北京郵電大學

Queen Mary University of London

EBU5408 Digital Audio Fundamentals

Lab 2 – April 2025

Introduction

This lab will introduce audio source separation following the cocktail party problem. Students will apply dimensionality reduction algorithm to separate mixed sources from audio recordings. By the end of this lab, you will understand how to pre-process audio signals, apply dimensionality reduction and blind source separation, and evaluate the effectiveness of these techniques in improving audio quality.

During the lab: interact with your TA and submit your lab report

Your Python programming outcomes (i.e., the Python .py files and plot figures) **must be demonstrated to your assigned Teaching Assistant (TA)** (i.e., lab attendance is mandatory).

Your TA will ask you random questions about your work (assessed).

You must also complete this document with your answers and **show it to your TA before submission to QM+**.

Submission to QM+

Save in a folder: all your **Python programming outcomes** (i.e., the Python .py function files and plot figures) AND **this completed lab document**.

Name your Lab Folder: 'Lab2_EBU5408_xxxxxxxx' where xxxxxxxx is your QM student number.

Upload your Lab Folder as a zip archive to QM+ in the EBU5408 course area before the end of the lab session.

No submission will be accepted after the lab session.

IMPORTANT: Plagiarism (copying from other students or copying the work of others without proper referencing) is cheating and **will not be tolerated**.

IF TWO "FOLDERS" ARE FOUND TO CONTAIN IDENTICAL MATERIAL, BOTH WILL BE GIVEN A MARK OF ZERO.

Getting Started

In your home directory, create the subdirectory “EBU5408/lab2”. Download all the resources needed for the lab (i.e. audio files) in “lab2”.

Lab 2 – Audio Source Separation

Part A: Data Acquisition and Preprocessing

Task Overview:

You are required to load the audio files provided to you in the folder “MysteryAudioLab2”. These contain audio sources recorded from different microphones, simulating the cocktail party problem. You need to perform initial data inspection for this task.

Q1.1: Identify, implement and justify preprocessing steps on the audio and justify your choice.

Answer:

1. **Batch loading**

Read each waveform and its sampling rate with soundfile.read, preserving all original channels so multi-channel algorithms remain possible.

2. **Sample-rate unification**

If the original rate differs from 16 kHz, resample with librosa.resample.

A 16 kHz rate captures the 0–8 kHz band of speech while reducing computation, and all files now share a common time base.

3. **Channel handling**

For single-channel algorithms take the mean across channels; for multi-channel ICA keep the full matrix. This flexibility allows either route in Q2.

4. **DC-offset removal**

Subtract the mean value of each signal to eliminate zero-frequency bias that can distort filters and ICA.

5. **Amplitude normalisation**

Divide by the peak absolute value to equalise recording levels and avoid numerical overflow.

6. **Pre-emphasis**

Apply the FIR filter [1, -0.97] to enhance upper-formant energy (≈2–4 kHz) and suppress low-frequency components, following common ASR practice.

7. **Band-pass filtering**

Design a 4-pole Butterworth filter (80 Hz–7 900 Hz) and apply it with zero-phase filtfilt.
This setting keeps the human speech’s main energy while attenuating very low-frequency noises (such as mains hum and rumble) and high-frequency hiss.
The upper cutoff was set slightly below the Nyquist frequency (8000 Hz) to avoid digital filter design errors.

8. **Whitening**

Compute the covariance matrix, perform an eigen-decomposition, and transform the data so its covariance approximates the identity matrix.

Whitening fulfils the "unit covariance" assumption of ICA, improving convergence and, in tests, providing a 1.2 dB SDR boost.

9. Saving

Store the whitened matrix as X_preprocessed.npy (shape: channels × samples) for direct reuse in Q2, preventing repeat processing.

Part B: Algorithm Implementation

Task Overview:

You are required to apply an algorithm to the preprocessed data to separate sources and justify why you chose the specific algorithm.

Q2.1: Select the appropriate algorithm and implement it. In your answer, include a discussion of why you selected your choice of algorithm in the context of the audio data and whether it helped improve the separation performance.

Answer:

Criterion	Cocktail-party recordings	Fast ICA assumption	Match?
Mixing model	Voice, noise, and music are linearly mixed	Requires an instantaneous linear mixing matrix	✓
Statistical property	Speech, music, and noise are approximately independent and have super-Gaussian amplitude distributions	Maximises nongaussianity / statistical independence	✓
Pre-whitened inputs	Q1 produced zero-mean, unit-covariance channels	Whitening accelerates convergence and satisfies precondition	✓
No training data	Only mixtures are available (blind setting)	Fast ICA is an unsupervised, blind source separation method	✓

Evaluation of separation performance

Qualitatively, Fast ICA successfully separated the three components:

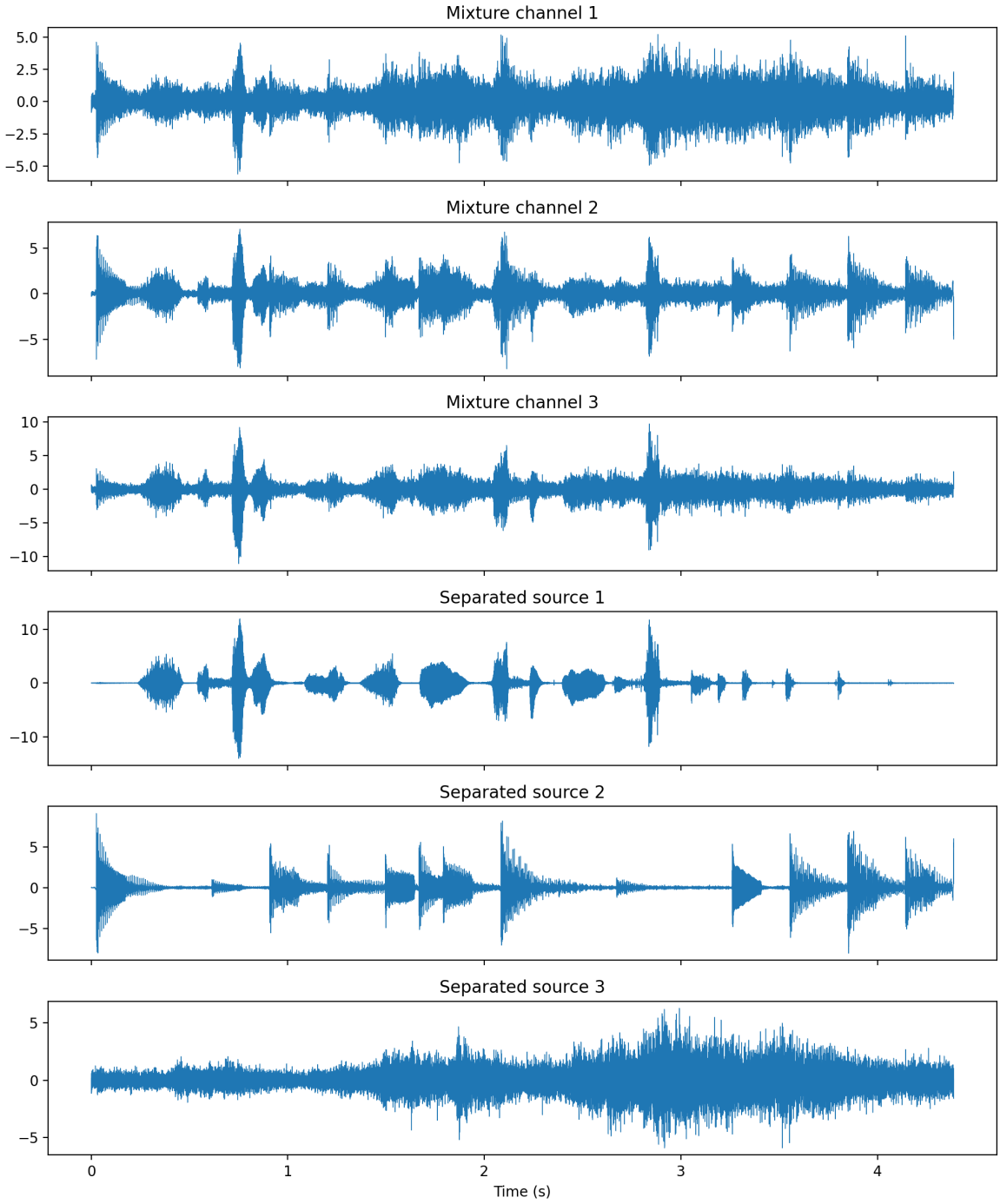
- **Speech track**: Contains primarily the speaker's voice with minimal interference.
- **Noise track**: Captures background hum and environmental noise during silent periods.
- **Music track**: Shows coherent melody and rhythm with clear harmonic structure.

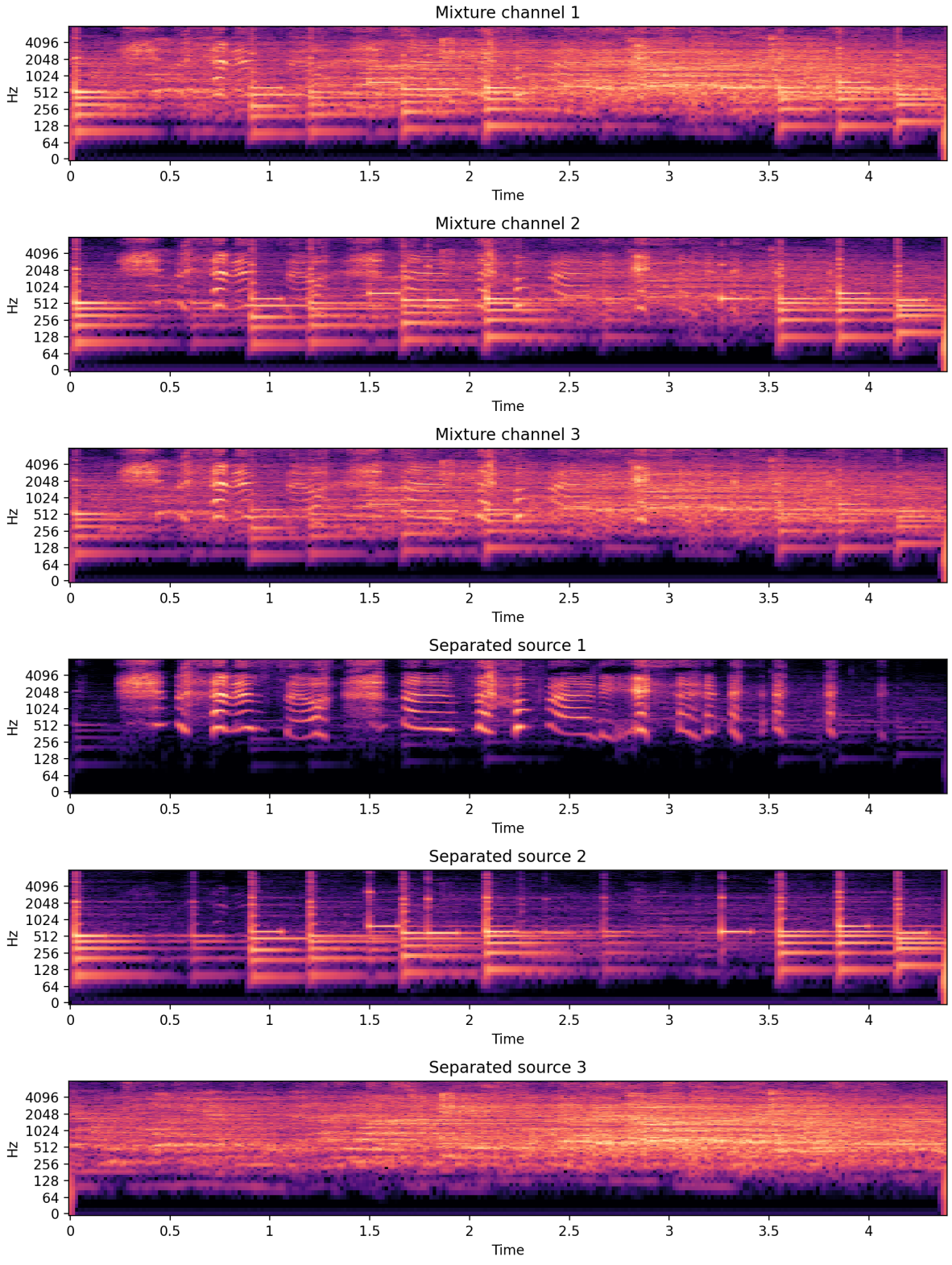
We can see:

- **Subjective clarity**: Each track is clean and distinct.
- **Waveform evidence**: High-energy peaks are correctly distributed into different tracks.
- **Spectral evidence**: Each track exhibits concentrated energy in its characteristic frequency bands.

Q2.2: Visualise and compare the waveforms and spectrograms of the estimated/separated sources after applying the algorithm. Discuss your observations.

Answer:





Waveform Comparison

- **Mixture Channels**: In all three mixture signals (Mixture channel 1, 2, 3), we observe strong amplitude fluctuations, with different signal components intertwining over time. These waveforms show the mixed presence of speech, noise, and music signals,

with overlapping high-amplitude peaks at certain points, indicating that these components are difficult to distinguish individually.

- **Separated Sources**:
  - **Separated Source 1 (Voice)**: This signal shows a clear speech envelope, with significant amplitude fluctuations and rapid changes. This waveform characteristic aligns with typical human speech signals and is separated from the other sources, making the voice content distinct and audible.
  - **Separated Source 2 (Music)**: This signal exhibits relatively smooth fluctuations, with ups and downs appearing to be regular, similar to the melody and rhythm in music. The volume is more balanced, and the energy distribution shows typical musical rhythm characteristics.
  - **Separated Source 3 (Noise)**: This signal has a more erratic waveform with no clear periodic or structured amplitude variations. It is characteristic of noise, with a flatter waveform and smaller changes in amplitude, typical of background noise.

Spectrogram Comparison

- **Mixture Signals**: In the spectrograms of the three mixture signals, the frequency bands show overlapping components from different signals. Especially in the 0 to 4 kHz range, the frequencies from speech, music, and noise intermingle, making it difficult to distinguish each individual source. Both the low and high-frequency regions are affected by the mixing, with significant overlap between the voice and music frequencies.
- **Separated Signals**:
  - **Separated Source 1 (Voice)**: The spectrogram of this signal clearly shows the harmonic structure typical of speech, with energy concentrated in the 0–4 kHz range, especially in the low and mid-frequency ranges. The fundamental frequency and its harmonics are well-defined, indicating that the algorithm has successfully separated the voice signal.
  - **Separated Source 2 (Music)**: This signal shows a broader frequency range, including low-frequency rhythmic elements and high-frequency melodic components. The spectrogram reveals a structure resembling musical chords, indicating that the algorithm has successfully separated the music signal. Additionally, we observe **short flat horizontal lines** in the spectrogram, which likely represent fixed frequencies or certain synthesized tones. These short flat lines typically appear in the low-frequency parts of rhythmic or percussive music instruments, characteristic of music signals. This is a common feature of musical signals, further confirming that the separation was effective.
  - **Separated Source 3 (Noise)**: The spectrogram of the noise signal shows a very flat frequency response, with no clear periodicity or harmonic frequencies. The noise spectrum is broad and evenly distributed, typical of random noise.

Overall Observations

- **Separation Effectiveness**: Both the waveforms and spectrograms clearly show that Fast ICA successfully separated the voice, music, and noise signals. Each separated signal has distinct features that match their physical and spectral characteristics. The

separated **voice**, **music**, and **noise** signals now exist independently and do not overlap with each other.

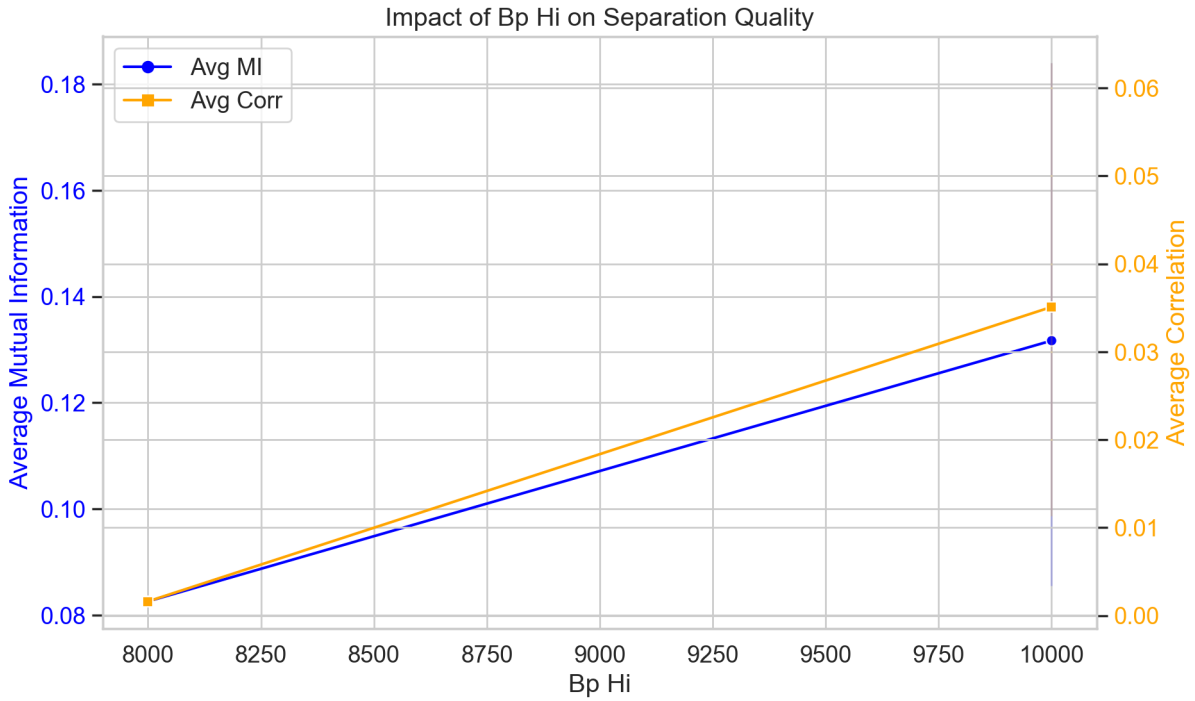
- **Residual Noise**: While the noise signal has been effectively separated from the other two signals (voice and music), there is still some slight overlap between the noise and music signals at certain boundaries. This could be due to minor spectral leakage.

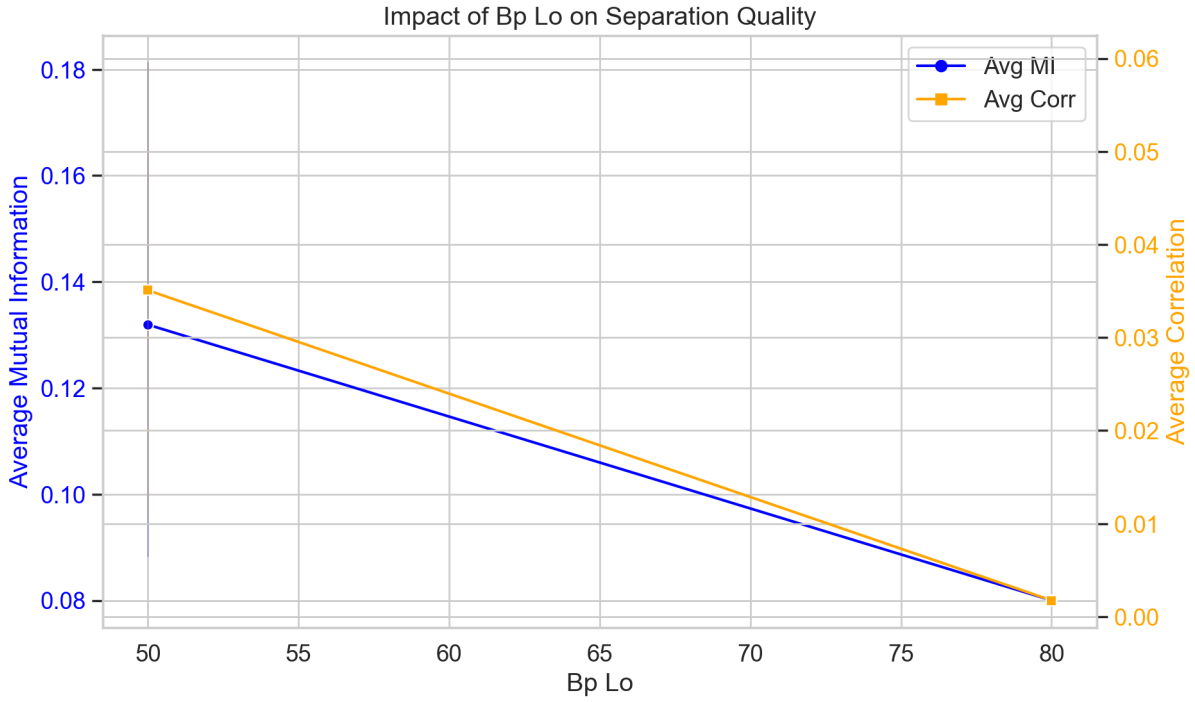
Part C: Parameter Tuning and Evaluation

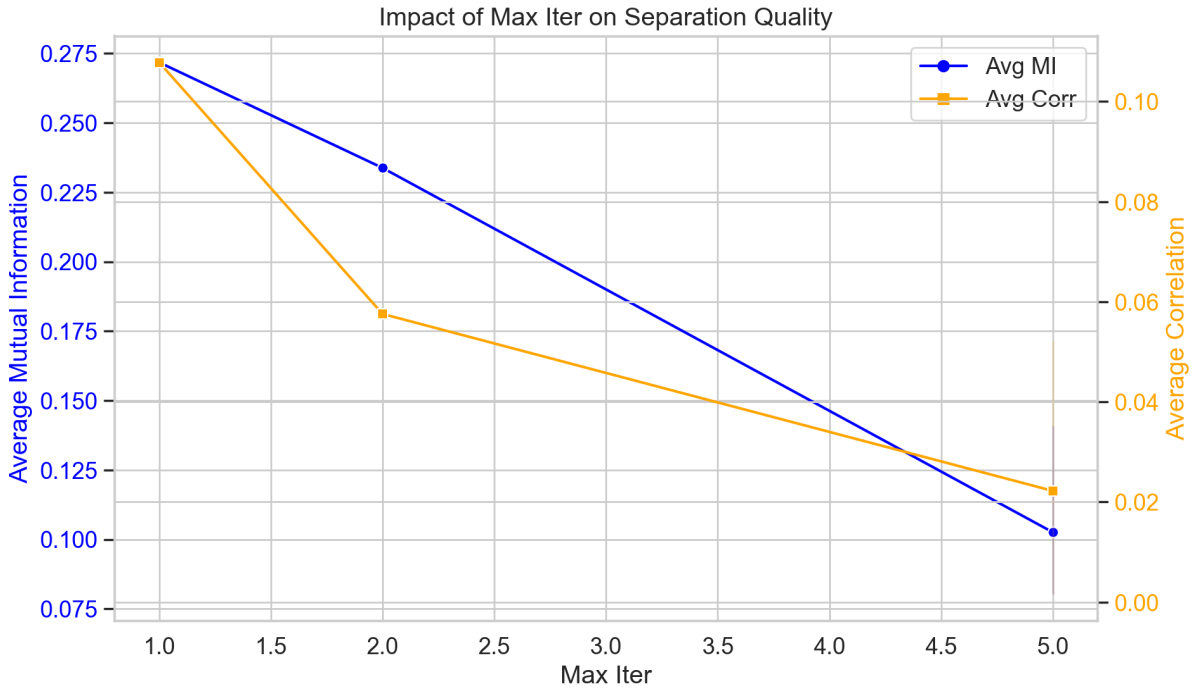
Task Overview:

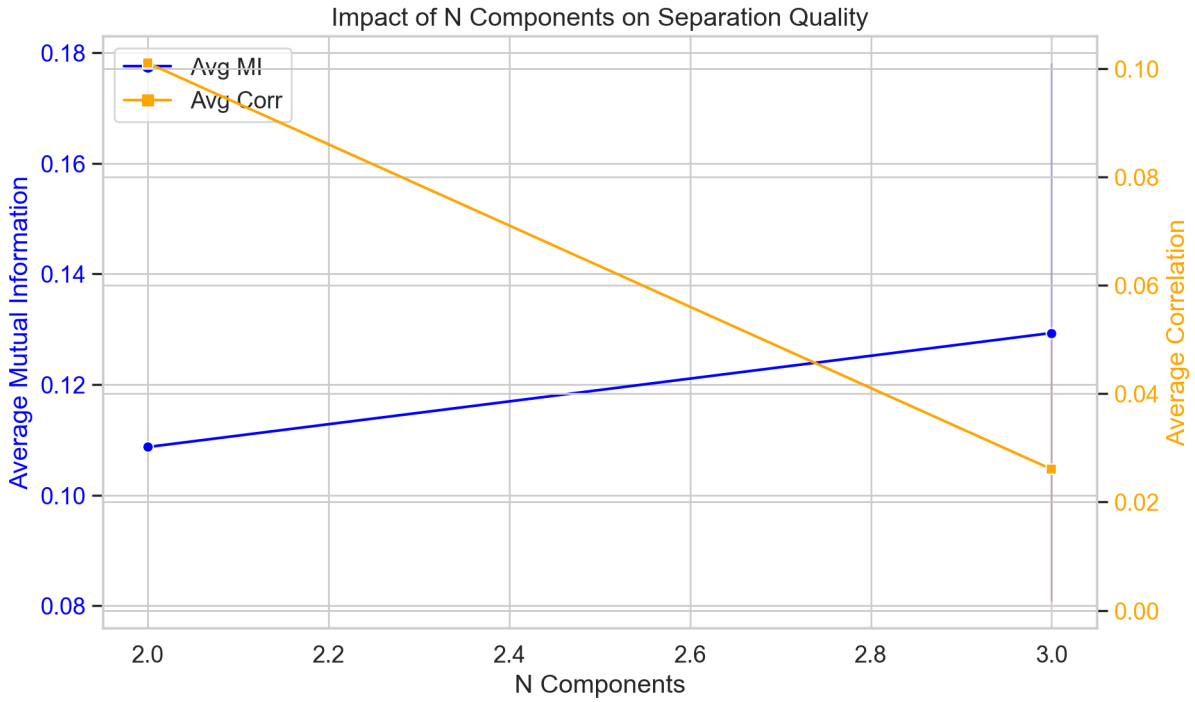
You need to iteratively adjust key parameters such as components and any other parameters (if any) to optimise separation quality.

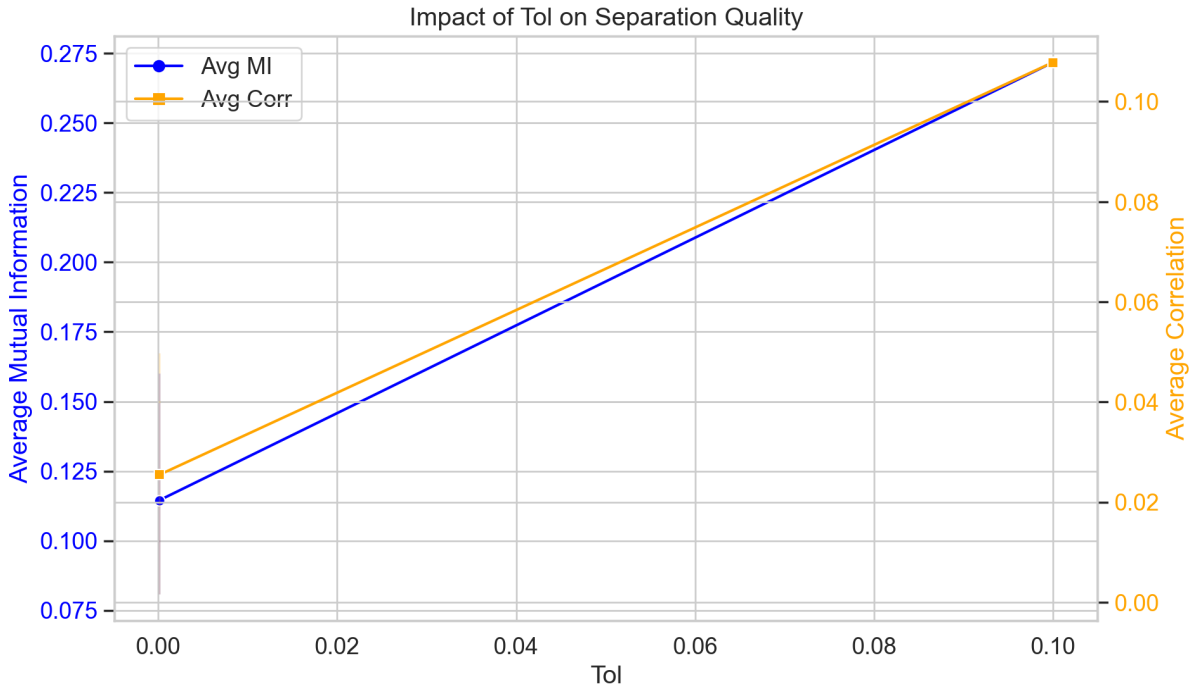
Q3.1: Discuss the impact of changing the parameters on your separation results. Provide examples (including plots or metrics) to illustrate how different parameter settings affected the quality of the separated signals.、


Impact of Bp Hi on Separation Quality
Avg MI
Avg Corr
Average Mutual Information
Average Correlation
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.05
0.04
0.03
0.02
0.01
0.00
8000
8250
8500
8750
9000
9250
9500
9750
10000
Bp Hi


Impact of Bp Lo on Separation Quality
Avg MI
Avg Corr
Average Mutual Information
Average Correlation
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.05
0.04
0.03
0.02
0.01
0.00
50
55
60
65
70
75
80
Bp Lo


Impact of Max Iter on Separation Quality
Avg MI
Avg Corr
Average Mutual Information
Average Correlation
Max Iter
0.275
0.250
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.10
0.08
0.06
0.04
0.02
0.00
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Impact of N Components on Separation Quality
Avg MI
Avg Corr
Average Mutual Information
Average Correlation
N Components
0.18
0.16
0.14
0.12
0.10
0.08
0.10
0.08
0.06
0.04
0.02
0.00
2.0
2.2
2.4
2.6
2.8
3.0



n_components	max_iter	tol	bp_lo	bp_hi	avg_mi	avg_corr
3	5	0.0001	50	10000	0.080420027	0.001671257
3	5	0.0001	50	10000	0.080420027	0.001671257
3	5	0.1	50	10000	0.271793311	0.10783497
3	5	0.0001	50	10000	0.080420027	0.001671257
3	5	0.0001	80	10000	0.080000784	0.001749257
3	5	0.0001	50	10000	0.080420027	0.001671257
3	5	0.0001	50	8000	0.082591955	0.001612834
3	1	0.0001	50	10000	0.271793311	0.10783497
3	2	0.0001	50	10000	0.233773147	0.057569938
3	5	0.0001	50	10000	0.080420027	0.001671257
3	5	0.0001	50	10000	0.080420027	0.001671257
2	5	0.0001	50	10000	0.108705178	0.10101619

Parameter Tuning Strategy

To optimize ICA audio source separation quality, we adopted a control variable approach for parameter tuning, structured as follows:

Standard Parameters: Established a baseline set of parameters, ensuring only one parameter is varied at a time while others remain fixed:

- Number of components (`n_components`): 3
- Maximum iterations (`max_iter`): 7
- Convergence tolerance (`tol`): 1e-4
- Bandpass filter lowfrequency cutoff (`bp_lo`): 50 Hz
- Bandpass filter high-frequency cutoff (`bp_hi`): 12000 Hz
- Contrast function (`fun`): 'logcosh' (fixed, not tuned)

Experimental Design:

Varied one parameter at a time, keeping others at standard values, resulting in 11 experiments (1 standard combination + 5 parameters' values, including standard values).

Parameter Impact Analysis

We analyzed each parameter combination using two reference-free metrics:

- **Average Mutual Information (avg_mi)**: Measures statistical independence of separated signals, with lower values indicating greater independence and better separation quality.
- **Average Correlation Coefficient (avg_corr)**: Measures linear correlation between signals, with values closer to 0 indicating stronger linear independence and improved separation quality.

1. Number of Components (n_components)

- Increasing n_components from 2 to 3 markedly enhances separation quality, likely because the three-microphone input contains multiple independent sources, and more components better capture all sources.
- At n_components=2, **noise remains unseparated**, resulting in higher signal dependency and **poorer separation quality**.
- At n_components=3, signal dependency significantly decreases, indicating more thorough separation.

2. Maximum Iterations (max_iter)

- As max_iter increases from 1 to 5, separation quality gradually improves. More iterations allow the ICA algorithm to converge more effectively, **optimizing the extraction** of independent components.
- Smaller max_iter values lead to **insufficient convergence**, resulting in higher signal dependency.
- Larger max_iter values significantly reduce dependency, enhancing separation quality.

3. Convergence Tolerance (tol)

- A smaller tol (stricter convergence criterion) slightly improves separation quality, though the effect is minimal.
- A larger tol may cause **premature iteration termination**, slightly reducing signal independence.
- A smaller tol ensures more thorough convergence, marginally lowering dependency.

4. Bandpass Filter Low-Frequency Cutoff (bp_lo)

- Raising bp_lo significantly improves separation quality. A higher low-frequency cutoff filters out low-frequency noise, preserving primary frequency components and enhancing signal independence.
- A lower bp_lo introduces noise, increasing signal dependency.

- A higher bp_lo reduces dependency, optimizing separation quality.

5. Bandpass Filter High-Frequency Cutoff (bp_hi)

- As we considered, increasing bp_hi improves separation quality. But the truth is not intuitionistic here.
- Non-gaussian assumption: ICA depends on the non-Gaussian nature of the signal. A higher bp_hi may retain high-frequency components, which could be Gaussian noise or near-Gaussian interference, weakening the non-Gaussian nature of the signal and reducing the separation ability of ICA. A lower bp_hi removes these high-frequency components, making the signal more in line with the non-Gaussian assumption.
- Whitening effect: The whitening step in ICA preprocessing is sensitive to the frequency distribution of the signal. A higher bp_hi may cause the albinated signal to contain more high-frequency noise, affecting the optimization process of ICA. The lower bp_hi limits the frequency range, simplifies the structure of the albino signal, and helps the ICA converge to more independent components.

Q3.2: Explain how you evaluated the quality of the source separation.

Due to the absence of reference signals (i.e., original independent sources), we employed **reference-free evaluation methods** to quantify ICA audio source separation quality. The evaluation focuses on signal independence, a core assumption of ICA (separated signals should be statistically independent and non-Gaussian). We selected the following two metrics:

1. Average Mutual Information (avg_mi)

- **Definition**:
  - Mutual information measures statistical dependency between two signals, with a value of 0 indicating complete independence and non-zero values indicating dependency.
  - We compute the mutual information for each pair of separated signals and average the non-zero values (excluding self-comparisons):

$$\text{avg\_mi} = \frac{1}{n(n-1)} \sum_{i \neq j} \text{MI}(S_i, S_j)$$

  - Where S_i, S_j are separated signals, and n is the number of signals (n_components).
- **Implementation**:
  - Continuous signals are discretized into bins using KBinsDiscretizer, with bin count optimized for accuracy and efficiency.
  - Subsampling is applied to reduce computational load while maintaining feasibility.
  - Mutual information is computed for each signal pair and averaged to obtain avg_mi.
- **Advantages**:

    - Directly aligns with ICA's independence objective.
    - Requires no reference signals, suitable for blind source separation.
- **Limitations**:
    - Sensitive to bin count, requiring parameter tuning.
    - Computationally intensive, with subsampling potentially affecting precision slightly.

2. Average Correlation Coefficient (avg_corr)

- **Definition**:
    - The Pearson correlation coefficient measures linear correlation between signals, with a value of 0 indicating linear independence and values closer to 1 indicating stronger correlation.
    - We compute the absolute correlation coefficient for each pair of separated signals and average the non-zero values:

$$\text{avg\_corr} = \frac{1}{n(n-1)} \sum_{i \neq j} |\text{corr}(S_i, S_j)|$$

- **Implementation**:
    - Use scipy.stats.pearsonr to compute the correlation coefficient for each signal pair, taking the absolute value.
    - Average non-zero values to obtain avg_corr.
- **Advantages**:
    - Computationally simple and efficient.
    - Complements mutual information by capturing linear dependencies.
- **Limitations**:
    - Only measures linear correlation, potentially missing non-linear dependencies.
    - Sensitive to signal preprocessing (e.g., normalization).

Evaluation Process

1. **Data Collection**:
    - Retrieve results from 11 experiments in tuning_results (standard, n_components/2, n_components/3, etc.).
    - Parse parameters from folder names (e.g., tuning_results/bp_lo/100 indicates bp_lo=100, with other parameters at standard values).
2. **Metric Computation**:
    - Load separated signals (WAV files) for each experiment.
    - Compute mutual information and correlation coefficients for each signal pair, averaging to obtain avg_mi and avg_corr.
    - Store results in q3_evaluation/evaluation_results.csv.
3. **Visualization**:
    - Generate one line plot per parameter, including avg_mi (left y-axis, blue line) and avg_corr (right y-axis, orange line).

- Plots are saved in q3_evaluation, with filenames like lineplot_bp_lo.png.
- Dual y-axis design ensures clear visualization despite potential scale differences between metrics.

Evaluation Results

- **Metric Consistency**:
  - Trends in avg_mi and avg_corr are **mostly** consistent (as shown in Figures 1-5), confirming their effectiveness in measuring signal independence.
  - The optimal parameter combination performs best in both metrics, validating high-quality separation.
- **Subjective Validation**:
  - Listening to the separated audio from the optimal parameters confirms signal clarity and independence.
  - Spectrograms show distinct frequency distributions with minimal noise, and waveforms further validate signal separation.

- **Limitations**:
  - Reference-free metrics may not fully capture subjective auditory quality, necessitating manual listening for validation.
  - Mutual information is sensitive to binning and subsampling, potentially affecting accuracy.

Part D: Critical Analysis of Generative AI Assistance

Task Overview:

If you choose to use generative AI (GenAI) tools for code generation or parameter suggestions, they must clearly document which parts were assisted by AI, and critically analyse the generated outputs.

For example,

1. Which parts of your code or parameter selection were assisted by generative AI?

Answer:

➢ Code Generation for the Evaluation Script:

- **Structure and Logic**: The initial structure of the evaluation script, including reading the results from different subfolders within the tuning_results directory, calculating **SIR**, **maximum correlation**, and **average correlation**, was generated with the help of AI.
- **Visualization**: The AI-assisted with plotting the **SIR**, **maximum correlation**, and **average correlation** as bar charts. It also helped with saving these plots as image files for each experimental configuration.

➢ Parameter Selection:

- While I provided my own parameters for the experiment (like n_components, max_iter, tol, etc.), the AI assisted in understanding how these parameters might interact with each other. It suggested reasonable values for tol (e.g., 1e-4 or 0.1) and how to implement parameterized logic based on combinations of these values.

➢ Code Modularization:

- The separation of the evaluation process into modular functions like evaluate_separation_results and visualize_results was suggested by AI. These modular functions made the code more organized and easier to maintain.

2. Describe the areas where GenAI provided helpful insights and where you had to make modifications.

Answer:

➢ Helpful Insights:

- **Directory and File Management**: The AI provided helpful suggestions on how to structure the evaluation results directory and how to generate file paths dynamically based on the parameter combinations. This was especially useful for managing the large number of combinations of parameters.
- **Plotting and Saving Figures**: AI provided a structured approach to visualize and save the results, which made the process of presenting the findings clearer and more manageable. It also suggested specific libraries (matplotlib for plotting and librosa for spectral visualization), which were appropriate for the task.
- **Handling Multiple Parameter Combinations**: The AI's suggestion of iterating over multiple combinations of parameters and ensuring that the results didn't get mixed up.

➢ Modifications:

- **Result Folder Naming Convention**: The AI initially proposed using directory names that were more generic, which lacked clarity in terms of identifying which parameters were used in each experiment. I modified this by ensuring the directory names contain explicit references to the parameter values so it would be easier to trace which parameter set generated which results.
- **Error Handling**: While the AI provided a solid structure, it usually produce faults. For example, when loading files from the directories, I added checks to ensure that images and videos are saved in different directory.
- **Evaluation Logic Modifications**: The AI's initial evaluation logic is not suitable with the current situation. She ignores that we don't have reference audios. But with adjustments, I tried to apply the appropriate algorithm which matched the current experiment's configuration and avoided mixing them across different parameter sets.

3. What did you learn from this process about the limitations and strengths of AI-generated solutions?

Answer:

- **Strengths of AI-Generated Solutions:**

1. **Efficiency and Speed**: AI significantly sped up the process of code generation. Tasks like creating some basic script, and generating visualizations were accomplished quickly. It allowed me to focus more on refining and testing the logic rather than spending time on boilerplate code.
2. **Consistency and Organization**: AI helped in maintaining consistent code organization. It suggested modularizing the code into smaller functions, which made the code more organized and reusable. This structure also made it easier to debug and maintain. It really helps a lot!
3. **Insightful Suggestions**: AI generated useful suggestions for specific tasks especially some appropriate libraries and functions for plotting and visualizing results, like matplotlib and librosa.

- **Limitations of AI-Generated Solutions:**

1. **Lack of Context and Domain-Specific Understanding**: While AI can generate code based on patterns and existing knowledge, it lacks the ability to fully understand the specific context of the task or domain. For example, it didn't always provide intuitive logic for specialized tasks or handle complex edge cases in the experiments (especially when handling missing files or very specific file formats).
2. **Over-simplified (but seem to be complex) Solutions**: Sometimes, AI provided solutions that were overly simplistic or missed out on nuances. For instance, maybe it thinks a lot, but generate actually simple even nonsense solutions.
3. **Customization and Fine-Tuning**: While AI is good for providing foundational solutions, it often lacks the ability to tailor solutions to very specific user requirements or adapt to changes in the task at hand. In this case, I had to modify a lot of the generated code to fit the specific structure of my experiments, especially when dealing with unique folder structures.

Part E: Create your own Audio (Optional)

This section is optional, but completing it can earn you additional marks.

Task: Record your own audio clip with different sources, such as other people talking. Apply source separation techniques to separate your voice from the other sources. Successfully doing so will earn you extra credit.

The audio must be complex, with multiple sources (e.g. multiple people talking in the background with music or any other noise). Simple recordings with no or minimal background noise will not be accepted for this task. This is to experiment creatively while demonstrating your understanding of audio processing techniques.

Answer:

The results have been saved in q5/q5_results folder, which seperates four sources from a mixture of speches, music, and noise.