Analysis of Battery Dataset Preprocessing Code

BatteryML Framework Analysis

September 15, 2025

1 Input Variables Across All Datasets

1.1 Common Input Variables (Present in Most Datasets)

- **Voltage (V)** - Battery terminal voltage
- **Current (A)** - Charge/discharge current
- **Time (s)** - Test time or cycle time
- **Cycle Index** - Cycle number
- **Charge Capacity (Ah)** - Cumulative charge capacity
- **Discharge Capacity (Ah)** - Cumulative discharge capacity
- **Temperature (°C)** - Cell temperature (when available)

1.2 Dataset-Specific Variables

CALCE Dataset:

- `Test_Time(s)`, `Current(A)`, `Voltage(V)`, `Cycle_Index`, `date`

HNEI Dataset:

- `Test_Time (s)`, `Current (A)`, `Voltage (V)`, `Cell_Temperature (C)`, `Discharge_Capacity (Ah)`, `Charge_Capacity (Ah)`, `Cycle_Index`

HUST Dataset:

- `Current (mA)`, `Time (s)`, `Voltage (V)` (converted from mA to A)

MATR Dataset:

- `I`, `V`, `Qc`, `Qd`, `Qdlin`, `T`, `Tdlin`, `dQdV`, `t` (from HDF5 files)
- Additional: Internal resistance, charge time, temperature statistics

OX Dataset:

- `Test_Time (s)`, `Current (A)`, `Voltage (V)`, `Cell_Temperature (C)`, `Discharge_Capacity (Ah)`, `Charge_Capacity (Ah)`, `Cycle_Index`

RWTH Dataset:

- `Zeit`, `Programmdauer`, `Strom`, `Spannung` (German column names)

SNL Dataset:

- `Test_Time (s)`, `Current (A)`, `Voltage (V)`, `Cell_Temperature (C)`, `Discharge_Capacity (Ah)`, `Charge_Capacity (Ah)`, `Cycle_Index`

UL-PUR Dataset:

- `Test_Time (s)`, `Current (A)`, `Voltage (V)`, `Cell_Temperature (C)`, `Discharge_Capacity (Ah)`, `Charge_Capacity (Ah)`, `Cycle_Index`

ARBIN/NEWARE Datasets:

- Configurable column mapping with standard variables plus additional fields like energy, step index, internal resistance

2 Preprocessing Steps Applied

2.1 Data Loading and Format Conversion

- **CALCE**: Handles both Excel (.xlsx/.xls) and text (.txt) files with different parsing logic
- **HUST**: Loads from pickle files with nested data structure
- **MATR**: Loads from HDF5 (.mat) files with complex nested structure
- **RWTH**: Multi-level zip file extraction and CSV parsing
- **ARBIN/NEWARE**: Configurable file format support (CSV, Excel)

2.2 Data Cleaning and Quality Control

2.2.1 Cycle Filtering:

- **CALCE**: Uses median filter with 21-point window to remove abnormal cycles (threshold: 3× median absolute deviation)
- **HNEI**: Skips first 12 problematic cycles, uses Hampel filter for outlier detection
- **HUST**: Skips first 2 cycles for specific cells
- **MATR**: Removes batteries that don't reach 80% capacity
- **RWTH**: Removes abnormal cycles using statistical methods
- **SNL**: Extensive list of cells to drop (24 specific cells), uses Hampel filter
- **UL-PUR**: Skips first 12 cycles, uses Hampel filter

2.2.2 Outlier Detection Methods:

- **Hampel Filter**: Used by HNEI, SNL, UL-PUR (threshold: 3× median absolute deviation)
- **Median Filter**: Used by CALCE
- **Statistical Methods**: Used by RWTH (window-based anomaly detection)

2.3 Data Transformation

2.3.1 Capacity Calculation:

- **CALCE, HUST, RWTH**: Calculate charge/discharge capacity using numerical integration: $Q[i] = Q[i-1] + I[i] \times (t[i] - t[i-1])/3600$
- **Other datasets**: Use pre-calculated capacity values

2.3.2 Unit Conversions:

- **CALCE**: mA → A (÷1000), mV → V (÷1000)
- **HUST**: mA → A (÷1000)
- **RWTH**: Time conversion from hours to seconds
- **NEWARE**: Configurable scaling factors

2.3.3 Cycle Index Organization:

- **CALCE**: Reorganizes cycle indices to be sequential
- **HNEI, UL-PUR**: Adjusts cycle numbers after skipping initial cycles

2.4 Missing Data Handling (NaN Values)

2.4.1 Forward/Backward Fill:

- **NEWARE**: Uses `ffill()` and `bfill()` for internal resistance data
- **ARBIN**: Similar approach for missing internal resistance values

2.4.2 Imputation Strategies:

- **HNEI, SNL, UL-PUR**: Forward imputation for missing cycles - uses data from the next available valid cycle
- **MATR**: Complex data merging between batches for cells that span multiple files

2.4.3 Data Validation:

- **RWTH**: Removes time anomalies (time jumps ¿ 1e5)
- **ARBIN/NEWARE**: Logs warnings for missing cycles but continues processing

2.5 Protocol and Metadata Assignment

Each dataset assigns specific:

- **Charge/Discharge Protocols**: C-rates, voltage limits, SOC ranges
- **Battery Specifications**: Form factor, materials, capacity
- **Operating Limits**: Voltage and current limits

3 Key Preprocessing Logic Details

3.1 Robust Outlier Detection

The preprocessing uses multiple statistical methods to identify and handle outliers:

- **Hampel Filter**: $|value - median| > 3 \times median\_absolute\_deviation$
- **Median Filter**: Removes values that deviate significantly from local median
- **Rolling Window Analysis**: Compares values with neighboring cycles

3.2 Cycle Imputation Strategy

For missing cycles, the system uses forward imputation:

1. Identifies missing cycles in the sequence
2. Finds the next valid cycle with data
3. Copies that cycle's data and reassigns the cycle number
4. This ensures continuous cycle numbering

3.3 Data Quality Gates

- **Capacity Thresholds**: Remove cycles with capacity ¡ 0.1 Ah (CALCE)
- **Cycle Life Requirements**: Remove batteries that don't reach 80% capacity (MATR)
- **Temperature Validation**: Skip cycles with missing temperature data
- **Time Continuity**: Remove records with anomalous time jumps

3.4 Format Standardization

All datasets are converted to a common `BatteryData` structure with:

- Standardized column names and units
- Consistent cycle numbering
- Unified metadata format
- Pickle serialization for efficient storage

The preprocessing pipeline is quite sophisticated, handling the diverse formats and quality issues present in real-world battery testing data while maintaining data integrity and providing robust outlier detection and imputation strategies.

Dataset Coverage of Common Variables

Variable	CALCE	HNEI	HUST	MATR	OX	RWTH	SNL	UL-PUR	ARBIN	N
voltage_in_V	✓	✓	✓	✓	✓	✓	✓	✓	✓	
current_in_A	✓	✓	✓	✓	✓	✓	✓	✓	✓	
time_in_s	✓	✓	✓	✓	✓	✓	✓	✓	✓	
charge_capacity_in_Ah	✓	✓	✓	✓	✓	✓	✓	✓	✓	
discharge_capacity_in_Ah	✓	✓	✓	✓	✓	✓	✓	✓	✓	
temperature_in_C	✗	✓	✗	✓	✓	✗	✓	✓	✓	
internal_resistance_in_ohm	✗	✗	✗	✓	✗	✗	✗	✗	✓	

Answer: No, not all datasets provide all common column values.

Datasets with Complete Common Variables (7/7)

- MATR: Has all common variables including temperature and internal resistance.
- ARBIN: Has all common variables including temperature and internal resistance.
- NEWARE: Has all common variables including temperature and internal resistance.

Datasets with Most Common Variables (6/7)

- HNEI: Missing internal resistance.
- OX: Missing internal resistance.
- SNL: Missing internal resistance.
- UL-PUR: Missing internal resistance.

Datasets with Limited Common Variables (5/7)

- CALCE: Missing temperature and internal resistance.
- HUST: Missing temperature and internal resistance.
- RWTH: Missing temperature and internal resistance.

Key Observations

- **Temperature Data:** 6 out of 10 datasets have temperature measurements.
  Available: HNEI, MATR, OX, SNL, UL-PUR, ARBIN, NEWARE.
  Missing: CALCE, HUST, RWTH.
- **Internal Resistance Data:** Only 3 out of 10 datasets have internal resistance.
  Available: MATR, ARBIN, NEWARE.
  Missing: All others.
- **Core Variables:** All datasets have the essential battery cycling variables: Voltage, Current, Time, Charge/Discharge Capacity.
- **Data Quality Impact:** The missing variables affect the types of analysis that can be performed:
  - Temperature-dependent models can only use 6 datasets.
  - Internal resistance-based features can only use 3 datasets.
  - Basic voltage/current/capacity analysis can use all 10 datasets.

This variable availability pattern reflects the different experimental setups and measurement capabilities of the various research groups that generated these datasets.