

Clustering-Based Spectrum Sensing with OFDM Modulation for Cognitive Radio Networks



Presented by - Harsh Raj






Motivation

- The exponential growth of wireless devices has led to increasing spectrum scarcity.
- **Cognitive Radio Networks (CRNs)** provide a dynamic solution by allowing **secondary users** to access underutilized frequency bands.
- This project explores how **unsupervised machine learning** can enhance **OFDM-based spectrum sensing**, making it more adaptive and robust.




Historical Context

- **Traditional spectrum sensing techniques such as:**



-  **Energy Detection**
-  **Matched Filtering**
-  **Cyclostationary Detection**

have limitations under **noise uncertainty** and **unknown signal conditions**.

 **Clustering**, an unsupervised learning approach, bypasses the need for fixed thresholds, making it suitable for dynamic environments—aligning with the growing use of AI in wireless communications.



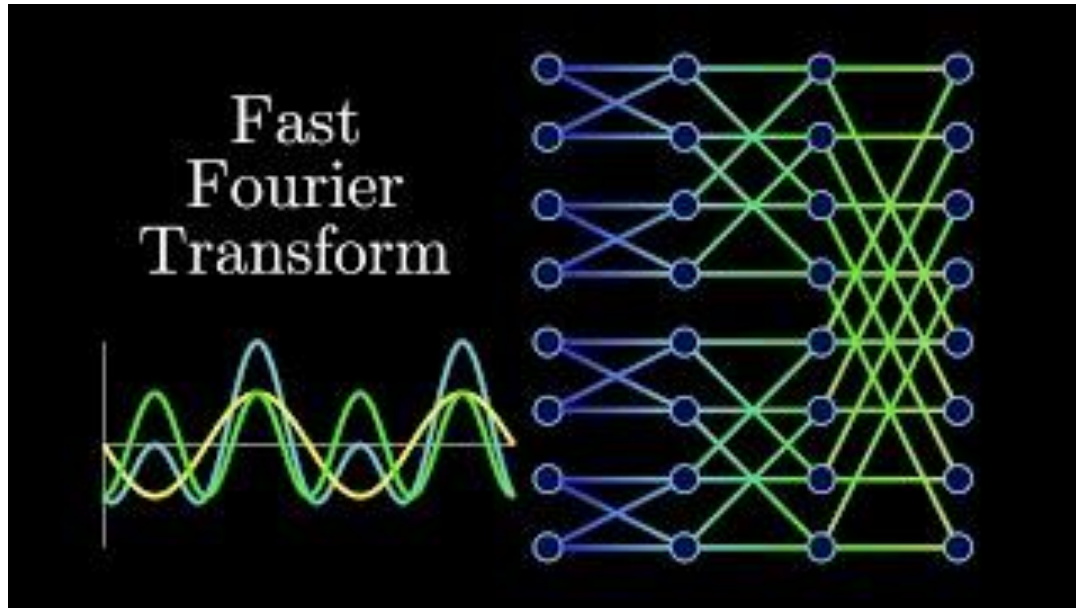
Key Learnings & Contributions

- Built a **complete OFDM transmitter and receiver** using **custom FFT/IFFT**
- Implemented **KMeans-based clustering** for threshold-free signal detection
- Visualized **signal energy, clustering output, and performance metrics**
- Evaluated system using:
 -  **ROC Curve**
 -  **Bit Error Rate (BER)**



FFT & IFFT Algorithm Explanation

- Following video by Reducible provides a clear and intuitive explanation of FFT and IFFT, crucial to OFDM systems:



FFT and IFFT algorithm from Scratch

$$\begin{pmatrix} F[0] \\ F[1] \\ F[2] \\ \vdots \\ F[N-1] \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1 \cdot 1} & \omega^{-1 \cdot 2} & \cdots & \omega^{-(N-1)} \\ 1 & \omega^{-2 \cdot 1} & \omega^{-2 \cdot 2} & \cdots & \omega^{-2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(N-1) \cdot 1} & \omega^{-(N-1) \cdot 2} & \cdots & \omega^{-(N-1)^2} \end{pmatrix} \begin{pmatrix} f[0] \\ f[1] \\ f[2] \\ \vdots \\ f[N-1] \end{pmatrix}$$

**Fourier
Transform
vector**

**DFT
Square Matrix**

**Data
vector**

We know that the fourier transform using this method take $O(N^2)$ time complexity.

But, in FFT and IFFT algorithm we reduce this complexity to $O(N \log N)$.

FFT and IFFT algorithm from Scratch

```
def FFT(P):
    n = len(P)
    if n == 1:
        return P

    omega = cmath.exp(2j * cmath.pi / n)

    P_e = P[0::2]
    P_o = P[1::2]

    y_e = FFT(P_e)
    y_o = FFT(P_o)

    y = [0] * n
    for j in range(n // 2):
        omega_j = omega**j
        y[j] = y_e[j] + omega_j * y_o[j]
        y[j + n // 2] = y_e[j] - omega_j * y_o[j]

    return y
```

```
def IFFT(P):
    n = len(P)
    if n == 1:
        return P

    omega_inv = cmath.exp(-2j * cmath.pi / n)

    P_e = P[0::2]
    P_o = P[1::2]

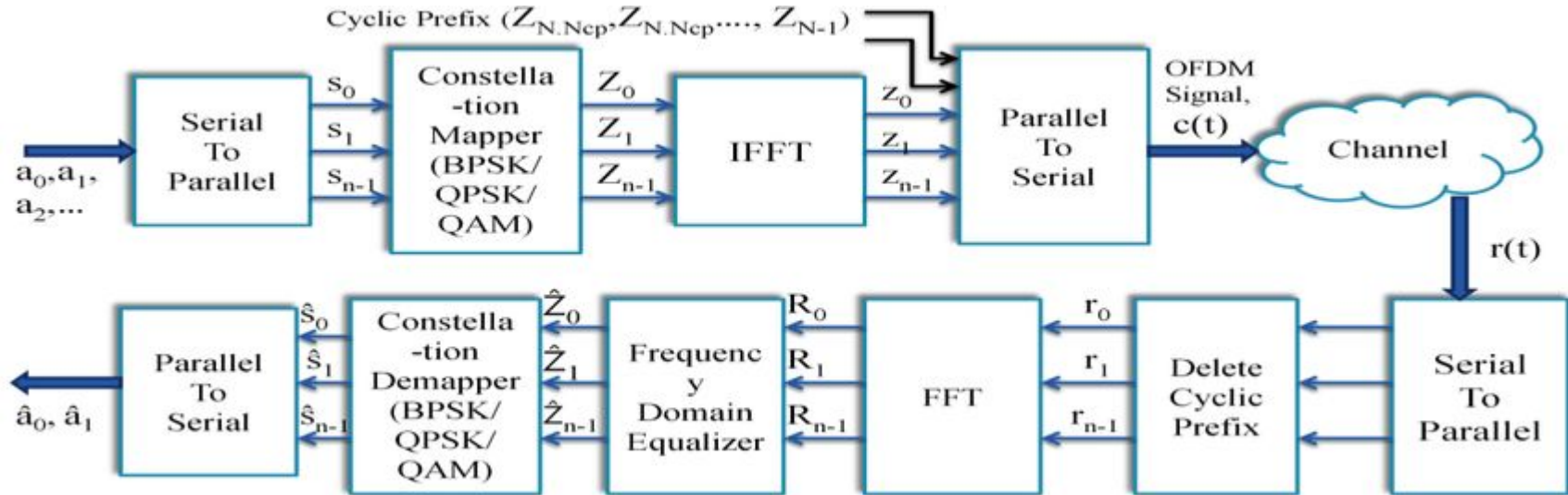
    y_e = IFFT(P_e)
    y_o = IFFT(P_o)

    y = [0] * n
    for j in range(n // 2):
        omega_j = omega_inv ** j
        y[j] = y_e[j] + omega_j * y_o[j]
        y[j + n // 2] = y_e[j] - omega_j * y_o[j]

    # Scale the result by 1/n
    return [val / n for val in y]
```



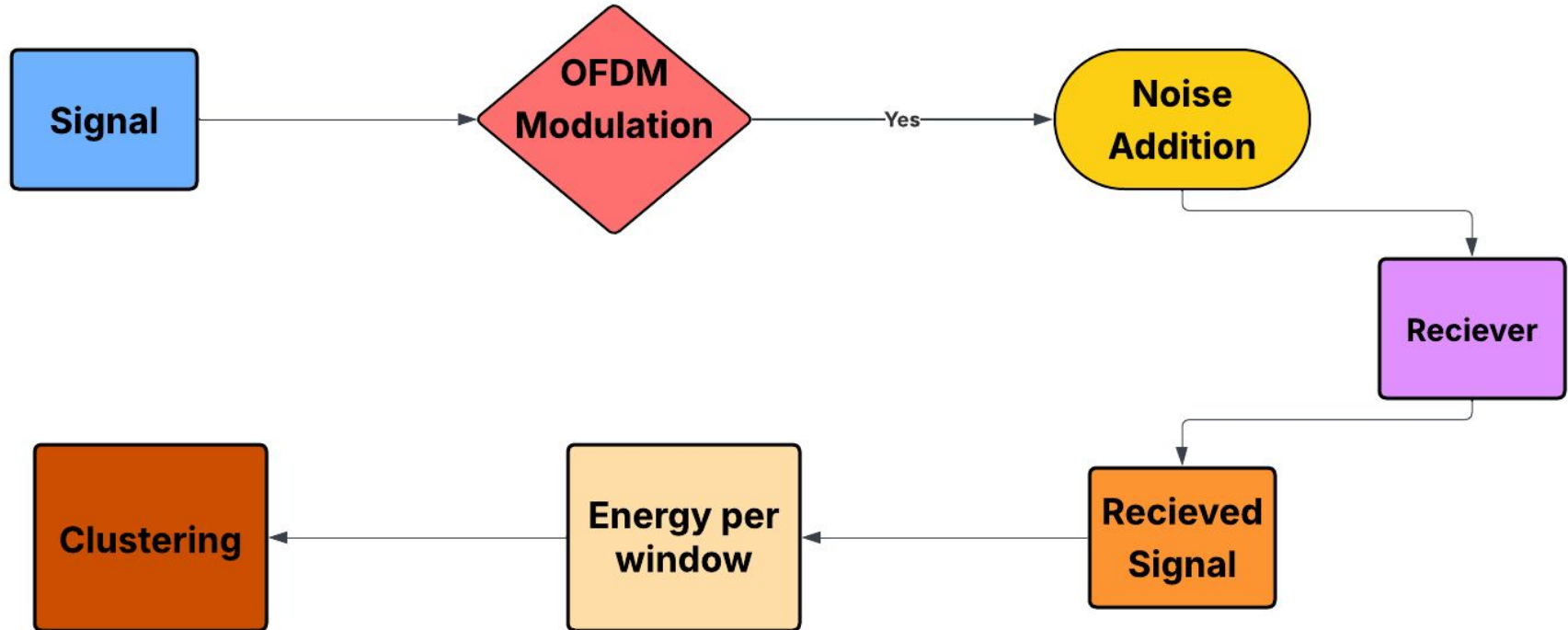
OFDM System Overview



Reference :

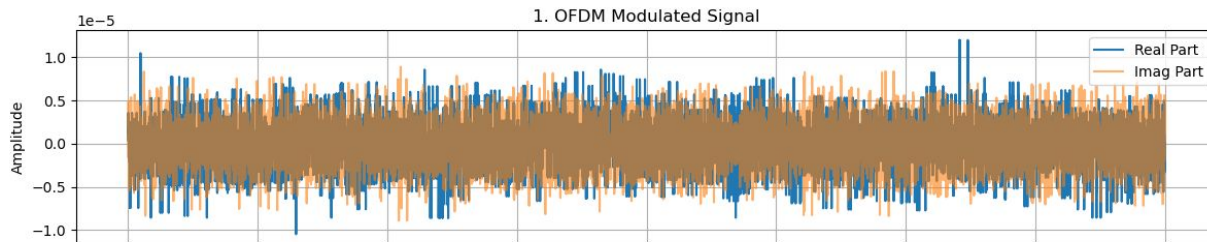
https://www.researchgate.net/figure/Transmitter-and-Receiver-architecture-of-OFDM_fig2_325283793

Framework

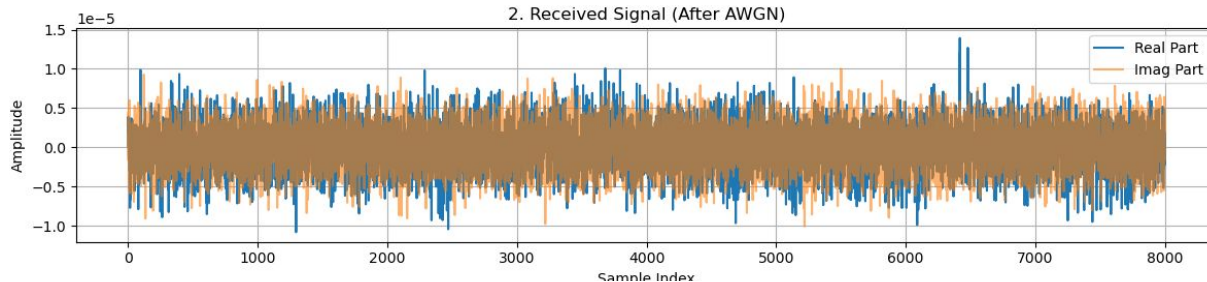


Signals Visualization

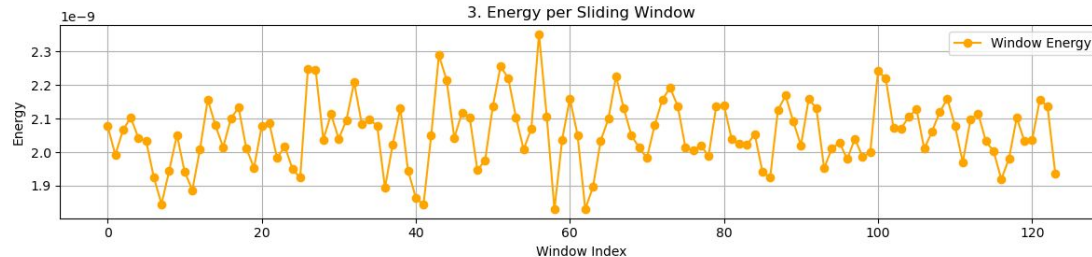
- 🎵 **Modulated Signal :**



- 📡 **Received Signal :**

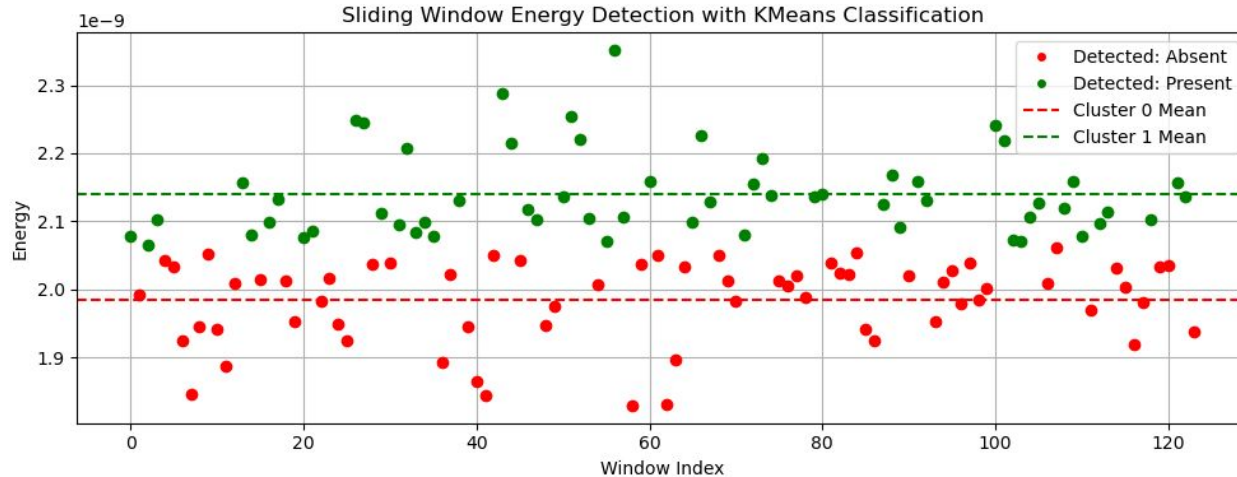


- ⚡ **Energy per Sliding Window :**





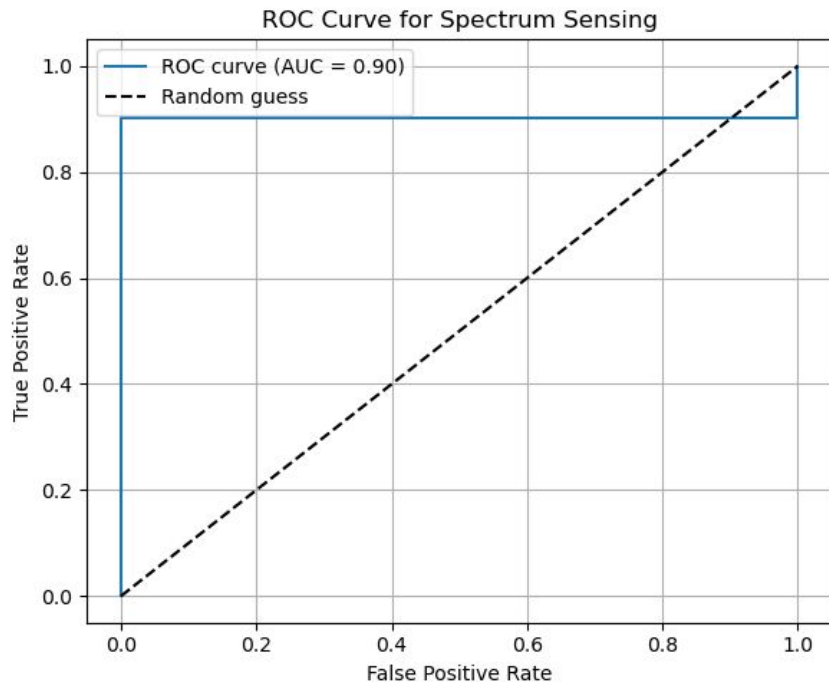
Energy and Clustering Visualization



We can see that above a threshold energy are classified as present and below that is classified as absent. Also the two dotted lines represents mean energy of that class.



ROC Curve



We can see that Area of ROC curve is 0.9 means our classifier is behaving far better than random guess.

Results and Future Works

- We obtained Block Error Rate nearly 0.

Future works :

- We can use multiple Receiver to solve real-world problems.
- We can use different Fusion Rules like Addition, Max, Majority Rule.
- Also co-operative sensing can be implemented using RL to select best neighbors and take all data and then we may use Unsupervised Learning or diff techniques to classify channel is free or busy.



Thank You!



Colab Notebook: [Click to Open](#)



Github Repository : [visit github](#)



Contact Me:



harsh.raj@iitg.ac.in



[linkedin.com/in/harsh-raj-13b100241](https://www.linkedin.com/in/harsh-raj-13b100241)