ECE594B

Spectrograms

April 9, 2021

Student:	Ivan Arevalo
Perm Number:	5613567
Email:	ifa@ucsb.com

Department of Electrical and Computer Engineering, UCSB

0 Motivation

The main objective of this assignment is to study how different choices of window shape, window length, and overlap affect the resulting spectrogram. I have used the first 3 seconds of audio from the Female_1_16k.wav file in the speech data folder for my experiments.

1 Female speech experiment

Figure 1 shows a wideband spectrogram with 50% overlap of the rectangular, hann, hamming, and blackman windows.

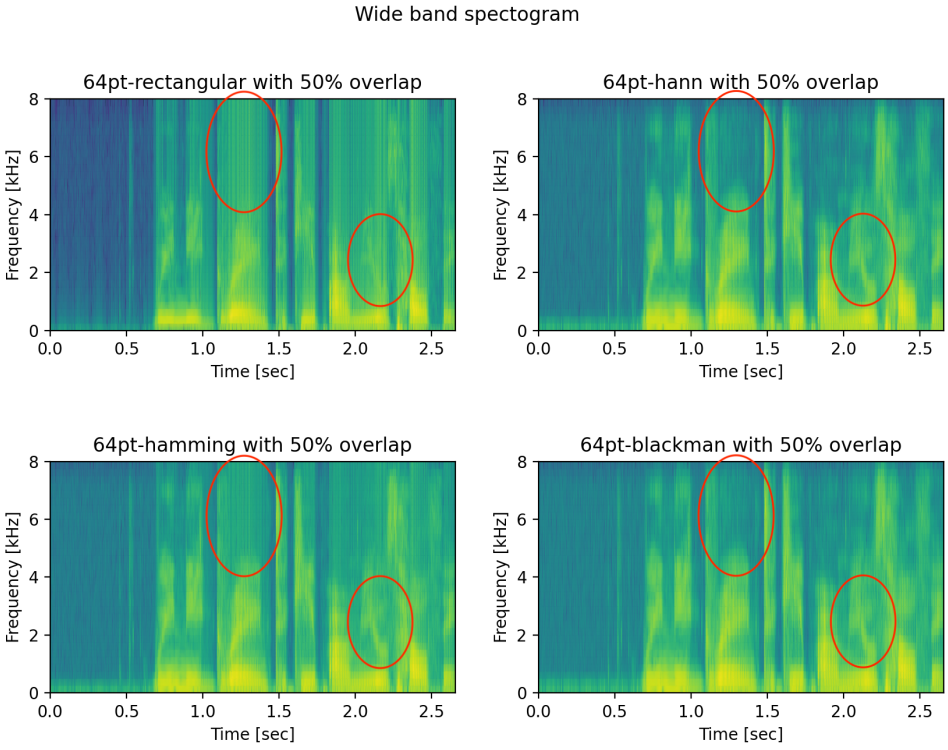

Figure 1: Wideband spectrogram for different window functions

We can observe in figure 1 that the rectangular window has higher frequencies present around the 1.3 second mark than the other windows. Given that the rectangular window does not mitigate discontinuities at the edges, its frequency response has higher sidelobes. This can be seen above by looking in the circled areas. The rectangular window seems to have a lot of noise compared to the hann and blackman windows which have crisper shapes in their spectrogram. In this experiment, hann seems to provide the best spectrogram.

Figure 2 shows the same comparison as figure 1 but with a narrow band spectrogram.

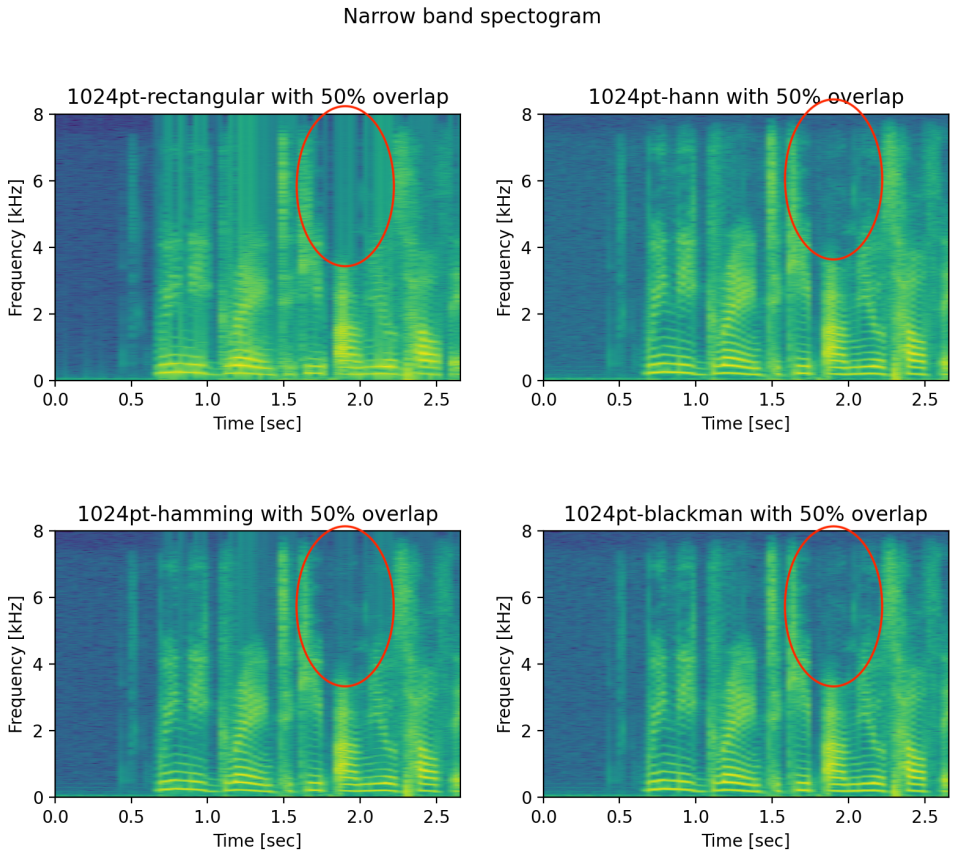

Figure 2: Narrowband spectrogram for different window functions

Similar to the wideband case in figure 1, the rectangular window seems to have high frequencies in areas where the other windows don't.

Given that hamming window is used so often, I will perform the rest the experiments on the hamming window and observe the how it affects the spectrogram.

Figure 3 compares the window length for the hamming window at 50% overlap

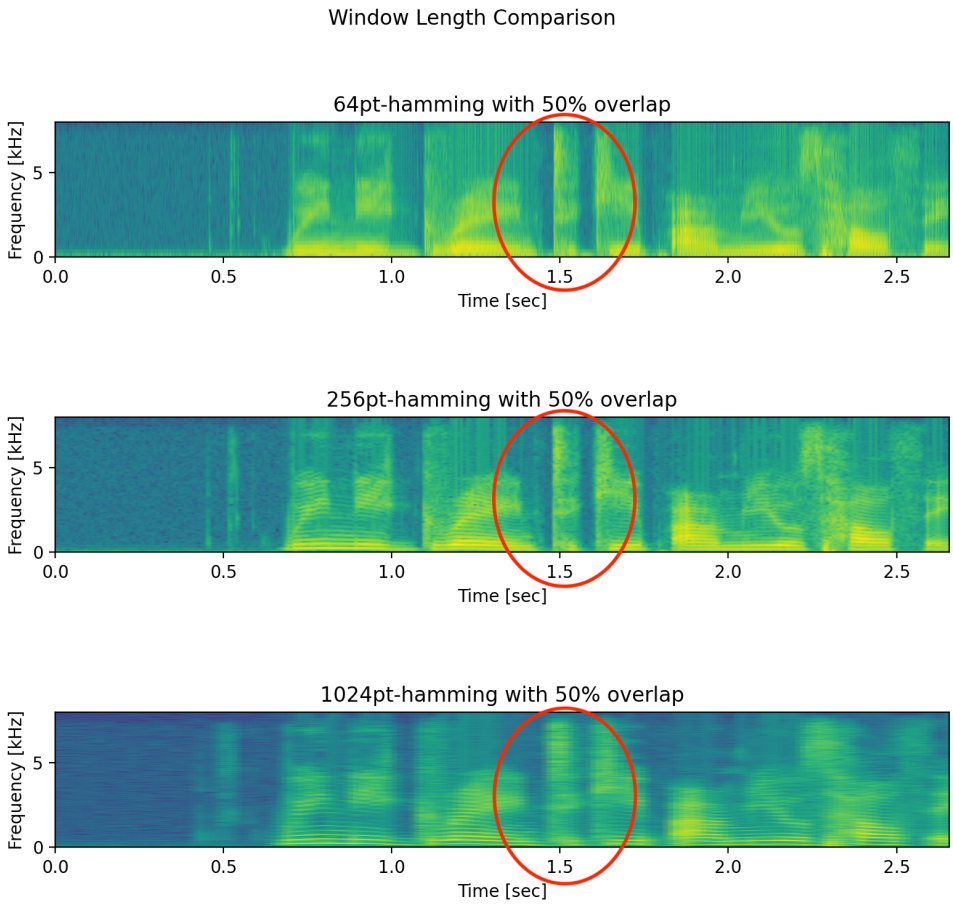

Figure 3: Window length comparison at 50% overlap

Comparing window lengths in figure 3, we can observe that narrow band spectrograms (1024 pts) have a higher resolution in the frequency-domain than wide band spectrograms (64 pts). This is expected given that the window looks at a bigger section of the original signal in order to determine its spectral distribution. Similarly, wideband spectrograms have a higher resoltuion in the time domain since the window function looks at a smaller section of the signal at time. This means that it can better track changes in the frequency domain.

Figure 4 compares window length spectrograms for the hamming window with no overlap.

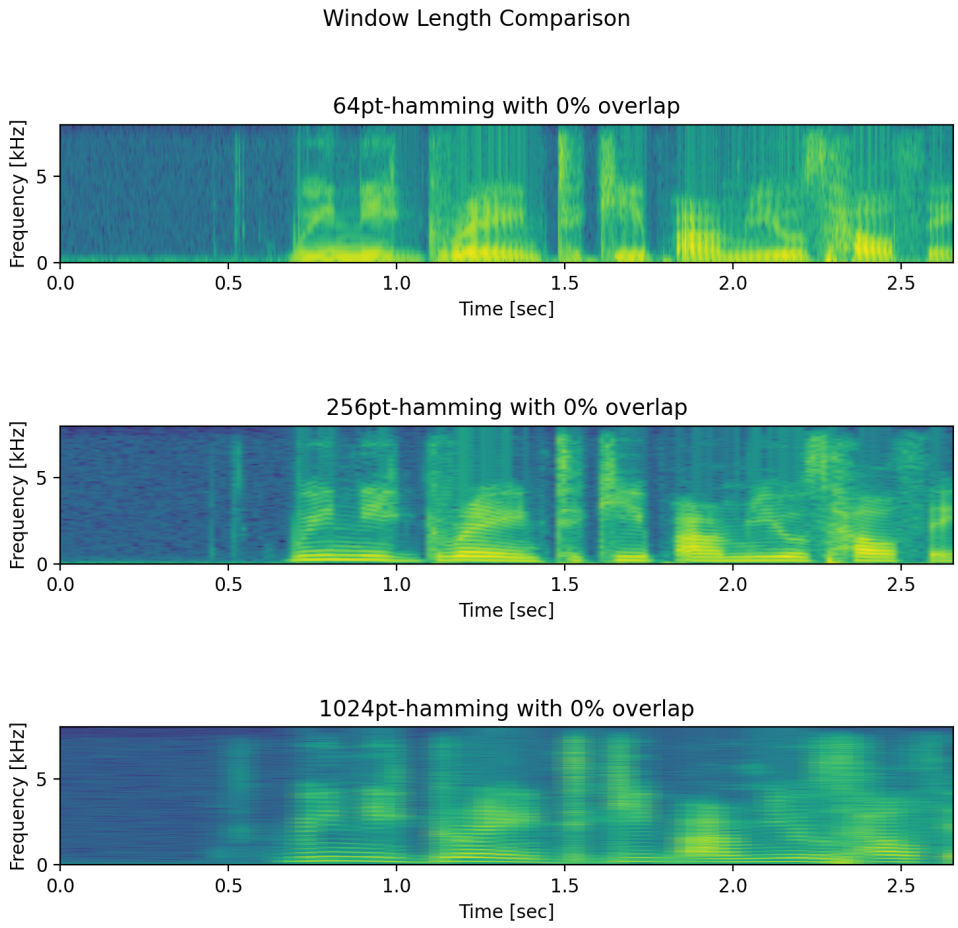

Figure 4: Window length comparison at 0% overlap

We can observe in figure 4 that the wideband spectrogram is able to track the sudden changes in spectral density around the 1.5 second mark, while the narrowband spectrogram blurred together this section of the spectrogram.

Figure 5 compares the effect of overlap on the wideband spectrogram for the hamming window .

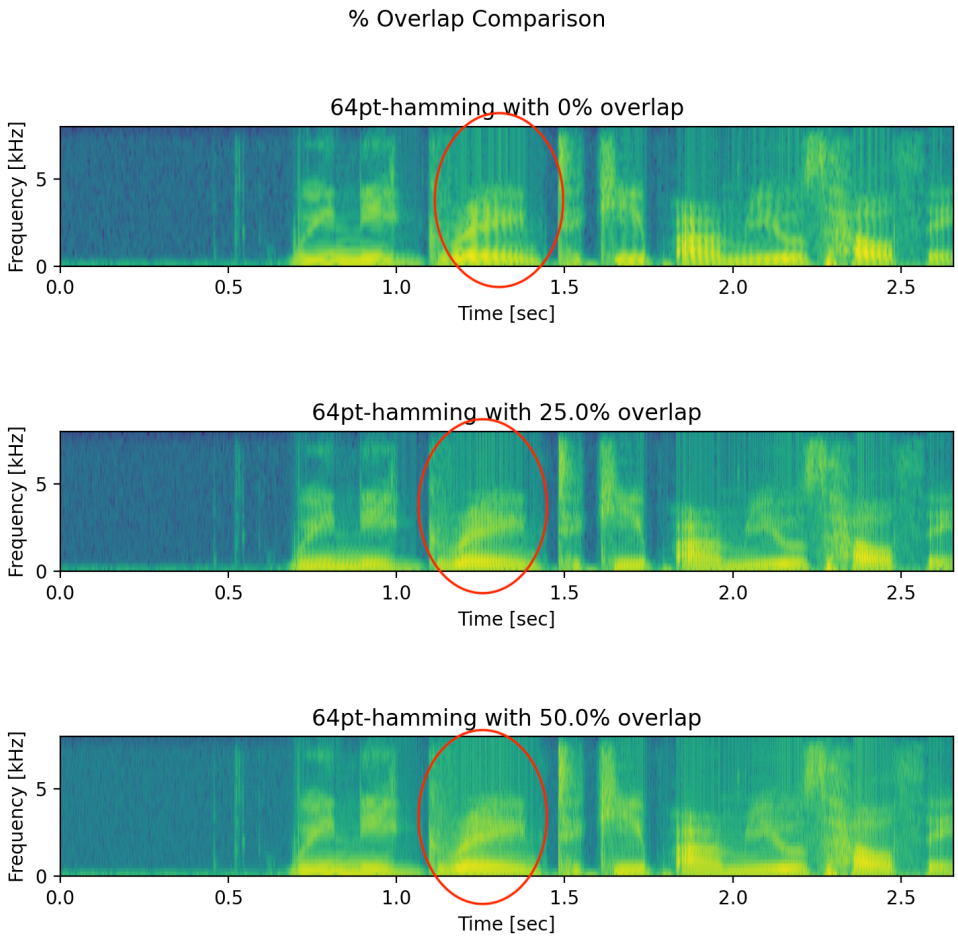

Figure 5: Overlap comparison for wideband spectrogram with hamming window

Figure 5 circles an area on each spectrogram where it is clear that overlap greatly influences the smoothness of the spectrogram. When we don't have any overlap, dips form around the edges of the windowed sections. This contributes to the choppy appearance of the 0% overlap spectrogram as oposed to the smoother looking spectrogram with 50% overlap.

Figure 6 compares overlaps for the narrow band spectrograms.

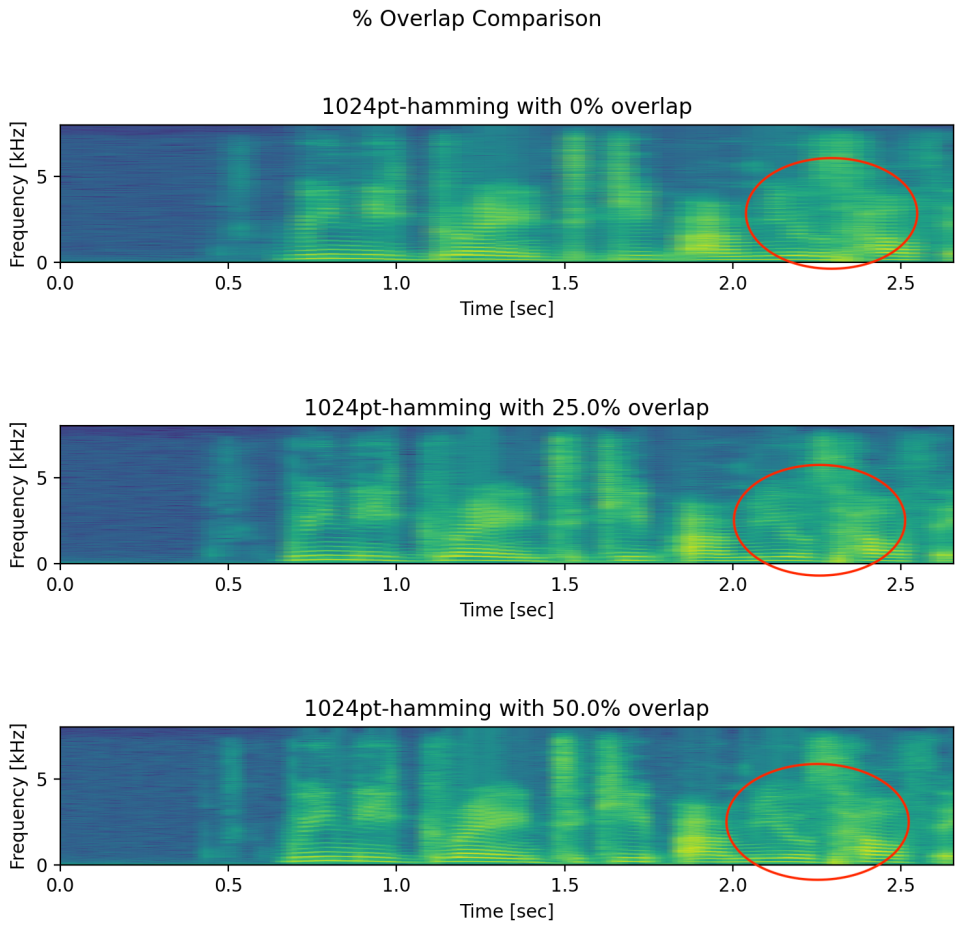

Figure 6: Overlap comparison for narrowband spectrogram with hamming window

It is harder to distinguish how overlap affects the narrow band spectrograms in figure 6. However, we can notice an area circled in red, where the spectral resolution greatly improves for the 50% percent overlap scenario.

2 Vowels experiment

I repeated the previous experiments with my own recording of all 5 vowels sounds. I found it easier to track the spectral peaks in this recording given the constant sounds. The blackman window yielded better spectrogram results for this recording.

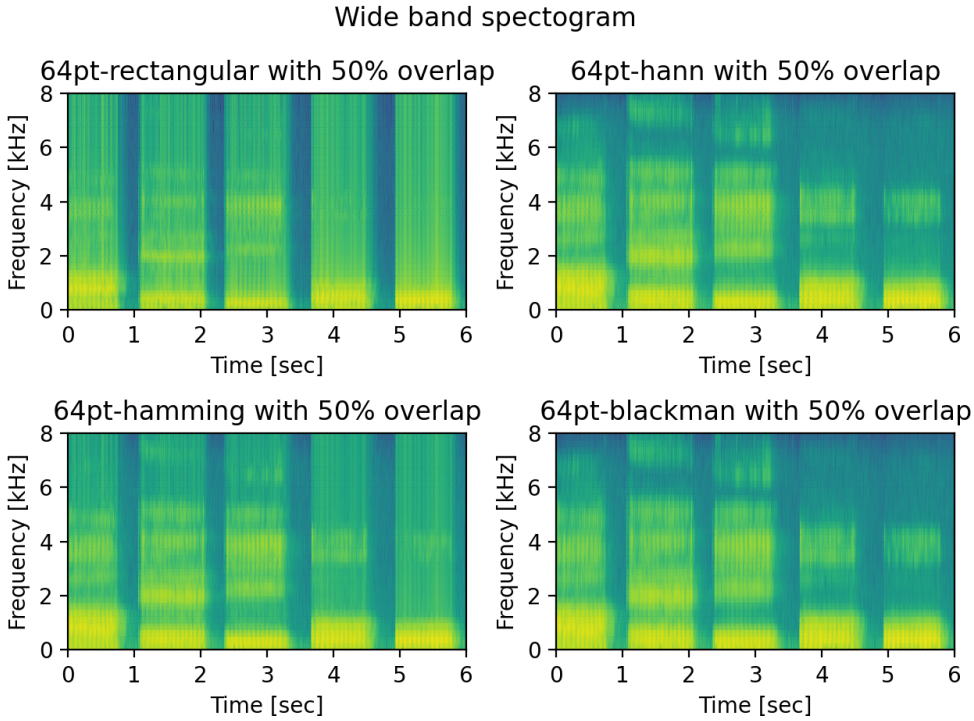

Figure 7: Wideband spectrogram for different window functions

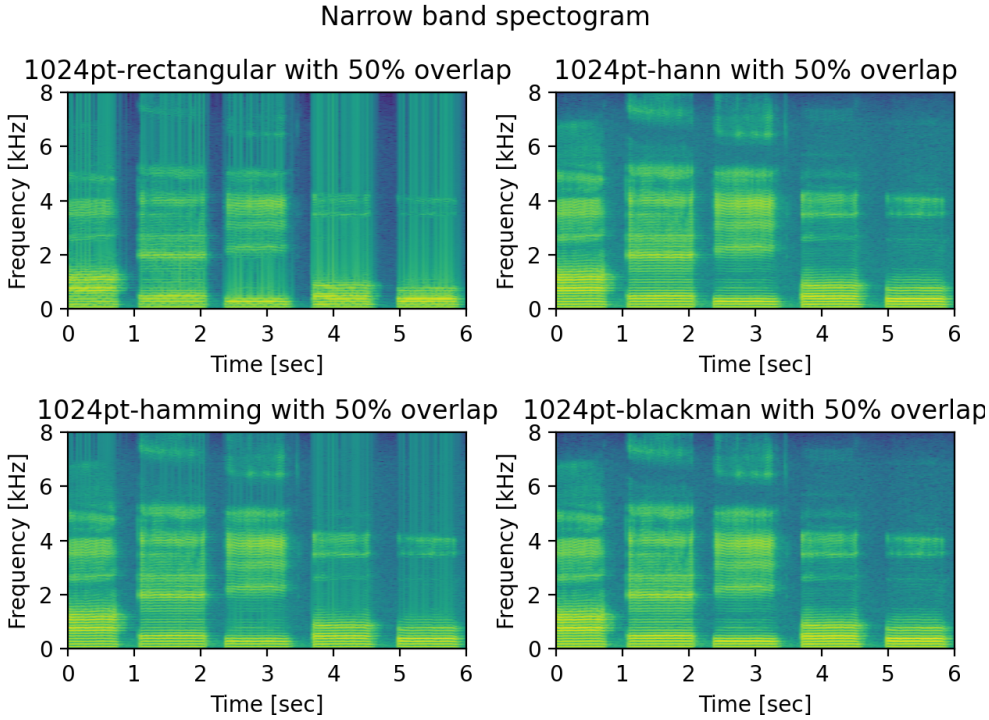

Figure 8: Narrowband spectrogram for different window functions

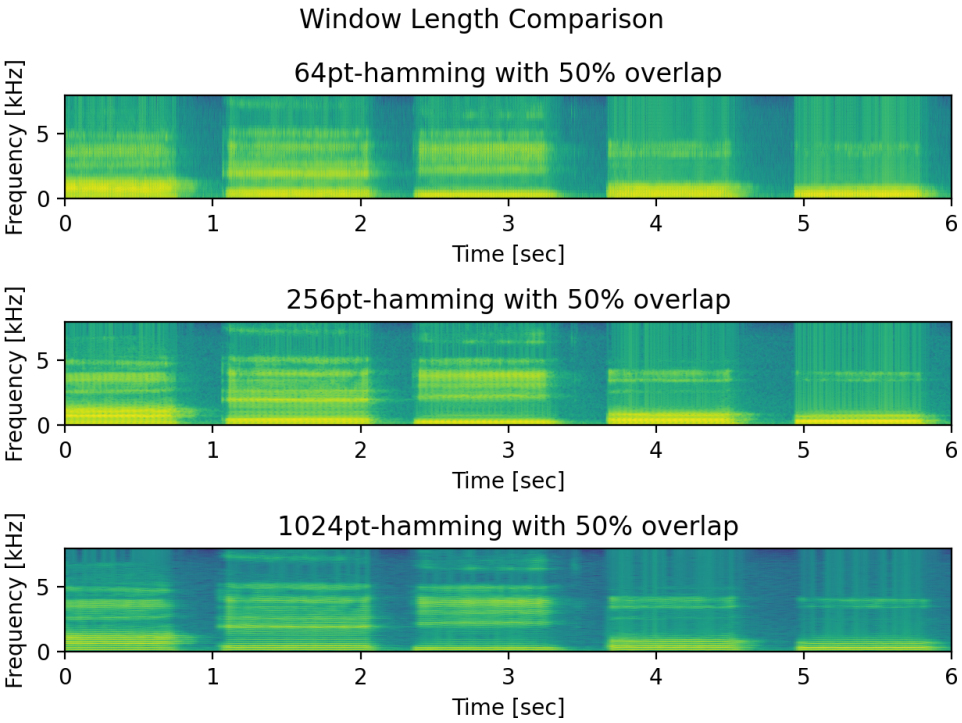

Figure 9: Window length comparison at 50% overlap

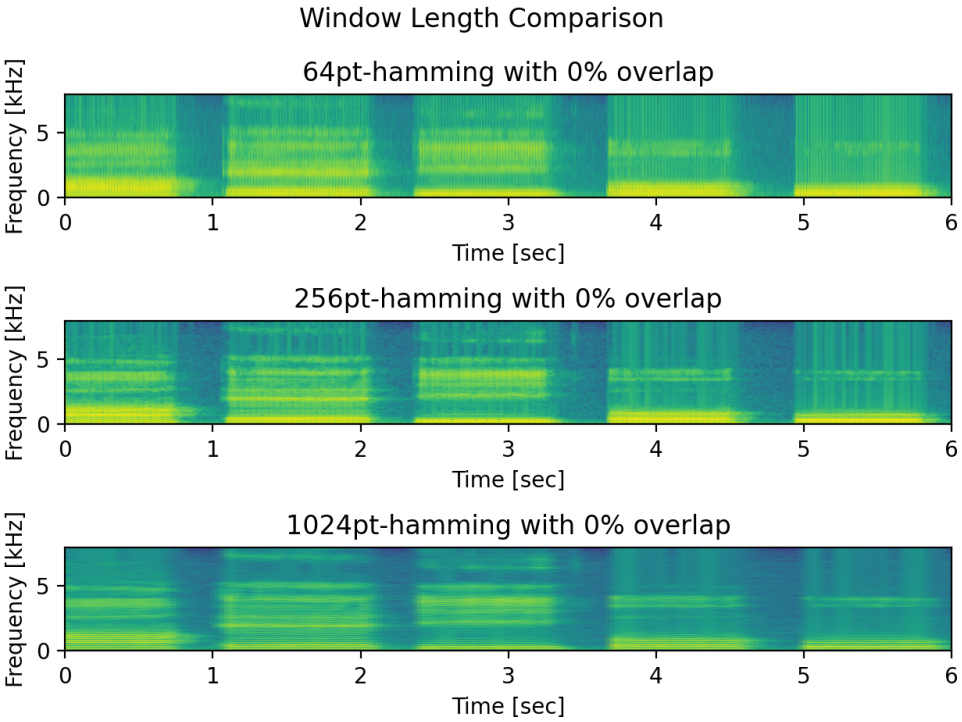

Figure 10: Window length comparison at 0% overlap

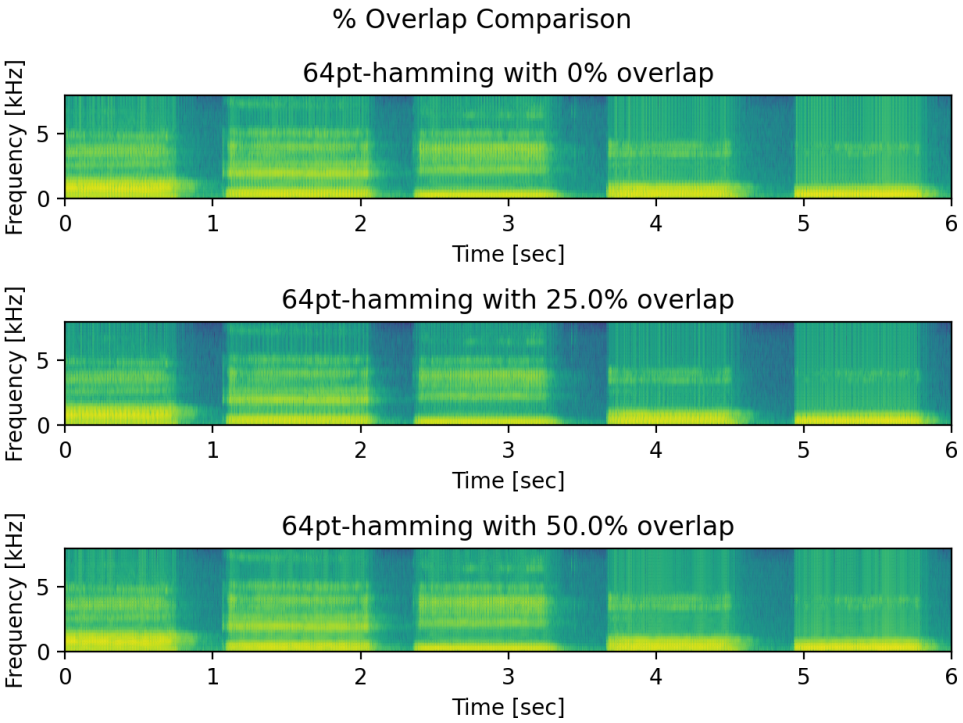

Figure 11: Overlap comparison for wideband spectrogram with hamming window

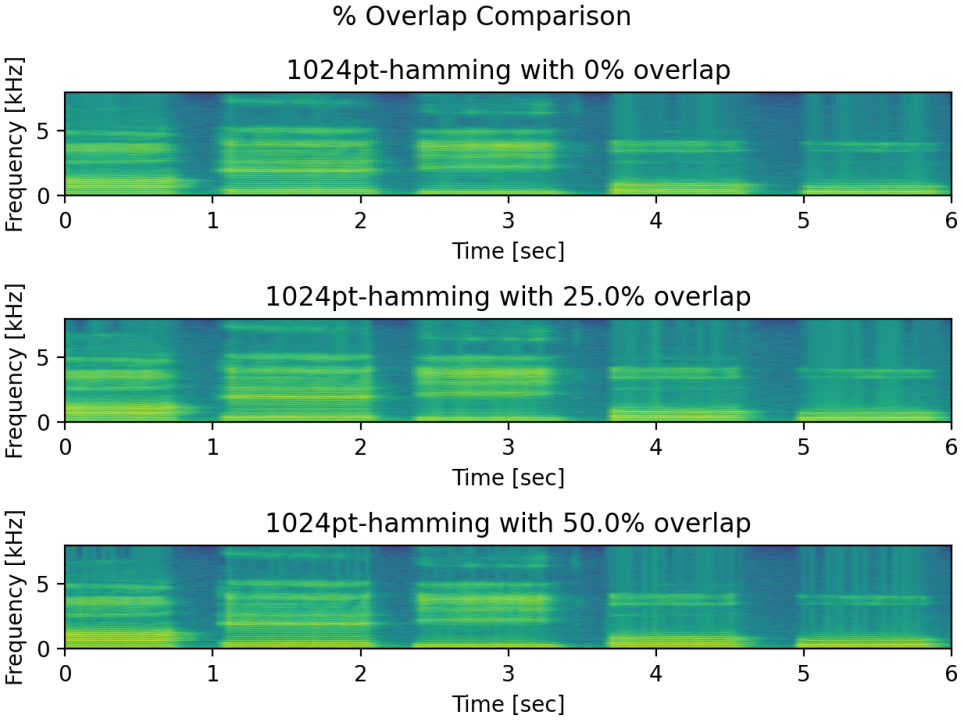

Figure 12: Overlap comparison for narrowband spectrogram with hamming window

3 Conclusion

In conclusion, it is clear that the window shape, length, and overlap greatly influences the quality of spectrograms that can be rendered. Rectangular windows proved to have noisy spectrograms with lower spectral resolution. Hann, hamming, and blackman windows seemed to have similar spectrograms under the same parameters. It would require deeper analysis on the specific application to determine which would work best. Percent overlap proved to be a very important parameter, specially when tracking the change in spectra is important. Finally, we got some intuition on how window length affects the spectrogram. Narrow band spectrograms gives us high frequency resolution while wideband spectrograms give us high time-domain resolution.

4 Code

```
import numpy as np
import matplotlib.pyplot as plt
import HW1.windows as windows
from scipy.io import wavfile
from scipy import signal
from matplotlib.colors import LogNorm

class Spectogram():
    def __init__(self, signal, fs):
        self.signal = signal
        self.fs = fs

    def generate_spectogram(self, window_type, win_size, overlap):

        win = windows.Window(win_size)
        window = np.zeros(win_size)
        if window_type == "hamming":
            window = win.hamming_window()
        elif window_type == "rectangular":
            window = win.rectangular_window()
        elif window_type == "triangular":
            window = win.triangular_window()
        elif window_type == "sine":
            window = win.sine_window()
        elif window_type == "hann":
            window = win.hann_window()
        elif window_type == "blackman":
            window = win.blackman_window()

        num_segments = int(1 + (len(self.signal)-win_size)/(win_size*(1-overlap)))
        spectogram = np.zeros((win_size//2 +1, num_segments))
        for i in range(num_segments):
            start_idx = int(win_size*(1-overlap)*i)
            end_idx = int(win_size*(1-overlap)*i + win_size)
            spectogram[:,i] = abs(np.fft.fft(self.signal[start_idx:end_idx]*window,
                win_size)[:win_size//2 +1])


        t = np.linspace(0, len(self.signal) / fs, spectogram.shape[1])
        f = np.linspace(0, int(fs / 2)/1000, spectogram.shape[0])


        return t, f, spectogram

    def plot(self, t, f, spectogram, fig=plt):
        print('inplot')
        fig.pcolormesh(t, f, spectogram, shading='gouraud', norm=LogNorm())
        fig.ylabel('Frequency [Hz]')
        fig.xlabel('Time [sec]')
        # plt.show()
```

```
if __name__ == '__main__':
    # fs, data = wavfile.read('./speech_data/P501_C_english_m2_IRS_08k.wav')
    fs, data = wavfile.read('./speech_data/Female_1_16k.wav')
    data = data[:len(data)//3] # Focusing on first third of signal

    windows_list = ['rectangular', 'hann', 'hamming', 'blackman']
    win_size = [64, 256, 1024]
    p_overlap = [0, 0.25, 0.5]


    spec = Spectogram(data, fs)


    # Experiment 1: compare all windows with 50% overlap for narrow and wide band
    fig, axs = plt.subplots(2,2)
    indices = [(0, 0), (0, 1), (1, 0), (1, 1)]
    for i, window in enumerate(windows_list):
        t, f, spectogram = spec.generate_spectogram(window, win_size[0], p_overlap[2])
        axs[indices[i]].pcolormesh(t, f, spectogram, shading='gouraud', norm=LogNorm())
        axs[indices[i]].set_ylabel('Frequency [kHz]')
        axs[indices[i]].set_xlabel('Time [sec]')
        axs[indices[i]].set_title(f"64pt-{window} with 50% overlap")
    plt.suptitle("Wide band spectogram")
    plt.tight_layout()
    plt.show()

    fig, axs = plt.subplots(2, 2)
    for i, window in enumerate(windows_list):
        t, f, spectogram = spec.generate_spectogram(window, win_size[2], p_overlap[2])
        axs[indices[i]].pcolormesh(t, f, spectogram, shading='gouraud', norm=LogNorm())
        axs[indices[i]].set_ylabel('Frequency [kHz]')
        axs[indices[i]].set_xlabel('Time [sec]')
        axs[indices[i]].set_title(f"1024pt-{window} with 50% overlap")
    plt.suptitle("Narrow band spectogram")
    plt.tight_layout()
    plt.show()

    # Experiment 2: compare different window sizes for the hamming window at 0% and 50%
        overlap.
    fig, axs = plt.subplots(3,1)
    indices = [(0, 0), (1,0), (2,0)]
    for i in range(len(win_size)):
        t, f, spectogram = spec.generate_spectogram(windows_list[2], win_size[i],
            p_overlap[0])
        axs[i].pcolormesh(t, f, spectogram, shading='gouraud', norm=LogNorm())
        axs[i].set_ylabel('Frequency [kHz]')
        axs[i].set_xlabel('Time [sec]')
        axs[i].set_title(f"{win_size[i]}pt-{windows_list[2]} with 0% overlap")
    plt.suptitle("Window Length Comparison")
    plt.tight_layout()
    plt.show()

    fig, axs = plt.subplots(3, 1)
```

```
    indices = [(0, 0), (1, 0), (2, 0)]
    for i in range(len(win_size)):
        t, f, spectogram = spec.generate_spectogram(windows_list[2], win_size[i],
            p_overlap[2])
        axs[i].pcolormesh(t, f, spectogram, shading='gouraud', norm=LogNorm())
        axs[i].set_ylabel('Frequency [kHz]')
        axs[i].set_xlabel('Time [sec]')
        axs[i].set_title(f"{win_size[i]}pt-{windows_list[2]} with 50% overlap")
    plt.suptitle("Window Length Comparison")
    plt.tight_layout()
    plt.show()

    # Experiment 3: compare different overlaps for the hamming window
    fig, axs = plt.subplots(3, 1)
    indices = [(0, 0), (1, 0), (2, 0)]
    for i in range(len(win_size)):
        t, f, spectogram = spec.generate_spectogram(windows_list[2], win_size[0],
            p_overlap[i])
        axs[i].pcolormesh(t, f, spectogram, shading='gouraud', norm=LogNorm())
        axs[i].set_ylabel('Frequency [kHz]')
        axs[i].set_xlabel('Time [sec]')
        axs[i].set_title(f"{win_size[0]}pt-{windows_list[2]} with {p_overlap[i]*100}%
            overlap")
    plt.suptitle("% Overlap Comparison")
    plt.tight_layout()
    plt.show()

    fig, axs = plt.subplots(3, 1)
    indices = [(0, 0), (1, 0), (2, 0)]
    for i in range(len(win_size)):
        t, f, spectogram = spec.generate_spectogram(windows_list[2], win_size[2],
            p_overlap[i])
        axs[i].pcolormesh(t, f, spectogram, shading='gouraud', norm=LogNorm())
        axs[i].set_ylabel('Frequency [kHz]')
        axs[i].set_xlabel('Time [sec]')
        axs[i].set_title(f"{win_size[2]}pt-{windows_list[2]} with {p_overlap[i]*100}%
            overlap")
    plt.suptitle("% Overlap Comparison")
    plt.tight_layout()
    plt.show()

    '''
    Alternative implementation

    spec = Spectogram(data, fs)
    t, f, spectogram = spec.generate_spectogram('hamming', win_size, p_overlap)
    spec.plot(t, f, spectogram)

    f, t, Sxx = signal.spectrogram(data, fs, window='hamming', nperseg=win_size,
        noverlap=int(win_size * p_overlap))
    fig = plt.figure()
    plt.pcolormesh(t, f, Sxx, shading='gouraud', norm=LogNorm())
    plt.ylabel('Frequency [Hz]')
    plt.xlabel('Time [sec]')
```

```
fig2 = plt.figure()
plt.specgram(data, Fs=fs, window=np.hamming(win_size), NFFT=win_size,
    noverlap=int(win_size * p_overlap))
plt.show()
'''
```