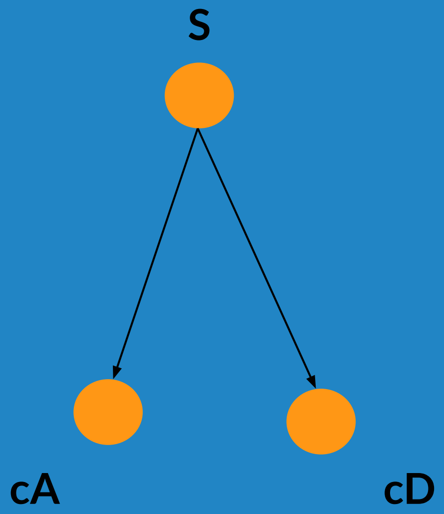
Discrete Wavelet Transform Analysis on Audio Signals using Python

1. Single-level Discrete Wavelet Transform



Single Level Wavelet Decomposition Tree

Coeffs = [cA, cD]

- **S** - Signal
- **cA** - Approximation Coefficients Vector
- **cD** - Detail Coefficients Vector

Support Libraries

- **PyWavelets**

An open source wavelet transform software for Python. It combines a simple high level interface with low level C and Cython performance.

https://pywavelets.readthedocs.io/en/latest/

- **scipy.io → wavfile**

Read and write audio data.

https://docs.scipy.org/doc/scipy/reference/generated/scipy.io.wavfile.read.html

- **MatplotLib**

Data visualization with Python.

https://matplotlib.org/stable/tutorials/introductory/pyplot.html

- **NumPy**

Performs a wide variety of mathematical operations on arrays.

https://numpy.org/doc/stable/user/quickstart.html

Sample Code

```
from scipy.io import wavfile
import matplotlib.pyplot as plt
import numpy as np
import pywt

samplerate, data = wavfile.read('sample1.wav');  # Reading the audio file
t = np.arange(len(data)) / float(samplerate);  # Getting Time
data = data/max(data);  # Normalize Audio Data

cA, cD = pywt.dwt(data, 'bior6.8', 'per');  # DWT

y = pywt.idwt(cA, cD, 'bior6.8', 'per')  # IDWT

wavfile.write('sampleR.wav', samplerate, y);  # writing y as Audio
wavfile.write('samplecD.wav', samplerate, cD);  # writing cD as Audio
```

Audio Files

https://drive.google.com/drive/folders/1ppfNKZDdFMEzvwdvp0qldSKW7pTCiqKy?usp=sharing

Original

Reconstructed Audio

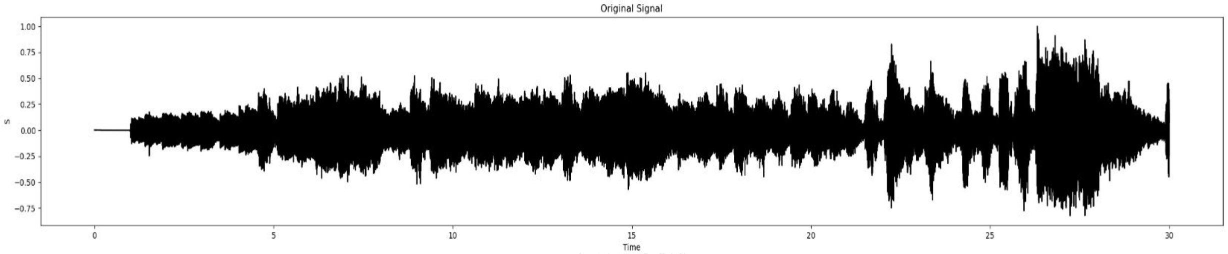
Detail Coeff. (cD) Audio

Detail Coeff. (cA) Audio

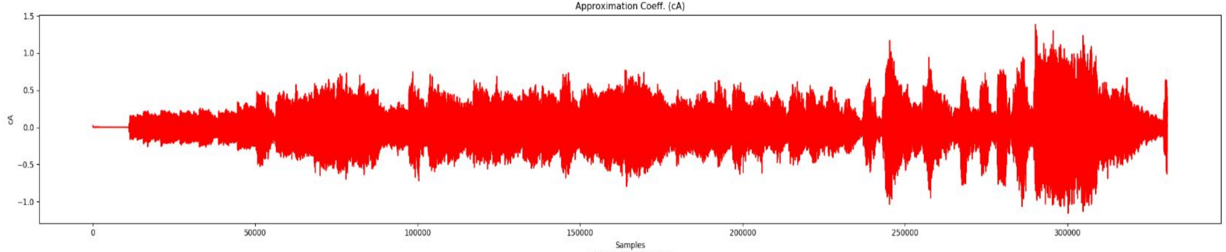
```
# Formatting for figure
L = len(data);
y = y[0:L];  # Matching length with input for plotting

plt.figure(figsize=(30, 20));

plt.subplot(4, 1, 1)
plt.plot(t, data, color='k');
plt.xlabel('Time');
plt.ylabel('S');
plt.title('Original Signal');

plt.subplot(4, 1, 2)
plt.plot(cA, color='r');
plt.xlabel('Samples');
plt.ylabel('cA');
plt.title('Approximation Coeff. (cA)');
```

```python
plt.subplot(4, 1, 3)
plt.plot(cD, color='g');
plt.xlabel('Samples');
plt.ylabel('cD');
plt.title('Detailed Coeff. (cD)');

plt.subplot(4, 1, 4)
plt.plot(t, y, color='b');
plt.xlabel('Time');
plt.ylabel('Value');
plt.title('Reconstructed Signal');

plt.savefig('plot.png', dpi=100)  # Saving plot as PNG image
plt.show()
```

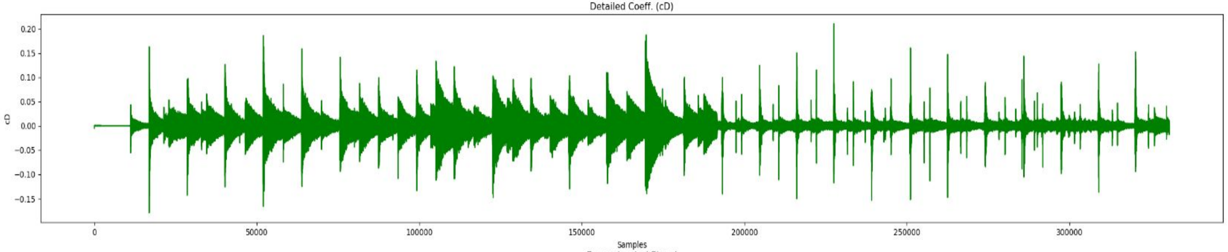
Original Signal



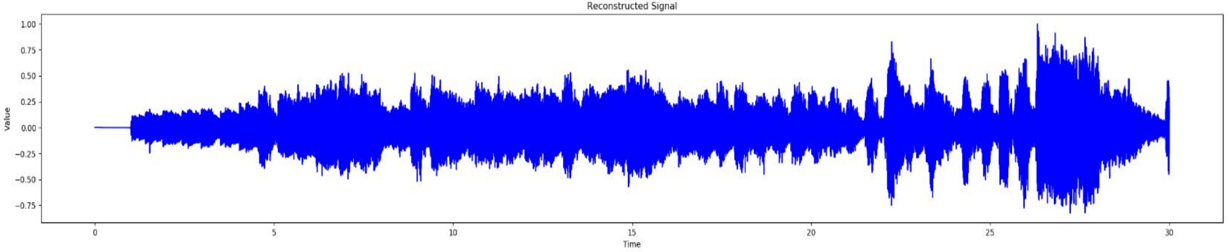
Approximation Coeff. (cA)



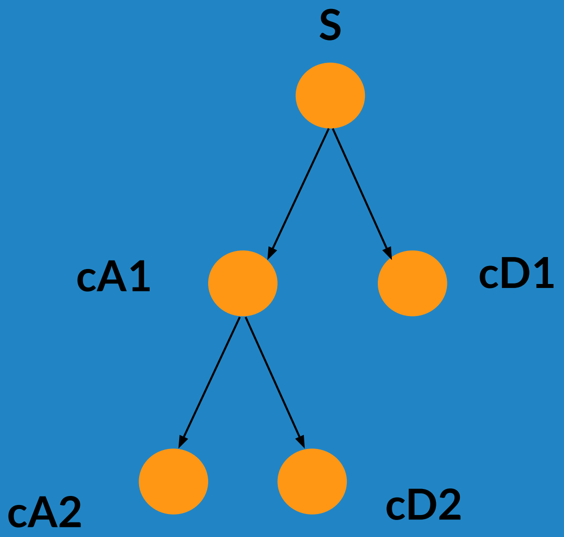
Detail Coeff. (cD)



Reconstructed Signal



2. Multi-level Discrete Wavelet Transform



2 Level Wavelet
Decomposition Tree

Coeffs = [cA2, cD2, cD1]

Sample Code

```
from scipy.io import wavfile
import matplotlib.pyplot as plt
import numpy as np
import pywt

samplerate, data = wavfile.read('sample.wav');  # Reading the audio file
t = np.arange(len(data)) / float(samplerate);  # Getting Time
data = data/max(data);  # Normalize Audio Data

coeffs = pywt.wavedec(data, 'bior6.8', mode='sym', level=2);  # DWT
cA2, cD2, cD1=coeffs

y = pywt.waverec(coeffs, 'bior6.8', mode='sym')  # IDWT

wavfile.write('sampleR.wav', samplerate, y);  # writing y as Audio
wavfile.write('samplecA2.wav', samplerate, cA2);  # writing cA2 as Audio
wavfile.write('samplecD2.wav', samplerate, cD2);  # writing cD2 as Audio
wavfile.write('samplecD1.wav', samplerate, cD1);  # writing cD1 as Audio
```

```python
# Formatting for figure
L = len(data);
y = y[0:L];  # Matching length with input for plotting

plt.figure(figsize=(30, 20));

plt.subplot(5, 1, 1)
plt.plot(t, data, color='k');
plt.xlabel('Time');
plt.ylabel('S');
plt.title('Original Signal');

plt.subplot(5, 1, 2)
plt.plot(cA2, color='r');
plt.xlabel('Samples');
plt.ylabel('cA2');
plt.title('Approximation Coeff. (cA2)');

plt.subplot(5, 1, 3)
plt.plot(cD2, color='g');
plt.xlabel('Samples');
plt.ylabel('cD2');
plt.title('Detailed Coeff. (cD2)');
```

```python
plt.subplot(5, 1, 4)
plt.plot(cD1, color='m');
plt.xlabel('Samples');
plt.ylabel('cD1');
plt.title('Detailed Coeff. (cD1)');

plt.subplot(5, 1, 5)
plt.plot(t, y, color='b');
plt.xlabel('Time');
plt.ylabel('Value');
plt.title('Reconstructed Signal');

plt.savefig('plot.png', dpi=100)  # Saving plot as PNG image
plt.show()
```

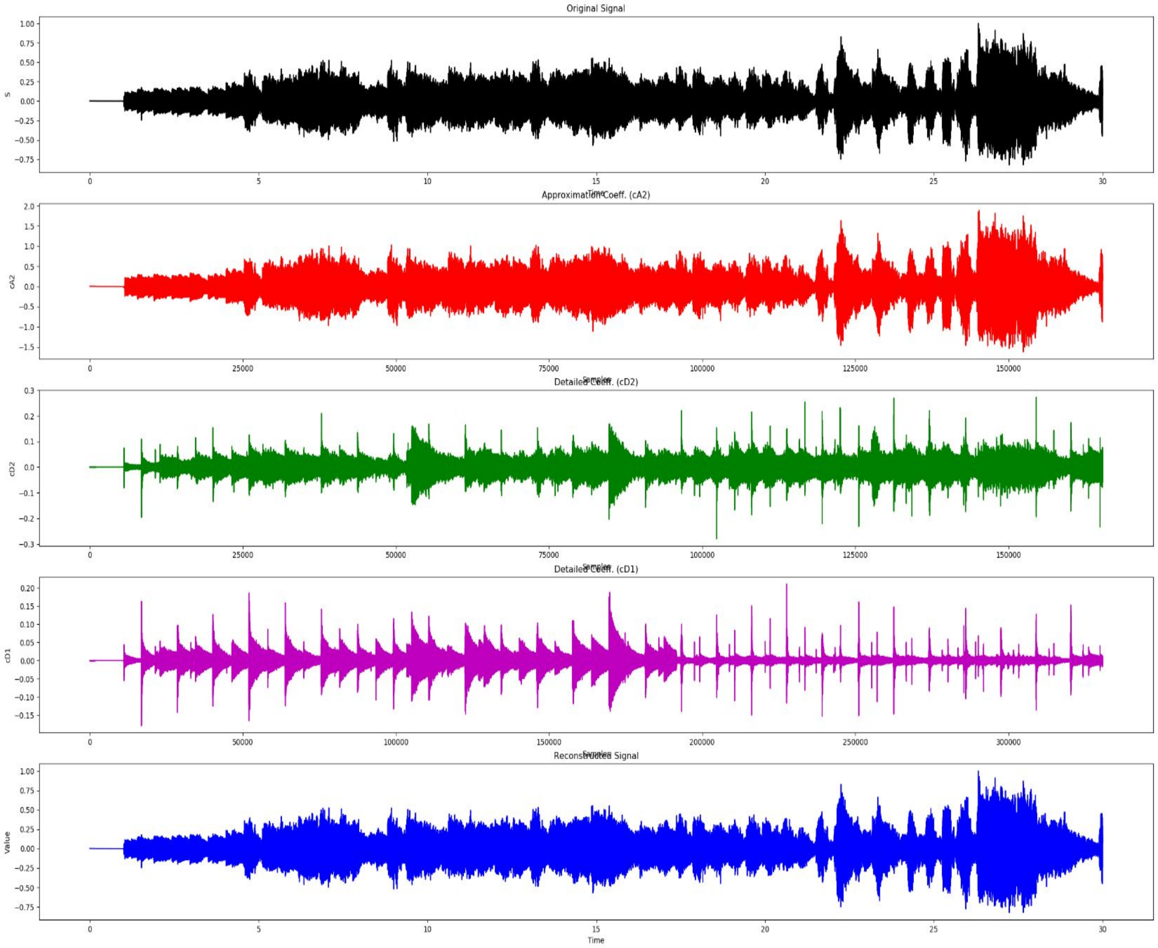
Original Signal

Approximation Coeff. (cA2)

Detail Coeff. (cD2)

Detail Coeff. (cD1)

Reconstructed Signal



Audio Files

https://drive.google.com/drive/folders/18t0iancm3JLJvBJsWCId8OOCvuE0gzJ-?usp=sharing

Original Audio

Reconstructed Audio

Detail Coeff. (cD2) Audio

Detail Coeff. (cA2) Audio

Detail Coeff. (cD1) Audio

Thanks!

Any questions?