EEG Preprocessing, EDA, and Dataloader Refactoring Report

1. Overview

The goal of this work was to redesign the EEG pipeline so that:

- The **dataloader** becomes fully independent.
- The **preprocessing, filtering, and EDA** stages operate as a modular preprocessing framework.
- The system produces reusable outputs that can be connected to any future model or dataloader.

Instead of tightly coupling data ingestion with preprocessing logic, the pipeline was restructured into a **producer–consumer architecture**:

- **Producer:** preprocessing + filtering + EDA pipeline
- **Consumer:** dataloader that reads cleaned outputs

This makes the EEG workflow scalable, reusable, and easier to maintain.

2. Classes and Files Created (and Why They Matter for EEG)

A. Preprocessing System

`preprocessor.py` → `EEGPreprocessor`

Handles:

- Loading data
- Resampling
- Re-referencing
- Filtering
- Writing cleaned derivatives

Importance for EEG:

- EEG signals require standardized preprocessing.
- Modular methods allow turning steps ON/OFF via YAML config.

`filtering.py` → `FilterApplier` + `WaveletDenoiser`

Implements:

- Bandpass filtering
- Notch filtering
- FIR/IIR selection
- Optional wavelet denoising

Why important in EEG:

EEG signals contain:

- line noise (50/60 Hz)
- slow drifts
- high-frequency artifacts

Filtering ensures:

- better signal-to-noise ratio
- meaningful frequency analysis

`time_domain.py` → `TimeDomainModule`

Implements:

- QC metrics
- Bad channel detection
- Epoching
- Artifact rejection
- Label interval processing

Importance:

EEG quality varies across channels and time.
This module ensures:

- noisy electrodes don't corrupt analysis
- standardized epoch segmentation

B. EDA System

`eda_engine.py` → `EDAEngine`

Central orchestrator that runs:

- Time-domain analysis
- Frequency-domain analysis
- Time-frequency analysis
- Plot generation

Importance:

EDA validates preprocessing by showing:

- signal quality
- power distribution
- temporal dynamics

`freq_analysis.py` → `FrequencyDomainAnalyzer`

Produces:

- PSD (Power Spectral Density)
- Bandpower statistics

Why important:

EEG is fundamentally frequency-driven:

- Delta / Theta / Alpha / Beta / Gamma bands
- Seizure patterns often appear in specific frequency ranges.

`timefreq_analysis.py` → `TimeFrequencyAnalyzer`

Produces:

- STFT spectrograms
- Morlet TFR maps

Why important:

Seizures are dynamic events.
Time-frequency analysis shows how power evolves over time.

`artifacts.py` → `ArtifactWriter`

Handles:

- saving JSON
- saving CSV
- saving plots

Importance:

Creates reproducible artifacts that can be used for:

- reports
- debugging
- model validation

C. Data Interface Layer

`bids_io.py` → `BIDSLoader`

Handles:

- discovering BIDS recordings
- loading raw EEG
- locating events.tsv

Importance:

Keeps pipeline compatible with standard EEG datasets.

`index_builder.py` → `WindowIndexBuilder`

Creates the bridge between preprocessing and dataloader.

Outputs:

window_index_train.csv

Why important:

This is the key architectural improvement.

It makes:

dataloader independent
preprocessing reusable
experiments reproducible

3. Outputs of the Code

After running the pipeline, the following outputs are generated:

Cleaned EEG Derivatives

results/preprocess/bids/sub-001/..._eeg.fif

These are fully preprocessed signals.

EDA Outputs

Located under:

results/preprocess/eda/

Includes:

- qc.json → signal quality metrics
- psd_mean.csv → frequency power distribution
- bandpower.csv → band energy
- stft_summary.csv → time-frequency summary
- raw_before.png / raw_after.png → signal comparison
- epochs.png → segmented EEG windows
- tfr_morlet.png → time-frequency representation

Dataloader Input Artifact

results/dataloader/window_index_train.csv

This connects preprocessing to model training.

4. How Preprocessing, Filtering, and EDA Were Used Together

Step 1 — Preprocessing

- Resampled EEG to target sampling rate.
- Applied bandpass + notch filters.
- Optional wavelet denoising.
- Marked bad channels via QC.

Goal: produce standardized EEG signals.

Step 2 — Time-Domain Analysis

- Computed variance and kurtosis.
- Identified noisy or flat channels.
- Created fixed-length epochs.

Goal: ensure clean segmentation.

Step 3 — Frequency-Domain Analysis

- Computed PSD.
- Extracted bandpower features.

Goal: understand spectral structure of EEG.

Step 4 — Time-Frequency Analysis

- Generated STFT spectrograms.
- Generated Morlet TFR maps.

Goal: visualize temporal evolution of brain activity.

Step 5 — Index Creation

- Used events.tsv to mark seizure intervals.
- Generated window index CSV.

Goal: decouple preprocessing from training.

5. Why This Architecture Is Important for EEG Research

This redesign introduces:

Modularity

Each component works independently.

Reproducibility

Cleaned FIF + index CSV ensure consistent datasets.

Flexibility

New preprocessing methods can be added without touching dataloader.

Scalability

Multiple datasets or experiments can reuse the same pipeline.

6. Final Summary

This work transformed the EEG processing workflow from a monolithic pipeline into a modular architecture:

- Preprocessing produces standardized EEG derivatives.
- EDA validates signal quality and frequency characteristics.
- IndexBuilder creates a dataset definition.
- The dataloader becomes a lightweight consumer.

This separation is critical for building robust EEG machine learning systems and enables faster experimentation, cleaner code organization, and more reliable analysis.