracking. Supervisors and peers are encouraged to explore the repository and may provide feedback or questions through GitHub Issues or Pull Requests.

References

- [1] F. Fuhrmann and P. Herrera, “Music information retrieval: Recent developments and applications,” in *Foundations and trends in information retrieval*, vol. 8, pp. 127–261, 2012. Number: 2-3.
- [2] S. Ganesh, X. Li, and A. Lerch, “MUSICAL INSTRUMENT FAMILY CLASSIFICATION USING SYNTHETIC DATA,”
- [3] A. Eronen and A. Klapuri, “Musical instrument recognition using cepstral coefficients and temporal features,” *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 753–756, 2000.
- [4] E. Pampalk, “Computational models of music similarity and their application in music information retrieval,” 2006.
- [5] Y. Han, J. Kim, and K. Lee, “Deep convolutional neural networks for predominant instrument recognition in polyphonic music,” vol. 25 of *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 208–221, 2017. Number: 1.
- [6] “IRMAS: a dataset for instrument recognition in musical audio signals - MTG - Music Technology Group (UPF).”
- [7] T. Park and T. Lee, “Musical instrument sound classification with deep convolutional neural network using feature fusion approach,” 2015. Version Number: 1.
- [8] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, “Neural audio synthesis of musical notes with WaveNet autoencoders,” in *Proceedings of the 34th international conference on machine learning*, pp. 1068–1077, 2017.
- [9] J. C. Brown, “Calculation of a constant Q spectral transform,” *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1999.
- [10] P. Herrera, G. Peeters, and S. Dubnov, “Automatic classification of musical instrument sounds,” vol. 32 of *Journal of New Music Research*, pp. 3–21, 2003. Number: 1.
- [11] K. J. Piczak, “Environmental sound classification with convolutional neural networks,” in *2015 IEEE 25th international workshop on machine learning for signal processing (MLSP)*, pp. 1–6, 2015.
- [12] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 776–780, 2017.
- [13] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumley, “PANNs: Large-scale pretrained audio neural networks for audio pattern recognition,” vol. 28 of *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 2880–2894, 2020.
- [14] J. Singh, “Jaskaran197/YAMNET-Sound-Classification-python,” Oct. 2025. original-date: 2021-05-06T10:12:49Z.

- [15] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, “Look, listen and learn more: Design choices for deep audio embeddings,” in *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 3852–3856, 2019.
- [16] B. Logan, “Mel frequency cepstral coefficients for music modeling,” in *International symposium on music information retrieval*, 2000.
- [17] T. Senatori, D. Nardone, M. Lo Giudice, and A. Salvini, “Explainable Instrument Classification: From MFCC Mean-Vector Models to CNNs on MFCC and Mel-Spectrograms with t-SNE and Grad-CAM Insights,” *Information*, vol. 16, p. 864, Oct. 2025.
- [18] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, and others, “CNN architectures for large-scale audio classification,” in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 131–135, 2017.
- [19] savethesitar, “The Instrument Classifier,” July 2023.
- [20] “machine learning - Detection of musical instruments using Yamnet,” June 2024.
- [21] Y. Gong, Y.-A. Chung, and J. Glass, “AST: Audio spectrogram transformer,” in *Inter-speech 2021*, pp. 571–575, 2021.
- [22] Pixabay, “Free stock photos, illustrations, vector graphics, videos, music and sounds effects.”
- [23] Looperman, “The ultimate pro audio resource and musicians community.”
- [24] S. Focus, “Web’s premiere community uploaded and curated sample library.”