Dataset Documentation

Solar-Battery Node AI

Power Health & Vision Anomaly Detection

AEROO Space AI Competition 2026

Talgar Private Boarding School No. 1

Dataset Version: 1.0 | Last Updated: January 2026

1. NASA Battery Degradation Dataset

1.1 Overview

Source: NASA Ames Prognostics Center of Excellence

Purpose: Train machine learning models for battery Remaining Useful Life (RUL) prediction

Access: https://www.kaggle.com/datasets/patrickfleith/nasa-battery-dataset

1.2 Dataset Characteristics

Property	Value
Total Cells	124 Li-ion cells (18650 format)
Cell Chemistry	Lithium-ion ($LiCoO_2$ cathode, graphite anode)
Nominal Capacity	2.0 Ah (rated)
Voltage Range	2.7V - 4.2V
Test Duration	2008-2018 (10 years)
Total Measurements	~168,000 charge-discharge cycles
Temperature Conditions	24°C, 43°C, 4°C (controlled chambers)
Data Format	CSV, MAT (MATLAB)

1.3 Features (Input Variables)

Feature	Unit	Range	Description
Voltage	V	2.7 - 4.2	Terminal voltage during operation
Current	A	-2.0 - 2.0	Charge current (+) or discharge current (-)
Temperature	°C	4 - 45	Cell surface temperature
Cycle Count	-	0 - 2000	Number of complete charge-discharge cycles
Capacity Fade	%	0 - 80	Percentage loss from nominal 2.0 Ah capacity
Impedance	mΩ	15 - 120	Internal resistance (DC measurement)

1.4 Target Variable

Remaining Useful Life (RUL):

- **Definition:** Percentage of battery life remaining before reaching end-of-life threshold (80% capacity)
- **Calculation:** RUL = 100 × (Current Capacity - 0.8 × Nominal Capacity) / (Nominal Capacity - 0.8 × Nominal Capacity)
- **Range**: 0-100%
- **Interpretation**:
  - 100%: Brand new cell
  - 70-100%: Excellent health
  - 40-70%: Good health
  - 20-40%: Fair health (replacement recommended)
  - 0-20%: Critical (imminent failure)

1.5 Data Preprocessing

```
# Data loading and preprocessing import pandas as pd from sklearn.preprocessing
import StandardScaler # Load CSV df = pd.read_csv('nasa_battery.csv') # Remove
outliers (IQR method) Q1 = df.quantile(0.25) Q3 = df.quantile(0.75) IQR = Q3 - Q1
df = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axis=1)] # Feature
normalization scaler = StandardScaler() features = ['voltage', 'current',
'temperature', 'cycle_count', 'capacity_fade'] df[features] =
scaler.fit_transform(df[features]) # Train-test split (80/20) from
sklearn.model_selection import train_test_split X_train, X_test, y_train, y_test =
train_test_split( df[features], df['rul'], test_size=0.2, random_state=42 )
```

1.6 Usage in Project

1. **Model Training:** Random Forest Regressor trained on 124 cells × ~1,350 cycles each = ~167,400 samples
2. **Validation:** 5-fold cross-validation with shuffle
3. **Test Performance:** 92.1% accuracy, MAE = 3.8%
4. **Deployment:** scikit-learn model serialized as pickle file, loaded by Flask backend

2. Roboflow Solar Panel Anomaly Dataset

2.1 Overview

Source: Roboflow Universe - Solar Panel Datasets

Purpose: Train computer vision models for solar panel anomaly detection

Access: https://universe.roboflow.com/gao-shou-zheng-b6xqc/solar-panel-oswal

2.2 Dataset Characteristics

Property	Value
Total Images	2,280 annotated images
Image Resolution	Variable (640×480 to 1920×1080)
Annotation Format	YOLO bounding boxes
Classes	4 (Normal, Dust, Crack, Coverage)
Train/Val/Test Split	70% / 20% / 10%
Source Type	Drone, satellite, ground camera imagery

2.3 Anomaly Classes

Class	Sample Count	Description
Normal	1,854	Clean, fully functional solar panels
Dust	1,102	Light to moderate dust accumulation
Crack	845	Physical damage, fractures, or breakage
Coverage	326	Debris, bird droppings, or obstruction

2.4 Data Augmentation

- **Rotation:** ±15° random rotation to simulate camera angle variation
- **Flip:** Horizontal and vertical flips for spatial invariance
- **Brightness:** ±25% adjustment to simulate lighting conditions
- **Contrast:** ±20% adjustment for varying weather
- **Noise:** Gaussian noise (σ=0.01) to simulate camera sensor imperfections
- **Crop:** Random 90% crops to focus on panel regions

> **Note**: All augmentations applied only to training set, not validation/test sets, to prevent data leakage.

2.5 Model Training & Deployment

```
# YOLOv8 training with Roboflow from ultralytics import YOLO # Load pretrained
model model = YOLO('yolov8n.pt') # Train on solar panel dataset results =
model.train( data='roboflow_solar.yaml', epochs=100, imgsz=640, batch=16 ) # Export
for STM32 NPU deployment model.export(format='tflite', int8=True, nms=True)
```

1. **Training:** YOLOv8n model trained for 100 epochs on Google Colab (Tesla T4 GPU)
2. **Quantization:** INT8 post-training quantization for STM32 NPU
3. **Optimization:** STEdgeAI Core v2.2.0 for NPU deployment
4. **Inference:** <100ms on STM32N6570-DK hardware

3. Dataset Summary

Both datasets provide complementary capabilities for the Solar-Battery Node AI system:

Dataset	Samples	Purpose	Performance
NASA Battery	168K cycles	Battery RUL prediction	92% accuracy
Roboflow Solar	2,280 images	Panel anomaly detection	<100ms inference

Dataset Documentation - Solar-Battery Node AI