P300 Dataset – Data Structure and Processing Pipeline

1. Dataset Overview

The dataset consists of 4 MATLAB .mat files, containing EEG recordings for two contributors (A and B):

Contributor	Training Set	Test Set	Characters
A	Subject_A_Train.mat	Subject_A_Test.mat	85 train / 100 test
B	Subject_B_Train.mat	Subject_B_Test.mat	85 train / 100 test

Dataset Tree Structure

```
Data
│
├── Contributor A
│     ├── Subject_A_Train.mat     (85 trials)
│     └── Subject_A_Test.mat      (100 trials)
│
└── Contributor B
      ├── Subject_B_Train.mat     (85 trials)
      └── Subject_B_Test.mat      (100 trials)
```

For training, we concatenate A_train + B_train, giving:
Total Training Trials = 170 character epochs

Total training characters (epochs):
85 (A) + 85 (B) = 170 character epochs (trials)

```
==================== Processing Contributor: I ====================
Loading data from: ../../data/contributor_I.mat
Number of trials based on Signal array: 85
Extracted Target Character String (length 85): EAEVQTDOJG8RBRGONCEDHCTUIDBPUHMEM6OUXOCFOUKWA4VJEF...
TargetChar string length matches number of signal trials.

 Data Info (Initial):
Contributor     Sampling Freq. (Hz)  Recording (min)      Chars      Spelled Word
==========================================================================================
I               240.00               46.01                85         EAEVQTDOJG8RBRGONCEDHCTUIDBPUH
                                                                     MEM6OUXOCFOUKWA4VJEFRZROLHYNQD
                                                                     W_EKTLBWXEPOUIKZERYOOTHQI

==================== Processing Contributor: II ====================
Loading data from: ../../data/contributor_II.mat
Number of trials based on Signal array: 85
Extracted Target Character String (length 85): VGREAAH8TVRHBYN_UGCOLO4EUERDOOHCIFOMDNU6LQCPKEIREK...
TargetChar string length matches number of signal trials.

 Data Info (Initial):
Contributor     Sampling Freq. (Hz)  Recording (min)      Chars      Spelled Word
==========================================================================================
II              240.00               46.01                85         VGREAAH8TVRHBYN_UGCOLO4EUERDOO
                                                                     HCIFOMDNU6LQCPKEIREKOYRQIDJXPB
                                                                     KOJDWZEUEWWFOEBHXTQTTZUMO
```

2. Data Structure of Each Variable

concatenate A_train + B_train, giving:

Variable	Dimension 1	Dimension 2	Dimension 3
Signal	170 trials	7,794 samples	64 channels
Flashing	170 trials	7,794 samples	–
StimulusCode	170 trials	7,794 samples	–
StimulusType	170 trials	7,794 samples	–
TargetChar	170 trials	7,794 samples	–

3. Meaning of Each Variable

3.1 Signal

Shape per trial: (7794 samples × 64 channels)
At each sample (time point), the EEG cap records 64 sensor values.

So: Signal[trial][sample][channel]

3.2 Flashing

Indicates if a row or column of the speller matrix is flashing.
  1 → Flash ON
  0 → No flash (matrix blank)
Used to detect flash start positions.

3.3 StimulusType

Indicates whether the flash corresponds to the target character.
  1 → Target flash (row/column contains the character the user focuses on)
  0 → Non-target flash
This is the classification label.

4. Downsampling (240 Hz → 120 Hz)

Initial frequency: 240 Hz
Target frequency: 120 Hz

4.1 Butterworth Bandpass Filter

A 4th-order Butterworth filter (0.1–20 Hz) is applied first:
  Low frequency: 0.1 Hz
  High frequency: 20 Hz

Filtering does not change the number of samples.

4.2 Downsampling

SCALE_FACTOR = 240 / 120 = 2
The downsampling keeps every 2nd sample: signals = signals[:, ::2, :]
Result:
  Before: 7794 samples
  After: 3897 samples

So, each trial now has: 3897 samples × 64 channels

5. Flash Detection and Window Extraction

5.1 Flash Start Detection

We scan the Flashing sequence of each trial.

```
Example:
Index:      0  1  2  3  4  ...
Flashing:  [ 1, 1, 1, 0, 0, 1, ... ]
             ^           ^
        Flash #1     Flash #2
```

A flash start occurs when:

  Flashing[i] == 1 AND Flashing[i-1] == 0
  Each detected flash triggers a window extraction.

5.2 Window Extraction

Window size: 78 samples

For each flash starting at index s:
  window = Signal[trial][s : s + 78 , :]

Example flash sequence:

  Flash at 0   → Window [0 : 78]
  Flash at 150   → Window [150 : 228]
  Flash at 280   → Window [280 : 358]
  ...
  Flash at 3200 → Window [3200 : 3278]

Flash Detection and Window Extraction

The algorithm scans through the entire trial timeline looking for **flash start events**:

```
Flash Pattern Example:
Sample Index:  0    1    2    3    4    5    ...  100  101  102  ...
Flashing:     [1]  [1]  [1]  [0]  [0]  [1]  ...  [0]  [1]  [1]  ...
               ^                   ^                   ^
            Flash #1            Flash #2            Flash #3
          (starts at 0)       (starts at 5)      (starts at 101)
```

Detailed Timeline Visualization:

```
Sample Index:
0                   78              150             228             280       358
|-------------------|               |---------------|               |---.......-----|
[    Window 1       ]               [   Window 2    ]               [  Window 3    ]
|   78 samples      |               |  78 samples   |               |  78 samples  |
|   64 channels     |               | 64 channels   |               |  64 channels |
└─ Flash detected   |               └─ Flash detected               └─ Flash detected
   at sample 0                         at sample 150                   at sample 280
```

6. Final Training Structure

Every character epoch (trial) contains: 30 flash windows
each window = (78 samples × 64 channels)

Example:

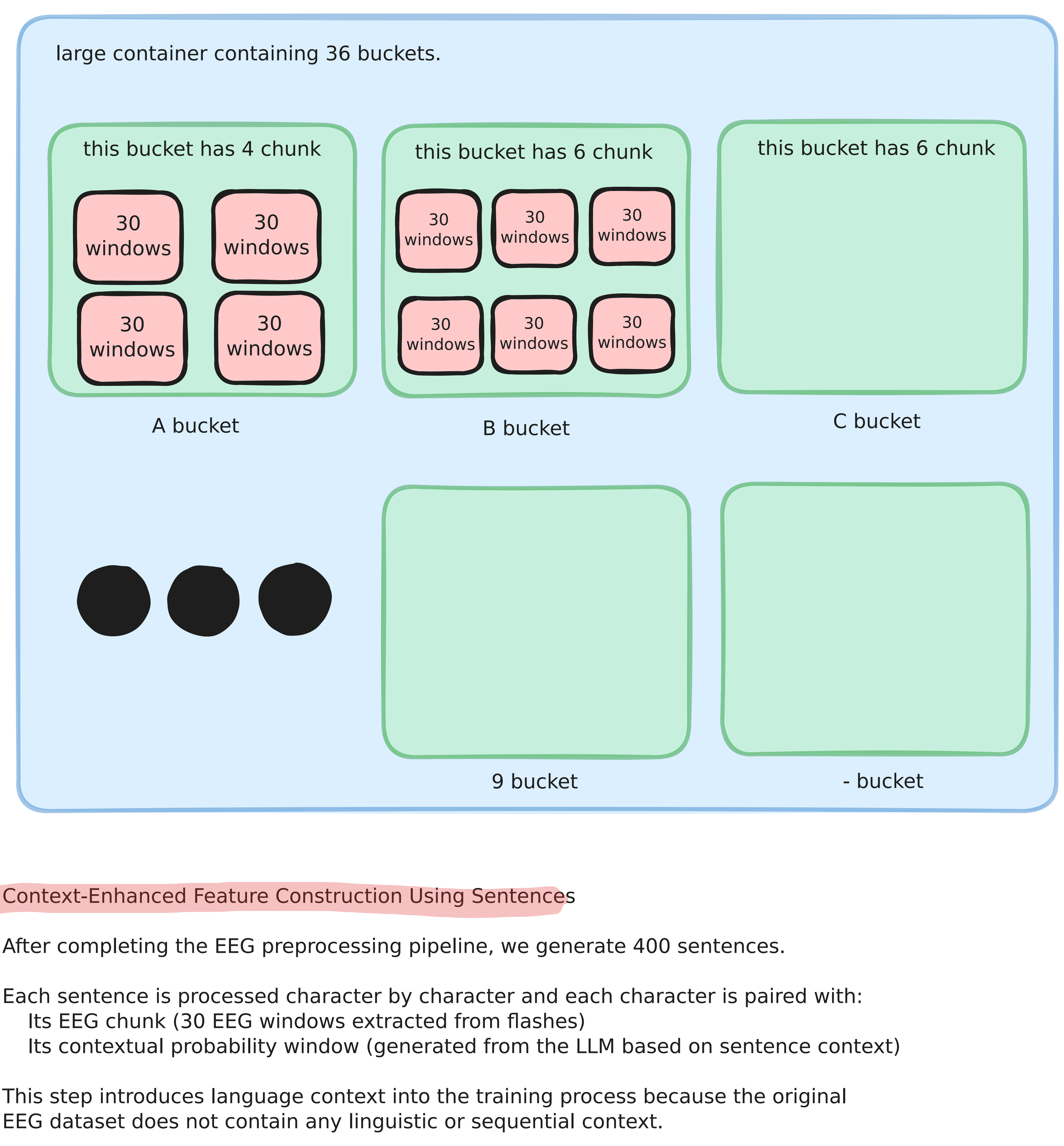
```
--- Verification (Combined Data) ---
Total number of windows per character across all contributors:
Character 'A': 120 windows
Character 'B': 180 windows
Character 'C': 180 windows
Character 'D': 240 windows
Character 'E': 480 windows
Character 'F': 120 windows
Character 'G': 120 windows
Character 'H': 240 windows
Character 'I': 180 windows
Character 'J': 120 windows
Character 'K': 180 windows
```

Character A appears 4 times: 4 trials × 30 windows = 120 windows for 'A'
Character B appears 6 times: 6 trials × 30 windows = 180 windows for 'B'

7. Summary - Final Data Organization After All Processing:

After completing all preprocessing steps (filtering → downsampling → flash detection → window extraction), the entire dataset is organized into a large container containing 36 buckets.

Each bucket corresponds to one character in the P300 speller matrix:
  A, B, C, D, E, F, G, H, I, J, K, L,
  M, N, O, P, Q, R, S, T, U, V, W, X,
  Y, Z, 1, 2, 3, 4, 5, 6, 7, 8, 9, _



8. Context-Enhanced Feature Construction Using Sentences

After completing the EEG preprocessing pipeline, we generate 400 sentences.

Each sentence is processed character by character and each character is paired with:
  Its EEG chunk (30 EEG windows extracted from flashes)
  Its contextual probability window (generated from the LLM based on sentence context)

This step introduces language context into the training process because the original EEG dataset does not contain any linguistic or sequential context.

By combining EEG activity with sentence-based context, the classifier learns that:
Characters do not occur independently

Some characters are more likely given the previous characters
EEG signals + language model context → better P300 prediction

Example 1: Sentence: "THE QUICK DOG JUMPED OVER"

Take the character "G" in the word "DOG".

Step 1 — Extract EEG Chunk
  "G" appears several times in the dataset.
  Select one chunk and map it to "G",Hence we have:
    EEG Chunk = 30 windows
    Each window = 78 × 64
  So "G" has → 30 EEG windows.

Step 2 — Create Contextual Input
  For the target character "G", the preceding text is:
    Context: "THE QUICK DO"
  This context is fed into the LLM (NLP model), which converts it into a probability vector representing how likely each of the 36 P300 characters is next.
  The output is a single context probability window, which becomes the 31st window for the character.

Final Feature Vector for "G"
  EEG: 30 windows (from EEG data)
  Context window: +1 window   (from LLM probabilities)
  Total windows: 31 windows

So the feature vector for character "G" becomes: 31 windows × (78 × 64)