INCOSE

WESTERN STATES REGIONAL CONFERENCE

Digital Twin Framework for Lifetime Predictions of Li-ion Batteries

INCOSE WSRC · 2025

AUTHORS

Jack Cavaluzzi, Kaled Houck, Porter Zohner, and Sonali Sinha Roy

PUBLICATION DATE

July, 2026

SUBMISSION #

21

INCOSE PUBLICATIONS

Digital Twin Framework for Lifetime Predictions of Li-ion Batteries

Jack Cavaluzzi
Idaho National Laboratory
PO Box 1625, Idaho Falls, ID 83401
(940) 206-9729
Jack.Cavaluzzi@inl.gov

Kaleb Houck
Idaho National Laboratory
PO Box 1625, Idaho Falls, ID 83401
(208) 716-3561
Kaleb.Houck@inl.gov

Porter Zohner
Idaho National Laboratory
PO Box 1625, Idaho Falls, ID 83401
(208) 526-3328
Porter.Zohner@inl.gov

Sonali Sinha Roy
Idaho National Laboratory
PO Box 1625, Idaho Falls, ID 83401
(208) 526-4091
Sonali.SinhaRoy@inl.gov

Abstract

Predicting battery lifetime is essential to ensure the reliability and economic viability of grid-scale energy-storage systems. Traditional accelerated testing methods offer only partial insight into battery degradation, especially when applied at the cell or module level, where they often ignore key system-level interactions. This paper presents an early-stage digital-twin framework developed as part of the Rapid Operational Validation Initiative effort sponsored by the United States Department of Energy, Office of Electricity. The framework integrates real-time battery-experiment data, physics-based simulations, and machine-learning models within a machine-learning operations pipeline to support the continuous prediction of battery health. The authors describe how synthetic data generated from the Python Battery Mathematical Modeling electrochemical model is used to bootstrap training, in addition to how the experimental data from the Idaho National Laboratory experimental testbeds are transformed for use in predictive models. The framework leverages Airflow for orchestration, MLflow for model tracking, and a DeepLynx digital-thread backend for system integration and visualization. Though model evaluation is ongoing, this paper outlines the system architecture, methodological rationale, and deployment strategy, and proposes two use cases that will validate the framework under both simulated and live conditions. This work contributes a reproducible, modular approach to artificial intelligence-enabled battery monitoring and serves as a foundation for future applied research.

Keywords

digital twin, artificial intelligence, batteries, accelerated testing

Introduction

The development of long-duration energy-storage (LDES) systems has accelerated rapidly in recent years, with many new technologies progressing

toward commercial deployment. These systems include lithium-ion (Li-ion) batteries, electrolyte-flow batteries, and other emerging technologies such as iron-air and thermal storage systems. First-of-a-kind LDES systems must demonstrate that the operational lifetimes of these new technologies are sufficient to ensure economic viability, which presents significant validation challenges.

For Li-ion battery systems, validation typically involves accelerated aging tests in which cells or packs are subjected to continuous charge-discharge cycles under adverse conditions. These tests aim to estimate the remaining useful life of a system by accelerating degradation. However, Li-ion systems are composed of individual cells, assembled into modules and packs, and further integrated with battery-management and thermal-control systems. While accelerated testing can be conducted on cells and packs, applying it to full-scale systems is prohibitively expensive and logistically complex. As a result, these tests consistently fail to capture such important system-level interactions as thermal behavior or operational dynamics.

Emerging software-based approaches offer a potential alternative to traditional validation methods. Tools, such as physics simulation, artificial intelligence (AI), and digital-twin technology, could provide asset-specific predictions of lifetime and performance. However, deploying these tools effectively will require a cohesive, scalable framework that supports rapid validation across diverse energy-storage technologies.

To address this need, the United States (U.S.) Department of Energy, Office of Electricity (DOE-OE) launched the Rapid Operational Validation Initiative (ROVI), which tasks six national laboratories with developing tools and frameworks for the validation of new energy-storage systems. A key component of ROVI is a digital-twin platform developed by the Idaho National Laboratory (INL). A digital twin is defined as "a set of virtual information constructs that mimic the structure, context, and behavior of a natural, engineered, or social system (or system of systems), is dynamically updated with data from its physical twin, has a predictive capability, and informs decisions that realize value. The bidirectional interaction between the virtual and the physical is central to the digital twin."

As part of this effort, INL operates an accelerated-testing test bed containing 35 Li-ion cells under a variety of test conditions. This test bed provides a unique opportunity to apply digital-twin methods across the entire life cycle of the assets being tested. Each of the 12 defined test conditions includes three cells—with one condition having two—thereby allowing for a comparative study under controlled experimental variability.

The motivation behind this work is to develop a digital-twin framework capable of predicting the state of health (SoH) of Li-ion batteries in realistic operational settings. The SoH represents the remaining capacity and performance of a battery, which naturally degrades over time. While the literature contains proposed digital-twin frameworks and machine-learning (ML) models trained on public datasets, these datasets are typically based on full charge-discharge cycles and are limited to only those cells tested to end-of-life under laboratory conditions. In contrast, real-world systems operate under partial cycling and variable loading, making the applicability of these datasets limited.

The framework presented in this paper is designed to overcome these limitations. It supports the creation of asset-specific digital twins for each cell being tested, combining real-time data streams and physics simulations tailored to the test conditions of each individual cell, and AI models trained initially on simulated data and updated throughout the test. Using this method allows us to evaluate whether AI models, informed by physics and updated in real-time, can produce meaningful predictions throughout an asset's life without requiring extensive historical data.

Methods

System Architecture

The digital-twin framework developed under the DOE ROVI project integrates an experimental

battery testing infrastructure with an industrial control system (ICS), a high-performance ML graphics processing unit (GPU) hosted on a government Azure demilitarized zone (DMZ), and orchestrated model-management services across segmented INL network environments. This architecture is designed to support automated, reproducible workflows for training and deploying ML models that estimate battery SoH and remaining useful life based on real-time data.

The system is composed of three primary components:

1. Experimental data-acquisition (DAQ) infrastructure (ICS network)
2. Model training and inference environment (GPU compute node)
3. Digital-twin service infrastructure (DMZ network).

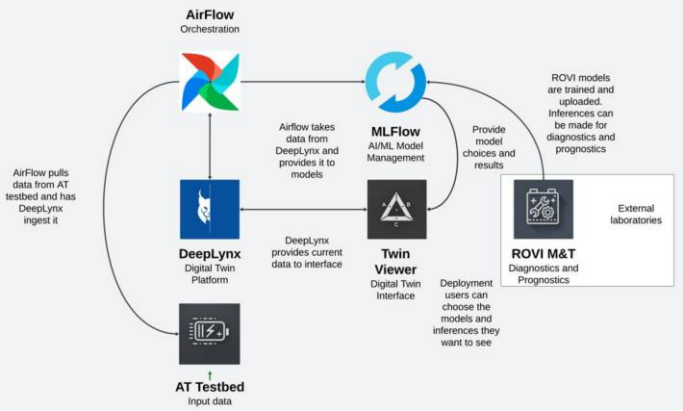


Figure 1. Digital-twin architecture used with accelerated testing.

As indicated in Figure 1, each component is connected through firewalled networks and orchestrated using a combination of automated workflows, model-management systems, and graph-based data stores.

At the DAQ layer, Li-ion battery experiments are conducted using a dedicated ICS with DAQ hardware. The DAQ system collects voltage, current, temperature, and other operational parameters at high-frequency (on the order of seconds) and stores these data on a Samba network share. This share serves as the transfer point between the isolated ICS network and the ML processing environment. The compute environment is a Linux-based GPU workstation equipped with dual NVIDIA RTX 4090 GPUs; this GPU compute system is co-located on the ICS network and hosts the Samba share where the DAQ results are sent.

To enable continuous training, a Samba share-file sensor, which was developed for this project as a custom Apache Airflow operator ("Apache Airflow," 2023), monitors the DAQ share for new experimental data. When new files are detected, they are automatically moved to a staging area on the GPU server. Once a predefined threshold of new data is available, an Airflow directed acyclic graph (DAG) is triggered. This DAG first saves the incoming data to DeepLynx, ensuring that experimental observations are stored with full context within the project's digital-thread repository. DeepLynx (Ritter, Darrington, and Browning, 2020) is a data-warehouse platform developed by INL for digital-twin applications that provides data management for both graph- and time-series data sources. The DAG then launches a training script that fits an ML-model architecture to the new data. Upon completion, the trained model, associated hyperparameters, and training data snapshot are logged in MLflow ("MLflow," 2023) to ensure reproducibility and traceability. When predictions are needed to assess future degradation under varying operational conditions, Airflow retrieves the trained model from MLflow and executes a prediction script on the GPU server. Then, the resulting forecast and evaluation metrics are stored back in both MLflow and DeepLynx, maintaining a complete chain of inference history.

The orchestration, model management, and visualization services reside on a separate INL DMZ network, which is segmented from the ICS network for cybersecurity and operational reliability. This DMZ hosts the central Airflow instance, the MLflow tracking server, a DeepLynx instance, and a web-based Digital-Twin Viewer application.

DeepLynx plays a central role in structuring and contextualizing data across the system. Battery-test assets, such as individual cells, cell packs, or larger system modules, are represented as nodes in a semantic-graph database. For instance, in the

experiments involving all 35 parallel test cells, each cell is categorized as a unique "BatteryCell" node in DeepLynx. Relationships between these nodes are encoded as edges to other nodes in the graph. DeepLynx also serves as a record system for both the raw time-series data and the ML-generated predictions. Time-series records are organized into distinct data sources per cell or test group and are linked to their corresponding asset nodes, supporting efficient querying and cross-referencing data for historical analysis.

Bootstrapping with Physics-Based Models

One of the primary challenges in developing a digital-twin model for a first-of-a-kind (FOAK) battery system is the absence of sufficient historical data to train predictive mockups. In conventional data-driven approaches, model performance is often constrained by the quantity and diversity of prior observations. However, in early-stage experimental systems—particularly those involving new chemistries or configurations—such data are typically unavailable. To address this challenge, the authors have adopted a hybrid approach that leverages physics-based simulation to bootstrap the ML-training process.

The authors use the Python Battery Mathematical Modeling package (PyBaMM) as the core simulation engine ("PyBaMM," 2025). PyBaMM is an open-source framework that provides a flexible interface for simulating Li-ion batteries based on a range of electrochemical and degradation models. It supports fine-grained control over model structure, parameter selection, solver behavior, and output handling, making it well-suited for research applications that integrate physics-based and data-driven workflows. In this framework, PyBaMM is used to simulate synthetic-degradation cycles under the conditions defined in the test plan. The simulations are used to generate a full time series of battery behavior, including capacity loss, internal resistance, and Li-inventory depletion. These outputs are then used as an input to the ML models, seeding them with a representative baseline of how a battery is expected to behave over time, even before real-experimental data become available.

Model and Parameterization

The simulation model employed in this work is based on the Doyle-Fuller-Newman electrochemical model (Doyle, Fuller, and Newman, 1993), which solves coupled partial-differential equations governing Li-ion concentration, reaction kinetics, and electrical behavior across the battery's anode, cathode, and separator regions. The specific implementation used here is adapted from PyBaMM's basic_dfn.py and has been parameterized for LiFePO4 (LFP) chemistry.

The baseline parameter set is drawn from Prada et al. (2013), which is widely used for modeling LFP cells. To enhance the model's ability to simulate long-term degradation behavior, these parameters are supplemented with additional terms from O'Kane et al. (2022). This extension enables the inclusion of side reactions, such as solid-electrolyte interphase (SEI) formation and other degradation mechanisms not present in the original Prada set. A list of all added parameters is included in the documentation planned to be open sourced at a later date (e.g., added_okane_params.txt).

In addition, the authors applied a set of custom updates through a dictionary override (i.e., my_custom_updates) that adjusts temperature, voltage cutoffs, and cell scale factors to better match the intended operating conditions. This allows the simulation to reflect the capacity and cycling behavior of test assets more realistically.

Covariates and Quantities of Interest

From the PyBaMM-generated results, a series of covariates were extracted that serve as input features for ML-training systems. These include both direct measurements and model-derived quantities:

- Cycle
- Step
- Terminal voltage (V)
- Current (A)
- Resistance (Ohm)

- Discharge capacity (A.h)
- Discharge voltage (W.h)
- Li-inventory loss (%)
- Local equivalent circuit models resistance (Ohm).

The quantity of interest (QoI) for prediction is the battery's remaining capacity when fully charged, representing its SoH over time. However, as is common in battery-health estimation, accurately defining and predicting capacity presents challenges. In real experiments, a battery's full-discharge capacity is typically known only at the end of a complete cycle, which complicates efforts to train models on per-timestep data. To address this, PyBaMM's built-in "Capacity" summary variable was used, which reports the capacity at the end of each cycle. Then, these values are interpolated to estimate a timestep-resolved capacity trend. While this provides a usable continuous target for training, it also introduces potential inaccuracies during partial cycles; this is a known limitation in the field and will be investigated further during planned future work.

Limitations and Future Validation

While the use of PyBaMM provides a powerful means to bootstrap training and ensure structural consistency in input features, the fidelity of the simulated data remains an open question. The current parameterization has not yet been fully validated against the real experimental results. Validation of the simulation model, particularly its ability to reproduce observed degradation trends, is a focus of future work. This includes comparing PyBaMM-generated capacity trajectories to experimental measurements from INL's testbed, and iteratively refining parameters to improve physical realism.

Nonetheless, the bootstrapping approach provides a foundation for enabling early digital-twin functionality, allowing model orchestration, data flow, and ML-training infrastructure to be tested and refined prior to the availability of live experimental data.

ML Models and MLOps Pipeline

Transformer-Based Time-Series Models

To support continuous, data-driven prediction of battery degradation over time, the MLOps pipeline used here incorporates modern deep-learning models that are well-suited for multivariate, temporal forecasting. Specifically, two transformer-based architectures are implemented via the open-source Darts Python library (Herzen et al., 2022): (1) the Temporal Fusion Transformer (TFT), and (2) Time-series Dense Encoder (TiDE). Both models are designed to handle multivariate inputs, temporal dependencies, and missing data, thus making them good candidates for real-world experimental systems where signals may be noisy, delayed, or intermittently missing.

TFT (Lim, Ank, Loeff, and Pfister, 2021; "Temporal Fusion Transformer (TFT)," 2021) is a hybrid architecture that combines recurrent layers with self-attention mechanisms. It was originally developed for interpretable multi-horizon forecasting and can capture both short- and long-term temporal dependencies in sequential data. TFT incorporates gating mechanisms and variable selection networks, enabling it to weigh the relevance of covariates over time and across features. This is particularly valuable in the setting used here, where the importance of variables such as current, temperature, or internal resistance may evolve as the battery degrades.

TiDE ("Time-series Dense Encoder (TiDE)," 2021) is a newer model that avoids recurrence entirely and instead uses stacked fully connected layers and temporal embeddings. TiDE is optimized for scalability and fast inference, making it suitable for real-time-prediction scenarios. While it lacks the interpretability features of TFT, it performs well on large datasets and has shown promise in forecasting tasks with complex seasonal or long-range patterns.

Both models are trained using simulation-bootstrapped or experimental data within the orchestrated MLOps pipeline. They accept a rich set of covariates derived from battery data as inputs—including current, voltage, resistance, and Li-inventory estimates—and outputs forecasts of battery-health indicators, such as remaining capacity. These two models represent the initial exploration of transformer-based forecasting approaches based on

early scoping work. However, further validation is required to assess their long-term suitability for battery-health prediction. The pipeline has been designed deliberately to be modular and extensible, allowing additional models to be integrated and benchmarked as the system matures, and more real-world data become available.

MLOps Orchestration with Airflow and MLflow

To support continuous, automated training and deployment of battery-health-prediction models, the authors have implemented a modular MLOps pipeline using Apache Airflow for orchestration and MLflow for experiment tracking and artifact management. This setup enables reproducible training, scalable deployment, and secure integration across INL's segmented-network architecture.

The pipeline begins with a custom Airflow file sensor that monitors a Samba share and a method of filesharing on Unix/Linux systems, which are exposed by the DAQ system. When new battery-experiment data are detected, they are moved to a staging area on the GPU server. Once sufficient new data are accumulated, a training DAG is triggered. This modular DAG structure allows for flexible chaining of tasks, making it easy to adapt the pipeline to different data sources or model types in future use cases.

Training tasks are executed remotely on the GPU node via Airflow's Secure Shell (SSH) and Secure File Transfer Protocol (SFTP) operators. Training scripts are transferred as part of the workflows and executed in a controlled Python environment activated via Conda ("Conda," 2022). Conda is a command-line tool from the open-source Anaconda data science and AI-distribution platform. These scripts preprocess the data, construct Darts Time-series objects, and fit a transformer-based model (e.g., TiDE or TFT) to the prepared input data. The training configuration—including covariates, time resolution, model hyperparameters, and chunk lengths—is passed via Airflow parameters to ensure full traceability.

Once training is complete, the trained model is serialized and logged to MLflow using a custom Python wrapper that conforms to the pyfunc interface in MLflow. In addition to the model itself, a variety of artifacts are logged, including processed training/validation data, covariate transforms and scalers, model hyperparameters, Conda environment, and all the Python code required to reproduce the training. This MLflow integration allows the trained models to be versioned, deployed, and queried independently of the orchestration pipeline, and supports downstream-batch or real-time inference workflows.

Prediction workflows are also managed by Airflow. Once a trained model is registered in MLflow, it can be pulled back to the GPU server using MLflow's model registry Application Programming Interface (API) and used for inference on the new input data. The resulting predictions are saved as Comma Separated Value (CSV) files and stored in MLflow. If the validation series are available, the evaluation metrics mean absolute error, root mean-square error, and symmetric mean absolute-percentage error are computed and saved to MLflow as well. Another DAG then uploads these prediction CSV results and attaches them to the corresponding battery asset nodes in the DeepLynx graph.

The integration of Airflow and MLflow allows for repeatable, end-to-end ML cycles. These include data detection and transformation, training, model logging, and predictive output, all while keeping workflows modular and inspectable. This design ensures that future model variants or retraining triggers can be easily integrated without restructuring the entire system.

Visualization and Digital-Thread Framework

DeepLynx is an open-source data-warehouse platform developed and maintained by INL. It serves as both a semantic-graph database and a time series data store. In the ROVI project, DeepLynx is used to store battery metadata, simulation outputs, model predictions, and training artifacts. This structure enables meaningful linkage between the individual

battery assets and their associated time-series data sources throughout their life cycles. DeepLynx also serves as the backend for the visualization interface. As shown in Figure 2, each blue node in the graph represents a battery cell while the green nodes represent time-series data sources. These sources are linked to their corresponding cells via edge relationships and may represent either real-time measurements or predictions.

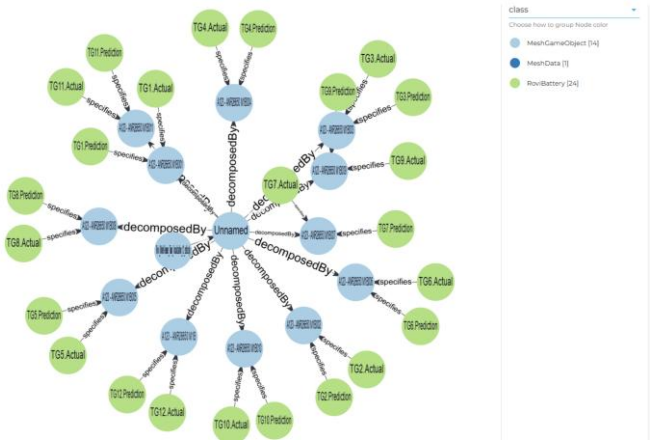


Figure 2. Semantic-graph data in DeepLynx used to organize simulated ROVI battery data.

To visualize these data, the authors developed a custom web-based application, the Digital-Twin Viewer, which acts as the primary user interface for exploring battery state and model predictions. Originally developed for another INL project, the viewer has been extended to support ROVI-specific features. It reads directly from the DeepLynx graph and presents battery data in an integrated interface capable of three-dimensional (3D) and two-dimensional (2D) visualization. As shown in Figure 3, the right pane provides a 3D view of the battery system, where users can view prediction results directly on the visual model. The left pane shows the detailed 2D time-series plots. When a user selects the "Informative" interaction mode and clicks a specific battery cell, the associated measured (or simulated) capacity data are plotted. In the 3D view, the predicted capacity values after 90 days are shown in blue while the actual values from the PyBaMM simulation are shown in green. This side-by-side comparison allows users to assess forecast performance and system status at the individual-cell level. The forecast window will be configurable in future versions.

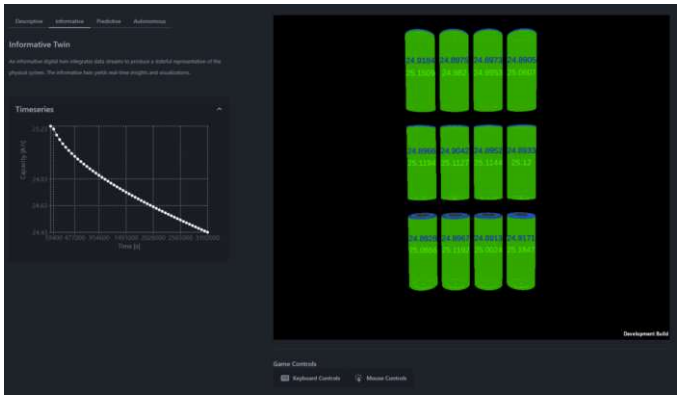


Figure 3. Screenshot of Digital-Twin Viewer Webapp showing battery simulated capacity in green and predicted capacity in blue.

Results

Implemented Architecture

The digital-twin framework described in this paper has been implemented and deployed across INL's internal ICS and DMZ network environments. Key infrastructure components, including the DAQ interface, GPU-based ML-training node, Airflow orchestration engine, MLflow model registry, and DeepLynx data graph, are all operational.

End-to-end workflows for model training and inference have been tested using simulated data. The authors expect to begin testing soon with the preliminary experimental data. The system can detect new data arriving via a Samba share, stage it for processing, and trigger a DAG in Apache Airflow that initiates model training on the GPU server. Trained models are logged using a custom MLflow wrapper that captures not only the trained model itself but also associated training scripts, covariates, scalers, input time series, and required Python packages via a Conda environment—everything needed to reproduce the training.

Initial training has been performed using synthetic data generated with the PyBaMM simulation package. Trained models, including transformer-based architectures such as TiDE and TFT, have been serialized, versioned, and stored successfully in

MLflow. Inference routines have also been validated using early experimental datasets. The resulting predictions are routed through Airflow and stored in DeepLynx for further analysis and visualization. While these results confirm the technical viability of the system architecture and integration, a full evaluation of model performance, sensitivity, and forecasting reliability remains part of the planned future work.

Preliminary Results

To generate the training and evaluation data, PyBaMM was used to simulate battery-degradation experiments for 12 distinct test groups, each representing different operating conditions. These simulations span a 12-month duration for each group. The synthetic data were uploaded to DeepLynx, where they were linked to the corresponding ROVIBattery nodes in the DeepLynx graph, as shown previously in Figure 2. Approximately 70% of the simulation data were used for model training. The remaining 30% was saved for evaluation and partial use as future covariates during prediction. A single transformer-based model is trained using this combined data and then applied to each test group individually to generate a set of 12 forecasts. Both the TFT and TiDE models were trained and evaluated in this workflow, and each model's output includes probabilistic-forecast intervals. For each group, prediction plots were produced that include forecasted trajectories and corresponding error bounds.

The training and prediction process is orchestrated entirely using Apache Airflow, with support for data movement, environment management, model execution, and result integration. The key DAGs involved in this process are:

deeplynx_time series_to_training_box

This DAG downloads the PyBaMM simulation data from DeepLynx and transfers it via Airflow's SFTP operator to the GPU training server. This DAG assumes the simulation results are already stored in DeepLynx and correctly linked to the ROVIBattery nodes within the system graph.

conda_packer and unpack_conda_env

Because the GPU server is isolated from the Internet, a two-step DAG process is used to create a portable Conda ("Conda," 2022) environment archive (i.e., tar.gz) on a connected machine and then unpacked on the GPU server to reproduce the software environment for training and inference.

pybamm_darts_fit

As shown in the workflow graph presented in Figure 4, this DAG transfers the selected model wrapper and training script for either TFT or TiDE to the GPU server using SFTP. It then launches the training process via SSH. Upon completion, the model, training dataset, hyperparameters, and scalers are logged to a remote MLflow instance, which serves as the centralized model registry. A screenshot of the MLflow logged model, execution environment, and artifacts is shown in Figure 5.

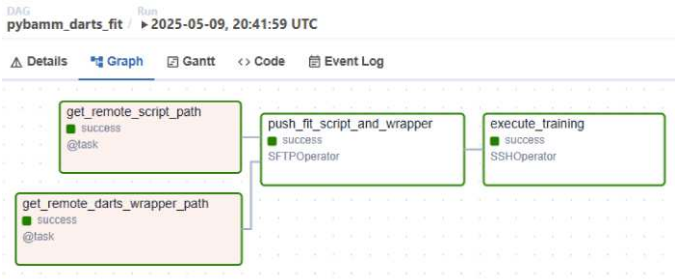


Figure 4. PyBaMM workflow DAG for model transfer, training, and logging.

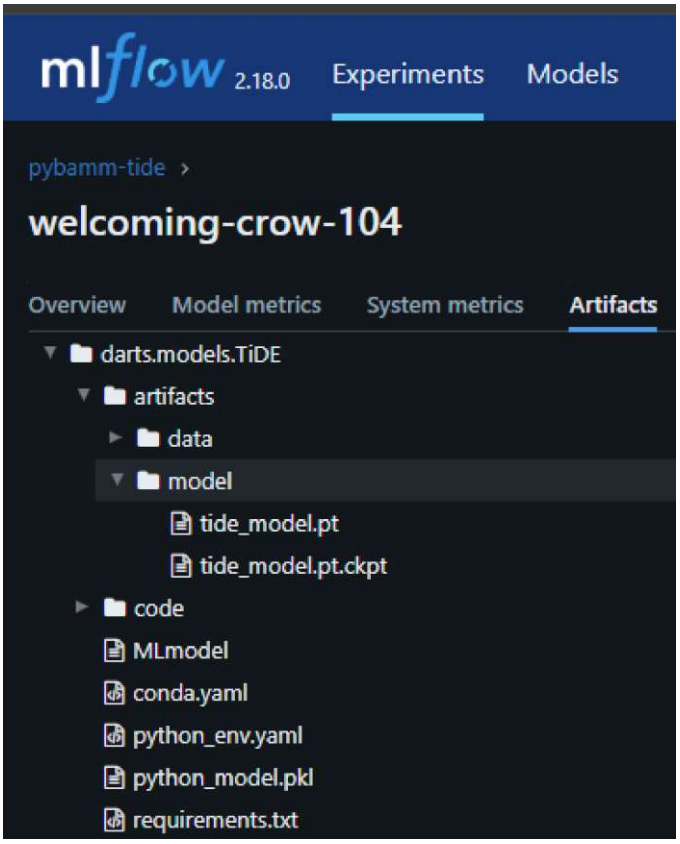


Figure 5. MLflow logged artifacts, including the model, training dataset, hyperparameters, and scalers.

pybamm_darts_predict_to_deeplynx_time series

This DAG initiates the prediction phase. It transfers a prediction script to the GPU server, which then pulls the trained model and data from the relevant MLflow run. The script generates predictions for each of the 12 test groups, saving the CSV outputs, visual plots, and evaluation metrics to MLflow. The CSV files are saved locally on the GPU server as well. Airflow then retrieves these prediction CSV files, uploads them to DeepLynx, and attaches them to the appropriate ROVIBattery nodes to maintain a complete data lineage

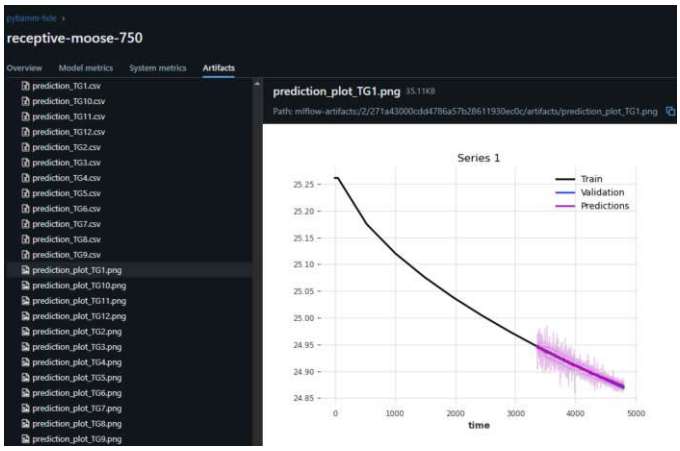


Figure 6. MLflow prediction run showing logged artifacts.

Discussion

Design Rationale and Systems-Engineering Considerations

The ROVI digital-twin framework was designed with an emphasis on modularity, traceability, and extensibility, in alignment with core systems-engineering principles. Its architecture separates orchestration, model management, and metadata integration into independent components, allowing individual modules to be developed and tested without requiring changes to the entire system. Airflow was selected for its ability to manage multistep workflows across diverse systems and interfaces. MLflow provides structured versioning for models and their associated artifacts while DeepLynx enables semantic modeling of experimental assets and data through a graph-based digital-thread framework. Together, these tools provide a transparent and reproducible infrastructure to manage ML development and deployment.

A central design feature is the use of simulation to address data scarcity at the beginning of experimental campaigns. By using physics-based models in PyBaMM to generate synthetic training data, the system can be brought online immediately, even before any real battery-degradation data are collected. This supports early-model training and enables full-pipeline testing from the outset. Unlike many digital-twin approaches that rely on large-historical

datasets, the ROVI framework is designed for deployment in data-limited environments. It supports real-time initialization and continuous improvement, making it well-suited to FOAK systems that lack legacy data.

From a systems-engineering perspective, this work highlights the importance of coordinated orchestration, metadata management, and flexible-system integration. Airflow ensures workflows are inspectable and auditable, while DeepLynx maintains relationships between the physical assets and their associated data, preserving semantic context across the pipeline. This architecture aims to support long-term scalability. Although the current implementation operates at the cell or test-group level, the metadata model and retraining infrastructure are built to be generalized to higher levels of integration, including battery modules, packs, and ultimately full-scale grid systems.

Future Work and Planned Validation Use Cases

While the current architecture supports training, inference, and logging of transformer-based models such as TiDE and TFT, a major next step is to broaden the scope of model exploration and evaluation. The pipeline has been intentionally designed to be modular and model-agnostic, allowing for the integration of alternative ML-model architectures to serve as baselines. Part of this future work will include work done by other national laboratories, such as Argonne with their Moirae collection of tools ("Moirae," 2025).

In addition to experimenting with different model types, future work will involve systematic-hyperparameter tuning. This will include evaluating variations in model parameters like forecast lengths, hidden-layer dimensions, dropout rates, and quantile targets. Another key direction is the examination of which covariates contribute most meaningfully to accurate predictions. The role of past covariates, such as internal resistance, current, voltage, or Li-inventory loss, must be rigorously evaluated across use cases. The authors also intend to determine whether future covariates, such as expected load conditions or projected ambient temperature, can be estimated reliably enough to improve long-term-forecast performance.

Perhaps most critically, future work will focus on selecting and validating appropriate QoIs for prediction. Traditional metrics, such as state of charge and SoH, are difficult to define with precision, even in well-instrumented laboratory experiments. The current approach interpolates battery capacity across timesteps based on PyBaMM cycle-level output, but this introduces uncertainty. The authors will assess whether other variables, such as resistance growth, total energy throughput, or degradation-mode indicators like SEI-layer thickness, can serve as more-stable and informative targets for the ML models.

Validation Strategy Using the Nissan Leaf Dataset

To assess the effectiveness of this digital-twin framework relative to more-traditional-modeling methods, the authors will conduct a controlled validation study using the publicly available Nissan Leaf battery dataset (INL Nissan Leaf–Pack Testing—DOE-VTO Battery Hub, 2019). This dataset includes real-world lifetime cycling data and offers a suitable benchmark for evaluating prediction quality over extended-time horizons. The validation will consist of four phases:

1. **Simulation Bootstrapping:** The authors will generate an initial synthetic training set using PyBaMM, simulating early battery degradation based on the chemistry and operational conditions relevant to the Leaf dataset. This synthetic data will serve as the starting point for model training, emulating the initialization of a digital twin prior to the availability of any real-world data.

2. **Incremental Streaming:** The actual Leaf dataset will be introduced to the system in staged batches. These batches will be released at fixed intervals to mimic live experimental-data collection. During this phase, the model will be retrained periodically, reflecting the operational

behavior of a digital twin deployed in a real-time environment.

3. **Parameter Adaptation:** As retraining progresses, the authors will examine how adjustments to training frequency, input-chunk length, and other hyperparameters affect predictive performance. This will help the authors understand whether retraining based on partial-lifetime data can be tuned effectively to maintain accuracy as new data arrive.

4. **Comparative Evaluation:** The authors will compare the performance of the digital-twin approach against the control scenarios. One control will consist of a TFT model trained once on the entire Leaf dataset without incremental updates, representing a conventional offline-modeling strategy. Another control will involve using the PyBaMM simulation alone, without any ML, representing a purely physics-based prediction approach.

Through this structured experiment, the authors aim to evaluate whether a physics-informed ML system can provide reliable predictions even when trained only on partial data, and whether it improves over time through retraining. The authors also seek to understand whether such a system can generalize to unseen future conditions more effectively than static models. This evaluation will help identify the advantages and limitations of live retraining workflows and inform how the system is deployed on INL's real-time-battery test beds. Additionally, this approach will mimic the work that needs to be done to create digital-twin models of the test articles at the accelerated-testing test bed. Ultimately, the results will contribute to the development of scalable, adaptive digital twins that can support long-term battery-health management across diverse system configuration.

Live Deployment in INL Battery Test Bed

The second use case involves full deployment in an INL experimental facility. Real-time-sensor data will be ingested from the DAQ system and processed through the pipeline in its live configuration. The digital twin will perform periodic retraining, log updated models to MLflow, and generate SoH predictions. These predictions will be stored in DeepLynx and visualized using the Digital-Twin Viewer. This deployment is expected to surface practical integration issues, such as retraining latency, system robustness under varying test conditions, and long-term data reliability. These insights will help guide the future evolution of the system and inform its eventual application to larger and novel complex battery systems. Together, these validation cases represent an essential step toward assessing the generalizability, reliability, and scalability of the proposed architecture.

Conclusions

This paper presents an early-stage digital-twin framework developed for the continuous prediction of battery health and lifetime, with a particular focus on FOAK systems where historical data are limited or unavailable. By integrating physics-based simulation, modern transformer-based ML, and an orchestrated MLOps pipeline using Airflow, MLflow, and DeepLynx, the system enables reproducible, modular, and scalable battery-forecasting workflows.

The implemented architecture supports bootstrapped model training from synthetic data, periodic retraining on streaming experimental data, and data integration through a graph-based digital-thread framework. While preliminary results validate the infrastructure and workflows, future work will focus on evaluating predictive performance, refining QoIs, and comparing a broader range of forecasting models.

Planned validation studies, including a controlled test using the Nissan Leaf dataset and deployment on INL's live-battery test bed, will assess the effectiveness of continuous retraining and model-informed initialization compared to traditional methods. This work contributes to the growing field of digital twins for energy systems by demonstrating a path from system design to adaptive, data-driven forecasting for long-term asset management.

Acknowledgement

This manuscript has been authored by Battelle Energy Alliance, LLC, under Contract No. DE-AC07-05ID14517 with the U.S. Department of Energy. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for U.S. Government purposes.

References

Airflow. (2023, April 20). Retrieved May 14, 2025, from https://github.com/apache/airflow Conda. (2022, April 3). Retrieved May 14, 2025, from https://github.com/conda/conda

Doyle, M., Fuller, T. F., & Newman, J. (1993). Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell. Journal of the Electrochemical Society, 140(6), 1526. https://doi.org/10.1149/1.2221597

Herzen, J., Lässig, F., Piazzetta, S. G., Neuer, T., Tafti, L., Raille, G.,... Huguenin, N. (2022). Darts: User-friendly modern machine learning for time series. Journal of Machine Learning Research, 23(124), 1–6.

Lim, B., Ank, S. Ö., Loeff, N., & Pfister, T. (2021). Temporal fusion transformers for interpretable multihorizon time series forecasting. International Journal of Forecasting, 374, 1748–1764. https://doi.org/10.1016/j.ijforecast.2021.03.012

Moirae — Moirae documentation. (2024). Retrieved May 20, 2025 from Github.io. https://rovi-org.github.io/auto-soh/

MLflow. (2023, April 20). Retrieved May 14, 2025, from https://github.com/mlflow/mlflow

National Academies. (2024). Foundational Research Gaps and Future Directions for Digital Twins. Retrieved from National Academies: https://nap.nationalacademies.org/catalog/26894/foundational-research-gaps-and-futuredirections-for-digital-twins

O'Kane, S. E., Ai, W., Madabattula, G., Alonso-Alvarez, D., Timms, R., Sulzer, V.,... Marinescu, M. (2022). Lithium-ion battery degradation: how to model it. Physical Chemistry Chemical Physics, 24(13), 7909–7922. https://doi.org/10.1039/D2CP00417H

Prada, E., Di Domenico, D., Creff, Y., Bernard, J., Sauvant-Moynot, V., & Huet, F. (2013). A simplified electrochemical and thermal aging model of LiFePO4-graphite Li-ion batteries: Power and capacity fade simulations. Journal of The Electrochemical Society, 160(4), A616. https://doi.org/10.1149/2.053304jes

PyBaMM. (2025, April 2). Retrieved May 14, 2025, from https://www.pybamm.org/

Ritter, C., Darrington, J., & Browning, J. (2020). DEEP LYNX: Digital Engineering Data Warehouse. Retrieved from Idaho National Laboratory: https://inl.elsevierpure.com/en/publications/deeplynx-digital-engineering-data-warehouse

Temporal Fusion Transformer (TFT). (2021, April 15). Retrieved May 14, 2025, from https://unit8co.github.io/darts/generated_api/darts.models.forecasting.tft_model.html Time-series Dense Encoder (TiDE). (2021, April 15). Retrieved May 14, 2025, from https://unit8co.github.io/darts/generated_api/darts.models.forecasting.tide_model.html

U.S. Department of Energy, Office of Electricity (DOE-OE). (2023a).

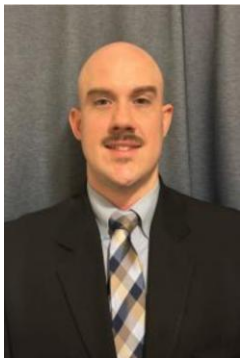
Rapid Operational Validation Initiative (ROVI). Retrieved from DOE-OE: https://www.energy.gov/oe/rapid-operationalvalidation-initiative-rovi 16

U.S. Department of Energy, Office of Electricity (DOE-OE). (2023b). Energy Department Selects Six National Laboratories to Validate Battery Energy Storage Performance. Retrieved from DOE-OE: https://www.energy.gov/oe/articles/energy-department-selects-six-nationallaboratories-validate-battery-energy-storage

U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy (DOE-EERE). (2019). INL Nissan Leaf – Pack Testing – Metadata. Retrieved from DOE-EERE: https://batterydata.energy.gov/dataset/metadata/inl-nissan-leaf-pack-testing

Ungurean, L., Cârstoiu, G., Micea, M. V., & Groza, V. (2017). Battery state of health estimation: A structured review of models, methods, and commercial devices. International Journal of Energy Research, 41(2), 151–181. https://doi.org/10.1002/er.3598

Biography



Jack Cavaluzzi

Dr. Jack Cavaluzzi is a modeling and simulation professional in the Autonomous Engineering department at INL, where he develops automated data pipelines using Apache Airflow and creates advanced data-visualization tools for 3D models. His current focus is on building a Digital-Twin Viewer web application that integrates the Unity and React tools to support the interactive analysis of complex engineering systems. Dr. Cavaluzzi holds bachelor's, master's, and doctoral degrees in nuclear engineering from Texas A&M University. Before joining INL in January 2022, he interned with the INL Advanced Scientific Computing Division, developing thermohydraulic turbomachinery models in C++ for RELAP-7 and THM, as part of his doctoral research.

Kaleb Houck

Kaleb Houck has more than 10 years of experience in integrating large-scale software systems. As a software architect and researcher in the INL Digital Engineering department, he leads digital-twin, digital-thread, and data-hub efforts. He is currently developing a digital twin for the Beartooth nuclear fuel-cycle test bed. Previously, Mr. Houck contributed to the creation of a LINK16 battlespace digital-twin system for NAVAIR to enhance integration and interoperability in complex systems. He holds a bachelor's degree in Computer Science from the University of Idaho, completed in 2007.

Porter Zohner

Porter Zohner is a digital-systems engineer at INL. He has a range of skills spanning multiple domains, such as extended-reality development, web-app visualization development, digital twins, cybersecurity, and data science. Mr. Zohner started as an intern in the INL's Information Management and Enterprise Architecture department in 2016. He is currently seeking a master's degree in artificial intelligence and machine learning from Drexel University. Mr. Zohner received his bachelor of science degree in computer information technology/computer science from Brigham Young University–Idaho in 2018. That same year, he started working full time at INL.

Sonali Sinha Roy

Sonali Sinha Roy is an artificial intelligence engineer in the Autonomous Engineering department at INL. In this role, she applies AI to automate engineering design, performs system-level modeling and simulations, and creates digital twins for large-scale energy systems. Dr. Sinha Roy holds a doctoral degree in aerospace engineering from Purdue University. Through her dissertation research, she developed a hierarchical state-based risk- and performance-assessment framework to aid in decision-making efforts during the design phase of complex multisystem space missions, such as the Mars Sample Return Campaign being conducted by the U.S. National Aeronautical and Space Administration (NASA) and the European Space Agency (ESA)