EC5011

DIGITAL SIGNAL PROCESSING

TASK 02 : SIMPLE AUDIO CLASSIFICATION USING FEATURE EXTRACTION

GROUP 56

2022/E/048 - BANDARA H.G.T.D.

2022/E/049 - HERATH H.M.I.U.

SEMESTER 05

23 JUN 2025

INTRODUCTION

This report focuses on the classification of audio signals using digital signal processing techniques, specifically feature extraction and filter design. The objective of the task is to accurately categorize unknown audio samples into predefined classes using extracted audio features and to design a reliable classification system for emergency vehicle sounds such as ambulances and firetrucks.

In Part 1 of the task, classification is performed by extracting distinguishing features from audio signals using techniques like Mel-Frequency Cepstral Coefficients (MFCC) and Fast Fourier Transform (FFT). These features are then compared using distance or similarity metrics such as Euclidean distance or cosine similarity to determine the closest match for unknown samples.

Part 2 involves the design of digital filters based on the spectral content of the audio signals. These filters are used to enhance classification accuracy by focusing on specific frequency ranges that best differentiate between the two sound classes. Energy ratios computed from the filtered signals are used as classification features, and thresholds are set to separate the classes effectively.

This report outlines the methodology, implementation details, and results obtained from both parts of the task, along with a discussion on the challenges encountered and possible improvements.

PART 01 : MFCC-BASED CLASSIFICATION

1. Introduction

Briefly state the problem:

- The objective is to classify unknown audio signals as belonging to either class_1 or class_2 using digital signal processing and feature extraction techniques.
- The approach involves extracting robust features from each audio file, comparing these features between known and unknown samples, and assigning each unknown file to the most similar known class.

2. Methodology

2.1. Feature Extraction

Process:

- For every audio file, extracted features that capture the unique characteristics of the sound.
- The two main methods suggested in the assignment were:
  - MFCC (Mel-Frequency Cepstral Coefficients)
  - FFT (Fast Fourier Transform)

Mathematical Explanation:

MFCCs:

- MFCCs are computed by:
    1. Framing: Divide the audio signal $x[n]$ into short frames.
    2. Windowing: Multiply each frame by a window function (e.g., Hamming).
    3. FFT: Compute the magnitude spectrum for each frame:

       $$X[k] = \sum_{n=0}^{N-1} x[n]w[n]e^{-j2\pi kn/N}$$

    4. Mel Filter Bank: Pass the spectrum through a bank of triangular bandpass filters spaced on the Mel scale.
    5. Logarithm: Take the logarithm of the filter bank energies.
    6. DCT: Apply the Discrete Cosine Transform to decorrelate the coefficients and obtain the MFCCs:

       $$c_m = \sum_{n=1}^{K} \log(E_n)\cos\left[\frac{\pi m}{K}(n-0.5)\right]$$

       where *En* is the energy in the *n*-th Mel filter.
- Why MFCCs?
  MFCCs are robust to noise and variation, and they mimic human auditory perception, making them ideal for distinguishing between classes of sounds.
- The FFT provides the frequency spectrum, but MFCCs are generally more effective for classification due to their perceptual basis.

2.2. Similarity/Distance Metrics

Process:

- For each unknown file, its feature vector was compared to those of all files in class_1 and class_2.
- A distance or similarity metric was used to measure how "close" the unknown file was to each class.

Mathematical Explanation:

- Euclidean Distance:
  For feature vectors f*u* (unknown) and f*k* (known):

  $$d(\mathbf{f}_u, \mathbf{f}_k) = \sqrt{\sum_{i=1}^{N} (f_{u,i} - f_{k,i})^2}$$

- Cosine Similarity:

$$\text{similarity}(\mathbf{f}_u, \mathbf{f}_k) = \frac{\mathbf{f}_u \cdot \mathbf{f}_k}{|\mathbf{f}_u||\mathbf{f}_k|}$$

- Manhattan Distance:

$$d(\mathbf{f}_u, \mathbf{f}_k) = \sum_{i=1}^{N} |f_{u,i} - f_{k,i}|$$

- Why Euclidean?
  Euclidean distance is simple, effective, and aligns with the assignment requirements.

2.3. Classification Logic

Process:

- For each unknown file:
  1. Its MFCC feature vector was extracted.
  2. The distance to every file in class_1 and class_2 was computed.
  3. The file was assigned to the class whose closest training file (minimum distance) was nearest.

Mathematical Rule:

- Let $D1$ be the minimum distance to any file in class_1 and $D2$ for class_2.
- Assign to:

$$\text{Class} = \arg\ \min\{D_1, D_2\}$$

3. Implementation Details

- Code Structure:

  1. All audio files from class_1, class_2, and unknown were loaded.
  2. The files were preprocessed: converted to mono and resampled to a common sampling rate.
  3. MFCC features were extracted for all files.
  4. For each unknown file, the Euclidean distance to all known files was computed.
  5. The class was assigned based on the minimum distance.

- Tools: MATLAB (using audioread, mfcc, mean, std, and KNN classifier).

Editor - C:\Users\user\Downloads\report_with_screen_shots\Part_01.m

Part_01.m | Part_02.m

```matlab
%% Part 1: Audio Classification Using Feature Extraction
clear; clc; close all;

% ========== PATH CONFIGURATION (Relative Paths) ==========
class1_path = 'class_1';
class2_path = 'class_2';
unknown_path = 'unknown';

% Verify paths exist
if ~exist(class1_path, 'dir')
    error('Class 1 folder not found. Make sure "class_1" folder exists in the same directory as this script.');
end
if ~exist(class2_path, 'dir')
    error('Class 2 folder not found. Make sure "class_2" folder exists in the same directory as this script.');
end
if ~exist(unknown_path, 'dir')
    error('Unknown folder not found. Make sure "unknown" folder exists in the same directory as this script.');
end

fprintf('=== PART 1: AUDIO CLASSIFICATION USING FEATURE EXTRACTION ===\n\n');

%% Step 1: Load and Extract Features with Consistent Sizing

% Get file lists
class1_files = dir(fullfile(class1_path, '*.wav'));
class2_files = dir(fullfile(class2_path, '*.wav'));
unknown_files = dir(fullfile(unknown_path, '*.wav'));

fprintf('Found %d files in class_1\n', length(class1_files));
fprintf('Found %d files in class_2\n', length(class2_files));
fprintf('Found %d files in unknown\n', length(unknown_files));

% Fixed parameters for consistent feature extraction
NUM_MFCC_COEFFS = 13;
TARGET_SAMPLE_RATE = 16000;

% Initialize feature storage
class1_features = [];
class2_features = [];

fprintf('\nExtracting MFCC features from Class 1 files...\n');
for i = 1:length(class1_files)
    file_path = fullfile(class1_path, class1_files(i).name);
```

```matlab
    try
        [audio, fs] = audioread(file_path);

        % Preprocessing for consistency
        if fs ~= TARGET_SAMPLE_RATE
            audio = resample(audio, TARGET_SAMPLE_RATE, fs);
            fs = TARGET_SAMPLE_RATE;
        end

        % Convert to mono if stereo
        if size(audio, 2) > 1
            audio = mean(audio, 2);
        end

        % Extract MFCC features
        mfcc_features = mfcc(audio, fs, 'NumCoeffs', NUM_MFCC_COEFFS);

        % Create fixed-size feature vector using statistical measures
        feature_vector = [
            mean(mfcc_features, 1), ...     % Mean of each coefficient (13 features)
            std(mfcc_features, 0, 1), ...  % Standard deviation (13 features)
            max(mfcc_features, [], 1), ... % Maximum values (13 features)
            min(mfcc_features, [], 1)      % Minimum values (13 features)
        ];

        class1_features = [class1_features; feature_vector];
        fprintf('  Processed: %s (Feature size: %d)\n', class1_files(i).name, length(feature_vector));

    catch ME
        fprintf('  Error processing %s: %s\n', class1_files(i).name, ME.message);
        continue;
    end
end

fprintf('\nExtracting MFCC features from Class 2 files...\n');
for i = 1:length(class2_files)
    file_path = fullfile(class2_path, class2_files(i).name);

    try
        [audio, fs] = audioread(file_path);

        % Preprocessing for consistency
        if fs ~= TARGET_SAMPLE_RATE
```

```matlab
            audio = resample(audio, TARGET_SAMPLE_RATE, fs);
            fs = TARGET_SAMPLE_RATE;
        end

        % Convert to mono if stereo
        if size(audio, 2) > 1
            audio = mean(audio, 2);
        end

        % Extract MFCC features
        mfcc_features = mfcc(audio, fs, 'NumCoeffs', NUM_MFCC_COEFFS);

        % Create fixed-size feature vector using statistical measures
        feature_vector = [
            mean(mfcc_features, 1), ...     % Mean of each coefficient
            std(mfcc_features, 0, 1), ...  % Standard deviation
            max(mfcc_features, [], 1), ... % Maximum values
            min(mfcc_features, [], 1)      % Minimum values
        ];

        class2_features = [class2_features; feature_vector];
        fprintf('  Processed: %s (Feature size: %d)\n', class2_files(i).name, length(feature_vector));

    catch ME
        fprintf('  Error processing %s: %s\n', class2_files(i).name, ME.message);
        continue;
    end
end

% Verify feature consistency
fprintf('\nFeature Matrix Summary:\n');
fprintf('Class 1 features: %d samples x %d features\n', size(class1_features, 1), size(class1_features, 2));
fprintf('Class 2 features: %d samples x %d features\n', size(class2_features, 1), size(class2_features, 2));

% Check if feature sizes match
if size(class1_features, 2) ~= size(class2_features, 2)
    error('Feature sizes do not match! Class1: %d, Class2: %d', ...
          size(class1_features, 2), size(class2_features, 2));
end
```

```matlab
%% Step 2: Classification of Unknown Files

fprintf('\nClassifying unknown files using Euclidean Distance...\n');
results = [];
expected_feature_size = size(class1_features, 2);

for i = 1:length(unknown_files)
    file_path = fullfile(unknown_path, unknown_files(i).name);

    try
        [audio, fs] = audioread(file_path);

        % Apply same preprocessing
        if fs ~= TARGET_SAMPLE_RATE
            audio = resample(audio, TARGET_SAMPLE_RATE, fs);
            fs = TARGET_SAMPLE_RATE;
        end

        if size(audio, 2) > 1
            audio = mean(audio, 2);
        end

        % Extract MFCC features with same structure
        mfcc_features = mfcc(audio, fs, 'NumCoeffs', NUM_MFCC_COEFFS);

        unknown_feature = [
            mean(mfcc_features, 1), ...
            std(mfcc_features, 0, 1), ...
            max(mfcc_features, [], 1), ...
            min(mfcc_features, [], 1)
        ];

        % Verify feature size consistency
        if length(unknown_feature) ~= expected_feature_size
            fprintf('  Warning: Feature size mismatch for %s. Expected: %d, Got: %d\n', ...
                    unknown_files(i).name, expected_feature_size, length(unknown_feature));
            continue;
        end

        % Calculate distances to all class 1 samples using Euclidean distance
        distances_class1 = [];
        for j = 1:size(class1_features, 1)
            dist = sqrt(sum((unknown_feature - class1_features(j, :)).^2));
```

```
                distances_class1 = [distances_class1; dist];
            end
            
            % Calculate distances to all class 2 samples
            distances_class2 = [];
            for j = 1:size(class2_features, 1)
                dist = sqrt(sum((unknown_feature - class2_features(j, :)).^2));
                distances_class2 = [distances_class2; dist];
            end
            
            % Find minimum distances
            min_dist_class1 = min(distances_class1);
            min_dist_class2 = min(distances_class2);
            
            % Classify based on minimum distance
            if min_dist_class1 < min_dist_class2
                predicted_class = 1;
                confidence = min_dist_class2 / (min_dist_class1 + min_dist_class2);
            else
                predicted_class = 2;
                confidence = min_dist_class1 / (min_dist_class1 + min_dist_class2);
            end
            
            % Store results
            results = [results; struct('filename', unknown_files(i).name, ...
                                       'predicted_class', predicted_class, ...
                                       'confidence', confidence, ...
                                       'dist_class1', min_dist_class1, ...
                                       'dist_class2', min_dist_class2)];
            
            fprintf('  %s -> Class %d (Confidence: %.3f)\n', ...
                    unknown_files(i).name, predicted_class, confidence);
            
        catch ME
            fprintf('  Error processing %s: %s\n', unknown_files(i).name, ME.message);
            continue;
        end
    end
```

```
%% Display Final Results
fprintf('\n=== PART 1 CLASSIFICATION RESULTS ===\n');
fprintf('%-25s %-15s %-12s %-12s %-12s\n', 'Filename', 'Predicted Class', 'Confidence', 'Dist Class1', 'Dist Class2');
fprintf('%s\n', repmat('-', 1, 80));
for i = 1:length(results)
    fprintf('%-25s %-15d %-12.3f %-12.3f %-12.3f\n', ...
            results(i).filename, results(i).predicted_class, ...
            results(i).confidence, results(i).dist_class1, results(i).dist_class2);
end

% Save results
save('part1_classification_results.mat', 'results', 'class1_features', 'class2_features');
fprintf('\nResults saved to part1_classification_results.mat\n');
fprintf('Part 1 completed successfully!\n\n');
```

Figure 01:MATLAB Codes for Part 01

4. Results

```
=== PART 1 CLASSIFICATION RESULTS ===
Filename                 Predicted Class Confidence   Dist Class1  Dist Class2
-----------------------------------------------------------------------------
A.wav                    2               0.734        9.578        3.471
B.wav                    1               0.907        1.817        17.706
C.wav                    1               0.972        60.970       2142.613
D.wav                    2               0.747        8.329        2.826
E.wav                    2               0.669        7.831        3.882
F.wav                    2               0.680        8.663        4.083
G.wav                    1               0.795        5.467        21.213
H.wav                    2               0.857        9.851        1.638
I.wav                    1               0.932        1.250        17.170
J.wav                    2               0.715        10.213       4.078
K.wav                    1               0.795        5.453        21.194
M.wav                    1               0.783        6.111        22.005
N.wav                    2               0.729        8.562        3.182
X.wav                    1               0.907        1.807        17.705
Y.wav                    2               0.683        8.519        3.958
Z.wav                    1               0.963        81.127       2098.766
```

Figure 02:Output Data Of Part 01

- Report overall accuracy:

$$\text{Accuracy} = \frac{\text{Number of correctly classified unknown files}}{\text{Total unknown files}} \times 100\%$$

- Confidence:
  Optionally, the confidence was defined as the ratio of the distances:

$$\text{Confidence} = \frac{\max(D_1, D_2)}{D_1 + D_2}$$

5. Conclusion

- Summary:
  - MFCCs with Euclidean distance provided robust classification.
  - The method is fast, interpretable, and reliable for real-time and batch classification.
- Challenges:
  - Overlapping classes or noisy recordings may reduce accuracy.
- Improvements/Alternatives:
  - Combining MFCCs with other features (e.g., spectral centroid, zero-crossing rate) could further improve performance.
  - Trying different classifiers (SVM, Random Forest) or distance metrics.

PART 2: FILTER DESIGN AND CLASSIFICATION OF EMERGENCY VEHICLE SOUNDS

Introduction

This section addresses the problem of distinguishing between ambulance and firetruck siren sounds using digital signal processing techniques. The goal is to analyze the frequency content of both classes, design filters to isolate distinguishing features, and build a robust classification pipeline that can accurately label unseen audio samples from the test set1.

B. Methodology

Feature Extraction

- **Spectral Analysis:**
  All training audio files were loaded and resampled to a common sampling rate of 16 kHz for consistency. The frequency content was analyzed using the Fast Fourier Transform (FFT) with an NFFT of 2048. The average spectra for both ambulance and firetruck classes were computed and visualized to identify frequency regions where the two classes differ significantly.

Filter Design

- **Filter Type and Order:**
  Based on the spectral analysis, three 8th-order IIR bandpass filters were designed using MATLAB's designfilt function. The chosen bands were:
  - Band 1: 500–1100 Hz
  - Band 2: 1300–2000 Hz
  - Band 3: 2200–3200 Hz
- **Justification:**
  These frequency ranges were selected as they showed the most pronounced differences in energy between the two classes in the training spectra. Using multiple bands allows the classifier to capture more nuanced differences than a single band.

Feature Construction

- **Energy Ratios:**
  For each audio file, the signal was filtered through each bandpass filter. The energy in each band was computed as the sum of squared amplitudes of the filtered signal. To form robust features, ratios between the band energies were calculated:
  - Band1/Band2
  - Band2/Band3
  - Band1/Band3

Classification

- **Classifier Choice:**
  A Linear Discriminant Analysis (LDA) classifier was trained on the feature vectors extracted from the training data. LDA was chosen for its transparency and effectiveness with linearly separable features, offering more robustness than a simple threshold-based approach.

- **Threshold Setting:**
  The LDA classifier implicitly sets decision boundaries based on the distribution of the feature vectors in the training set.

C. Implementation Details

- **Code Structure:**
  The code is organized into clear sections:
  1. Path setup and data loading
  2. Frequency content analysis and visualization
  3. Filter design and response plotting
  4. Feature extraction from filtered energies
  5. Classifier training (LDA)
  6. Testing and evaluation on the test set

- **Classification Logic:**
  For each test file, the same preprocessing and feature extraction steps are applied. The trained LDA model predicts the class label based on the extracted features.

- **Tools Used:**
  All implementation was done in MATLAB, using built-in functions such as audioread, fft, designfilt, filtfilt, and fitcdiscr.

```
clear; clc; close all;

%% --- 1. Path Setup ---
train_amb = fullfile('filter','train','ambulance');
train_fire = fullfile('filter','train','firetruck');
test_amb = fullfile('filter','test','ambulance');
test_fire = fullfile('filter','test','firetruck');
fs_target = 16000;

%% --- 2. Analyze Frequency Content (Training Data) ---
fprintf('Analyzing frequency content of training data...\n');
amb_train_files = dir(fullfile(train_amb,'*.wav'));
fire_train_files = dir(fullfile(train_fire,'*.wav'));
NFFT = 2048;
amb_spectra = zeros(NFFT/2+1, numel(amb_train_files));
fire_spectra = zeros(NFFT/2+1, numel(fire_train_files));
for i = 1:numel(amb_train_files)
    [audio, fs] = audioread(fullfile(train_amb, amb_train_files(i).name));
    audio = preprocess_audio(audio, fs, fs_target);
    S = abs(fft(audio, NFFT));
    amb_spectra(:,i) = S(1:NFFT/2+1);
end
for i = 1:numel(fire_train_files)
    [audio, fs] = audioread(fullfile(train_fire, fire_train_files(i).name));
    audio = preprocess_audio(audio, fs, fs_target);
    S = abs(fft(audio, NFFT));
    fire_spectra(:,i) = S(1:NFFT/2+1);
end
mean_amb = mean(amb_spectra,2);
mean_fire = mean(fire_spectra,2);
freqs = linspace(0, fs_target/2, NFFT/2+1);

% Plot average spectra for both classes
figure;
plot(freqs, mean_amb, 'b', 'LineWidth',1.5); hold on;
plot(freqs, mean_fire, 'r', 'LineWidth',1.5);
xlabel('Frequency (Hz)'); ylabel('Magnitude');
legend('Ambulance','Firetruck'); title('Average Spectrum (Training)');
grid on;

%% --- 3. Design Multiple Bandpass Filters (Filter Bank) ---
% Choose 3 frequency bands based on spectrum plot
bands = [500 1100; 1300 2000; 2200 3200]; % [low high] for each band
numBands = size(bands,1);
fprintf('Designing %d bandpass filters...\n', numBands);
bpFilters = cell(numBands,1);
for b = 1:numBands
    bpFilters{b} = designfilt('bandpassiir','FilterOrder',8, ...
        'HalfPowerFrequency1',bands(b,1),'HalfPowerFrequency2',bands(b,2), ...
        'SampleRate',fs_target);
end

% Plot filter responses
figure;
for b = 1:numBands
    [h, f] = freqz(bpFilters{b}, 1024, fs_target);
    plot(f, 20*log10(abs(h)), 'DisplayName',sprintf('Band %d: %d-%d Hz',b,banc
end
xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
title('Bandpass Filter Responses'); legend; grid on;

```

```
%% --- 4. Compute Filtered Energies and Feature Vectors ---
fprintf('Extracting filter-bank energy features for training data...\n');
train_features = [];
train_labels = [];
for i = 1:numel(amb_train_files)
    [audio, fs] = audioread(fullfile(train_amb, amb_train_files(i).name));
    audio = preprocess_audio(audio, fs, fs_target);
    energies = zeros(1,numBands);
    for b = 1:numBands
        energies(b) = band_energy(audio, bpFilters{b});
    end
    % Use ratios between bands as features
    feat = [energies(1)/energies(2), energies(2)/energies(3), energies(1)/ener
    train_features = [train_features; feat];
    train_labels = [train_labels; 1];
end
for i = 1:numel(fire_train_files)
    [audio, fs] = audioread(fullfile(train_fire, fire_train_files(i).name));
    audio = preprocess_audio(audio, fs, fs_target);
    energies = zeros(1,numBands);
    for b = 1:numBands
        energies(b) = band_energy(audio, bpFilters{b});
    end
    feat = [energies(1)/energies(2), energies(2)/energies(3), energies(1)/ener
    train_features = [train_features; feat];
    train_labels = [train_labels; 2];
end

% Plot feature distributions
figure;
gscatter(train_features(:,1),train_features(:,2),train_labels, 'br','ox');
xlabel('Energy Ratio Band1/Band2'); ylabel('Energy Ratio Band2/Band3');
title('Training Feature Scatter Plot'); legend('Ambulance','Firetruck'); grid

%% --- 5. Train Linear Classifier (LDA) ---
% Still transparent, but more robust than a single threshold
lda = fitcdiscr(train_features, train_labels);

%% --- 6. Test and Evaluate ---
fprintf('Classifying test files...\n');
amb_test_files = dir(fullfile(test_amb,'*.wav'));
fire_test_files = dir(fullfile(test_fire,'*.wav'));
test_files = [arrayfun(@(f) fullfile(test_amb, f.name), amb_test_files, 'Unifo
              arrayfun(@(f) fullfile(test_fire, f.name), fire_test_files, 'Uni
test_labels = [ones(1,numel(amb_test_files)), 2*ones(1,numel(fire_test_files))
test_features = [];
for i = 1:numel(test_files)
    [audio, fs] = audioread(test_files{i});
    audio = preprocess_audio(audio, fs, fs_target);
    energies = zeros(1,numBands);
    for b = 1:numBands
        energies(b) = band_energy(audio, bpFilters{b});
    end
    feat = [energies(1)/energies(2), energies(2)/energies(3), energies(1)/ener
    test_features = [test_features; feat];
end
[pred, score] = predict(lda, test_features);

% Print results
class_names = {'ambulance','firetruck'};
```

```
correct = sum(pred(:)' == test_labels);
fprintf('\n%-25s %-12s %-12s %-10s\n','File','True Class','Prediction','Score
fprintf('%s\n',repmat('-',[1 60]));
for i = 1:numel(test_files)
    fprintf('%-25s %-12s %-12s %-10.2f\n', ...
        test_files{i}(max(1,end-20):end), ...
        class_names{test_labels(i)}, ...
        class_names{pred(i)}, ...
        max(score(i,:)));
end
acc = 100*correct/numel(test_files);
fprintf('\nFinal Accuracy: %.2f%% (%d/%d)\n', acc, correct, numel(test_files)

% (Optional) Plot test features
figure;
gscatter(test_features(:,1),test_features(:,2),pred, 'br','ox');
xlabel('Energy Ratio Band1/Band2'); ylabel('Energy Ratio Band2/Band3');
title('Test Feature Scatter Plot (Predicted Classes)'); legend('Ambulance','F

%% --- Helper Functions ---
function audio = preprocess_audio(audio, fs, fs_target)
    if size(audio,2)>1, audio = mean(audio,2); end
    if fs ~= fs_target, audio = resample(audio, fs_target, fs); end
    audio = audio / (max(abs(audio)) + eps);
end
function e = band_energy(audio, filt)
    filtered = filtfilt(filt, audio);
    e = sum(filtered.^2);
end
```

Figure 03: Code Set For Part_02

Results

- Present a table: For each test file, show the file name, true class, predicted class, energy ratio, and confidence (distance from threshold).
- Report overall accuracy:

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of test files}} \times 100\%$$

```
Analyzing frequency content of training data...
Designing 3 bandpass filters...
Extracting filter-bank energy features for training data...
Classifying test files...

File                      True Class    Prediction    Score
------------------------------------------------------------
bulance\sound_171.wav     ambulance     ambulance     0.58
bulance\sound_172.wav     ambulance     firetruck     0.50
bulance\sound_173.wav     ambulance     firetruck     0.55
bulance\sound_174.wav     ambulance     firetruck     0.56
bulance\sound_175.wav     ambulance     ambulance     0.83
bulance\sound_176.wav     ambulance     ambulance     0.99
bulance\sound_177.wav     ambulance     ambulance     0.55
bulance\sound_178.wav     ambulance     firetruck     0.57
bulance\sound_179.wav     ambulance     firetruck     0.53
bulance\sound_180.wav     ambulance     ambulance     0.59
bulance\sound_181.wav     ambulance     firetruck     0.57
bulance\sound_182.wav     ambulance     ambulance     0.50
bulance\sound_183.wav     ambulance     firetruck     0.51
bulance\sound_184.wav     ambulance     firetruck     0.56
bulance\sound_185.wav     ambulance     firetruck     0.51
bulance\sound_186.wav     ambulance     firetruck     0.51
bulance\sound_187.wav     ambulance     firetruck     0.57
bulance\sound_188.wav     ambulance     ambulance     0.69
bulance\sound_189.wav     ambulance     firetruck     0.52
bulance\sound_190.wav     ambulance     firetruck     0.51
bulance\sound_191.wav     ambulance     ambulance     0.91
bulance\sound_192.wav     ambulance     ambulance     0.64
bulance\sound_193.wav     ambulance     ambulance     0.61
bulance\sound_194.wav     ambulance     firetruck     0.55
bulance\sound_195.wav     ambulance     ambulance     0.99
bulance\sound_196.wav     ambulance     ambulance     0.54
bulance\sound_197.wav     ambulance     firetruck     0.51
bulance\sound_198.wav     ambulance     ambulance     0.51
bulance\sound_199.wav     ambulance     ambulance     0.59
bulance\sound_200.wav     ambulance     firetruck     0.53
retruck\sound_366.wav     firetruck     firetruck     0.57
retruck\sound_367.wav     firetruck     firetruck     0.55
retruck\sound_368.wav     firetruck     firetruck     0.58
retruck\sound_369.wav     firetruck     firetruck     0.57
retruck\sound_370.wav     firetruck     firetruck     0.56
retruck\sound_371.wav     firetruck     firetruck     0.58
retruck\sound_372.wav     firetruck     firetruck     0.57
retruck\sound_373.wav     firetruck     firetruck     0.55
retruck\sound_374.wav     firetruck     firetruck     0.57
retruck\sound_375.wav     firetruck     firetruck     0.56
retruck\sound_376.wav     firetruck     firetruck     0.57
retruck\sound_377.wav     firetruck     firetruck     0.58
retruck\sound_378.wav     firetruck     firetruck     0.57
retruck\sound_379.wav     firetruck     ambulance     0.66
retruck\sound_380.wav     firetruck     firetruck     0.57
retruck\sound_381.wav     firetruck     firetruck     0.58
```

```
retruck\sound_382.wav     firetruck    firetruck    0.57
retruck\sound_383.wav     firetruck    firetruck    0.58
retruck\sound_384.wav     firetruck    firetruck    0.56
retruck\sound_385.wav     firetruck    firetruck    0.56
retruck\sound_386.wav     firetruck    firetruck    0.57
retruck\sound_387.wav     firetruck    firetruck    0.58
retruck\sound_388.wav     firetruck    firetruck    0.55
retruck\sound_389.wav     firetruck    ambulance    0.50
retruck\sound_390.wav     firetruck    firetruck    0.58
retruck\sound_391.wav     firetruck    firetruck    0.57
retruck\sound_392.wav     firetruck    firetruck    0.56
retruck\sound_393.wav     firetruck    firetruck    0.56
retruck\sound_394.wav     firetruck    firetruck    0.55
retruck\sound_395.wav     firetruck    firetruck    0.57
retruck\sound_396.wav     firetruck    firetruck    0.53
retruck\sound_397.wav     firetruck    ambulance    0.52
retruck\sound_398.wav     firetruck    firetruck    0.57
retruck\sound_399.wav     firetruck    firetruck    0.57
retruck\sound_400.wav     firetruck    ambulance    0.97

Final Accuracy: 69.23% (45/65)
fx >>
```

Figure 04: Output For Part_02

Conclusion

The filter-bank approach, combined with energy ratio features and LDA classification, provided a transparent and effective solution for distinguishing ambulance and firetruck sirens. The main strengths of this method are its interpretability and reliance on signal processing principles.

Challenges included selecting optimal frequency bands and ensuring robustness to variations in siren recordings.

Potential improvements could involve:

- Using more specific features
- Exploring non-linear classifiers for potentially higher accuracy
- Applying data augmentation to increase robustness to noise and recording conditions.