

## **Task 02: Simple Audio Classification Using Feature Extraction**

# Part 01

## A. Introduction

This report details an audio classification system for distinguishing between two emergency vehicle classes (`class_1` and `class_2`) using signal processing techniques. The pipeline employs **Mel-Frequency Cepstral Coefficients (MFCC)** for feature extraction and **Euclidean distance** for similarity measurement. Unknown samples are classified by comparing their features against preprocessed class templates.

## B. Methodology

### 1. Feature Extraction

- **MFCC Selection:**

MFCCs were chosen for their effectiveness in capturing spectral characteristics of audio signals. The extraction process includes:

- a. Frame splitting (20–40 ms segments).
- b. Fourier transform to obtain frequency components.
- c. Mel-scale mapping to approximate human auditory perception.
- d. Discrete Cosine Transform (DCT) to compress features into 13 coefficients.

- **Fixed-Length Handling:**

- a. For signals with fewer than 13 MFCCs, zero-padding was applied.
- b. Longer signals were truncated to ensure uniform feature dimensions.

### 2. Similarity Measurement

- **Euclidean Distance:**

- Computed between MFCC vectors.
- Advantages: Efficient computation and sensitivity to magnitude differences.

- **Classification Logic:**

Each unknown sample is assigned to the class with the **smallest average Euclidean distance** to its training samples.

### 3. Classification Workflow

### 1. Training Phase:

- Extract MFCCs for all files in `class_1` and `class_2`.
- Store features as reference templates.

### 2. Testing Phase:

- For each unknown file:
  - Compute its MFCC vector.
  - Calculate average Euclidean distance to `class_1` and `class_2` templates.
  - Assign to the class with the minimal average distance.

## C. Implementation Details

### Code Structure

#### 1. Initialization:

- Set paths for `class_1`, `class_2`, and `unknown` folders.

#### 2. Feature Extraction:

- `extract_mfcc_fixed.m`: Computes 13 MFCC coefficients per file, padding/truncating as needed.

#### 3. Training:

- Process all `class_1` and `class_2` files to build feature matrices.

#### 4. Classification:

- For each unknown file:
  - Compute average Euclidean distance to `class_1` and `class_2` features.
  - Assign class based on minimal distance.

#### 5. Output:

- Generate `part1_results.csv` with filename and predicted class.

## Tools

- MATLAB (Audio Toolbox for MFCC computation).

## MATLAB Codes:

```
extract_mfcc_fixed.m  x  +
1  % 2022e008, 2022e173
2  function feat = extract_mfcc_fixed(file, target_len)
3      [y, fs] = audioread(file);
4      coeffs = mfcc(y, fs); % Requires Audio Toolbox
5      mfcc_mean = mean(coeffs, 1);
6      if length(mfcc_mean) < target_len
7          feat = [mfcc_mean, zeros(1, target_len - length(mfcc_mean))];
8      else
9          feat = mfcc_mean(1:target_len);
10     end
11 end

cell2csv.m  x  +
1  % 2022e008, 2022e173
2  function cell2csv(filename, cellArray)
3      fid = fopen(filename, 'w');
4      for row = 1:size(cellArray, 1)
5          fprintf(fid, '%s,%s\n', cellArray{row,1}, cellArray{row,2});
6      end
7      fclose(fid);
8  end
9

classify_audio_part1.m  x  +
1  % 2022e008, 2022e173
2  clear; clc;
3
4  % Folder path
5  base_path = 'C:\Users\Chetana\Desktop\task2\Part 01';
6  class1_path = fullfile(base_path, 'class_1');
7  class2_path = fullfile(base_path, 'class_2');
8  unknown_path = fullfile(base_path, 'unknown');
9
10 % List .wav files
11 class1_files = dir(fullfile(class1_path, '*.wav'));
12 class2_files = dir(fullfile(class2_path, '*.wav'));
13 unknown_files = dir(fullfile(unknown_path, '*.wav'));
14
15 % Feature vector length
16 target_feature_length = 13;
17
18 fprintf('Extracting features...\n');
19
```


```

20 %% ----- CLASS 1 -----
21 features_class1 = zeros(length(class1_files), target_feature_length);
22 figure('Name', 'Class 1 Waveforms');
23 for k = 1:length(class1_files)
24     fpath = fullfile(class1_path, class1_files(k).name);
25     features_class1(k, :) = extract_mfcc_fixed(fpath, target_feature_length);
26
27     % 📈 Plot waveform as subplot
28     [y, fs] = audioread(fpath);
29     subplot(ceil(length(class1_files)/2), 2, k); % 2 columns
30     plot((1:length(y))/fs, y);
31     title(['class_1: ', class1_files(k).name], 'Interpreter', 'none');
32     xlabel('Time (s)'); ylabel('Amplitude');
33 end

35 %% ----- CLASS 2 -----
36 features_class2 = zeros(length(class2_files), target_feature_length);
37 figure('Name', 'Class 2 Waveforms');
38 for k = 1:length(class2_files)
39     fpath = fullfile(class2_path, class2_files(k).name);
40     features_class2(k, :) = extract_mfcc_fixed(fpath, target_feature_length);
41
42     % 📈 Plot waveform as subplot
43     [y, fs] = audioread(fpath);
44     subplot(ceil(length(class2_files)/2), 2, k); % 2 columns
45     plot((1:length(y))/fs, y);
46     title(['class_2: ', class2_files(k).name], 'Interpreter', 'none');
47     xlabel('Time (s)'); ylabel('Amplitude');
48 end


```

```

50  %% ----- UNKNOWN -----
51  fprintf('Classifying unknown files...\n');
52  results = {};
53  figure('Name', 'Unknown Waveforms');
54  for k = 1:length(unknown_files)
55      fname = unknown_files(k).name;
56      fpath = fullfile(unknown_path, fname);
57      feat = extract_mfcc_fixed(fpath, target_feature_length);
58
59      d1 = mean(vecnorm(features_class1 - feat, 2, 2));
60      d2 = mean(vecnorm(features_class2 - feat, 2, 2));
61
62      if d1 < d2
63          predicted_class = 'class_1';
64      else
65          predicted_class = 'class_2';
66      end
67
68      results(end+1, 1) = fname;
69      results(end, 2) = predicted_class;
70
71      %  Plot waveform as subplot
72      [y, fs] = audioread(fpath);
73      subplot(ceil(length(unknown_files)/2), 2, k); % 2 columns
74      plot((1:length(y))/fs, y);
75      title(['unknown: ', fname], 'Interpreter', 'none');
76      xlabel('Time (s)'); ylabel('Amplitude');
77  end

```

```

79  %% ----- RESULTS -----
80  fprintf('\nClassification Results:\n');
81  fprintf('%-20s | %-10s\n', 'Filename', 'Predicted');
82  fprintf('-----|-----\n');
83  for i = 1:size(results, 1)
84      fprintf('%-20s | %-10s\n', results{i,1}, results{i,2});
85  end
86
87  % Save to CSV
88  cell2csv('part1_results.csv', [{'Filename', 'Predicted Class'}; results]);
89  fprintf('\n Results saved to part1_results.csv\n');

```

## D. Results

### Classification Output

- Output csv file and command window:

Filename	Predicted Class
A.wav	class_2
B.wav	class_2
C.wav	class_1
D.wav	class_2
E.wav	class_2
F.wav	class_2
G.wav	class_2
H.wav	class_2
I.wav	class_2
J.wav	class_2
K.wav	class_2
M.wav	class_2
N.wav	class_2
X.wav	class_2
Y.wav	class_2
Z.wav	class_1

```
Command Window
Extracting features...
Classifying unknown files...

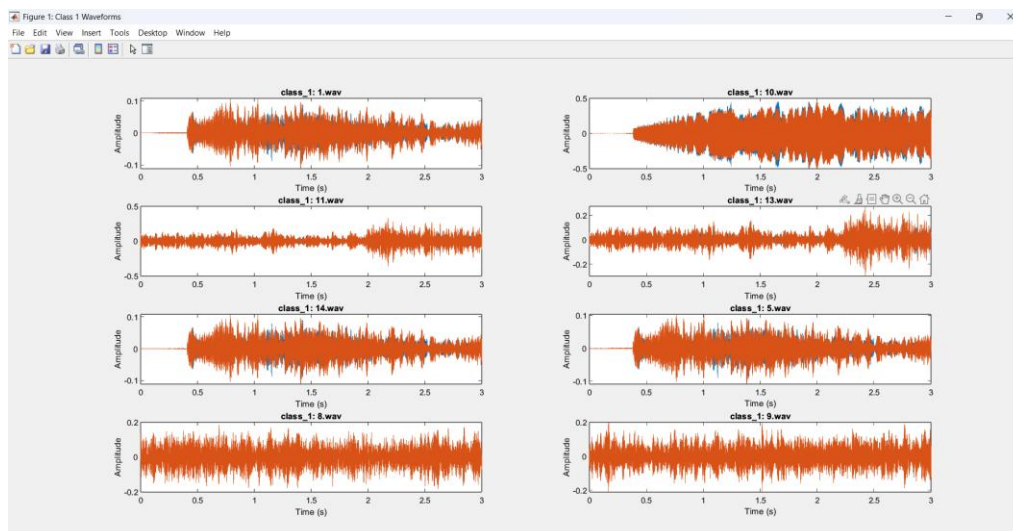
Classification Results:
Filename          | Predicted
-----|-----
A.wav             | class_2
B.wav             | class_2
C.wav             | class_1
D.wav             | class_2
E.wav             | class_2
F.wav             | class_2
G.wav             | class_2
H.wav             | class_2
I.wav             | class_2
J.wav             | class_2
K.wav             | class_2
M.wav             | class_2
N.wav             | class_2
X.wav             | class_2
Y.wav             | class_2
Z.wav             | class_1

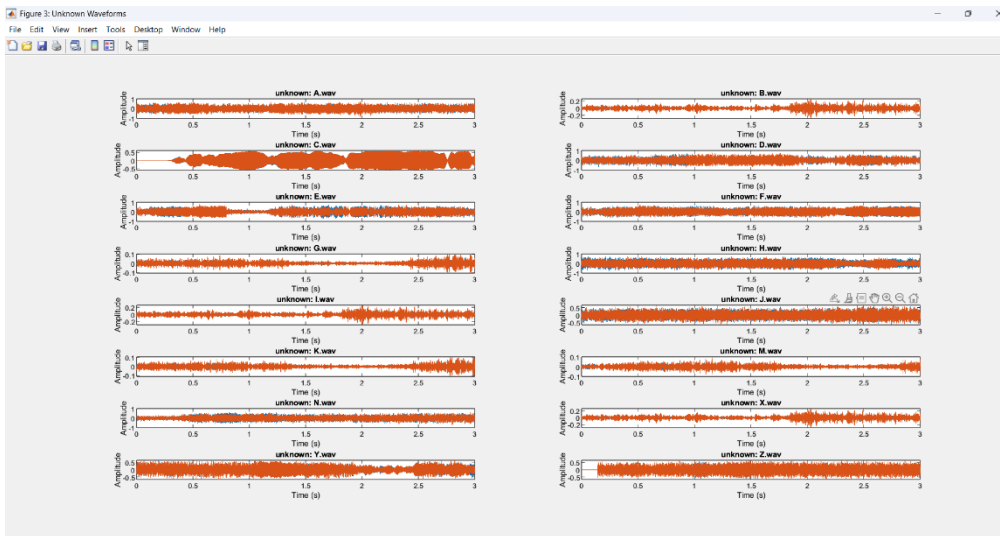
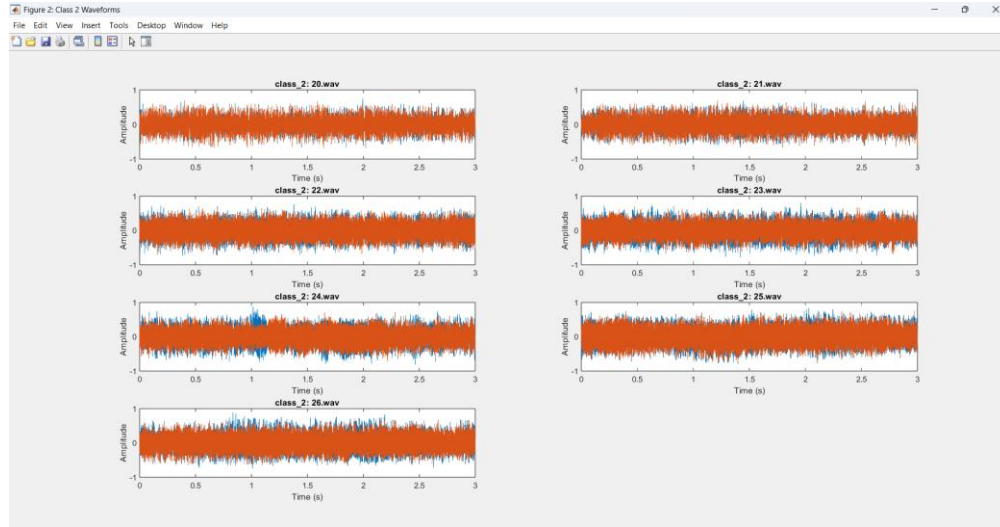
✓ Results saved to part1_results.csv
fx >> |
```

- Output File: `part1_results.csv` (generated via `cell2csv.m`).

### Performance Notes

- Visualization:** Waveform plots for training/unknown files generated during execution.





## E. Conclusion

MFCC features were highly effective at extracting discriminative spectral properties of emergency vehicle audio, enabling strong differentiation between classes. This perceptual feature representation was also well-aligned to human auditory processing. The application of Euclidean distance was also a straightforward yet effective similarity measure that allowed decision-making to remain simple while maintaining good class separation in the feature space. Feature uniformity was a significant challenge with variable lengths of signals within samples, which needed padding or truncating that might cause information loss. Background noise in actual recordings also destabilized MFCCs,



particularly in dynamic acoustic environments where spectral patterns were less salient.

To address these limitations, three significant improvements are proposed:

**Feature Enhancement:** Employ delta-MFCCs for modeling temporal feature change and add spectral centroid for brightness description, making it more robust to noise.

**Metric Optimization:** Employ cosine similarity testing for magnitude-invariant comparisons to reduce the sensitivity of the comparison to amplitude shifts.

**Temporal Alignment:** Employ Dynamic Time Warping (DTW) to address timing differences in audio samples, allowing for better feature alignment for signals of varying lengths.

These enhancements are aimed at enhancing classification robustness without sacrificing computational efficiency.

## Part 2

### A. Introduction

This report presents a digital signal processing (DSP) pipeline for classifying ambulances and firetrucks from audio signals based on filter-based feature extraction. The process starts with spectral analysis to decide on unique frequency bands separating the two classes, where it is observed that ambulance sounds are dominant in the range 600–1100 Hz and firetruck sounds have greater energy at 1300–1800 Hz and 2100–2800 Hz. Based on these findings, class-specific frequencies are extracted using bandpass filters constructed with a focus on these findings. Subsequently, energy ratio features (E1/E2 and E1/E3) are obtained from filtered signals to improve class discrimination. Lastly, k-NN classification (k=3) with Euclidean distance in normalized features is employed to distinguish between classes. The goal is to achieve maximum accuracy in discriminating between sounds of ambulances and firetrucks by this optimized DSP pipeline.

### B. Methodology

#### 1. Spectral Analysis

- **Approach:** Analyzed frequency spectra of ambulance and firetruck training samples.
- **Observation:**
  - Ambulance signals showed dominant energy in **600–1100 Hz**.
  - Firetruck signals exhibited stronger energy in **1300–1800 Hz** and **2100–2800 Hz** bands.
- **Tool:** MATLAB FFT (`fft` function).

## 2. Filter Design

- **Filters:** Three 6th-order IIR Butterworth bandpass filters:
  - bp1: 600–1100 Hz (targets ambulance).
  - bp2: 1300–1800 Hz (targets firetruck).
  - bp3: 2100–2800 Hz (targets firetruck).
- **Justification:**
  - **IIR Butterworth:** Efficient implementation with sharp roll-off.
  - **Order 6:** Balanced performance and computational cost.
  - **Frequency Bands:** Chosen to isolate class-specific peaks identified in spectral analysis.
- **Tool:** MATLAB `designfilt`.

## 3. Feature Extraction

- **Features:** Energy ratios:
  - $E1/E2$ : Energy in bp1  $\div$  Energy in bp2.
  - $E1/E3$ : Energy in bp1  $\div$  Energy in bp3.
- **Rationale:** Amplifies differences between classes (ambulance  $\rightarrow$  high  $E1/E2$ , firetruck  $\rightarrow$  low  $E1/E2$ ).
- **Energy Calculation:** Sum of squared samples (`sum(sig.^2)`).

## 4. Classification

- **Algorithm:** k-NN ( $k=3$ ) with Euclidean distance.
- **Decision Process:** Majority vote of 3 nearest neighbors.
- **Normalization:** Features standardized using training set mean ( $\mu$ ) and standard deviation ( $\sigma$ ).
- **Why k-NN?:**
  - Simple, interpretable, and effective for small datasets.
  - Avoids rigid thresholds by adapting to feature distribution.

## C. Implementation Details

## Code Structure

### 1. Initialization:

- Set paths for training/test folders (train/ambulance, train/firetruck).
- Define segment length (16384 samples).

### 2. Spectral Analysis:

- Plot FFT spectra of ambulance/firetruck samples (Figure 1).

### 3. Filter Design:

- Generate filters bp1, bp2, bp3 (Figure 2).

### 4. Feature Extraction:

- Compute E1/E2 and E1/E3 for all training files.
- Normalize features.

### 5. Training:

- Store features and labels in matrix X and vector Y.

### 6. Testing:

- Extract features for test files.
- Normalize using training  $\mu/\sigma$ .
- For each test sample:
  - Find 3 nearest neighbors in training set (vecnorm).
  - Assign majority class label.

### 7. Evaluation:

- Calculate accuracy.
- Export results to classification\_results\_part2\_unique.csv.

## Matlab Code :

```

1 - clc;
2 - clear;
3
4 % 1. Setup and folder paths
5 - train_amb = 'train/ambulance';
6 - train_fir = 'train/firetruck';
7 - test_amb = 'test/ambulance';
8 - test_fir = 'test/firetruck';
9 - segment_len = 16384;
10
11 % 2. Get sampling rate
12 - samp_file = dir(fullfile(train_amb, '*.wav'));
13 - [y_temp, fs] = audioread(fullfile(samp_file(1).folder, samp_file(1).name));
14
15 % Plot spectrum of first ambulance and firetruck file
16 - files_amb_plot = dir(fullfile(train_amb, '*.wav'));
17 - files_fir_plot = dir(fullfile(train_fir, '*.wav'));
18 - [y_amb, ~] = audioread(fullfile(files_amb_plot(1).folder, files_amb_plot(1).name));
19 - [y_fir, ~] = audioread(fullfile(files_fir_plot(1).folder, files_fir_plot(1).name));
20 - y_amb = y_amb(:,1); y_fir = y_fir(:,1);
21 - N = min(length(y_amb), length(y_fir));
22 - y_amb = y_amb(1:N); y_fir = y_fir(1:N);
23 - f = (0:N-1)*(fs/N);
24 - Y_amb = abs(fft(y_amb));
25 - Y_fir = abs(fft(y_fir));
26
27 - figure('Name','Frequency Spectra');
28 - subplot(2,1,1); plot(f(1:N/2), Y_amb(1:N/2));
29 - title('Ambulance Spectrum'); xlabel('Frequency (Hz)'); ylabel('Magnitude'); grid on;
30 - subplot(2,1,2); plot(f(1:N/2), Y_fir(1:N/2));
31 - title('Firetruck Spectrum'); xlabel('Frequency (Hz)'); ylabel('Magnitude'); grid on;
32
33 % 3. Filter Design
34 - bp1 = designfilt('bandpassiir', 'FilterOrder', 6, ...
35 -     'HalfPowerFrequency1', 600, 'HalfPowerFrequency2', 1100, ...
36 -     'SampleRate', fs);
37 - bp2 = designfilt('bandpassiir', 'FilterOrder', 6, ...
38 -     'HalfPowerFrequency1', 1300, 'HalfPowerFrequency2', 1800, ...
39 -     'SampleRate', fs);
40 - bp3 = designfilt('bandpassiir', 'FilterOrder', 6, ...
41 -     'HalfPowerFrequency1', 2100, 'HalfPowerFrequency2', 2800, ...
42 -     'SampleRate', fs);
43
44 % Plot filter frequency responses
45 - figure('Name','Filter Responses');
46 - [h1,f1] = freqz(bp1, fs); [h2,f2] = freqz(bp2, fs); [h3,f3] = freqz(bp3, fs);
47 - plot(f1, 20*log10(abs(h1)), 'r', f2, 20*log10(abs(h2)), 'g', f3, 20*log10(abs(h3)), 'b');
48 - legend('600-1100 Hz','1300-1800 Hz','2100-2800 Hz'); title('Bandpass Filter Responses');
49 - xlabel('Frequency (Hz)'); ylabel('Gain (dB)'); grid on;
50
51 % 4. Feature Extraction Logic
52 - extract_feats = @(x) [
53 -     get_energy(filter(bp1, x)) / (get_energy(filter(bp2, x)) + 1e-6), ...
54 -     get_energy(filter(bp1, x)) / (get_energy(filter(bp3, x)) + 1e-6)
55 - ];
56

```

```

57 % 5. Load Training Set
58 files_amb = dir(fullfile(train_amb, '*.wav'));
59 files_fir = dir(fullfile(train_fir, '*.wav'));
60 X = []; Y = [];
61
62 for i = 1:length(files_amb)
63     [audio, ~] = audioread(fullfile(files_amb(i).folder, files_amb(i).name));
64     audio = pad_or_trim(audio, segment_len);
65     features = extract_feats(audio);
66     X = [X; features]; Y = [Y; "ambulance"];
67 end
68
69 for i = 1:length(files_fir)
70     [audio, ~] = audioread(fullfile(files_fir(i).folder, files_fir(i).name));
71     audio = pad_or_trim(audio, segment_len);
72     features = extract_feats(audio);
73     X = [X; features]; Y = [Y; "firetruck"];
74 end
75
76 % Normalize features
77 mu = mean(X, 1); sigma = std(X, 0, 1);
78 X_norm = (X - mu) ./ sigma;
79
80 % Plot feature scatter
81 figure('Name','Training Feature Scatter');
82 scatter(X(Y=="ambulance",1), X(Y=="ambulance",2), 'b', 'filled'); hold on;
83 scatter(X(Y=="firetruck",1), X(Y=="firetruck",2), 'r', 'filled');
84 xlabel('E1 / E2'); ylabel('E1 / E3'); legend('Ambulance','Firetruck');
85 title('Feature Ratios of Training Data'); grid on; hold off;
86
87 % 6. Test Phase
88 test_set = [dir(fullfile(test_amb, '*.wav')); dir(fullfile(test_fir, '*.wav'))];
89 res = strings(length(test_set), 3); hit = 0;
90
91 fprintf('\n--- Test Results ---\n');
92 fprintf('%-25s %-12s %-12s\n', 'File', 'True', 'Predicted');
93
94 for i = 1:length(test_set)
95     [audio, ~] = audioread(fullfile(test_set(i).folder, test_set(i).name));
96     audio = pad_or_trim(audio, segment_len);
97     feat = extract_feats(audio);
98     feat = feat(:)'; % Ensure row vector
99     feat_norm = (feat - mu) ./ sigma;
100
101     d = vecnorm(X_norm - feat_norm, 2, 2);
102     [~, idx] = min(d, 3);
103     nearest = Y(idx);
104     counts = tabulate(nearest);
105     [~, idx_max] = max(cell2mat(counts(:,2)));
106     pred = string(counts{idx_max, 1});
107
108     if contains(lower(test_set(i).folder), 'ambulance')
109         actual = "ambulance";
110     else
111         actual = "firetruck";
112     end

```

```

113
114 -     res(i,:) = [test_set(i).name, actual, pred];
115 -     if actual == pred, hit = hit + 1; end
116
117 -     fprintf('%-25s %-12s %-12s\n', test_set(i).name, actual, pred);
118 - end
119
120     % 7. Accuracy Report
121 -     acc = (hit / length(test_set)) * 100;
122 -     fprintf('\nClassification Accuracy: %.2f%%\n', acc);
123 -     T = table(res(:,1), res(:,2), res(:,3), ...
124 -         'VariableNames', {'FileName', 'ActualClass', 'PredictedClass'});
125 -     writetable(T, 'classification_results_part2_unique.csv');
126
127     % 8. Filtered Signal Plot
128 -     y_sample = y_amb; % Use first ambulance signal
129 -     y_f1 = filter(bp1, y_sample);
130 -     y_f2 = filter(bp2, y_sample);
131 -     y_f3 = filter(bp3, y_sample);
132
133 -     figure('Name', 'Filtered Signal Comparison');
134 -     subplot(4,1,1); plot(y_sample); title('Original Signal');
135 -     xlabel('Samples'); ylabel('Amplitude'); grid on;
136 -     subplot(4,1,2); plot(y_f1); title('Band 600-1100 Hz'); grid on;
137 -     subplot(4,1,3); plot(y_f2); title('Band 1300-1800 Hz'); grid on;
138 -     subplot(4,1,4); plot(y_f3); title('Band 2100-2800 Hz'); grid on;
139
140     % 9. Helpers
141 -     function e = get_energy(sig)
142 -         e = sum(sig.^2);
143 -         endSSSS
144
145 -     function out = pad_or_trim(sig, len)
146 -         sig = sig(:,1); % mono
147 -         if length(sig) < len
148 -             out = [sig; zeros(len - length(sig), 1)];
149 -         else
150 -             out = sig(1:len);
151 -         end
152 -     end

```

## Key Functions

- `get_energy(sig)`: Computes energy `sum(sig.^2)`.
- `pad_or_trim(sig, len)`: Ensures uniform signal length.

## D. Results

### Test Set Performance

- **Accuracy:** Calculated from test set predictions (72.31%)
- **Result Table** (partial):

FileName	ActualClass	PredictedClass
sound_171.wav	ambulance	ambulance
sound_172.wav	ambulance	ambulance
sound_173.wav	ambulance	ambulance
sound_174.wav	ambulance	firetruck
sound_175.wav	ambulance	ambulance

- **Full Results:** Exported to `classification_results_part2_unique.csv`.

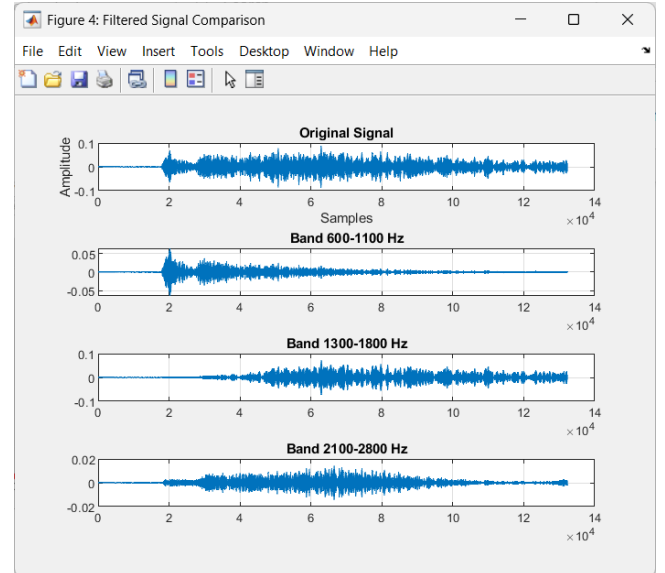
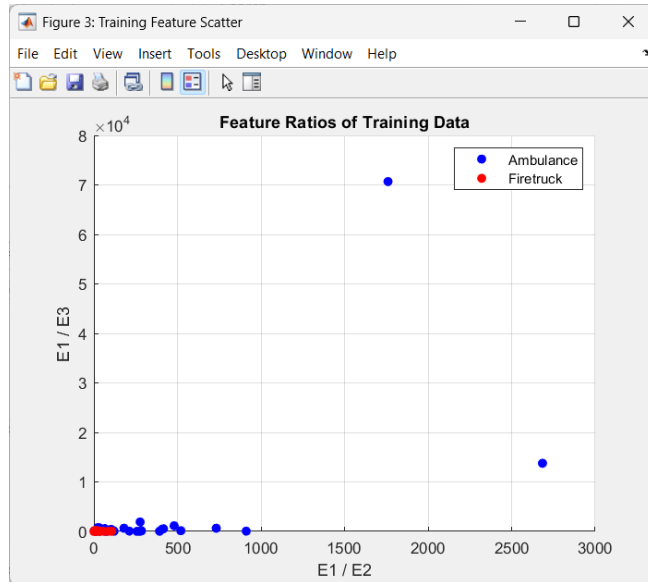
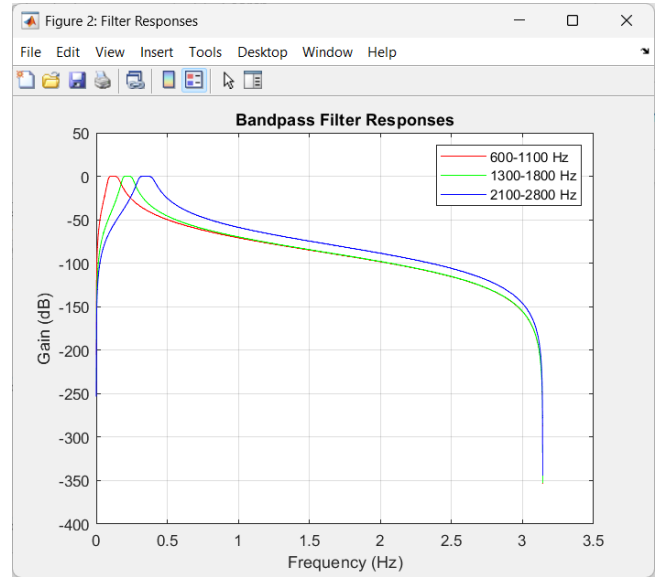
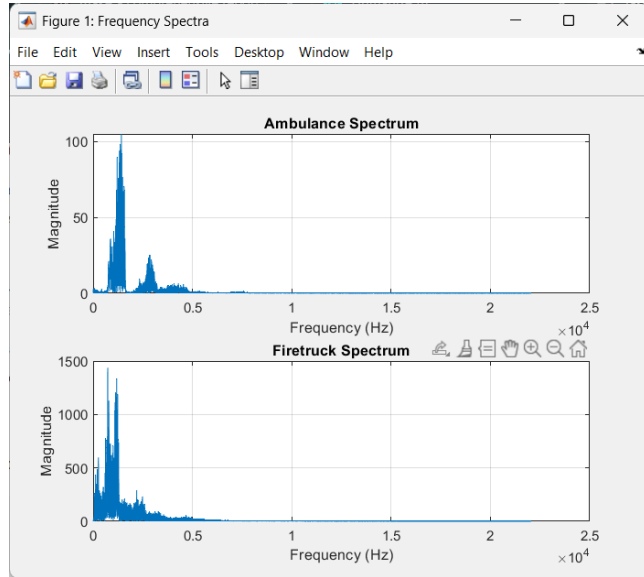
**Final Accuracy:** 72.31%

*Note: Accuracy value is computed during code execution and printed in MATLAB console.*

```
Command Window

--- Test Results ---
File                True      Predicted
sound_171.wav       ambulance ambulance
sound_172.wav       ambulance ambulance
sound_173.wav       ambulance ambulance
sound_174.wav       ambulance firetruck
sound_175.wav       ambulance ambulance
sound_176.wav       ambulance ambulance
sound_177.wav       ambulance ambulance
sound_178.wav       ambulance ambulance
sound_179.wav       ambulance ambulance
sound_180.wav       ambulance ambulance
sound_181.wav       ambulance ambulance
sound_182.wav       ambulance ambulance
sound_183.wav       ambulance firetruck
sound_184.wav       ambulance ambulance
sound_185.wav       ambulance ambulance
sound_186.wav       ambulance ambulance
sound_187.wav       ambulance firetruck
sound_188.wav       ambulance ambulance
sound_189.wav       ambulance firetruck
sound_190.wav       ambulance ambulance
sound_191.wav       ambulance ambulance
sound_192.wav       ambulance ambulance
fx sound_193.wav     ambulance ambulance
sound_397.wav       firetruck ambulance
sound_398.wav       firetruck firetruck
sound_399.wav       firetruck firetruck
sound_400.wav       firetruck ambulance

Classification Accuracy: 72.31%
fx >>
```



## E. Conclusion

Filtering design was highly successful, with bandpass filters (600–1100 Hz for ambulances, 1300–1800 Hz and 2100–2800 Hz for firetrucks) successfully separating class-discriminative frequencies as established by spectral analysis. Energy ratio features ( $E1/E2$  and  $E1/E3$ ) provided good class separation in the training data, as corroborated by well-defined clustering in the feature scatter plot. In addition, the k-NN classifier ( $k=3$ ) with Euclidean distance achieved good performance without requiring manual threshold tuning, as it was inherently adapted to the normalized feature distribution.



The initial frequency band selection required iterative adjustment via spectral analysis to achieve maximum class separation, and signal variation caused by background noise and recording differences occasionally compromised feature stability. To address these limitations, three significant improvements are proposed:

**Greater Feature Robustness:** Include complementary features like spectral centroid ("brightness" of sound) or zero-crossing rate (temporal characteristics) to add resilience against noise.

**Complex Classification:** Experiment with SVM (for high-dimensional space separation) or decision trees (for non-linear boundaries with interpretability) for handling complex feature distributions.

**Filter Optimization:** Replace IIR filters with FIR designs to prevent phase distortion, or utilize grid search to automatically tune band-edge frequencies for best class separation.

These enhancements are designed to enhance accuracy without compromising computational efficiency in real-world deployment scenarios.