Department of Computer Science

Lab Manual

of

SPEECH PROCESSING AND RECOGNITION

MAI574

Class Name: V MScAIML

Master of Science Artifitial Intelligence and Machine Learning

2025-26

Prepared by: Sivesh Pb **Verified by: Faculty name:**

Department Overview

Department of Computer Science of CHRIST (Deemed to be University) strives to shape outstanding computer professionals with ethical and human values to reshape nation's destiny. The training imparted aims to prepare young minds for the challenging opportunities in the IT industry with a global awareness rooted in the Indian soil, nourished and supported by experts in the field.

Vision

The Department of Computer Science endeavours to imbibe the vision of the University "**Excellence and Service"**. The department is committed to this philosophy which pervades every aspect and functioning of the department.

Mission

"To develop IT professionals with ethical and human values". To accomplish our mission, the department encourages students to apply their acquired knowledge and skills towards professional achievements in their career. The department also moulds the students to be socially responsible and ethically sound.

Introduction to the Programme

Machines are gaining more intelligence to perform human like tasks. Artificial Intelligence has spanned across the world irrespective of domains. MSc (Artificial Intelligence and Machine Learning) will enable to capitalize this wide spectrum of opportunities to the candidates who aspire to master the skill sets with a research bent. The curriculum supports the students to obtain adequate knowledge in the theory of artificial intelligence with hands-on experience in relevant domains with tools and techniques to address the latest demands from the industry. Also, c andidates gain exposure to research models and industry standard application development in specialized domains through guest lectures, seminars, industry offered electives, projects, internships, etc.

Programme Objective

● To acquire in-depth understanding of the theoretical concepts in Artificial Intelligence and Machine Learning
● To gain practical experience in programming tools for Data Engineering, Knowledge Representation, Artificial intelligence, Machine learning, Natural Language Processing and Computer Vision.
● To strengthen the research and development of intelligent applications skills through specialization based real time projects.
● To imbibe quality research and develop solutions to the social issues.

Programme Outcomes:
PO1 : Conduct investigation and develop innovative solutions for real world problems in industry and research establishments related to Artificial Intelligence and Machine Learning
PO2 : Apply programming principles and practices for developing automation solutions to meet future business and society needs.

PO3 : Ability to use or develop the right tools to develop high end intelligent systems
PO4 : Adopt professional and ethical practices in Artificial Intelligence application development
PO5 : Understand the importance and the judicious use of technology for the sustainability of the environment.

MAI574– SPEECH PROCESSING AND RECOGNITION

Total Teaching Hours for Semester: 75 (3+4)
Max Marks:150 **Credits: 5**

Course Objectives

This course enables the learners to understand fundamentals of speech recognition, speech production and representation. It also enables the learners to impart knowledge on automatic speech recognition and pattern comparison techniques. This course helps the learners to develop automatic speech recognition model for different applications.

Course Outcomes

After successful completion of this course students will be able to

CO1: Understand the speech signals and represent the signal in time and frequency domain.
CO2: Analyze different signal processing and speech recognition methods.
CO3: Implement pattern comparison techniques and Hidden Markov Models (HMM)
CO4: Develop speech recognition system for real time problems.

Unit-1 **Teaching Hours: 15**

FUNDAMENTALS OF SPEECH RECOGNITION

Introduction- The Paradigm for Speech Recognition- Brief History of speech recognition research- The Speech Signal: The process of speech production and perception in human beings- the speech production system- representing speech in time and frequency domain-speech sounds and features.

Lab Programs:

1. Implement the task that takes in the audio as input and converts it to text.
2. Apply Fourier transform and calculate a frequency spectrum for a signal in the time domain.

Unit-2 **Teaching Hours: 15**

2.1 APPROACHES TO AUTOMATIC SPEECH RECOGNITION BY MACHINE:

The acoustic phonetic approach-The pattern recognition approach-The artificial intelligence approach.

2.2 SIGNAL PROCESSING AND ANALYSIS METHODS FOR SPEECH RECOGNITION:

Introduction- spectral analysis models- the bank of filters front end processor- linear predictive coding model for speech recognition- vector quantization.

Lab Programs:

3. Implement sampling and quantization techniques for the given speech signals.

4. Explore linear predictive coding model for speech recognition

Unit-3 **Teaching Hours: 15**

PATTERN COMPARISON TECHNIQUES

Speech detection- distortion measure- Mathematical consideration- Distortion measure – Perceptual consideration- Spectral Distortion Measure- Incorporation of spectral dynamic feature into distortion measure- Time alignment and normalization.

Lab Programs:

0. Demonstrate different pattern in the given speech signal

0. Implement time alignment and normalization techniques

Unit-4 **Teaching hours: 15**

THEORY AND IMPLEMENTATION OF HIDDEN MARKOV MODELS:

Introduction- Discrete time Markov processes- Extension to hidden Markov Models- Coin -toss models- The urn and ball model- Elements of a Hidden Markov Model- HMM generator of observation- The three basic problems for HMM's- The Viterbi algorithm- Implementation issues for HMM's.

Lab Programs:

7. Implement simple hidden Markov Model for a particular application.

8.Apply Viterbi dynamic programming algorithm to find the most likely sequence of hidden states.

Unit-5 **Teaching Hours: 15**

TASK ORIENTED APPLICATION OF AUTOMATIC SPEECH RECOGNITION:

Task specific voice control and dialog- Characteristics of speech recognition applications- Methods of handling recognition error- Broad classes of speech recognition applications- Command and control applications- Voice repertory dialer- Automated call-type recognition- Call distribution by voice commands- Directory listing retrieval- Credit card sales validation.

Lab Programs:

9. Demonstrate automatic speech recognition for Call distribution by voice commands.

10. Apply speech recognition system to access telephone directory information from spoken spelled names (Directory listing retrieval).

Text Books and Reference Books

[1] Fundamentals of Speech Recognition, Lawrence R Rabiner and Biing- Hwang Juang. Prentice-Hall Publications, 20209.

[2] Introduction to Digital Speech Processing, Lawrence R. Rabiner, Ronald W. Schafer, Now Publishers, 2015.

Essential Reading / Recommended Reading

[1] Intelligent Speech Signal Processing, Nilanjan Dey, Academic Press, 2019.

[2] Speech Recognition-The Ultimate Step-By-Step Guide, Gerardus Blokdyk, 5STARCooks,2021.

[3] Automatic Speech Recognition- A Deep Learning Approach, Dong Yu, Li Deng, Springer-Verlag London, 2015.

Web Resources:

1. www.w3cschools.com
2. https://www.simplilearn.com/tutorials/python-tutorial/speech-recognition-in-python
3. https://realpython.com/python-speech-recognition/
4. https://cloud.google.com/speech-to-text/docs/tutorials
5. https://www.coursera.org/courses?query=speech%20recognition
6. https://pylessons.com/speech-recognition

CO – PO Mapping

	PO1	PO2	PO3	PO4	PO5
CO1	3	1			1
CO2	2	2			1
CO3	1	3	1		2
CO4	1	1	3	1	3

LIST OF PROGRAMS

MCA 2023-2024

Sl. no	Title of lab Experiment	Page number	RBT	CO
1	Sampling and Reconstruction of Speech Signals	7	L3	CO1, 3
2			L3	CO2, 3
3			L3	CO3
4			L3	CO3
5			L3	CO3
6			L3	CO3, 4
7			L4	CO3
8			L3	CO3
9			L3	CO4
10			L5	CO4

Evaluation Rubrics:

(1) Implementation: 5 marks.

(2) Complexity and Validation: 3 marks.

(3) Documentation & Writing the inference: 2 marks.

Submission Guidelines:

- Make a copy of the lab manual template with your <name_reg:no_subject name >,
- Copy the given question and the answer (lab code) with results, followed by the conclusion of that lab. Title the lab as lab number.
- Keep updating your lab manual and show the lab manual of that particular lab for evaluation.
- Create a Git Repository in your profile <SPR lab-reg no> . Follow a different branch for each lab <Lab 1, Lab 2…>, and push the code to Git. The link should be provided in Google Classroom along with the PDF of the lab manual.
- Upload the PDF to Google Classroom before the deadline.

Lab 1

Lab Exercise I: Sampling and Reconstruction of Speech Signals

Aim

To study sampling and reconstruction of speech signals at different sampling rates, evaluate reconstruction using zero-order hold and linear interpolation, and implement the source-filter model to analyze the effect of filtering, sampling, and reconstruction on speech quality.

(1) Implement sampling and quantization techniques for the given speech signals.

(a) Plot the time domain representation of the original speech signal.

(b) Sample the speech signal at different sampling rates (e.g., 8kHz, 16kHz, and 44.1kHz).

(c) Plot sampled speech signal for each of these sampling rates.

(d) Using the sampled signals from above, reconstruct the signal using:

(i) Zero-order hold (nearest-neighbor interpolation)

(ii) Linear interpolation.

(e) Calculate the Mean Squared Error (MSE) between the original and the reconstructed signals for both methods.

Write an inference on how sampling rates affect the quality and accuracy of the reconstructed speech signal.

(2) Implement the source-filter model for a given speech signal and analyze the impact of sampling and reconstruction on the quality of the speech signal.

(a) Generate a synthetic speech signal using the source-filter model.

(i) Create a source signal (e.g., a glottal pulse train for voiced sounds or white noise for unvoiced sounds).

(ii) Apply a filter that models the vocal tract, represented by an all-pole filter or an FIR filter with formants (resonances of the vocal tract).

(b) Plot the generated speech signal and analyze the effect of the filter on the original source.

(c) Sample the speech signal generated above at different sampling rates (e.g., 8 kHz, 16 kHz, 44.1 kHz).

(d) Reconstruct the signal using a suitable interpolation method (e.g., zero-order hold, linear interpolation).

(e) Compute the Mean Squared Error (MSE) between the original and reconstructed speech signals.

Write an inference on tasks such as creating the source-filter model, different sampling rates, and reconstruction of the sampled signals.

Code with Results

1) Sampling and Quantization

The following code loads a speech signal from a `.wav` file, samples it at different rates, and reconstructs it using zero-order hold and linear interpolation.

```
import numpy as np

import matplotlib.pyplot as plt

from scipy.io import wavfile

from scipy.signal import spectrogram



# Load the speech signal

sampling_rate, audio_data = wavfile.read('speech.wav')
```

```
# Create a time vector

time = np.linspace(0, len(audio_data) / sampling_rate, len(audio_data))



# Plot the waveform

plt.figure(figsize=(12, 6))

plt.plot(time, audio_data)

plt.xlabel('Time (s)')

plt.ylabel('Amplitude')

plt.title('Speech Signal Waveform')

plt.grid(True)

plt.show()



# Compute and plot the spectrogram

frequencies, times, Sxx = spectrogram(audio_data, fs=sampling_rate)



plt.figure(figsize=(12, 6))

plt.pcolormesh(times, frequencies, 10 * np.log10(Sxx))

plt.ylabel('Frequency (Hz)')

plt.xlabel('Time (s)')
plt.title('Speech Signal Spectrogram')

plt.colorbar(label='Intensity (dB)')

plt.show()



# Define downsampling rates

downsampling_rate_1 = 2 # Half of the original sampling rate

downsampling_rate_2 = 4 # Quarter of the original sampling rate



# Function to implement zero-order hold reconstruction

def zero_order_hold(signal, original_len):

    reconstructed_signal = np.zeros(original_len)

    step = original_len // len(signal)

    for i in range(len(signal)):

        start = i * step

        end = min((i + 1) * step, original_len)

        reconstructed_signal[start:end] = signal[i]

    return reconstructed_signal



# Function to implement linear interpolation reconstruction

def linear_interpolation(signal, original_len):
    original_indices = np.arange(len(signal))

    new_indices = np.linspace(0, len(signal) - 1, original_len)

    reconstructed_signal = np.interp(new_indices, original_indices, signal)

    return reconstructed_signal


# Downsample and reconstruct for the first rate

downsampled_audio_1 = audio_data[::downsampling_rate_1]

zoh_reconstructed_1 = zero_order_hold(downsampled_audio_1, len(audio_data))

linear_reconstructed_1 = linear_interpolation(downsampled_audio_1, len(audio_data))


# Downsample and reconstruct for the second rate

downsampled_audio_2 = audio_data[::downsampling_rate_2]

zoh_reconstructed_2 = zero_order_hold(downsampled_audio_2, len(audio_data))

linear_reconstructed_2 = linear_interpolation(downsampled_audio_2, len(audio_data))


# Calculate Mean Squared Error

mse_zoh_1 = np.mean((audio_data - zoh_reconstructed_1)**2)

mse_linear_1 = np.mean((audio_data - linear_reconstructed_1)**2)

mse_zoh_2 = np.mean((audio_data - zoh_reconstructed_2)**2)

mse_linear_2 = np.mean((audio_data - linear_reconstructed_2)**2)
```

```
print(f"MSE for Zero-Order Hold (Rate 2): {mse_zoh_1}")

print(f"MSE for Linear Interpolation (Rate 2): {mse_linear_1}")

print(f"MSE for Zero-Order Hold (Rate 4): {mse_zoh_2}")

print(f"MSE for Linear Interpolation (Rate 4): {mse_linear_2}")




# Plot the original and reconstructed signals

plt.figure(figsize=(15, 10))



plt.subplot(5, 1, 1)

plt.plot(time, audio_data)

plt.title('Original Signal')

plt.ylabel('Amplitude')

plt.grid(True)



plt.subplot(5, 1, 2)

plt.plot(time, zoh_reconstructed_1)

plt.title(f'Zero-Order Hold Reconstruction (Rate {downsampling_rate_1})')

plt.ylabel('Amplitude')
plt.grid(True)



plt.subplot(5, 1, 3)

plt.plot(time, linear_reconstructed_1)

plt.title(f'Linear Interpolation Reconstruction (Rate {downsampling_rate_1})')

plt.ylabel('Amplitude')

plt.grid(True)



plt.subplot(5, 1, 4)

plt.plot(time, zoh_reconstructed_2)

plt.title(f'Zero-Order Hold Reconstruction (Rate {downsampling_rate_2})')

plt.ylabel('Amplitude')

plt.grid(True)



plt.subplot(5, 1, 5)

plt.plot(time, linear_reconstructed_2)

plt.title(f'Linear Interpolation Reconstruction (Rate {downsampling_rate_2})')

plt.xlabel('Time (s)')

plt.ylabel('Amplitude')

plt.grid(True)
plt.tight_layout()

plt.show()
```

—---

Results:

The Mean Squared Error (MSE) values for the different reconstruction methods and sampling rates are:

- **MSE for Zero-Order Hold (Rate 2): 15091657.31**
- **MSE for Linear Interpolation (Rate 2): 646943.95**
- **MSE for Zero-Order Hold (Rate 4): 17685475.68**
- **MSE for Linear Interpolation (Rate 4): 1070193.91**

The plots generated show the original speech signal and the signals reconstructed using both zero-order hold and linear interpolation at two different downsampling rates.

—---

(2) Source-Filter Model

The following code implements a basic source-filter model to generate a synthetic speech signal, and then samples and reconstructs this signal.

```
# Generate a glottal pulse train as the source signal

def generate_glottal_pulse(duration, fs, f0):

    t = np.arange(0, duration, 1/fs)

    pulse_train = np.zeros_like(t)

    period = int(fs / f0)
    for i in range(0, len(t), period):

        pulse_train[i] = 1

    return t, pulse_train



duration = 1.0 # seconds

fs = 16000 # sampling frequency

f0 = 150 # fundamental frequency

t, source_signal = generate_glottal_pulse(duration, fs, f0)



plt.figure(figsize=(12, 4))

plt.plot(t, source_signal)

plt.title('Glottal Pulse Train (Source Signal)')

plt.xlabel('Time (s)')

plt.ylabel('Amplitude')

plt.xlim(0, 0.1)

plt.grid(True)

plt.show()



# Create a vocal tract filter (all-pole filter)

from scipy.signal import lfilter, freqz
```

```
# Formant frequencies and bandwidths for a vowel-like sound

formants = [(800, 80), (1200, 100), (2500, 150)] # (frequency, bandwidth) in Hz

a = np.array([1])

for f, bw in formants:

    r = np.exp(-np.pi * bw / fs)

    theta = 2 * np.pi * f / fs

    a_k = np.array([1, -2 * r * np.cos(theta), r**2])

    a = np.convolve(a, a_k)


# Apply the filter to the source signal

synthesized_speech = lfilter([1], a, source_signal)



# Plot the filter's frequency response

w, h = freqz([1], a, worN=8000)

plt.figure(figsize=(12, 6))

plt.plot(0.5 * fs * w / np.pi, 20 * np.log10(abs(h)))

plt.title('Vocal Tract Filter Frequency Response')

plt.xlabel('Frequency (Hz)')

plt.ylabel('Gain (dB)')
plt.grid(True)

plt.show()



# Plot the synthesized speech

plt.figure(figsize=(12, 4))

plt.plot(t, synthesized_speech)

plt.title('Synthesized Speech Signal')

plt.xlabel('Time (s)')

plt.ylabel('Amplitude')

plt.grid(True)

plt.show()



# Define downsampling rates

downsampling_rate_1 = 2

downsampling_rate_2 = 4



# Downsample and reconstruct for the first rate

downsampled_synthesized_1 = synthesized_speech[::downsampling_rate_1]

zoh_reconstructed_synthesized_1 = zero_order_hold(downsampled_synthesized_1,
len(synthesized_speech))
```

```
linear_reconstructed_synthesized_1 = linear_interpolation(downsampled_synthesized_1, len(synthesized_speech))


# Downsample and reconstruct for the second rate

downsampled_synthesized_2 = synthesized_speech[::downsampling_rate_2]

zoh_reconstructed_synthesized_2 = zero_order_hold(downsampled_synthesized_2, len(synthesized_speech))

linear_reconstructed_synthesized_2 = linear_interpolation(downsampled_synthesized_2, len(synthesized_speech))


# Calculate Mean Squared Error

mse_zoh_synthesized_1 = np.mean((synthesized_speech - zoh_reconstructed_synthesized_1)**2)

mse_linear_synthesized_1 = np.mean((synthesized_speech - linear_reconstructed_synthesized_1)**2)

mse_zoh_synthesized_2 = np.mean((synthesized_speech - zoh_reconstructed_synthesized_2)**2)

mse_linear_synthesized_2 = np.mean((synthesized_speech - linear_reconstructed_synthesized_2)**2)


print(f"MSE for Synthesized ZOH (Rate {downsampling_rate_1}): {mse_zoh_synthesized_1}")
print(f"MSE for Synthesized Linear (Rate {downsampling_rate_1}): {mse_linear_synthesized_1}")

print(f"MSE for Synthesized ZOH (Rate {downsampling_rate_2}): {mse_zoh_synthesized_2}")

print(f"MSE for Synthesized Linear (Rate {downsampling_rate_2}): {mse_linear_synthesized_2}")


# Plot the original and reconstructed synthesized speech signals

plt.figure(figsize=(15, 10))


plt.subplot(5, 1, 1)

plt.plot(t, synthesized_speech)

plt.title('Original Synthesized Signal')

plt.ylabel('Amplitude')

plt.grid(True)


plt.subplot(5, 1, 2)

plt.plot(t, zoh_reconstructed_synthesized_1)

plt.title(f'Synthesized ZOH Reconstruction (Rate {downsampling_rate_1})')

plt.ylabel('Amplitude')

plt.grid(True)
plt.subplot(5, 1, 3)

plt.plot(t, linear_reconstructed_synthesized_1)

plt.title(f'Synthesized Linear Reconstruction (Rate {downsampling_rate_1})')

plt.ylabel('Amplitude')

plt.grid(True)



plt.subplot(5, 1, 4)

plt.plot(t, zoh_reconstructed_synthesized_2)

plt.title(f'Synthesized ZOH Reconstruction (Rate {downsampling_rate_2})')

plt.ylabel('Amplitude')

plt.grid(True)



plt.subplot(5, 1, 5)

plt.plot(t, linear_reconstructed_synthesized_2)

plt.title(f'Synthesized Linear Reconstruction (Rate {downsampling_rate_2})')

plt.xlabel('Time (s)')

plt.ylabel('Amplitude')

plt.grid(True)
```

plt.tight_layout()

plt.show()

—--

Results:

The MSE values for the reconstructed synthetic speech signals are:

- **MSE for Synthesized ZOH (Rate 2): 0.00030**
- **MSE for Synthesized Linear (Rate 2): 0.00002**
- **MSE for Synthesized ZOH (Rate 4): 0.00037**
- **MSE for Synthesized Linear (Rate 4): 0.00004**

Plots were generated showing the glottal pulse train, the frequency response of the vocal tract filter, the original synthesized speech signal, and the reconstructed signals at different downsampling rates.

Conclusion /inference

Here are the key takeaways from the experiments:

- **Effect of Sampling Rate: Higher sampling rates (lower downsampling rates) result in a more accurate reconstruction of the speech signal, as indicated by the lower Mean Squared Error (MSE). This is because higher sampling rates capture more detail from the original signal, leading to less information loss.**
- **Reconstruction Methods: Linear interpolation consistently outperforms zero-order hold in terms of reconstruction accuracy, resulting in a significantly lower MSE. Zero-order hold creates a blocky, staircase-like signal, while linear interpolation provides a smoother and more faithful representation of the original signal.**
- **Source-Filter Model: The source-filter model successfully generated a synthetic speech signal. The glottal pulse train served as the excitation source, and the vocal tract filter shaped this source to create vowel-like sounds with clear formants.**
- **Combined Effects: The experiments with the source-filter model confirmed the findings from the natural speech signal. Lower downsampling rates and the use of linear interpolation for reconstruction led to a much lower MSE and a higher quality synthesized speech signal. The distortion of the signal was more pronounced at higher downsampling rates.**

In summary, this lab successfully demonstrated the principles of speech signal sampling and reconstruction. It highlighted the importance of a sufficiently high sampling rate and the superiority of linear interpolation over zero-order hold for signal reconstruction. The source-filter model provided a practical application of these concepts and showed how filtering, sampling, and reconstruction all play a role in the quality of synthesized speech.