Train-Test Split Analysis for BatteryML

Analysis Report

September 15, 2025

1 Base Architecture

All splitters inherit from `BaseTrainTestSplitter` which:

1. **Loads cell data paths** from directories or file lists
2. **Abstracts the split method** that returns `(train_cells, test_cells)`
3. **Handles file discovery** automatically (finds all `.pkl` files in directories)

2 Split Types by Dataset

2.1 HUST Dataset - Fixed Cell ID Split

Listing 1: HUST Split Logic

```
class HUSTTrainTestSplitter:
    train_ids = [
        '1-3', '1-4', '1-5', '1-6', '1-7', '1-8', '2-2', '2-3',
        '2-4', '2-6', '2-7', '2-8', '3-2', '3-3', '3-4', '3-5',
        # ... 53 total train cells
    ]
    # All other cells go to test set
```

Logic:

- **Predefined train set**: 53 specific cell IDs (e.g., '1-3', '2-4')
- **Test set**: All remaining cells not in train_ids
- **Split ratio**: ~53 train / ~24 test = ~69% train, 31% test

Example:

- Train: `HUST_1-3.pkl`, `HUST_2-4.pkl`, etc.
- Test: `HUST_1-1.pkl`, `HUST_1-2.pkl`, etc.

2.2 MATR Dataset - Multiple Split Strategies

2.2.1 Primary Test Split (`MATRPrimaryTestTrainTestSplitter`)

Listing 2: MATR Primary Split

```
train_ids = [
    'b1c1', 'b1c3', 'b1c5', 'b1c7', 'b1c11', 'b1c15',
    # ... 45 total train cells from batches 1-2
```

```
]
test_ids = [
    'b1c0', 'b1c2', 'b1c4', 'b1c6', 'b1c9', 'b1c14',
    # ... 46 total test cells from batches 1-2
]
# Note: b2c1 is removed as outlier
```

2.2.2 Secondary Test Split (`MATRSecondaryTestTrainTestSplitter`)

Listing 3: MATR Secondary Split

```
train_ids = [
    # Same 45 cells from batches 1-2
]
test_ids = [
    'b3c0', 'b3c1', 'b3c3', 'b3c4', 'b3c5', 'b3c6',
    # ... 44 cells from batch 3
]
```

2.2.3 CLO Test Split (`MATRCLOTestTrainTestSplitter`)

Listing 4: MATR CLO Split

```
train_ids = [
    'b4c43', 'b1c7', 'b3c9', 'b2c6', 'b2c21', 'b4c27',
    # ... 100 cells from batches 1-4 (mixed)
]
test_ids = [
    'b3c41', 'b3c22', 'b1c6', 'b3c32', 'b4c19', 'b4c22',
    # ... 100 cells from batches 1-4 (mixed)
]
```

Logic:

- **Batch-based splitting**: Different batches used for train/test
- **Outlier handling**: Problematic cells (like b2c1) are excluded
- **Multiple strategies**: Primary, Secondary, and CLO splits for different evaluation scenarios

2.3 SNL Dataset - Fixed Cell ID Split

Listing 5: SNL Split Logic

```
test_ids = [
    'SNL_18650_LFP_25C_0-100_0.5-1C_a',
    'SNL_18650_LFP_25C_0-100_0.5-2C_a',
    # ... 25 specific test cells
]
# All other cells go to train set
```

Logic:

- **Predefined test set**: 25 specific cell IDs with different chemistries and conditions
- **Train set**: All remaining cells
- **Diverse test set**: Includes LFP, NCA, NMC with different temperatures and C-rates

2.4 Combined Datasets - Fixed Cell ID Split

2.4.1 CRUH Split (CALCE + RWTH + UL-PUR + HNEI)

Listing 6: CRUH Split

```
test_ids = [
    'RWTH_015', 'UL-PUR_N15-OV3_18650_NCA_23C_0-100_0.5-0.5C_c',
    'HNEI_18650_NMC_LCO_25C_0-100_0.5-1.5C_j', 'RWTH_048',
    # ... 33 cells from all datasets
]
```

2.4.2 CRUSH Split (CALCE + RWTH + UL-PUR + SNL + HNEI)

Listing 7: CRUSH Split

```
test_ids = [
    'SNL_18650_NMC_35C_0-100_0.5-1C_c',
    'UL-PUR_N15-OV3_18650_NCA_23C_0-100_0.5-0.5C_c',
    # ... 72 cells from all datasets
]
```

2.4.3 MIX100 Split (All Datasets)

Listing 8: MIX100 Split

```
test_ids = [
    "HUST_1-1", "MATR_b3c42", "RWTH_011", "RWTH_032",
    # ... 150 cells from all datasets
]
```

Logic:

- **Cross-dataset splitting**: Cells from multiple datasets in both train and test
- **Balanced representation**: Each dataset contributes to both splits
- **Diverse test sets**: Different chemistries, temperatures, and cycling conditions

2.5 Random Split (Generic)

Listing 9: Random Split

```
class RandomTrainTestSplitter:
    def __init__(self, train_test_split_ratio=0.6, seed=0, cell_to_drop=None):
        # 60% train, 40% test by default
        # Optional cell filtering
        # Reproducible with seed
```

Logic:

- **Random splitting**: Shuffles all cells and splits by ratio
- **Configurable ratio**: Default 60% train, 40% test
- **Reproducible**: Uses seed for consistent results
- **Filtering**: Can exclude specific cells

3 Splitting Logic Details

3.1 Cell ID Extraction

Listing 10: Cell ID Extraction Examples

```
# For HUST: filename.stem.split('_')[1] -> '1-3'
# For MATR: filename.stem.split('_')[1] -> 'b1c1'
# For SNL: filename.stem -> 'SNL_18650_LFP_25C_0-100_0.5-1C_a'
```

3.2 Split Assignment

Listing 11: Split Assignment Logic

```
for filename in self._file_list:
    if filename.stem in test_ids:  # or train_ids
        self.test_cells.append(filename)
    else:
        self.train_cells.append(filename)
```

3.3 Quality Control

- **Outlier removal**: Problematic cells (like MATR b2c1) are excluded
- **Data validation**: Ensures all specified cells exist
- **Balance checking**: Verifies train/test split sizes

4 Split Statistics by Dataset

Table 1: Train-Test Split Statistics

Dataset	Train Cells	Test Cells	Split Ratio	Strategy
HUST	53	∼24	69%/31%	Fixed train IDs
MATR Primary	45	45	50%/50%	Batch-based
MATR Secondary	45	44	50%/50%	Cross-batch
MATR CLO	100	100	50%/50%	Mixed batches
SNL	∼36	25	59%/41%	Fixed test IDs
CRUH	∼75	33	69%/31%	Cross-dataset
CRUSH	∼200	72	74%/26%	Cross-dataset
MIX100	∼350	150	70%/30%	Cross-dataset

5 Key Design Principles

5.1 Reproducibility

- **Fixed splits**: Same cells always in train/test
- **No randomness**: Deterministic assignment
- **Version control**: Split definitions are code, not random

5.2 Realistic Evaluation

- **Temporal separation**: Different batches for train/test (MATR)
- **Condition diversity**: Different cycling conditions in test set
- **Chemistry variety**: Multiple battery types in test set

5.3 Fair Comparison

- **Consistent splits**: Same splits used across all models
- **Balanced representation**: Each dataset contributes to both splits
- **Outlier handling**: Problematic cells are excluded

5.4 Cross-Dataset Generalization

- **Multi-dataset splits**: Train on some datasets, test on others
- **Domain adaptation**: Test model generalization across different battery types
- **Real-world simulation**: Mimics real deployment scenarios

6 Example: HUST Split in Action

Listing 12: HUST Split Example

```
# Input: Directory with 77 HUST cells
# HUST_1-1.pkl, HUST_1-2.pkl, ..., HUST_10-8.pkl

# Process:
train_cells = []
test_cells = []

for filename in ['HUST_1-1.pkl', 'HUST_1-2.pkl', ...]:
    cell_id = filename.stem.split('_')[1]  # '1-1', '1-2', etc.

    if cell_id in ['1-3', '1-4', '1-5', ...]:  # 53 predefined IDs
        train_cells.append(filename)
    else:
        test_cells.append(filename)

# Result:
# train_cells: 53 files (HUST_1-3.pkl, HUST_1-4.pkl, ...)
# test_cells: 24 files (HUST_1-1.pkl, HUST_1-2.pkl, ...)
```

7 Summary

This splitting strategy ensures **reproducible, realistic, and fair evaluation** of battery degradation models across different datasets and experimental conditions. The approach balances:

- **Consistency**: Same splits across all experiments
- **Realism**: Mimics real-world deployment scenarios

- **Fairness**: Balanced representation of different battery types
- **Generalization**: Cross-dataset evaluation capabilities

The train-test split system provides a robust foundation for evaluating battery degradation models across diverse experimental conditions and battery chemistries.