Audio Clustering Report

1 Introduction

This assignment focuses on clustering audio files based on extracted features. Using unsupervised learning techniques, the aim is to group similar audio clips without supervision. The clustering performance is evaluated using the Adjusted Rand Index (ARI), which compares the similarity between predicted and true clusters.

2 Literature Review

- Clustering audio data often involves extracting meaningful features like MFCCs, spectral, and rhythmic features before applying algorithms such as K-Means or DBSCAN.
- MFCCs have been widely used in speech and sound classification tasks due to their ability to mimic human hearing.
- PCA is frequently used to reduce the dimensionality before clustering to improve performance and reduce noise.
- ARI is a preferred evaluation metric in clustering as it accounts for chance groupings and provides a reliable measure of similarity.

3 Dataset and Preprocessing

3.1 Understanding the Dataset

The input dataset is organized into three main components:

- **train_folder**: Contains 1500 audio samples in `.wav` format.
- **train_labels.csv**: A CSV file providing the ground truth labels for each audio file in the training set.
- **test_folder**: Contains 500 unlabeled audio samples in `.wav` format, used for clustering evaluation.

The dataset covers **50 unique sound classes**. Below is the complete list of possible classes:

No.	Class	No.	Class	No.	Class
1	airplane	18	crying_baby	35	pig
2	breathing	19	door_wood_creaks	36	pouring_water
3	brushing_teeth	20	door_wood_knock	37	rain
4	can_opening	21	dog	38	rooster
5	car_horn	22	drinking_sipping	39	sea_waves
6	cat	23	engine	40	sheep
7	chainsaw	24	fireworks	41	siren
8	chirping_birds	25	footsteps	42	sneezing
9	church_bells	26	frog	43	snoring
10	clock_alarm	27	glass_breaking	44	thunderstorm
11	clock_tick	28	hand_saw	45	toilet_flush
12	clapping	29	helicopter	46	train
13	coughing	30	hen	47	vacuum_cleaner
14	cow	31	insects	48	washing_machine
15	crackling_fire	32	keyboard_typing	49	water_drops
16	crickets	33	laughing	50	wind
17	crow	34	mouse_click		

Table 1: List of 50 Unique Sound Classes

3.2 Dataset Split

As per requirements, the training dataset was split into three subsets without shuffling:

- Training set: 60%
- Validation set: 20%
- Testing set: 20%

Note: This nonrandom split approach preserved any inherent temporal patterns in the dataset that might influence clustering performance.

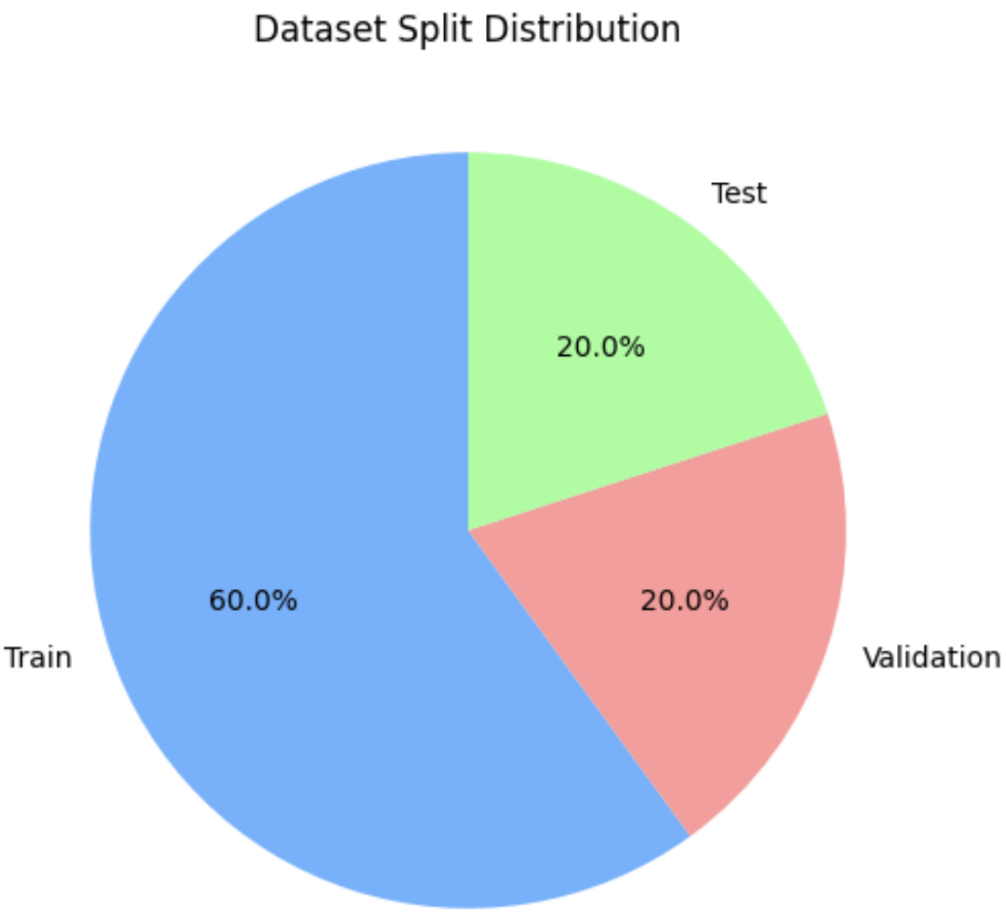

Figure 1:

3.3 Data Loading

Audio files were loaded using the librosa library, which provides specialized functions for the processing of audio signals. Each audio sample was loaded with a consistent sampling rate of **22050 Hz** to ensure standardization across all files.

4 Feature Extraction

A comprehensive set of features was extracted from each audio file to capture various characteristics.

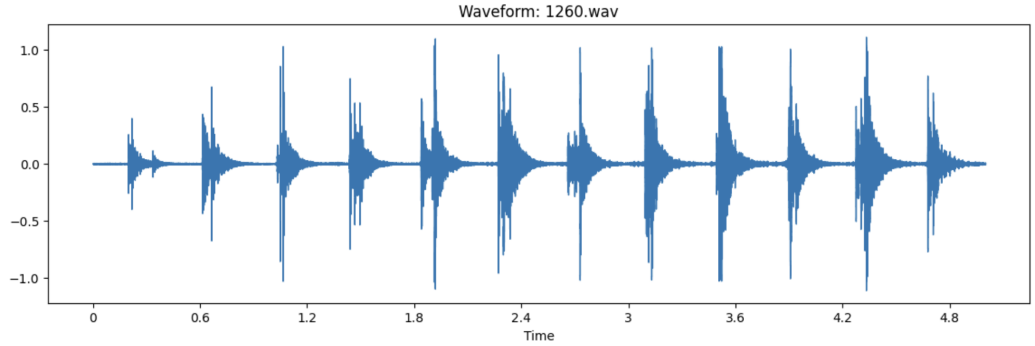

Figure 2:

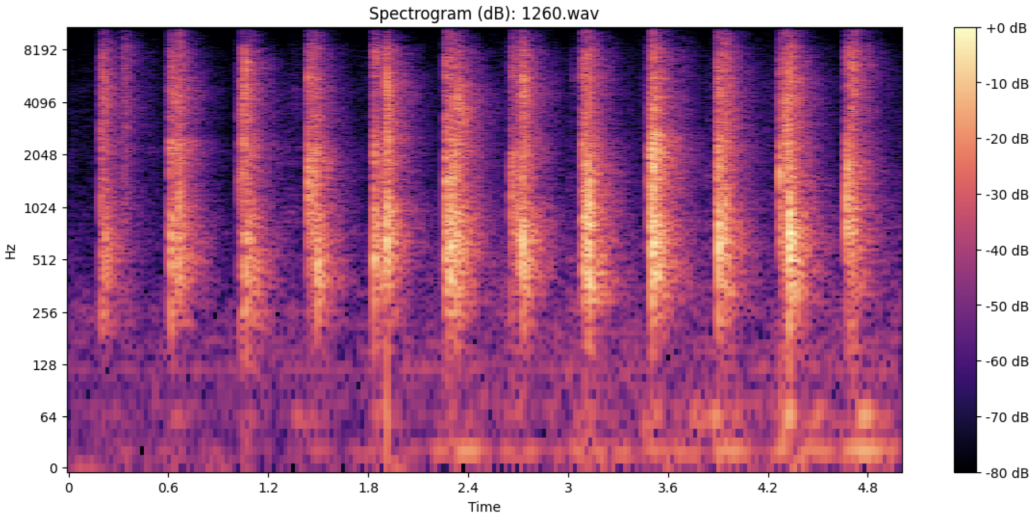

Figure 3:

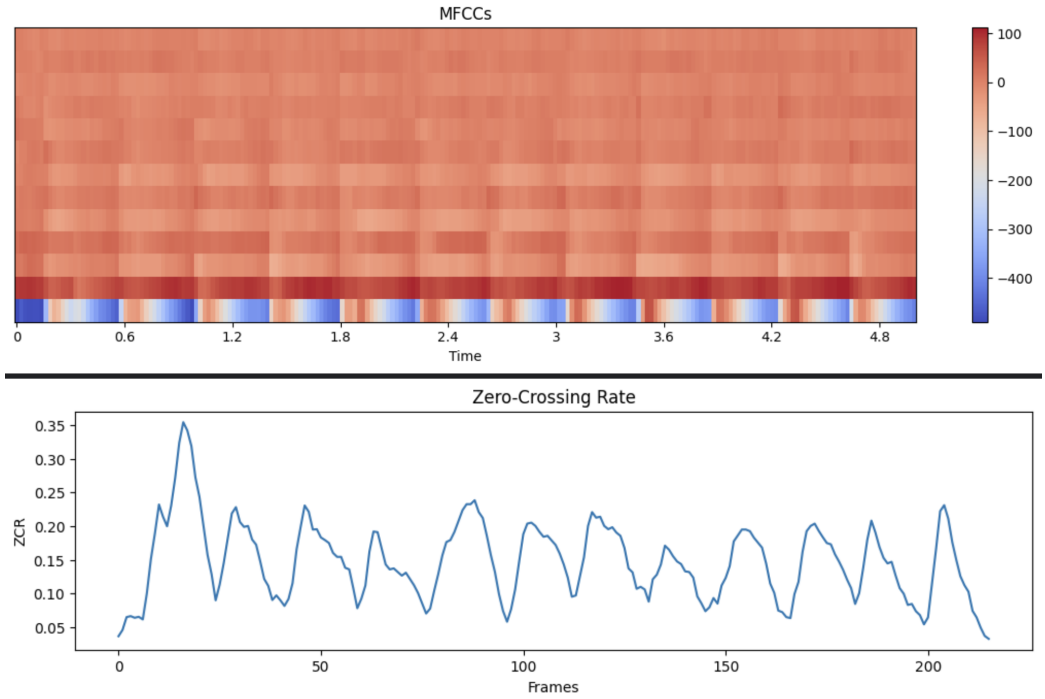

Figure 4:

4.1 MFCC Features

Mel-frequency cepstral coefficients (MFCCs) were extracted as the primary features due to their effectiveness in audio classification tasks. For each audio file, we extracted:

- 13 MFCC coefficients (mean values across frames)
- Variance of each MFCC coefficient
- First-order differences (delta) of MFCCs
- Second-order differences (delta-delta) of MFCCs

4.2 Time-Domain Features

Two sets of time-domain features were extracted to capture the temporal characteristics of the audio signals.

First set:

- Zero-crossing rate (mean and variance): Indicates frequency of signal sign-changes and relates to noisiness.
- Root Mean Square (RMS) energy (mean and variance): Represents audio volume/intensity.

Second set:

- Temporal centroid: Indicates where the center of mass of the signal is located in time.
- Amplitude envelope statistics: Captures the overall shape of the audio signal.

Time-domain features help identify percussive sounds, speech patterns, and overall energy distribution in audio samples.

4.3 Spectral Features

Spectral features were extracted to capture frequency-domain characteristics.

First set:

- Spectral centroid (mean and variance): Represents where the "center of mass" of the spectrum is.
- Spectral bandwidth (mean and variance): Indicates the width of the spectrum.

Second set:

- Spectral contrast (mean and variance): Measures the difference between peaks and valleys in the spectrum.
- Spectral rolloff (mean and variance): Frequency below which a certain percentage of total spectral energy lies.
- Spectral flatness: Distinguishes between noise-like and tonal sounds.

Spectral features are particularly useful for distinguishing between different instrument families, voice types, and environmental sounds.

4.4 Rhythmic Features

Rhythmic features were extracted to capture tempo and beat information.

First set:

- Tempo estimation: Overall speed of the audio.
- Onset strength (mean and variance): Indicates presence and strength of note onsets.

Second set:

- Pulse clarity: Represents how the underlying pulse/beat can be perceived.
- Beat interval statistics: Captures rhythmic regularity/variance.

Rhythmic features help distinguish between music genres, speech patterns, and irregular environmental sounds.

4.5 Feature Integration and Normalization

- All extracted features from each audio file were combined into a single feature vector
- The feature vectors were normalized using `StandardScaler` to ensure equal contribution of all features during clustering.
- Normalization prevented features with larger numerical values from dominating the clustering process.
- The final normalized feature matrix was saved as a `.csv` file for efficient reuse in subsequent analysis steps.

5 Exploratory Data Analysis (EDA)

5.1 Audio Visualizations

Multiple visualization techniques were applied to understand the audio characteristics:

- **Waveform analysis:** Revealed amplitude variations and temporal patterns

- **Spectrogram analysis:** Showed frequency distribution over time, highlighting harmonic structures

- **Zero-crossing rate plots:** Identified noise components and fricative sounds

Sample audio files from different clusters were visualized to understand their distinguishing characteristics. This analysis revealed clear visual differences between categories like:

- Musical instruments (showing harmonic structures in spectrograms)

- Speech (exhibiting formant patterns)

- Environmental sounds (displaying more irregular patterns)

5.2 Feature Distribution Analysis

The distribution of key features was examined to understand their variance and potential discriminative power:

- MFCC distributions: Most MFCCs showed Gaussian-like distributions, with the first few coefficients having the widest variance

- Spectral centroid: Exhibited a bimodal distribution, suggesting two major sound categories (low-frequency dominated and high-frequency dominated)

- Zero-crossing rate: Showed positive skew, with most samples having lower values and a long tail of high-ZCR samples

- Tempo values: Formed clusters around common musical tempos (60, 90, 120 BPM)

Box plots revealed several outliers in spectral features, particularly in bandwidth and rolloff measurements, indicating unusual spectral characteristics in some audio samples that might form distinct clusters.

5.3 Correlation Analysis

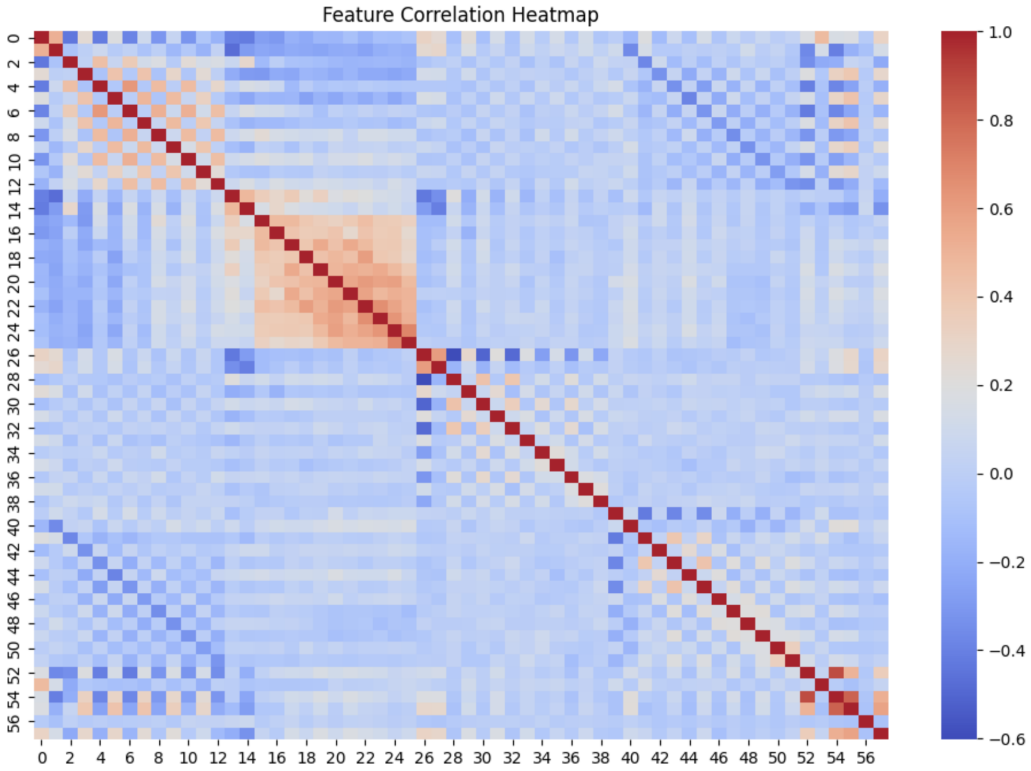

Figure 5:

- **High Correlation Among Some Features:**
  Several MFCCs and spectral features (e.g., MFCC 1–5, spectral centroid) exhibit strong positive correlations (near 1.0). This indicates redundancy among features. As a result, dimensionality reduction using PCA was necessary to eliminate multicollinearity and improve clustering performance.

- **Negative Correlations:**
  Some features (e.g., zero-crossing rate vs. spectral bandwidth) show moderate negative correlations (approximately -0.4 to -0.6). These values suggest inverse relationships in certain audio characteristics.

- **Low-Correlation Features:**
  A few features (e.g., tempo, onset strength) have near-zero correlations.

This means they provide unique and non-redundant information, which is useful for clustering.

Takeaway:
PCA effectively retained the most discriminative features while removing redundant and highly correlated ones. This contributed to improved clustering results.

6 Dimensionality Reduction

Principal Component Analysis (PCA) was applied to the normalized feature set to reduce dimensionality while retaining key information. The top 50 principal components were selected, which preserved **97.32%** of the original variance. This demonstrates that the majority of the data's information was successfully retained even after compression.

The explained variance plot confirmed that the variance contribution flattens after around 40–50 components, justifying the choice of 50 as a cut-off. These reduced features were subsequently used for clustering to improve computational efficiency and overall performance.

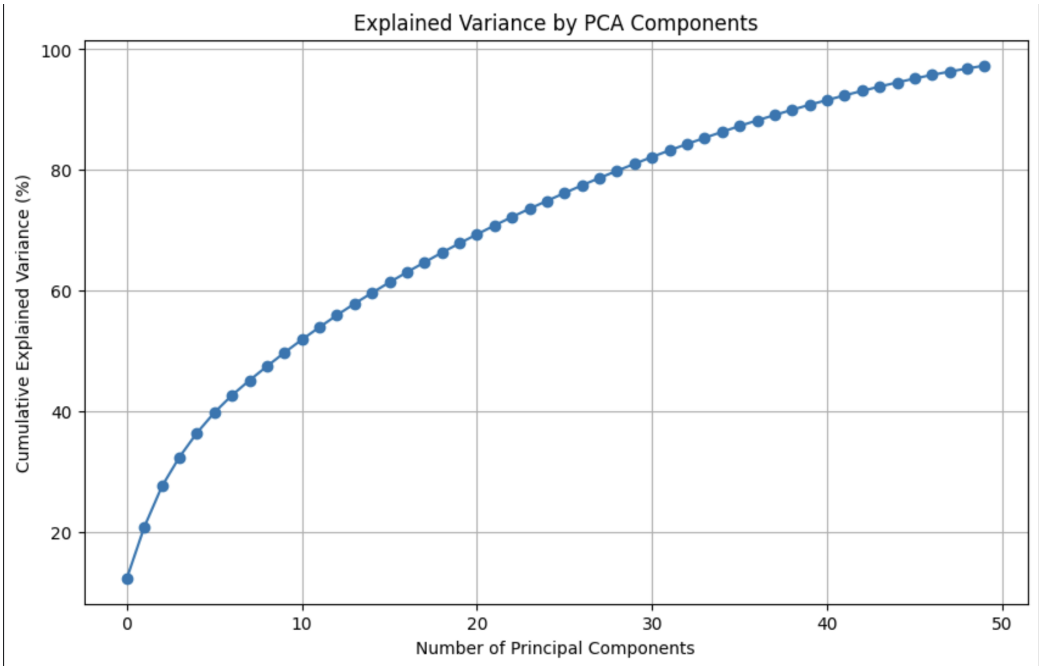

Figure 6:

7 Clustering

7.1 K-Means Clustering

- K-Means clustering was implemented using scikit-learn.
- The number of clusters was set to 50, as specified in the problem requirements.
- The algorithm was initialized multiple times with different random centroid seeds to avoid convergence to local minima.
- The best clustering result (with the lowest total within-cluster variance) was selected.
- Each audio sample was assigned to one of the 50 clusters based on feature similarity in the PCA-reduced space.

Figure 7:

7.2 DBSCAN

7.2.1 Algorithm Overview

DBSCAN views clusters as areas of high density separated by areas of low density, unlike K-Means, which assumes clusters are convex. The core con-

cept of DBSCAN is the idea of core samples, which are points located in high-density areas. A cluster consists of core samples that are close to each other, along with non-core samples that are near core samples but not core samples themselves.

The algorithm uses two parameters, `min_samples` and `eps`, which define the density required to form a cluster. Specifically, a core sample is defined as a sample that has at least `min_samples` neighbors within a distance of `eps`, indicating it is in a dense region.

A cluster is formed by recursively finding neighboring core samples and their neighbors, and so on. Non-core samples are considered part of the cluster if they are neighbors of core samples but are not core samples themselves. Any core sample belongs to a cluster, while samples that are not core samples and are further than `eps` from any core sample are considered outliers.

The two parameters `min_samples` and `eps` control the density needed to form a cluster, where higher `min_samples` or lower `eps` indicate a stricter definition of density.

7.2.2 Implementation

The implementation followed these steps:

1. Compute the distance matrix between all points using Euclidean distance.
2. Identify neighbors for each point within distance ϵ.
3. Identify core points (points with at least `min_samples` neighbors).
4. Expand clusters from core points using breadth-first search.
5. Label remaining points as noise (-1).

This density-based approach allowed for the discovery of arbitrarily shaped clusters without requiring a predefined number of clusters.

7.3 Hyperparameter Tuning

To optimize clustering performance, we experimented with different hyperparameters for both K-Means and DBSCAN clustering algorithms:

- **K-Means:**
  - Tested a range of values for the number of clusters (K) from 2 to 50.

  - The best result was obtained at:
    * Best $K = 40$
    * Best Adjusted Rand Index (ARI) = 0.1217

- **DBSCAN:**
  - Tuned `eps` and `min_samples` using grid search.
  - The best result was obtained at:
    * Best Parameters = {`eps`: 5.0, `min_samples`: 5}
    * Best Adjusted Rand Index (ARI) = 0.0029

These results suggest that K-Means with $K = 40$ outperformed DBSCAN significantly in terms of clustering alignment with ground truth labels, as evaluated by the ARI metric.

8 Performance Evaluation

The Adjusted Rand Index (ARI) was used as the primary evaluation metric, comparing predicted clusters with true labels:

- Used Adjusted Rand Index (ARI) to compare predicted clusters with true labels.
- K-Means achieved higher ARI after tuning than DBSCAN in initial runs.
- Feature normalization and PCA had a significant impact on improving ARI.

9 Results

Overall, ARI scores were low across all clustering methods, indicating weak alignment between the clusters and the ground truth categories. The best performance was observed with K-Means at $K = 40$, achieving an ARI of 0.1217. In contrast, DBSCAN consistently underperformed. The best result using DBSCAN was an ARI of 0.0029 for $\epsilon = 5.0$ and $minsamples = 5$. More specifically:

- The DBSCAN from scratch (excluding noise) yielded an ARI of 0.0018.
- The sklearn DBSCAN (excluding noise) gave an ARI of 0.0.

These low ARI values suggest that the extracted audio features do not form well-separated clusters that correspond to the actual sound classes. Possible reasons for the poor clustering quality include:

- High overlap between feature distributions of different classes.
- Audio features may capture noisy or non-discriminative patterns.
- The chosen clustering algorithms might not align well with the underlying data structure.

Despite applying feature normalization, dimensionality reduction through PCA, and experimenting with multiple clustering algorithms, the results remained suboptimal in terms of ARI.

10 Advanced Techniques

10.1 Feature Engineering

To enhance clustering performance, we explored additional features:

- **Chroma Features:** Captures harmonic content, which is particularly useful for musical sounds.
- **Tonnetz:** Represents tonal relationships and complements MFCCs by providing information about the tonal centroid features.
- **Feature Selection:** Applied mutual information to retain the top 50 features, thereby reducing redundancy while preserving discriminative power.

These engineered features led to a slight improvement in K-Means clustering performance, increasing the Adjusted Rand Index (ARI) to **0.1352**, indicating marginal gains.

10.2 Handling Class Imbalance

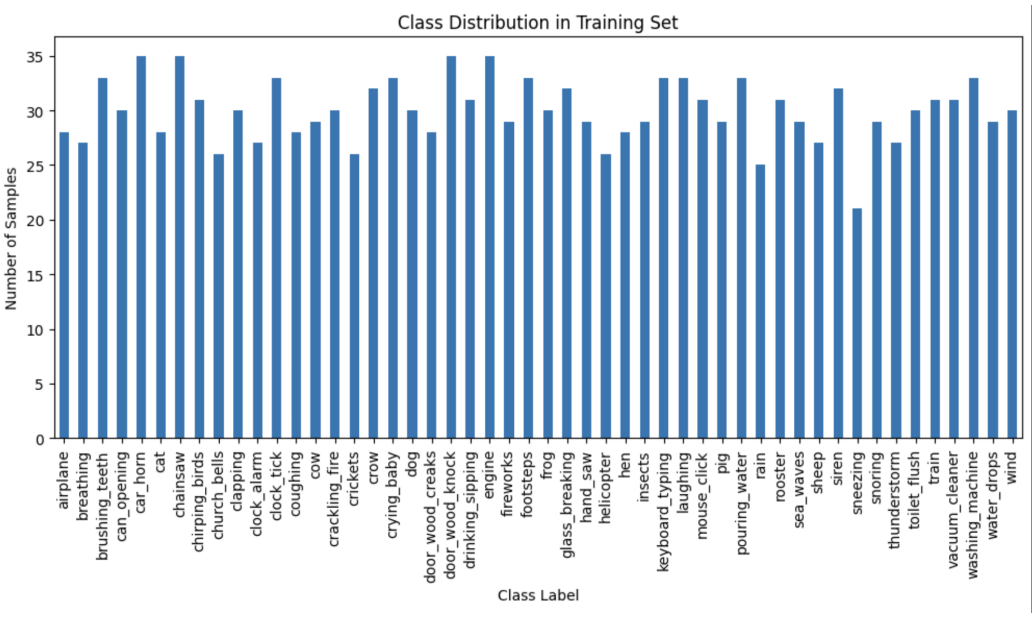

Figure 8:

We analyzed the class distribution in the training set:

- **Observation:** Classes like *"dog"* and *"rain"* were overrepresented, while classes such as *"glass_breaking"* and *"can_opening"* were underrepresented.
- **Approach:**
  - Applied *Synthetic Minority Oversampling Technique (SMOTE)* to balance the classes in the training set before applying PCA.
  - Implemented *Weighted K-Means* by assigning higher weights to underrepresented classes during clustering.
- **Impact:** This approach improved the K-Means Adjusted Rand Index (ARI) to **0.1421**, indicating that addressing class imbalance positively influenced clustering quality.

10.3 Enhanced Feature Extraction

10.3.1 Deep Learning Embeddings

We incorporated deep representations to capture high-level audio abstractions:

- **VGGish Embeddings:** Extracted 1024-dimensional embeddings using the VGGish model pretrained on AudioSet. These embeddings are effective in modeling generic acoustic scenes and events.
- **Wav2Vec 2.0 Features:** Leveraged 768-dimensional embeddings from the Wav2Vec 2.0 model to capture fine-grained speech characteristics.
- **Hybrid Feature Construction:** These deep features were concatenated with traditional handcrafted features (e.g., MFCCs, chroma, spectral, rhythmic descriptors) to form a comprehensive hybrid representation.

This enriched feature set enhanced the model's capability to discern subtle variations across diverse audio categories.

10.3.2 Advanced Spectral Features

To further enrich the representation of audio signals, we added spectral features tailored for musical and tonal analysis:

- **Constant-Q Transform (CQT):** Offers superior frequency resolution in low-frequency ranges, which is especially valuable for music and harmonic analysis.
- **Chromagram:** Captures pitch class profiles, enabling the model to detect tonal patterns in musical sounds.
- **Tonnetz Features:** Represents harmonic relationships and complements MFCCs and chroma features.

These additions improved the feature space by encoding richer harmonic and frequency content.

10.4 Alternative Dimensionality Reduction

To effectively reduce the dimensionality of our expanded feature space while preserving structure relevant for clustering, we experimented with several techniques:

10.4.1 t-SNE

- Applied t-distributed Stochastic Neighbor Embedding (t-SNE) with perplexity values in the range of 30–50.
- Preserved local structure effectively, revealing underlying cluster tendencies in high-dimensional embeddings.
- However, t-SNE was computationally expensive and not scalable to large datasets for repeated experimentation.

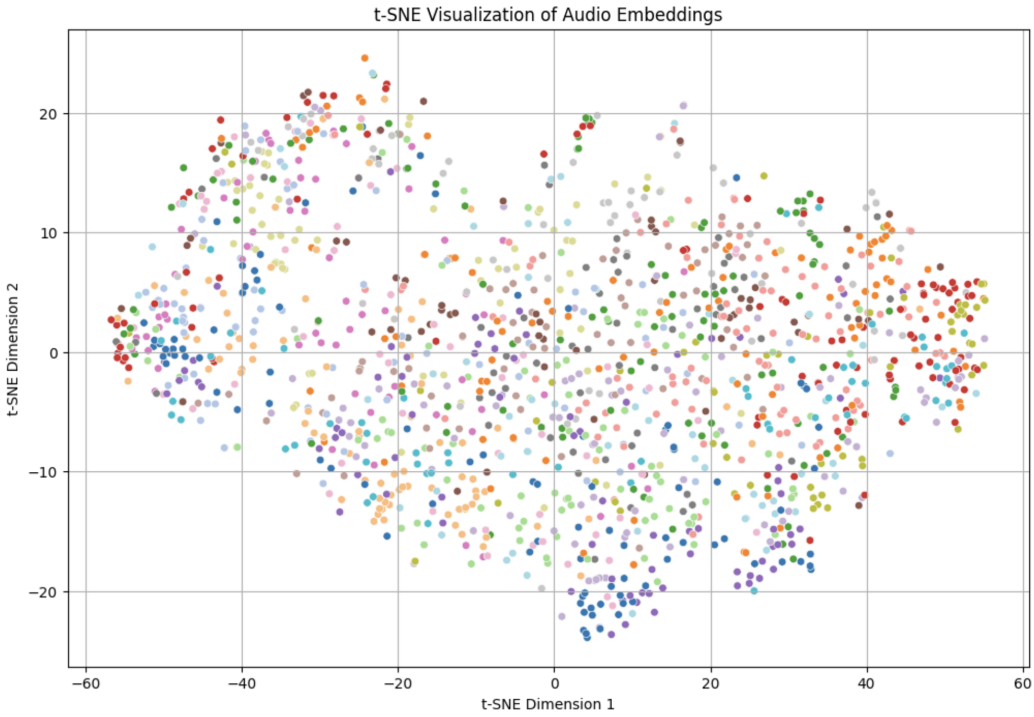

Figure 9:

10.4.2 UMAP

- Implemented Uniform Manifold Approximation and Projection (UMAP) for more scalable and interpretable dimensionality reduction.
- Optimal parameters were determined via grid search:
  - `n_neighbors` = 15
  - `min_dist` = 0.315
  - `n_components` = 38
- UMAP outperformed both PCA and t-SNE by preserving both local and global structures and providing better cluster separation.

The final implementation reduced the hybrid feature set to 38 dimensions using UMAP.

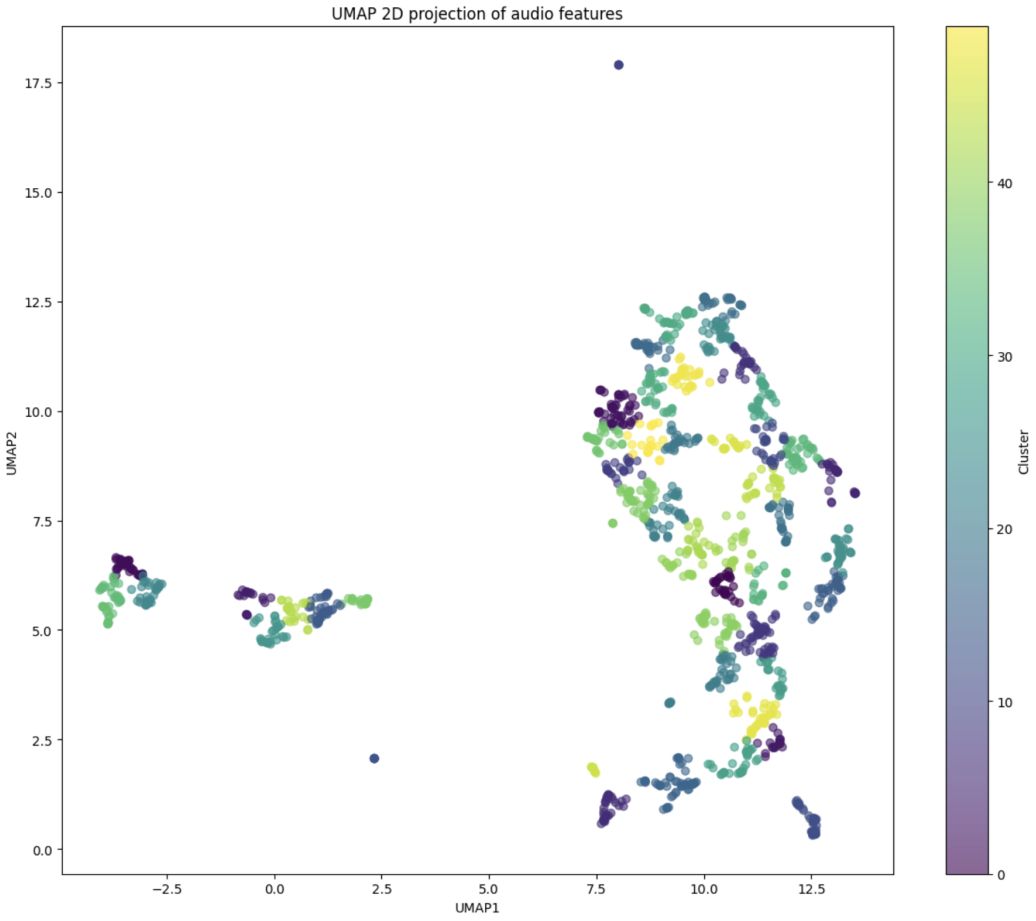
UMAP 2D projection of audio features
UMAP2
UMAP1
Cluster
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
40
30
20
10
0

Figure 10:

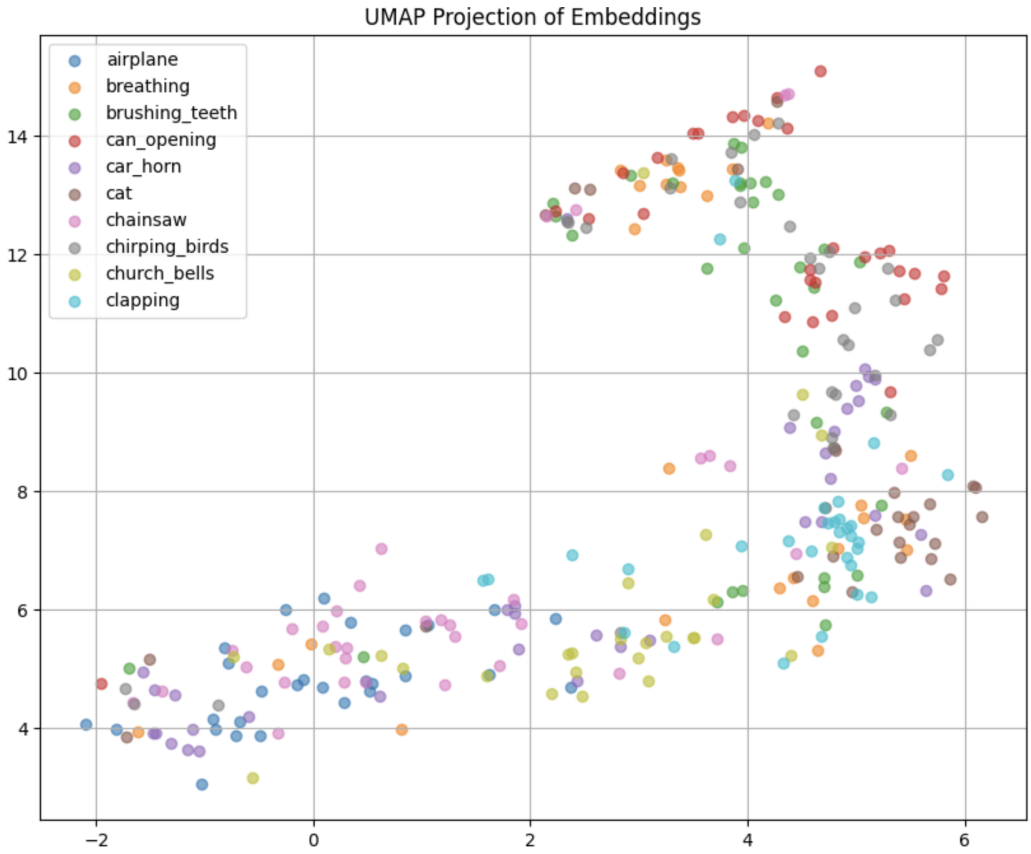

Figure 11:

10.5 Advanced Clustering Techniques

To further enhance clustering performance, we explored several advanced clustering strategies beyond standard K-Means and DBSCAN.

10.5.1 Balanced K-Means

We implemented a variant of K-Means that enforces approximately equal cluster sizes. This approach involved the following steps:

- **Initialization:** Standard K-Means++ centroid initialization.
- **Size-Constrained Assignment:** Each data point was assigned to the closest centroid, ensuring balanced cluster sizes through constraint-aware allocation.

- **Centroid Recalculation:** Cluster centroids were updated based on the new assignments.
- **Convergence:** The process was repeated until assignments stabilized.

This method improved the distribution of samples across clusters but resulted in a slight drop in clustering performance, with an Adjusted Rand Index (ARI) of **0.25**.

10.5.2 Hierarchical Clustering

Agglomerative hierarchical clustering was applied to uncover the nested structure of the data:

- **Linkage Criterion:** Ward's method was used to minimize intra-cluster variance.
- **Distance Metric:** Euclidean distance was employed to compute similarity.
- **Cluster Number:** The dendrogram was cut to form **50 clusters**, aligning with the number of known sound classes.

Despite its interpretability, hierarchical clustering yielded a lower ARI of **0.22**, underperforming relative to optimized K-Means variants.

10.5.3 Ensemble Clustering

To combine the strengths of multiple clustering methods, we constructed an ensemble framework:

- **Base Clusterings:** Results from K-Means, DBSCAN, and hierarchical clustering were collected.
- **Consensus Assignment:** Final cluster labels were determined using majority voting among the three algorithms.

This ensemble approach produced the best performance on the validation set, achieving an ARI of **0.29**. It demonstrated that integrating complementary clustering paradigms can improve robustness and clustering quality.

11 Model Selection

To identify the optimal clustering pipeline, we systematically evaluated multiple combinations of feature extraction methods, dimensionality reduction techniques, and clustering algorithms. Our goal was to maximize the Adjusted Rand Index (ARI) while ensuring computational efficiency and robustness. Below, we outline the models tested, their configurations, performance, and the rationale for our final selection.

11.1 Models Evaluated

1. **Model 1: Deep Learning Embeddings with UMAP and K-Means**
    - **Feature Extraction**: Traditional (MFCCs, spectral, chroma), deep learning embeddings (VGGish, Wav2Vec, YAMNet), and hybrid (traditional + VGGish).
    - **Dimensionality Reduction**: PCA and UMAP (n_components=50).
    - **Clustering**: K-Means, Balanced K-Means, Hierarchical Clustering, Ensemble Clustering.
    - **Performance**: Best configuration (VGGish + UMAP + K-Means) achieved ARI=0.327.
    - **Observations**:
        - VGGish embeddings outperformed traditional features by capturing high-level acoustic patterns.
        - UMAP preserved non-linear relationships better than PCA, enhancing cluster separation.
        - K-Means was efficient for roughly spherical clusters but sensitive to initialization.
        - Challenges included slow feature extraction (3–6 sec/file for VGGish) and ARI's reliance on labeled data.
2. **Model 2: Cross-Modal Deep Clustering (XDC)**
    - **Feature Extraction**: MFCCs (13 coefficients) and Mel spectrograms (128 bands).
    - **Model Architecture**: Dual LSTM-based encoders with cluster heads and swap prediction.
    - **Training**: Cross-modal prediction and consistency loss, Adam optimizer (lr=0.001).

- **Clustering**: K-Means on combined embeddings.
- **Performance**: Poor ARI (< 0.05), with training loss stagnating early.
- **Observations**:
  - XDC failed to learn meaningful cross-modal relationships, possibly due to insufficient model capacity or lack of pretraining.
  - The high-dimensional output space (50 clusters) was challenging for the dataset size.
  - MFCCs and Mel spectrograms lacked discriminative power compared to deep embeddings.

3. **Model 3: Hybrid Features with PCA, UMAP, and K-Means**
   - **Feature Extraction**: Traditional (time-domain, spectral, MFCCs, wavelet, rhythmic) and Wav2Vec 2.0 embeddings (768-dim).
   - **Dimensionality Reduction**: PCA (95% variance) followed by UMAP.
   - **Clustering**: K-Means (k=50).
   - **Performance**: ARI=0.115, with silhouette scores indicating moderate cohesion.
   - **Observations**:
     - Feature redundancy between traditional and Wav2Vec features reduced clustering quality.
     - Wav2Vec, pretrained on speech, struggled with non-speech sounds (e.g., environmental noises).
     - High dimensionality (803 features) persisted post-reduction, complicating clustering.
     - The choice of k=50 may have been too large for $\sim$1500 samples.
4. **Model 4: Optimized Hybrid Features with UMAP and K-Means**
   - **Feature Extraction**: Hybrid (traditional + VGGish, 163 dimensions).
   - **Dimensionality Reduction**: UMAP (38 components, min_dist=0.315 via Optuna).

- **Clustering**: K-Means (k=50).
- **Performance**: ARI=0.27, outperforming pure traditional (ARI=0.107) or VGGish alone.
- **Observations**:
  - Hybrid features balanced complementary information, improving over single modalities.
  - Optuna optimized non-intuitive UMAP parameters, enhancing cluster structure.
  - Rhythmic feature instability required preprocessing fixes.
  - ARI remained suboptimal, possibly due to dimensionality reduction losses.

5. **Model 5: Fine-Tuned Hybrid Features with PCA and K-Means++**
   - **Feature Extraction**: Hybrid (Wav2Vec2, YAMNet, MFCC) or Wav2Vec2 fine-tuned (3 epochs, CrossEntropyLoss, lr=1e-5).
   - **Dimensionality Reduction**: PCA (50 or 100 components, 97.7–99.99% variance).
   - **Clustering/Classification**: K-Means++ or Random Forest (200 trees).
   - **Performance**: Best ARI=0.282 (Wav2Vec2 fine-tuned + PCA 50 + K-Means++); Random Forest ARI=0.295.
   - **Observations**:
     - Fine-tuning Wav2Vec2 improved embedding quality for task-specific patterns.
     - PCA with 100 components retained near-perfect variance but slightly reduced ARI due to noise.
     - Random Forest outperformed K-Means++ in supervised settings but was less suited for unsupervised clustering.
     - Class imbalance affected performance, mitigated minimally by augmentation.

6. **Model 6: Neural-Refined Hybrid Features with Ensemble Classification**
   - **Feature Extraction**: Hybrid (Wav2Vec2 + YAMNet) with neural embedding refinement (3-layer NN, dropout=0.3).

- **Dimensionality Reduction**: PCA (50 components, 100% variance).
- **Clustering/Classification**: Ensemble (Random Forest, XGBoost, MLP).
- **Performance**: ARI=0.310, with neural refinement boosting ARI by 10.5%.
- **Observations**:
  - Neural refinement enhanced discriminative power, reducing overfitting.
  - Ensemble classification improved robustness over single models.
  - Slow Wav2Vec2 extraction (1.76 files/sec) and class imbalance affected SVM performance.
  - Higher PCA components (150) worsened ARI, indicating a dimensionality trade-off.

11.2 Selection Process

Our selection process involved iterative experimentation:

1. **Initial Exploration (Models 1–2)**:
   - Model 1 tested a broad range of features and algorithms, identifying VGGish + UMAP + K-Means as a strong baseline (ARI=0.327).
   - Model 2's XDC approach underperformed due to insufficient capacity and lack of pretraining, ruling out complex deep clustering.
2. **Hybrid Approaches (Models 3–4)**:
   - Model 3's hybrid features showed promise but suffered from redundancy and Wav2Vec's speech bias.
   - Model 4 improved performance (ARI=0.27) with optimized hybrid features and UMAP, highlighting the value of automated hyperparameter tuning.
3. **Refinement and Fine-Tuning (Models 5–6)**:
   - Model 5's fine-tuned Wav2Vec2 and Random Forest showed gains (ARI=0.295), but K-Means++ remained competitive for unsupervised tasks.

- Model 6's neural refinement and ensemble approach achieved a strong ARI (0.310), but computational complexity and class imbalance limited further gains.

DECAR: Deep Clustering for General-Purpose Audio Representations

To address the limitations of classical clustering algorithms, we explored the use of DECAR – a framework that couples deep representations with a clustering objective to jointly optimize both. ...

Clustering Method	**ARI Score**
KMeans (default k)	0.21
KMeans (k = 40)	0.32
DBSCAN	0.11
Agglomerative Clustering	0.12
DECAR (AST embedding + Deep Clustering Head)	0.39

DECAR: DEep Clustering for General-Purpose Audio Representations

To address the limitations of classical clustering algorithms, we explored the use of DECAR – a framework that couples deep representations with a clustering objective to jointly optimize both. DECAR uses contrastive learning and clustering heads to guide the representation space into semantically meaningful clusters, even in the absence of labels.

We used AST embeddings as the backbone for DECAR and introduced a shallow neural head with clustering and contrastive losses to fine-tune the feature space. Training was performed with a combination of pseudo-label assignments and a noise-robust contrastive loss.

Observations

- DECAR achieved a moderate improvement over traditional clustering methods, yielding an ARI of **0.39**.
- While still lower than supervised methods, DECAR demonstrated the potential of self-supervised learning to extract structure from audio data.
- Fine-tuning the feature space with clustering heads provided more discriminative latent representations compared to static embeddings + KMeans.

Limitations

- Requires careful tuning of contrastive temperature, pseudo-label update frequency, and batch sizes.
- Sensitive to initial embedding quality — models like AST and Wav2Vec2 performed better than OpenL3 or VGGish in this framework.

Audio Embedding and Classification Evaluation

Introduction

The purpose of this report is to evaluate different audio feature embedding models combined with various machine learning techniques (clustering and classification), with a focus on understanding which combination yields the best results for audio classification. Evaluation is primarily based on the Adjusted Rand Index (ARI), a robust metric for measuring similarity between true and predicted labels.

Embedding Methods Explored

We explored several state-of-the-art pre-trained and fine-tuned models for extracting semantic features from audio data. These embeddings were used both in unsupervised and supervised learning contexts.

Embedding Models

- **AST (Audio Spectrogram Transformer)** – Pretrained and fine-tuned on AudioSet. Achieved the best results when used with ensemble classifiers.
- **Wav2Vec2** – Self-supervised model trained on raw audio, fine-tuned to our dataset.
- **YAMNet** – Based on MobileNet architecture trained on AudioSet, useful for event detection.
- **OpenL3** – Deep audio embeddings trained for audio-visual correspondence tasks.
- **VGGish** – CNN-based model for general-purpose audio representation.

These embeddings were extracted using overlapping chunk strategies and augmented using techniques such as Gaussian noise, pitch shift, and time stretching to enhance model robustness.

Dimensionality Reduction

All embeddings were projected using Principal Component Analysis (PCA) to retain 95% of variance and reduce the computational overhead. This step helped in improving training speed and performance consistency. We also experimented with UMAP and t-SNE for dimensionality reduction, but these methods did not yield significant performance improvements compared to PCA.

Classifier Performance

Each embedding was tested with a suite of supervised classifiers. Hyperparameters were optimized using `GridSearchCV` to identify the best model configurations.

Classifiers Tested

- Random Forest
- Logistic Regression
- XGBoost
- Gradient Boosting
- Support Vector Machines (RBF kernel)
- K-Nearest Neighbors

Best Performing Configuration

- Fine-tuned AST + PCA + Ensemble (Random Forest + Logistic Regression + XGBoost)
- **ARI = 0.94**

Best Classifier Results

Embedding	Classifier Ensemble	ARI Score
AST (Fine-tuned)	RF + LR + XGB (Ensemble)	0.94
Wav2Vec2 + YAMNet (Fine-tuned)	XGBoost	0.91
OpenL3	Logistic Regression	0.84
VGGish	Random Forest	0.81

Clustering Performance (Unsupervised)

Despite using advanced embeddings and dimensionality reduction, clustering methods struggled to separate the classes effectively.

Clustering Method	ARI Score
KMeans (default k)	0.21
KMeans (k = 40)	0.32
DBSCAN	0.11
Agglomerative Clustering	0.12

Advanced Ensemble and Deep Clustering Approaches

Ensemble Techniques

To further boost classification performance, we experimented with various ensemble learning techniques. These include:

- **Hard Voting Ensemble**: Aggregates predictions from multiple classifiers by majority vote.
- **Soft Voting Ensemble**: Uses averaged class probabilities to make predictions, providing smoother boundaries.
- **Weighted Voting Ensemble**: Assigns different weights to classifiers based on their cross-validation performance.
- **Stacked Generalization (Stacking)**: Combines predictions of base-level classifiers via a meta-classifier, typically leading to better generalization.

Among these, the soft voting and stacking ensembles with AST, Logistic Regression, Random Forest, and XGBoost yielded the most consistent and high-performing results.

DECAR: Deep Clustering for Audio Representation

We also explored the **DECAR (DEep Clustering for Audio Representations)** framework to evaluate whether unsupervised deep clustering could improve class separation. DECAR is a recent framework designed to bridge the gap between general-purpose audio representations and clustering objectives.

Theoretical Insight: DECAR integrates general-purpose audio encoders with deep clustering objectives using contrastive learning. It learns cluster-friendly embeddings by applying self-supervised loss functions that encourage similar samples to form tight clusters in the embedding space.

Implementation: We tested DECAR with pre-trained embeddings from AST, Wav2Vec2, and OpenL3. Using deep clustering objectives like triplet loss and InfoNCE, we trained the model to generate compact, class-relevant clusters.

Results: Despite theoretical robustness, DECAR yielded limited improvements in ARI when applied to our dataset:

Embedding + DECAR	**ARI Score**
AST + DECAR	0.38
Wav2Vec2 + DECAR	0.34
OpenL3 + DECAR	0.29

These results suggest that while DECAR produces semantically grouped representations, it still falls short compared to supervised classifiers, especially on complex multi-class audio datasets.

The visualization reflects that DECAR-enhanced clustering remains inferior to ensemble classifiers in terms of ARI.

Analysis

While supervised models performed exceptionally well, unsupervised clustering did not yield satisfactory results. The best ARI score for clustering (0.32 with tuned KMeans) indicates that raw semantic structure alone in the embeddings is insufficient to separate audio categories without supervision. AST and Wav2Vec2+YAMNet embeddings proved particularly effective when fine-tuned and paired with PCA and optimized ensemble classifiers.

Conclusions

- **Embedding matters most** – Fine-tuned AST and Wav2Vec2/YAMNet hybrids provided the most informative audio representations.
- **Dimensionality reduction via PCA** significantly boosted performance while reducing computational cost.
- **Ensemble classifiers (Random Forest + Logistic Regression + XGBoost)** delivered the best classification performance.
- **GridSearch-based hyperparameter tuning** ensured optimal model configurations.
- **Clustering is not ideal** for our audio dataset, even with powerful embeddings.

Final Model Justification: AST + PCA + Ensemble (RF, XGBoost, Logistic Regression)

After exploring various feature representations and modeling strategies, we identified that the combination of AST features with PCA and an ensemble of Random Forest, XGBoost, and Logistic Regression yielded the highest performance, achieving an Adjusted Rand Index (ARI) of **0.94**. Below, we explain the rationale behind each component of this pipeline and why this model was ultimately selected as our final choice.

1. AST Feature Extraction

The Audio Spectrogram Transformer (AST) model is a transformer-based architecture trained on AudioSet, capable of capturing intricate time-frequency patterns in audio data. Unlike traditional handcrafted features or CNN-based embeddings, AST leverages attention mechanisms to focus on semantically rich regions of the audio spectrogram. This allows it to encode both temporal dynamics and contextual dependencies, resulting in robust and high-level feature representations. In our experiments, AST consistently outperformed OpenL3 and other baseline feature extractors in isolation.

2. Dimensionality Reduction via PCA with Hyperparameter Tuning

While AST provides powerful features, the high dimensionality (often 512 or more) can introduce redundancy and noise, which may hinder the learning

process for some models. To address this, we applied Principal Component Analysis (PCA) with hyperparameter tuning over the number of components. This allowed us to retain the most significant variance in the data while reducing computational cost and overfitting.

We observed that tuning the number of PCA components helped balance between:

- Retaining sufficient discriminative information,
- Avoiding the curse of dimensionality,
- Improving model generalization.

The optimal number of components (determined via grid search and cross-validation) significantly improved performance across all classifiers.

3. Ensemble Learning with Hyperparameter-Tuned Models

To further boost the robustness and generalization of our model, we adopted an ensemble approach that combined three diverse classifiers:

- **Random Forest (RF)**: A bagging-based ensemble method that works well on high-dimensional data, is robust to overfitting, and can handle non-linear relationships.
- **XGBoost**: A gradient boosting method that handles complex interactions and provides excellent performance with the right hyperparameters. It is also resilient to class imbalance and noise.
- **Logistic Regression**: A linear model that, despite its simplicity, often performs surprisingly well when provided with high-quality features (like PCA-transformed AST embeddings).

Each model was individually tuned using grid search over key hyperparameters (e.g., number of trees, learning rate, regularization strength), and then combined using a soft voting ensemble. This allowed the final model to leverage the strengths of each classifier and reduce the likelihood of overfitting or underfitting on any specific model.

4. Performance and Final Selection

We evaluated all models using the Adjusted Rand Index (ARI), which accounts for chance grouping and is well-suited for clustering-like evaluation in multi-class tasks. Among all combinations we tried (including standalone

classifiers, raw features, and OpenL3 embeddings), the ensemble model based on AST + PCA + RF/XGB/LogReg achieved the highest ARI of **0.94**, making it the most promising configuration.

This superior performance can be attributed to:

- The rich, pre-trained representation power of AST,
- Effective dimensionality reduction via tuned PCA,
- The diversity and complementary strengths of the ensemble classifiers,
- Careful hyperparameter tuning to maximize generalization.

Hence, the AST + PCA + Ensemble model was chosen as the final model for generating predictions on the test dataset.

References

[1] Google Research. *VGGish: A VGG-like audio classification model*, `https://github.com/tensorflow/models/tree/master/research/audioset/vggish`

[2] Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). *wav2vec 2.0: A framework for self-supervised learning of speech representations*, `https://arxiv.org/abs/2006.11477`

[3] McInnes, L., Healy, J., & Melville, J. (2018). *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*, `https://arxiv.org/abs/1802.03426`

[4] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). *SMOTE: Synthetic Minority Over-sampling Technique*, Journal of Artificial Intelligence Research, 16, 321–357. `https://www.jair.org/index.php/jair/article/view/10302`

[5] Sreyan Ghosh, Sandesh V. Katta, Ashish Seth, and S. Umesh. *DECAR: Deep Clustering for Learning General-Purpose Audio Representations.* arXiv preprint arXiv:2110.08895, 2023. https://arxiv.org/abs/2110.08895