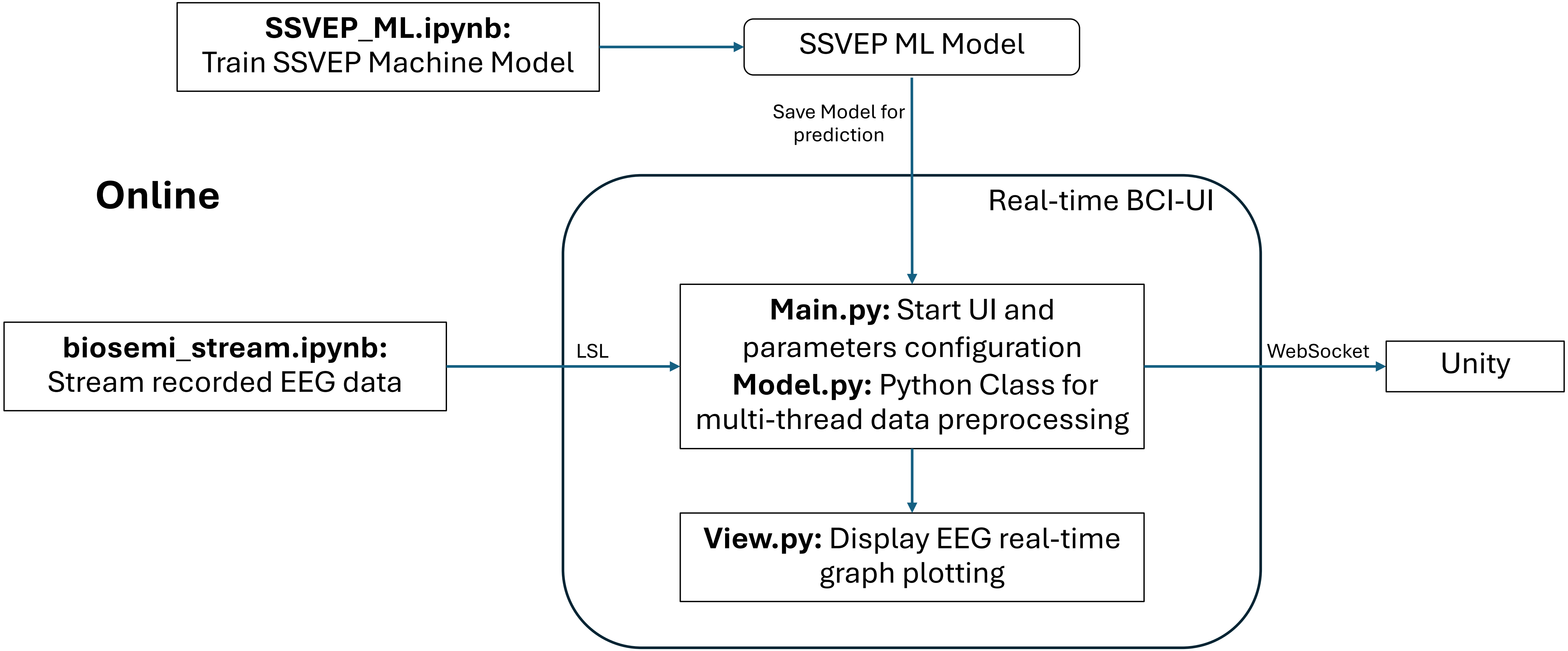
BCI
Real-time preprocess and classification
Example usage of SSVEP

Overview

Offline

Online



SSVEP_ML.ipynb: Train SSVEP Machine Model

SSVEP ML Model

Save Model for prediction

Real-time BCI-UI

biosemi_stream.ipynb: Stream recorded EEG data

LSL

Main.py: Start UI and parameters configuration

Model.py: Python Class for multi-thread data preprocessing

View.py: Display EEG real-time graph plotting

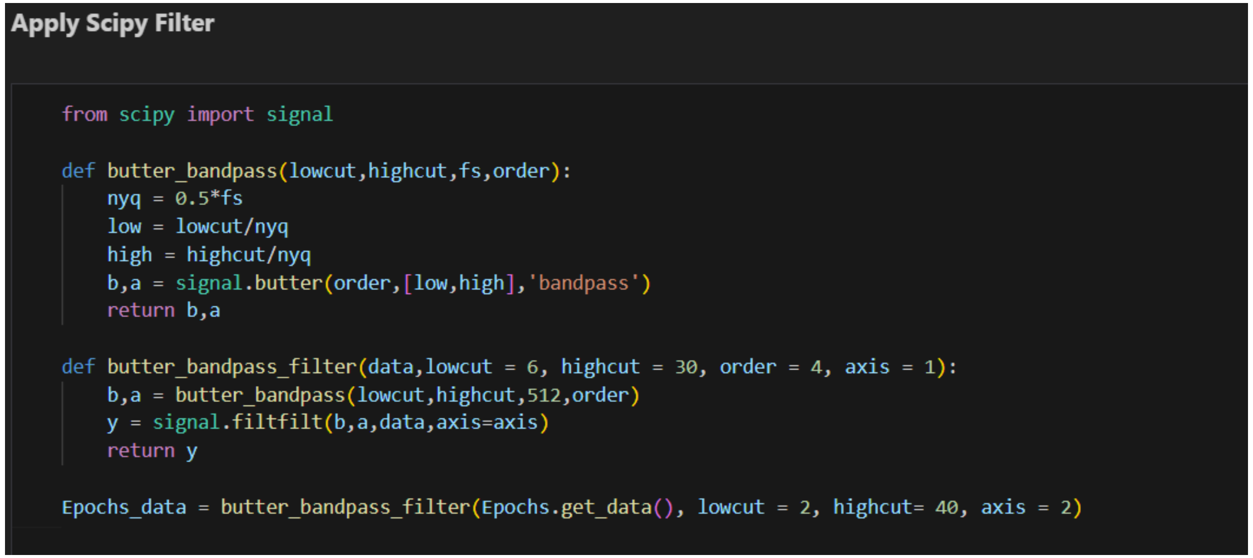
WebSocket

Unity

Changing in offline analysis

SSVEP_ML.ipynb

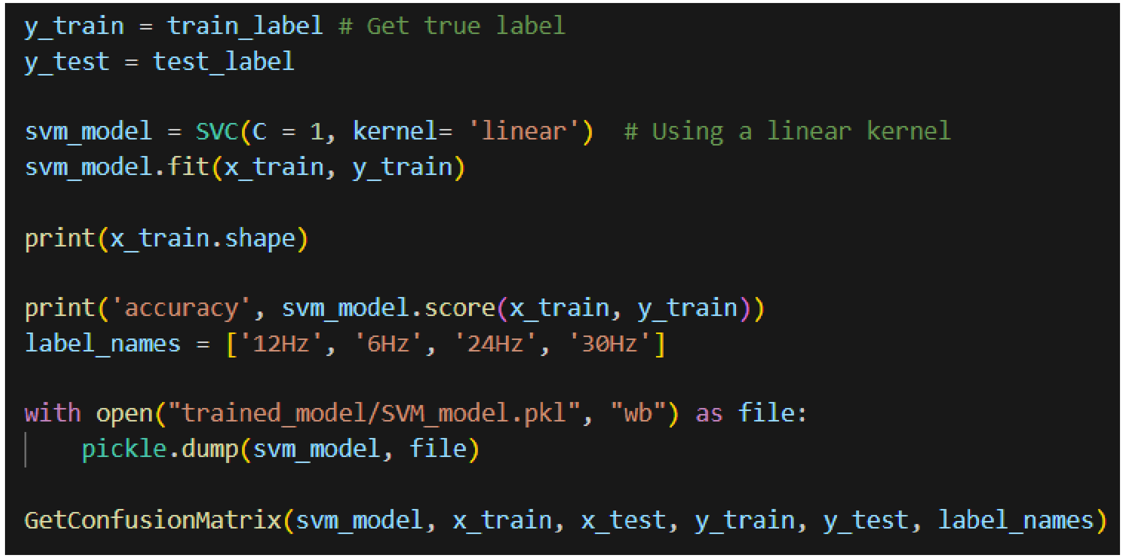
1. Changing Filter Method



Apply Scipy Filter

```
from scipy import signal

def butter_bandpass(lowcut,highcut,fs,order):
    nyq = 0.5*fs
    low = lowcut/nyq
    high = highcut/nyq
    b,a = signal.butter(order,[low,high],'bandpass')
    return b,a

def butter_bandpass_filter(data,lowcut = 6, highcut = 30, order = 4, axis = 1):
    b,a = butter_bandpass(lowcut,highcut,512,order)
    y = signal.filtfilt(b,a,data,axis=axis)
    return y

Epochs_data = butter_bandpass_filter(Epochs.get_data(), lowcut = 2, highcut= 40, axis = 2)
```

2. Saving ML model

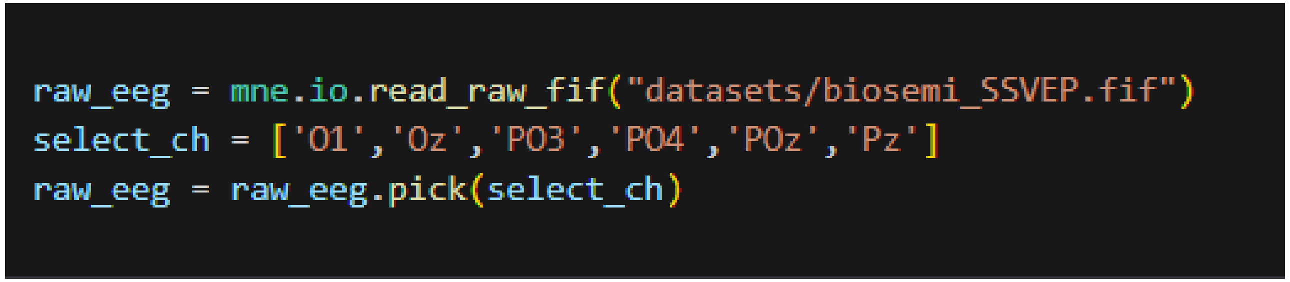


```
y_train = train_label # Get true label
y_test = test_label

svm_model = SVC(C = 1, kernel= 'linear')  # Using a linear kernel
svm_model.fit(x_train, y_train)

print(x_train.shape)

print('accuracy', svm_model.score(x_train, y_train))
label_names = ['12Hz', '6Hz', '24Hz', '30Hz']

with open("trained_model/SVM_model.pkl", "wb") as file:
    pickle.dump(svm_model, file)

GetConfusionMatrix(svm_model, x_train, x_test, y_train, y_test, label_names)
```

EEG data Streaming

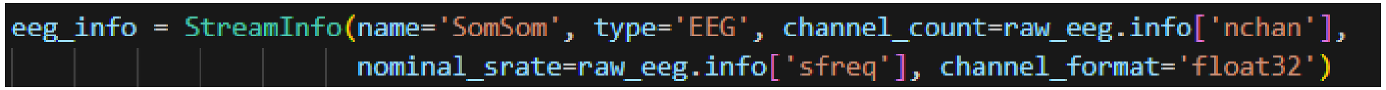
biosemi_stream.ipynb

1. Select target EEG channels

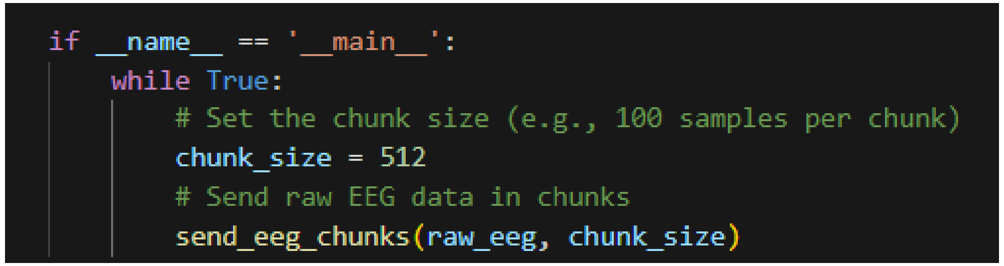


```python
raw_eeg = mne.io.read_raw_fif("datasets/biosemi_SSVEP.fif")
select_ch = ['O1','Oz','PO3','PO4','POz','Pz']
raw_eeg = raw_eeg.pick(select_ch)
```

2. Change name of StreamInfo and chunk_size same as your sampling rate



```python
eeg_info = StreamInfo(name='SomSom', type='EEG', channel_count=raw_eeg.info['nchan'],
                      nominal_srate=raw_eeg.info['sfreq'], channel_format='float32')
```



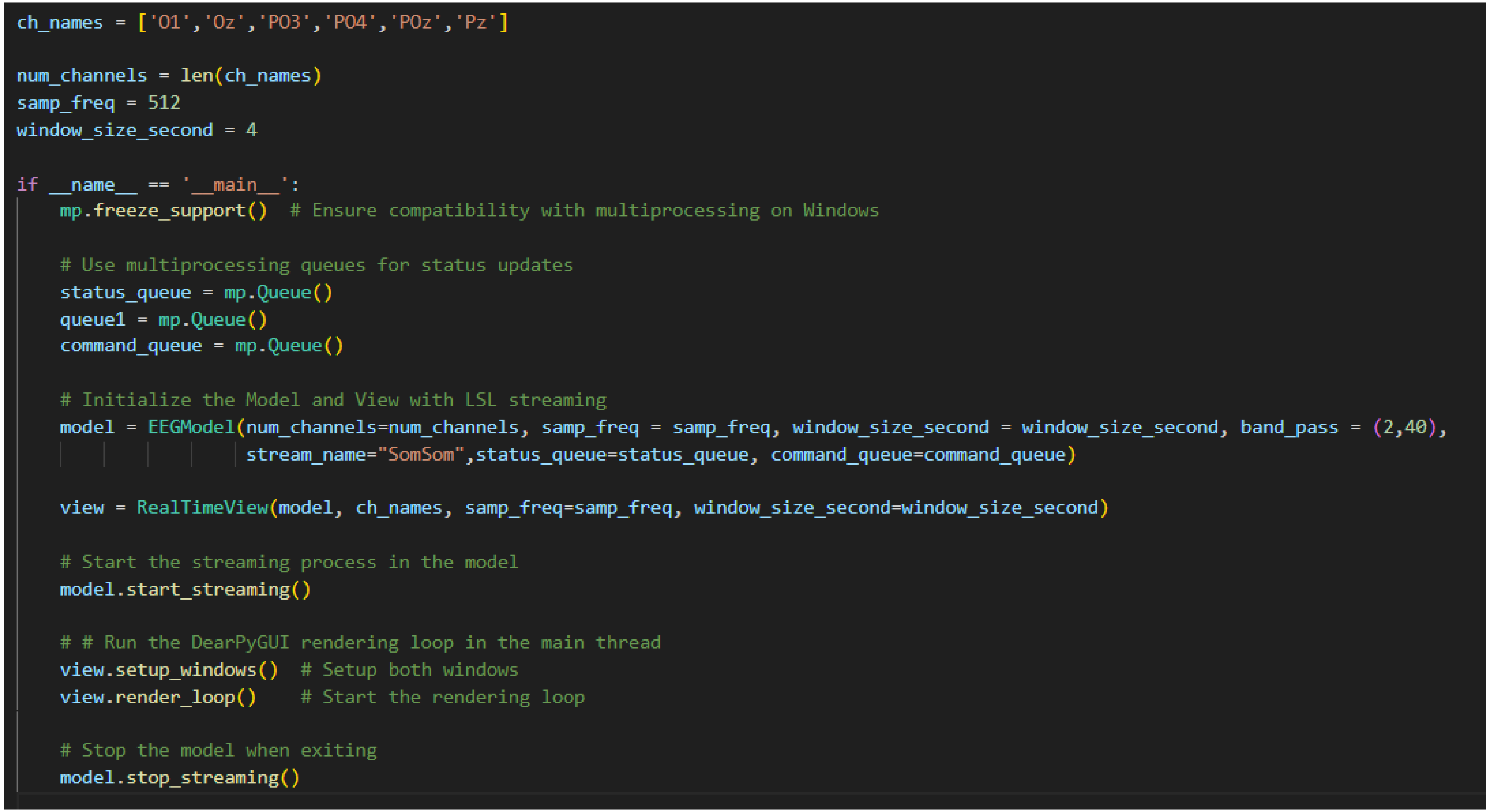
```python
if __name__ == '__main__':
    while True:
        # Set the chunk size (e.g., 100 samples per chunk)
        chunk_size = 512
        # Send raw EEG data in chunks
        send_eeg_chunks(raw_eeg, chunk_size)
```

Real-time BCI-UI

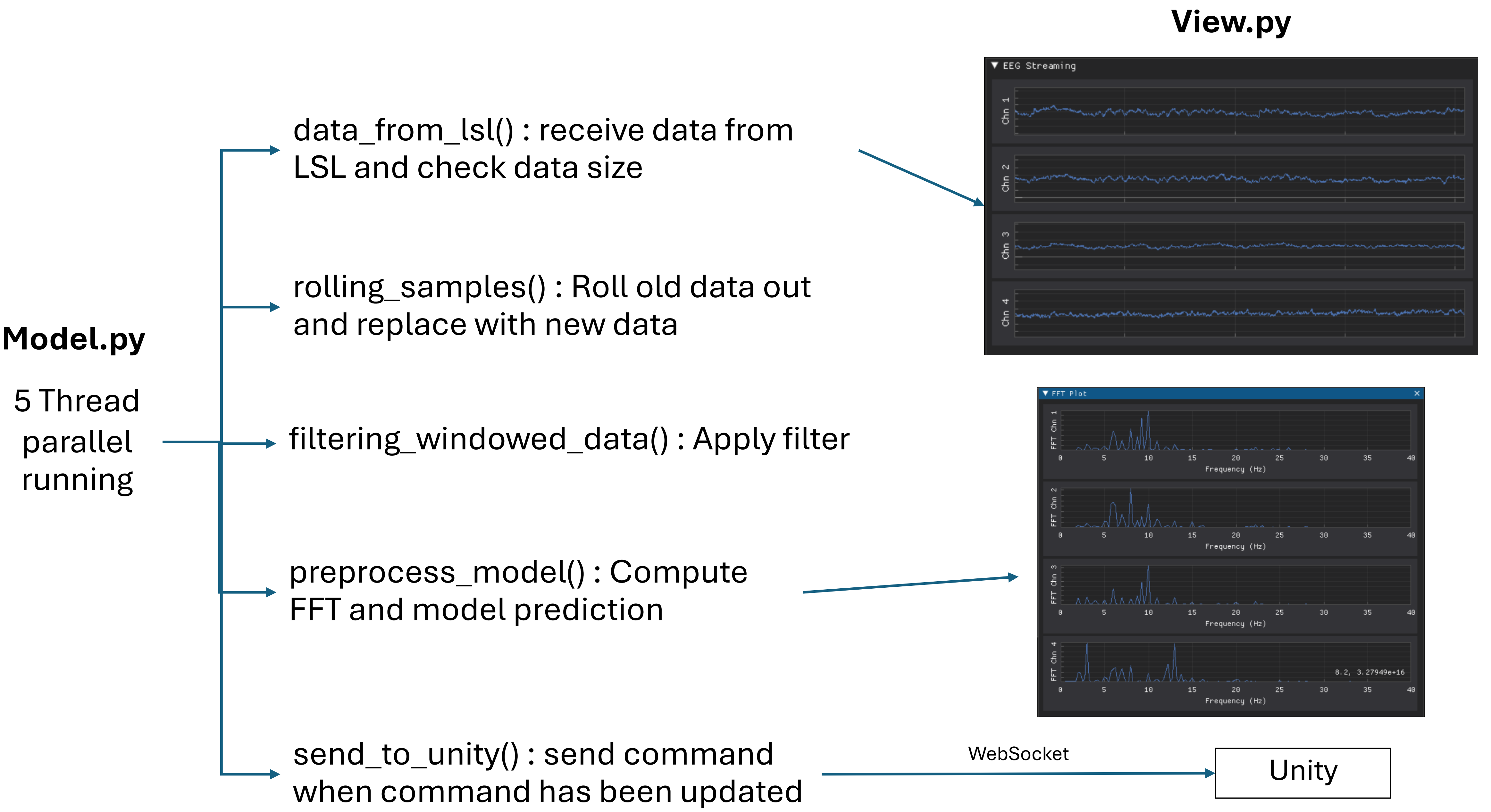
Main.py

Config parameters

Don’t forget to match your stream name



```
ch_names = ['O1','Oz','PO3','PO4','POz','Pz']

num_channels = len(ch_names)
samp_freq = 512
window_size_second = 4

if __name__ == '__main__':
    mp.freeze_support()  # Ensure compatibility with multiprocessing on Windows

    # Use multiprocessing queues for status updates
    status_queue = mp.Queue()
    queue1 = mp.Queue()
    command_queue = mp.Queue()

    # Initialize the Model and View with LSL streaming
    model = EEGModel(num_channels=num_channels, samp_freq = samp_freq, window_size_second = window_size_second, band_pass = (2,40),
                     stream_name="SomSom",status_queue=status_queue, command_queue=command_queue)

    view = RealTimeView(model, ch_names, samp_freq=samp_freq, window_size_second=window_size_second)

    # Start the streaming process in the model
    model.start_streaming()

    # # Run the DearPyGUI rendering loop in the main thread
    view.setup_windows()  # Setup both windows
    view.render_loop()    # Start the rendering loop

    # Stop the model when exiting
    model.stop_streaming()
```



Model.py

5 Thread parallel running

data_from_lsl() : receive data from LSL and check data size

rolling_samples() : Roll old data out and replace with new data

filtering_windowed_data() : Apply filter

preprocess_model() : Compute FFT and model prediction

send_to_unity() : send command when command has been updated

View.py

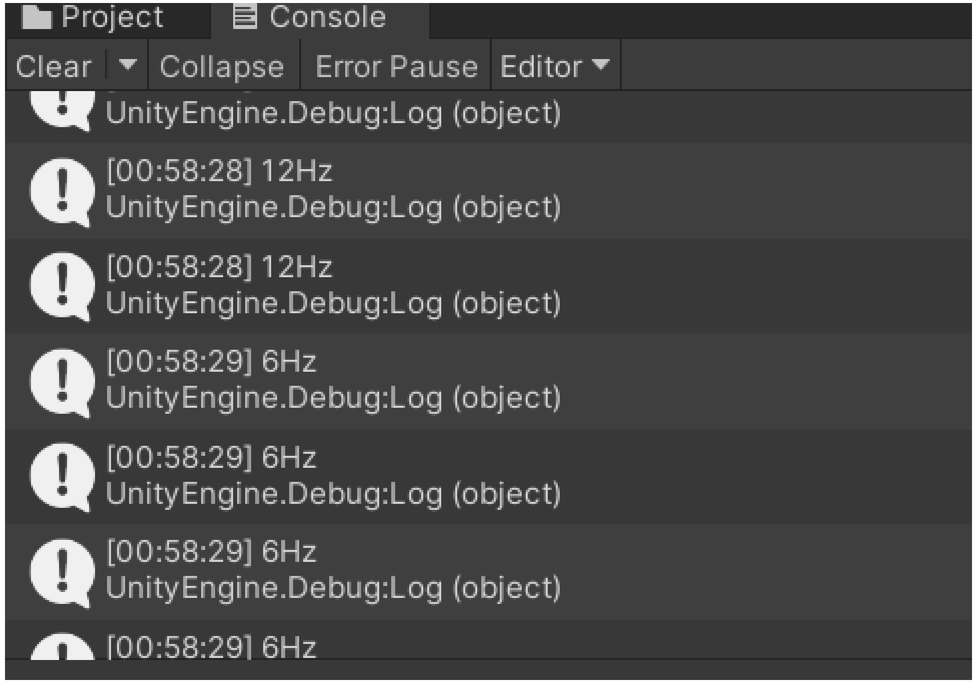
WebSocket

Unity

Unity

If you want to receive command in Unity, copy this code to your C# script

Print Log in callback function

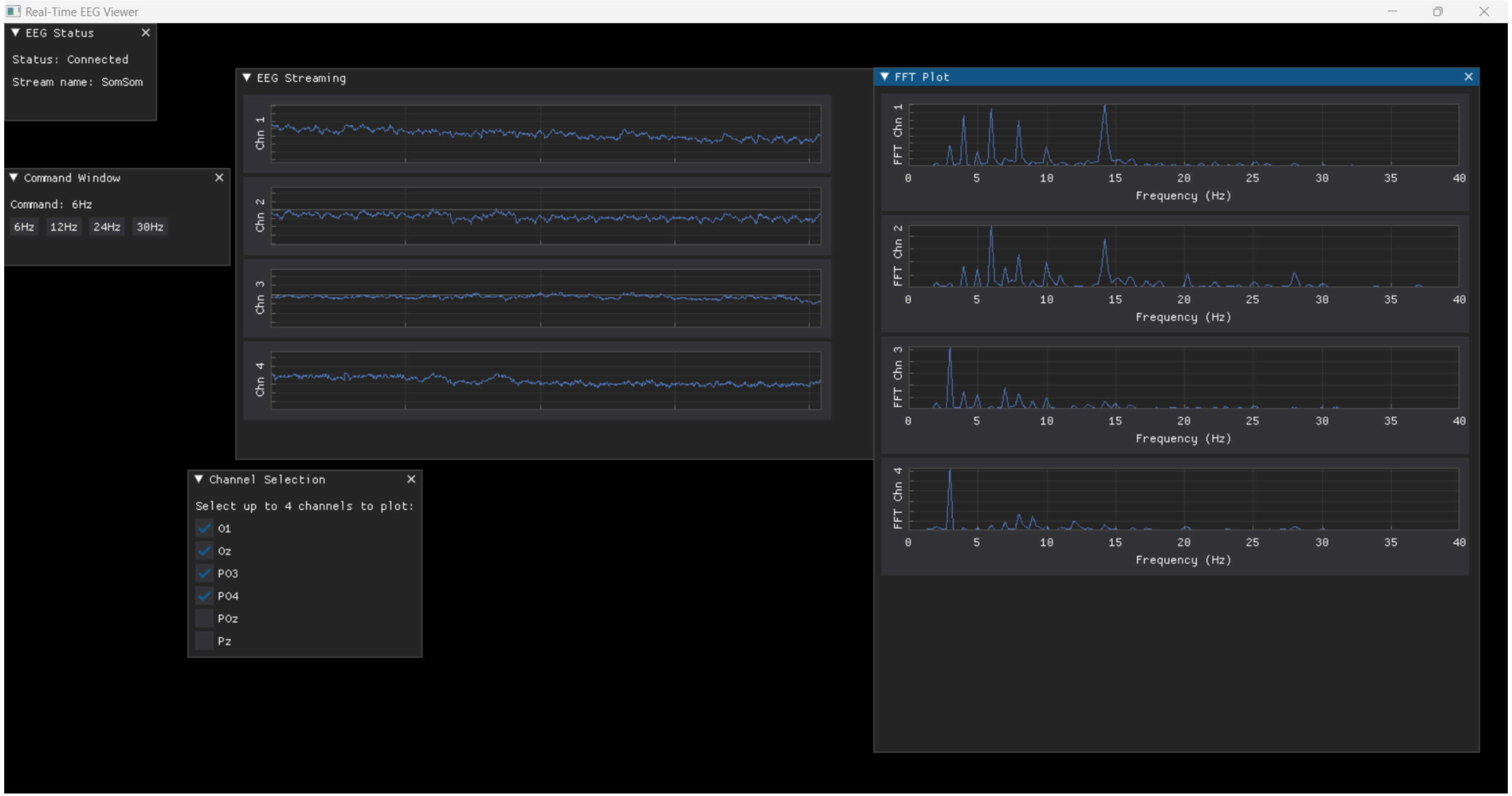


```
UnityEngine.Debug:Log (object)
[00:58:28] 12Hz
UnityEngine.Debug:Log (object)
[00:58:28] 12Hz
UnityEngine.Debug:Log (object)
[00:58:29] 6Hz
UnityEngine.Debug:Log (object)
[00:58:29] 6Hz
UnityEngine.Debug:Log (object)
[00:58:29] 6Hz
UnityEngine.Debug:Log (object)
[00:58:29] 6Hz
```

```csharp
using UnityEngine;
using UnityEngine.UI;
using System;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.Threading;

public class PythonToUnity1 : MonoBehaviour
{
    Thread receiveThread;
    UdpClient client;
    public int port = 1880;
    private bool startRecieving = true;
    private string data;
    // Start is called before the first frame update
    void Start()
    {
        receiveThread = new Thread(
            new ThreadStart(ReceiveData));
        receiveThread.IsBackground = true;
        receiveThread.Start();
    }

    public void ReceiveData()
    {
        client = new UdpClient(port);
        while (startRecieving)
        {
            try
            {
                IPEndPoint anyIP = new IPEndPoint(IPAddress.Any, 0);
                byte[] dataByte = client.Receive(ref anyIP);
                data = Encoding.UTF8.GetString(dataByte);
                Debug.Log(data);
            }
            catch (Exception err)
            {
                print(err.ToString());
            }
        }
    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

Files Running Steps

1. Run biosemi_stream.ipynb
2. Run Main.py



Status: Connected

Stream name: SomSom

Command: 6Hz

Select up to 4 channels to plot:

☑ O1

☑ Oz

☑ PO3

☑ PO4

□ POz

□ Pz