TU Delft

Delft University of Technology

Document Version
Final published version

Licence
Dutch Copyright Act (Article 25fa)

Citation (APA)
Yu, R., Kleine, V., Gromotka, P., Rudolf, T., Eisenmann, A., Chandra Mouli, G. R., Palensky, P., & Cremer, J. L. (2026). Probabilistic Hysteresis Factor Prediction for Electric Vehicle Batteries with Graphite Anodes Containing Silicon. *IEEE Transactions on Transportation Electrification* , *12*(4), 5686-5695. https://doi.org/10.1109/TTE.2026.3673281

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

Probabilistic Hysteresis Factor Prediction for Electric Vehicle Batteries With Graphite Anodes Containing Silicon

Runyao Yu, Viviana Kleine, Philipp Gromotka, Thomas Rudolf, Adrian Eisenmann, Gautham Ram Chandra Mouli, *Senior Member, IEEE*, Peter Palensky, *Senior Member, IEEE*, and Jochen L. Cremer, *Senior Member, IEEE*

***Abstract*—Batteries with silicon–graphite-based anodes, which offer higher energy density and improved charging performance, introduce pronounced voltage hysteresis, making state-of-charge (SoC) estimation particularly challenging. Existing approaches to modeling hysteresis rely on exhaustive high-fidelity tests or focus on conventional graphite-based lithium-ion batteries, without considering uncertainty quantification or computational constraints. This work introduces a data-driven approach for probabilistic hysteresis factor prediction, with a particular emphasis on applications involving silicon–graphite anode-based batteries. A data harmonization framework is proposed to standardize heterogeneous driving cycles across varying operating conditions. Statistical learning and deep learning models are applied to assess performance in predicting the hysteresis factor with uncertainties while considering computational efficiency. Extensive experiments are conducted to evaluate the generalizability of the optimal model configuration in unseen vehicle models through retraining, zero-shot prediction, fine-tuning, and joint training. By addressing key challenges in SoC estimation, this research facilitates the adoption of advanced battery technologies.**

***Index Terms*—Deep Learning, heterogeneous driving cycle, hysteresis factor, probabilistic prediction, silicon–graphite.**

I. Introduction

BATTERIES are one of the key components of EVs. Central to battery management is the estimation of state-of-charge (SoC), which indicates the remaining usable battery energy. A more precise SoC estimation reduces the required SoC buffer, which is typically set to account for estimation uncertainties and protect the battery from aging. As a result, the SoC operating range can be expanded, enabling a greater depth of discharge and increasing the usable capacity. Especially, original equipment manufacturers (OEMs) continuously seek to improve the driving range. If the additional 2% SoC is utilized to extend the range of vehicles, assuming an energy consumption of 19.8 kWh/100 km (comparable to an electric compact SUV), the range will be increased by approximately 10 km. Given that range anxiety remains a major barrier to EV adoption [1], [2], [3], even small improvements can contribute to broader consumer acceptance. In addition to estimation accuracy, BMS is often memory- and computation-constrained in highly competitive cost structures as in the automotive industry, requiring efficient models suitable for limited random access memory (RAM) and read-only memory (ROM) capacities.

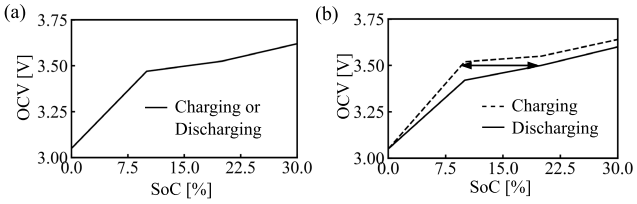


Fig. 1. Illustration of OCV–SoC curves. (a) OCV–SoC relationship provided by the cell manufacturer. (b) OCV–SoC relationship with voltage hysteresis, which is particularly significant for silicon–graphite anode-based batteries.

Traditionally, cell manufacturers provide the open-circuit voltage (OCV)-SoC reference table to correct the SoC value after battery relaxation, as illustrated in Fig. 1(a), where a unique OCV corresponds to a unique SoC. However, OCV–SoC reference tables often depend on battery chemistry, as different electrode materials exhibit distinct voltage characteristics. Common materials in automotive applications are graphite for the anode [4], and LFP, NMC, and LMO for the cathode [5]. Due to the limited energy density of graphite anodes, silicon has gained significant popularity as an alternative to graphite anodes [6]. Numerous automobile manufacturers announce plans to integrate silicon technology into their batteries, as integrating silicon as an anode enables longer vehicle ranges and faster charging times [7]. However, silicon–graphite anode-based batteries have certain drawbacks, with pronounced voltage hysteresis being a major limitation.

Voltage hysteresis represents a phenomenon where the OCV–SoC behavior differs based on the charge direction, as shown in Fig. 1(b). During hysteresis, the OCV curve

Received 23 June 2025; revised 11 September 2025 and 1 January 2026; accepted 9 March 2026. Date of publication 12 March 2026; date of current version 22 July 2026. *(Corresponding author: Runyao Yu.)*

Runyao Yu and Jochen L. Cremer are with Delft University of Technology, 2628 CD Delft, The Netherlands, and also with the AIT Austrian Institute of Technology, 1210 Vienna, Austria (e-mail: runyao.yu@tudelft.nl; j.l.cremer@tudelft.nl).

Viviana Kleine is with Delft University of Technology, 2628 CD Delft, The Netherlands, and also with the Porsche Engineering Group, 71287 Weissach, Germany (e-mail: v.kleine@student.tudelft.nl).

Philipp Gromotka, Thomas Rudolf, and Adrian Eisenmann are with the Porsche Engineering Group, Weissach, Germany (e-mail: philipp.gromotka@porsche-engineering.de; thomas.rudolf@porsche-engineering.de; adrian.eisenmann@porsche-engineering.de).

Gautham Ram Chandra Mouli and Peter Palensky are with Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: g.r.chandramouli@tudelft.nl; p.palensky@tudelft.nl).

Digital Object Identifier 10.1109/TTE.2026.3673281

2332-7782 © 2026 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: TU Delft Library. Downloaded on August 07,2026 at 06:23:01 UTC from IEEE Xplore. Restrictions apply.

for charging consistently lies above that for discharging, making it challenging to determine the exact SoC from a given OCV value. For example, at an OCV of 3.5 V, the SoC could correspond to either 10% or 20%. A hysteresis factor is introduced to address the issue, providing a quantifiable approach to SoC estimation within the hysteresis region, and is represented by a scalar continuous weighting toward the charging or discharging characteristic curve. The nonlinear and history-dependent nature of hysteresis further increases the computational complexity of SoC estimation models.

In prior work, various models have been developed to incorporate hysteresis voltage into SoC estimation, which can be categorized into three main approaches: ECMs, physics-based models, and machine-learning-based models. ECMs [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22] incorporate hysteresis through various methods, such as defining a hysteresis factor to blend charge and discharge OCV curves or introducing additional parameters into the (dis)charging equations to account for hysteresis effects. Physics-based models [23] focus on modeling the anode's open-circuit potential and isolating the hysteresis factor based on delithiation and lithiation processes. However, these models typically require special tests, such as constant current pulse tests, which are time-consuming and expensive [24]. To overcome these limitations, machine-learning models [24], [25] have been developed, leveraging data-driven techniques such as random forests and long short-term memory (LSTM) networks to predict the hysteresis factor. Each of these modeling approaches aims to improve SoC estimation accuracy under hysteresis, with varying levels of model complexity and data requirements.

However, prior studies face several limitations that hinder their practical deployment and generalizability. First, data processing pipelines are often vaguely described and inconsistent across works [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], lacking a unified framework to systematically harmonize heterogeneous driving cycles under varying driving conditions. Second, most studies focus exclusively on graphite anodes, with only [23] explicitly addressing silicon–graphite chemistries, limiting the applicability to emerging battery technologies. Third, existing approaches often produce point estimates without the quantification of predictive uncertainties using machine-learning models [24], [25], which can inherently adapt to nonstationary behaviors observed in real-world conditions. Without probabilistic modeling, however, it remains challenging to quantify prediction confidence under such dynamic scenarios. Fourth, hardware constraints are typically overlooked. Without accounting for computational limitations, model implementations can lead to unrealistic computational demands, making them impractical for deployment in BMS with restricted processing power. Fifth, the generalization of hysteresis prediction models to unseen vehicle models remains largely unexplored. Given the variability in vehicle configurations, energy consumption profiles, and cell chemistries, assessing cross-vehicle transferability is essential.

To address the challenges of SoC estimation in the presence of voltage hysteresis—particularly in silicon–graphite anode-based batteries—we present a probabilistic machine-learning approach built on a data harmonization framework. Our method explicitly accounts for predictive uncertainty and computational constraints, and is validated for generalization across diverse EV models. The key contributions of this article are as follows.

1) We propose a data harmonization framework to systematically process heterogeneous driving cycles and provide a valuable analysis of fleet data, yielding insights into real-world battery dynamics.
2) We formulate hysteresis factor prediction in silicon–graphite anode-based batteries as a probabilistic modeling task, systematically benchmarking statistical and deep learning approaches under computational constraints, thereby advancing methods for uncertainty-aware battery state estimation.
3) We provide a systematic evaluation of multiple transfer learning strategies (retraining, zero-shot prediction, fine-tuning, and joint training) across vehicle models with different cell chemistries and electrical specifications, deriving insights into the principles of model generalization for battery hysteresis prediction.

The paper is structured as follows. Section II introduces the proposed data harmonization framework for heterogeneous driving cycles. Section III presents the machine-learning methods. The evaluation criteria are detailed in Section IV. The experimental setup, along with the results for probabilistic hysteresis factor prediction, is discussed in Sections V and VI. Section VII summarizes the findings and provides concluding remarks. Finally, Section VIII describes limitations and future work.

II. Data Harmonization Framework for Heterogeneous Driving Cycles

Let us consider driving cycles data consist of multiple time-series features, characterized by varying feature sets, a mixture of active and inactive periods, different sampling rates, diverse temporal lengths, and nonuniform units and scales, leading to a heterogeneous data structure. In detail, the heterogeneous driving cycles are formulated as a feature matrix $X = \{X^{(i)} \in \mathbb{R}^{T_i \times F_i}\}_{i=1}^{D}$, where D is the number of driving cycles, T_i is the number of time steps (or sequence length) in the sample i, and F_i is the number of features. Note that both T_i and F_i may vary across samples, reflecting differences in duration and available features. We propose an approach to harmonize the heterogeneous driving cycles, illustrated in Fig. 2(a).

A. Data Filtering

Various features are available, such as battery current, cell voltage, and SoC. Table I overviews the features used in different hysteresis models from the literature. In alignment with prior work and engineering considerations, features influenced by the hysteresis factor and battery current, such as SoC and capacity, are excluded to minimize algorithmic bias. Additionally, OCV hysteresis curves are excluded, as

Authorized licensed use limited to: TU Delft Library. Downloaded on August 07,2026 at 06:23:01 UTC from IEEE Xplore. Restrictions apply.

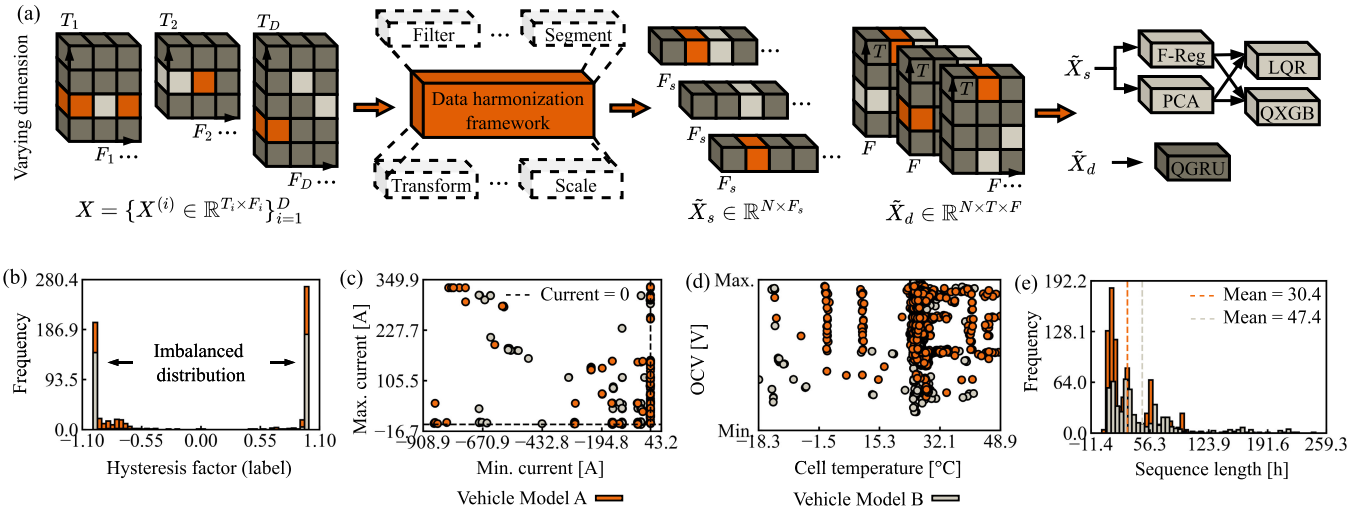


Fig. 2. Overview of harmonization framework and exemplary driving cycles of two vehicle models. (a) Visualization of the harmonization framework. Heterogeneous driving cycle data are harmonized to produce unified representations: $\tilde{X}_s$ and $\tilde{X}_d$, which are fed into corresponding machine-learning models. (b) Distribution of the hysteresis factor (range in [–1, 1]), showing a highly imbalanced distribution for vehicle mode A and vehicle model B. Less than 70% of the data available for vehicle model A is present in vehicle model B. (c) Scatter plot of maximum versus minimum current. The dashed lines indicate the zero-current baseline, separating charging (negative current) and discharging (positive current) regions, where the vehicle model A exhibits a more aggressive charging and discharging behavior. (d) Scatter plot of normalized OCV against cell temperature, capturing diverse thermal conditions and spanning a wide range from –18.3 °C to 48.9 °C. Most samples are concentrated between 20 °C and 35 °C, suggesting that typical operations occur within standard ambient or thermally managed conditions. (e) Distribution of sequence lengths of driving cycles, exhibiting a heterogeneous temporal structure. The vehicle model A tends to be shorter on average, with a mean sequence length of 30.4 h, while the vehicle model B shows longer sequences, averaging 47.4 h. This suggests that different vehicle models are associated with different operating regimes.

TABLE I
RELEVANT FEATURES ACCORDING TO DIFFERENT HYSTERESIS MODELS

Model	Hysteresis Factor	Battery Current	Capacity	SoC	OCV Hysteresis Curves	Cell Voltage
Plett [8]	✓	✓				
Roscher [20]		✓	✓			
Xie [13]		✓			✓	
Ko [15]		✓	✓	✓		
Xie [12]	✓	✓	✓	✓		
Plett [26]				✓	✓	
Baronti [27]				✓	✓	
Dong [17]				✓	✓	
Kim [19]				✓		✓
Antony [18]		✓	✓			
Zhou [21]				✓	✓	✓
Yu [22]						✓
Gao [23]					✓	✓
Li [25]				✓		✓
Xu [24]		✓				✓

these curves are only used as a precursor to calculate voltage changes, and this work predicts the hysteresis factor directly. Moreover, the cell temperature is included, as some models from the literature indirectly considered the temperature by performing parameter identification under different temperature conditions. Therefore, the input features considered in this work are battery current, cell voltage, and cell temperature.

The analysis of these features extracted from two vehicle models is shown in Fig. 2(b)–(e). Note that the battery current is used as a proxy for the cell current. Furthermore, if an identified feature, such as cell temperature or cell voltage, is missing, the corresponding data sample is excluded. Therefore, the feature matrix is updated to $X' = \{X'^{(i)} \in \mathbb{R}^{T_i \times F}\}_{i=1}^{D'}$, where F becomes constant between samples, and $D' < D$ as only partial samples contain all the needed features.

B. Data Segmentation

Driving cycles often contain inactive periods, such as vehicle standby or battery relaxation phases. We apply a data segmentation strategy to identify and retain relevant data segments, illustrated in Fig. 3. A valid segment starts after a SoC correction if a relaxation phase occurs, during which the battery rests without charge or discharge, allowing the voltage to stabilize toward its open-circuit value. The segment is retained until the next SoC correction occurs. Any segment

Authorized licensed use limited to: TU Delft Library. Downloaded on August 07,2026 at 06:23:01 UTC from IEEE Xplore. Restrictions apply.

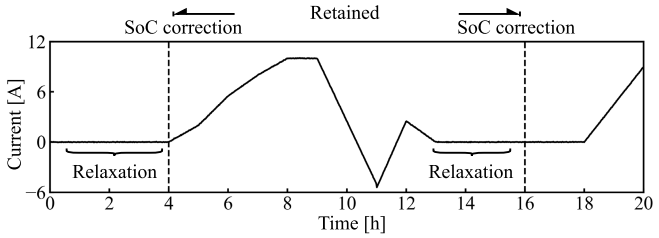


Fig. 3. Example illustration of segmentation. A valid segment starts at the fourth hour, where a relaxation phase is followed by a SoC correction. The segment between hours 16 and 20 is not retained, as it lacks a closing SoC correction.

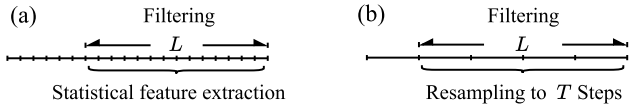


Fig. 4. Illustration of two transformation approaches. (a) Transformation for statistical learning models. (b) Transformation for deep learning models.

where the battery current remains zero throughout or the duration is shorter than 10 s is discarded. Each segmented data instance is treated as a separate sample. Therefore, the number of samples after segmentation is higher than D', as one driving cycle may contain several segments. This segmentation results in a new feature matrix $X'' = \{X''^{(i)} \in \mathbb{R}^{T_i' \times F}\}_{i=1}^{N}$, where N is the number of samples after segmentation.

C. Data Transformation

Some driving cycles may originally be sampled at higher or lower rates due to differences in data logging equipment, vehicle models, or testing protocols, causing temporal inconsistency. To ensure uniformity and comparability across the dataset, all features are resampled to a fixed frequency of 10 Hz (10 time steps per second). In addition, the segmented data described in Section II-B consists of samples with varying sequence lengths. The variability in sequence length poses challenges for machine-learning models, as different models expect different input formats. We introduce two data transformation approaches depending on the requirements of the target model, shown in Fig. 4.

1) For Statistical Learning Models: Statistical learning models require a 2-D input matrix. First, a *temporal filter* is applied on $X''^{(i)}$ to keep only the last L time steps from each sample. This step results in a new feature matrix $X''' \in \mathbb{R}^{N \times L \times F}$, where all sequences are truncated to a consistent temporal length across samples. The temporal filter helps to study the optimal look-back window size given constrained computational power. Then, the statistical feature matrix $X_s \in \mathbb{R}^{N \times F_s}$ is produced by computing *statistical features*, such as mean, min, and max, summarized in Table II, from each feature along the temporal dimension.

2) For Deep Learning Models: Deep learning models for time series often require a 3-D tensor. The same temporal filter is applied to X'' to produce X'''. The statistical feature extraction is unnecessary since deep learning models inherently learn nonlinear representations. However, under limited computational power, it is essential to reduce the resolution of the data. To this end, the truncated feature matrix X''' is resampled at a fixed number of evenly spaced time steps T, resulting in a uniform 3-D input tensor $X_d \in \mathbb{R}^{N \times T \times F}$. The hyperparameter T controls the granularity of the time series representation: smaller values of T capture broader temporal patterns, while larger values retain finer dynamics. All combinations of L and T will be evaluated empirically to identify the optimal configuration for model performance.

TABLE II
STATISTICAL FEATURES AND THEIR MATHEMATICAL FORMULATION

Name	Mathematical definition
Minimum value	$\min_{\ell=0}^{L} x_\ell$
Maximum value	$\max_{\ell=0}^{L} x_\ell$
Absolute minimum value	$\min_{\ell=0}^{L} \lvert x_\ell \rvert$
Absolute maximum value	$\max_{\ell=0}^{L} \lvert x_\ell \rvert$
Sum of absolute changes	$\sum_{\ell=0}^{L-1} \lvert x_{\ell+1} - x_\ell \rvert$
Mean of changes	$\frac{1}{L} \sum_{\ell=0}^{L-1} (x_{\ell+1} - x_\ell)$
Mean of absolute changes	$\frac{1}{L} \sum_{\ell=0}^{L-1} \lvert x_{\ell+1} - x_\ell \rvert$
Absolute energy	$\sum_{\ell=0}^{L} x_\ell^2$
Sum of values	$\sum_{\ell=0}^{L} x_\ell$
Mean	$\frac{1}{L} \sum_{\ell=0}^{L} x_\ell$
Complexity	$\sqrt{\sum_{\ell=0}^{L-1} (x_{\ell+1} - x_\ell)}$
Sum of positive values	$\sum_{\ell=0}^{L} x_\ell,\ \text{where } x_\ell > 0$
Sum of negative values	$\sum_{\ell=0}^{L} x_\ell,\ \text{where } x_\ell < 0$
Mean of positive values	$\frac{1}{L} \sum_{\ell=0}^{L} x_\ell,\ \text{where } x_\ell > 0$
Mean of negative values	$\frac{1}{L} \sum_{\ell=0}^{L} x_\ell,\ \text{where } x_\ell < 0$
Number of zero elements	$\sum_{\ell=0}^{L} \mathbf{I}(x_\ell = 0)$
Root mean square	$\sqrt{\frac{1}{L} \sum_{\ell=0}^{L} x_\ell^2}$
First value	x_0
Last value	x_L

D. Data Scaling

Features with different units and scales may introduce unintended biases, with models implicitly assigning greater importance to larger magnitude inputs. Scaling also improves numerical stability and accelerates convergence during training [28].

Authorized licensed use limited to: TU Delft Library. Downloaded on August 07,2026 at 06:23:01 UTC from IEEE Xplore. Restrictions apply.

To ensure uniformity, all features are standardized to consistent units—for example, cell voltage is converted from millivolts (mV) to volts (V). Similar unit adjustments are applied to battery current, eliminating mismatches and enhancing comparability across samples.

Subsequently, we normalize the feature values to the range $[0, 1]$ using min–max scaling.

1) For Statistical Learning Models: We apply the standard `MinMaxScaler` implementation from `scikit-learn` to obtain the scaled feature matrix $\tilde{X}_s$, ensuring consistent and reproducible feature scaling.

2) For Deep Learning Models: We implement a customized min–max scaler. Given the input tensor $X_d \in \mathbb{R}^{N\times T\times F}$, the scaled data $\tilde{X}_d$ are calculated as

$$\tilde{X}_d = \frac{X_d - X_d^{\min}}{X_d^{\max} - X_d^{\min}} \tag{1}$$

where $X_d^{\min}, X_d^{\max} \in \mathbb{R}^{1\times 1\times F}$ represent the minimum and maximum values of each feature. This ensures that the feature matrix is scaled within $[0, 1]$.

III. Machine-Learning Methods

To enable uncertainty-aware predictions, all models in this work are trained to predict multiple target quantiles instead of a single point estimate. Due to limited hardware resources, we apply three machine-learning models for comparison, categorizing them into two types. The first type, statistical learning models, includes linear quantile regression (LQR) and quantile XGBoost (QXGB). These models operate on 2-D statistical feature inputs and are combined with dimensionality reduction techniques, namely principal component analysis (PCA) and F-regression, to either compress or select features prior to modeling. The second type is a deep learning model, the quantile gated recurrent unit (QGRU), which processes 3-D sequential inputs without the need for statistical feature extraction. The objective is to identify the optimal model configuration while evaluating its computational requirements under constrained hardware resources, as typically encountered in vehicle applications.

A. Principal Component Analysis

PCA is used to reduce the dimensionality of the scaled statistical feature matrix $\tilde{X}_s \in \mathbb{R}^{N\times F_s}$ by projecting it onto a lower dimensional orthogonal subspace that captures the directions of maximum variance. Let $W_{\text{PCA}} \in \mathbb{R}^{F_s\times K}$ be the matrix of the top K eigenvectors of the empirical covariance matrix of $\tilde{X}_s$, where $K < F_s$. The choice of K is determined based on explained variance.

After dimensionality reduction, the compressed features are given by

$$\tilde{X}_{\text{PCA}} = \tilde{X}_s W_{\text{PCA}} \tag{2}$$

where $\tilde{X}_{\text{PCA}} \in \mathbb{R}^{N\times K}$ is then used as the input to statistical models such as LQR or QXGB.

This method decorrelates the input features and often improves the robustness and generalization of downstream models by eliminating noise and redundant information. PCA also enables more efficient storage and faster computation, which is particularly beneficial in memory-constrained environments such as in-vehicle systems [29].

B. F-Regression

F-regression evaluates the linear dependency between each feature and the target using the F-statistic. For each feature $\tilde{x}_j \in \mathbb{R}^N$ (i.e., the jth column of $\tilde{X}_s$), we perform a linear regression against the target $y \in \mathbb{R}^N$ and compute the F-statistic $\mathcal{F}_j$ and p-value p_j

$$\mathcal{F}_j = \frac{\text{Var}\left(\mathbb{E}\left[y \mid \tilde{x}_j\right]\right)}{\mathbb{E}\left[\text{Var}\left(y \mid \tilde{x}_j\right)\right]}, \quad p_j = \mathbb{P}\left(\mathcal{F} > \mathcal{F}_j\right) \tag{3}$$

where a smaller p-value p_j indicates stronger statistical relevance to the target. The top K features with the lowest p-values are selected to form the reduced matrix $\tilde{X}_{\text{F-reg}} \in \mathbb{R}^{N\times K}$, which is used as the input to LQR or QXGB.

This method allows for a simple yet effective feature selection process, prioritizing features that are most predictive under a linear assumption. Unlike PCA, which produces linear combinations of all features, F-regression retains the original feature meanings, making the model more interpretable. Furthermore, by discarding weakly correlated features, F-regression can help mitigate overfitting and reduce computational cost, particularly when the original feature set is large.

C. Linear Quantile Regression

LQR is chosen as a simple yet effective baseline due to its low computational complexity, fast training time, and interpretability [30]. Unlike ordinary least squares (OLSs), which estimate the conditional mean of the target variable, LQR directly estimates conditional quantiles, making it well-suited for probabilistic prediction tasks under high-variance input data. It models the τth conditional quantile of the target as a linear function of the input features

$$\hat{y}_{\tau,i} = \tilde{X}_{\text{red}}^T \beta_\tau \tag{4}$$

where $\tilde{X}_{\text{red}} \in \mathbb{R}^{N\times K}$ represents the reduced feature matrix obtained from either PCA ($\tilde{X}_{\text{PCA}}$) or F-regression ($\tilde{X}_{\text{F-reg}}$), and $\beta_\tau \in \mathbb{R}^K$ is the parameter vector estimated for quantile τ.

The model parameters are learned by minimizing the quantile loss $L_\tau(y_i, \hat{y}_{\tau,i})$ described in Section IV-A, which penalizes under- and over-predictions asymmetrically depending on the target quantile.

Due to its linear structure, LQR can be trained efficiently even on large datasets and provides easily interpretable coefficients that reflect the marginal effect of each feature on different parts of the target distribution. However, its modeling capacity is limited by the assumption of linearity, making it less flexible than more complex models such as tree-based methods or neural networks.

D. Quantile XGBoost

QXGB is selected due to its strong performance in capturing nonlinear relationships by leveraging gradient-boosted decision trees [31]. The predicted quantile is given by

$$\hat{y}_{\tau,i} = f_\tau\left(\tilde{X}_{\text{red}}\right) \tag{5}$$

Authorized licensed use limited to: TU Delft Library. Downloaded on August 07,2026 at 06:23:01 UTC from IEEE Xplore. Restrictions apply.

where $\tilde{X}_{\text{red}} \in \mathbb{R}^{N\times K}$ is the reduced feature matrix produced by PCA or F-regression, and f_τ denotes an ensemble of boosted decision trees trained for quantile τ. Specifically, the model consists of M additive trees

$$f_\tau\left(\tilde{X}_{\text{red}}\right) = \sum_{m=1}^{M} h_m\left(\tilde{X}_{\text{red}}\right) \tag{6}$$

where each base learner h_m is trained iteratively to minimize the quantile loss $L_\tau(y_i, \hat{y}_{\tau,i})$ as defined in Section IV-A.

E. Quantile Gated Recurrent Unit

Recurrent neural networks (RNNs) are well-suited for time-series data. Although both QLSTM and QGRU can model sequential dependencies, QGRU has been shown to be more parameter-efficient [32], making it a better choice given our hardware constraints.

The QGRU model processes scaled sequential inputs $\tilde{X}_d \in \mathbb{R}^{N\times T\times F}$. The update equations for the QGRU are

$$z_t = \sigma\left(W_z\tilde{X}_t + U_z h_{t-1} + b_z\right) \tag{7}$$

$$r_t = \sigma\left(W_r\tilde{X}_t + U_r h_{t-1} + b_r\right) \tag{8}$$

$$\tilde{h}_t = \tanh\left(W_h\tilde{X}_t + U_h\left(r_t \odot h_{t-1}\right) + b_h\right) \tag{9}$$

$$h_t = (1 - z_t)\odot h_{t-1} + z_t \odot \tilde{h}_t \tag{10}$$

where $\tilde{X}_t \in \mathbb{R}^{N\times F}$ represents the input at time step t and $h_t \in \mathbb{R}^{N\times H}$ denotes the hidden state. The learnable parameters include the weight matrices $W_z, W_r, W_h \in \mathbb{R}^{F\times H}$ and $U_z, U_r, U_h \in \mathbb{R}^{H\times H}$, and the bias terms $b_z, b_r, b_h \in \mathbb{R}^{H}$. The activation functions are the sigmoid function $\sigma(\cdot)$ for the update gate z_t and reset gate r_t, and the hyperbolic tangent function $\tanh(\cdot)$ for computing the candidate hidden state $\tilde{h}_t$. The operator $\odot$ denotes element-wise multiplication.

The predicted quantile is obtained as

$$\hat{y}_{\tau,i} = g_\tau\left(h_T\right) \tag{11}$$

where g_τ is a fully connected layer mapping the final hidden state h_T to the quantile output.

The QGRU model is trained by minimizing the same quantile loss $L_\tau(y_i, \hat{y}_{\tau,i})$ from Section IV-A.

IV. Metrics

We consider three key metrics: average quantile loss (AQL), random access memory (RAM), and read-only memory (ROM) to assess the model's prediction performance and computational efficiency.

A. Average Quantile Loss

AQL is utilized to estimate the conditional quantiles of the target distribution [33]. Given a quantile level $\tau \in (0, 1)$, the quantile loss function L_τ is defined as

$$L_\tau\left(y_i, \hat{y}_{\tau,i}\right) = \begin{cases} \tau\cdot\left(y_i - \hat{y}_i\right), & \text{if } y_i \geq \hat{y}_i \\ (1-\tau)\cdot\left(\hat{y}_i - y_i\right), & \text{if } y_i < \hat{y}_i \end{cases} \tag{12}$$

where y_i denotes the true value, and $\hat{y}_i$ represents the predicted quantile for the ith sample. This loss function asymmetrically penalizes errors: when predicting upper quantiles ($\tau > 0.5$), under-predictions receive a higher penalty, whereas for lower quantiles ($\tau < 0.5$), over-predictions are penalized more.

TABLE III
HYPERPARAMETER SEARCH RANGE

Model	Hyperparameter	Range
LQR	Reg. L1	[0, 1]
QXGB	Learning Rate	[0.005, 0.1]
	Max. Depth	[3, 5]
	Subsample	[0.6, 1]
	Reg. Lambda	[0.1, 10]
	Reg. Alpha	[0.1, 3]
	Min. Child Weight	[1, 4]
	Colsample by Tree	[0.7, 0.9]
	Num. Boost Rounds	[50, 1000]
QGRU	Learning Rate	[0.0001, 0.01]
	Hidden Size	[4, 64]
	Num. Layers	[1, 2]
	Batch Size	[8, 32]

The AQL is calculated as the mean quantile loss over all predicted quantiles and samples

$$\text{AQL} = \frac{1}{|\mathcal{Q}|N}\sum_{\tau\in\mathcal{Q}}\sum_{i=1}^{N} L_\tau\left(y_i, \hat{y}_{\tau,i}\right) \tag{13}$$

where $\mathcal{Q}$ is the set of quantile levels considered, and N denotes the total number of samples. A lower AQL value indicates improved overall quantile prediction performance.

B. Random Access Memory

RAM is used to measure the real-time computational workload and data handling efficiency of machine-learning models, measured in megabytes (MB). As a volatile memory, RAM temporarily stores intermediate computations, activations, and model states during training and inference. In the context of EVs, hardware resources are limited, making RAM consumption a critical factor for deploying machine-learning models. An efficient model should minimize RAM usage while maintaining predictive performance, ensuring compatibility with on-board computing limitations.

C. Read-Only Memory

ROM is used to measure the storage requirements of machine-learning models, particularly for deployment in resource-limited environments, measured in MB. Unlike RAM, ROM is nonvolatile, meaning it retains data even after power loss. It stores the trained model's weights, biases, and static parameters required for inference. For the hysteresis factor prediction in batteries, minimizing ROM usage is essential to fit within the storage constraints of EV hardware. A compact model with low ROM requirements ensures efficient deployment, preserving space for other essential vehicle control and battery management functions.

V. Probabilistic Hysteresis Factor Prediction

The experiment is conducted in two stages: first, a *sequence-level prediction* is performed to identify the optimal model and

Authorized licensed use limited to: TU Delft Library. Downloaded on August 07,2026 at 06:23:01 UTC from IEEE Xplore. Restrictions apply.

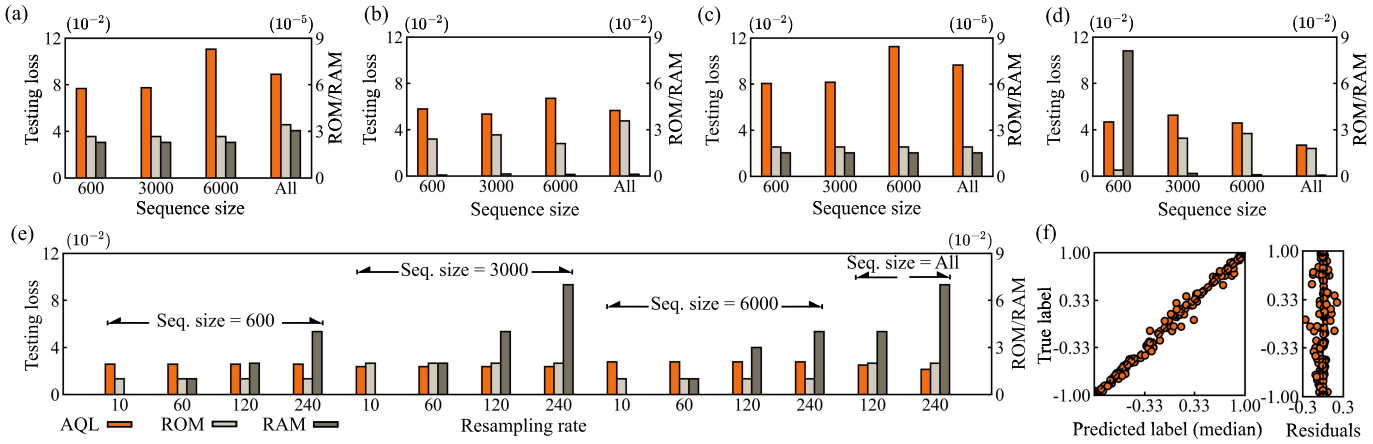


Fig. 5. Comparison of model performance and computational efficiency for different modeling strategies on sequence-level and subsequence-level prediction tasks. (a) PCA + LQR on sequence-level prediction. (b) PCA + QXGB on sequence-level prediction. (c) F-Reg + LQR on sequence-level prediction. (d) F-Reg + QXGB on sequence-level prediction. (e) QGRU on sequence-level prediction. Different resampling rates are explored for each sequence size. (f) Scatter plots of true label versus predicted (median) label (left) and residuals (right) for QGRU on subsequence-level prediction. Tight clustering along the diagonal and low residuals demonstrate the strong predictive performance of QGRU.

TABLE IV
MODEL PERFORMANCE COMPARISON ON TESTING DATA

Dim. Red.	Model	Seq. Size	Resamp. Rate	AQL ↓	ROM ↓	RAM ↓
PCA	LQR	600	–	7.67×10^{-2}	2.67×10^{-5}	2.29×10^{-5}
	QXGB	3000	–	5.34×10^{-2}	2.67×10^{-2}	1.35×10^{-3}
F-Reg	LQR	600	–	8.06×10^{-2}	$\mathbf{1.91 \times 10^{-5}}$	$\mathbf{1.53 \times 10^{-5}}$
	QXGB	All	–	2.67×10^{-2}	1.79×10^{-2}	6.80×10^{-4}
–	QGRU	All	240	$\mathbf{2.15 \times 10^{-2}}$	2.00×10^{-2}	7.00×10^{-2}

Subsequence-Level Prediction

Dim. Red.	Model	Subseq. Size	Resamp. Rate	AQL ↓	ROM ↓	RAM ↓
–	QGRU	6000	240	$\mathbf{2.68 \times 10^{-3}}$	$\mathbf{1.00 \times 10^{-2}}$	$\mathbf{4.00 \times 10^{-2}}$
	QGRU*	6000	240	4.65×10^{-3}	1.00×10^{-2}	4.00×10^{-2}

its configuration. Then, the identified model is applied in a *subsequence-level prediction* setting. The target quantiles are set at 5%, 50%, and 95%, respectively. The dataset is split into 80%, 10%, and 10% for training, validation, and testing, respectively. The hyperparameters are optimized based on an exhaustive random grid search with fivefold cross-validation, employing scikit-learn's RandomizedSearchCV. The search space of hyperparameters is listed in Table III, totaling more than 1000 trials.

A. Sequence-Level Prediction

In this approach, each sample includes the full sequence. Each sequence is truncated to keep the last $L \in \{600, 3000, 6000, \mathit{All}\}$ time steps, corresponding to $S \in \{60, 300, 600, \mathit{All}\}$ s.

Each truncated sequence is resampled at a fixed number of time steps $T \in \{10, 60, 120, 240\}$ to ensure consistent input dimensions, as described in Section II-C2.

The performance comparison is summarized in Table IV, where the best results are highlighted in **bold**. Note that the table only reports the best configuration for each model, that is, the combination of sequence size and resampling rate that yields the lowest AQL. Further details on the full set of configurations and their performance can be found in Fig. 5.

Among statistical learning models, QXGB with F-Reg achieves a strong tradeoff, reaching an AQL of 2.67×10^{-2} with ROM and RAM consumption of 1.79×10^{-2} MB and 6.80×10^{-4} MB, respectively, making it a competitive choice for resource-constrained scenarios. In contrast, LQR models show poorer predictive performance, with an AQL value of 7.67×10^{-2}. Moreover, increasing the sequence size does not lead to a consistent decrease in testing loss, indicating that the LQR is unable to fully leverage longer input sequences. However, LQR can still be viable for extremely low-resource applications given its minimal ROM and RAM usage. Notably, QGRU achieves the lowest AQL of 2.15×10^{-2} while maintaining reasonable ROM and RAM usage. This demonstrates the strong representational capacity of deep learning models compared to traditional statistical approaches. It is seen that the optimal resampling rate of 240 yields the lowest AQL for QGRU. Increasing the resampling rate could potentially further reduce AQL; however, this may result in significantly higher memory usage, which could be prohibitive in resource-constrained environments.

Authorized licensed use limited to: TU Delft Library. Downloaded on August 07,2026 at 06:23:01 UTC from IEEE Xplore. Restrictions apply.

Overall, the QGRU stands out as the most accurate model with a tolerable computational cost, whereas QXGB demonstrates a favorable tradeoff between predictive performance and memory usage, making it suitable for moderately constrained environments. LQR models are more appropriate for extremely limited applications, where computational efficiency is prioritized over predictive performance.

B. Subsequence-Level Prediction

In practical deployments, hysteresis factors are predicted continuously over time. To mimic this setting, each sample is first divided into fixed-length subsequences, rather than applying a temporal filter to truncate the sequences, as introduced in Section II-C2. Building on Section V-B, where QGRU demonstrated the best AQL loss, we further evaluate QGRU for subsequence-level prediction. In detail, each subsequence spans a fixed length of $L \in \{100, 600, 3000, 6000\}$ time steps from each sample, corresponding to $S \in \{10, 60, 300, 600\}$ s. Then, the subsequences are similarly resampled at a fixed number $T \in \{10, 60, 120, 240\}$. Each subsequence inherits the same label as in the sequence-level prediction setting, where a single label is assigned to the entire sequence.

Results in Table IV show that the best configuration uses subsequences of 6000 time steps with a resampling rate of 240, which satisfies the hardware requirements of our setup—ROM under 1×10^{-2} MB and RAM under 4×10^{-2} MB. This configuration offers a balance between predictive performance and deployment feasibility. It is worth noting that further improvements may be achieved by increasing the time steps or the resampling rate.

We further evaluate QGRU under two configurations: nonautoregressive and autoregressive. In the nonautoregressive setting, each subsequence is treated independently. In the autoregressive setting, temporal dependencies are introduced by feeding the previous subsequence's prediction as an additional input feature to the next.

Results in Table IV indicate that the QGRU* (autoregressive) yields higher test loss than QGRU (nonautoregressive). This suggests that the accumulation of prediction errors over subsequences degrades performance and is not sufficiently compensated by implicitly learned temporal covariates. We regard high input variance data in combination with the quantile task setting as the possible explanation.

VI. Generalization Across Vehicle Models

This section evaluates the generalization capability of the best model (QGRU) under unseen vehicle models. Specifically, the vehicle model B is introduced, as visualized in Fig. 2. Both models correspond to fully EVs; however, they differ in terms of cell chemistry and electrical specifications.[1] We consider the following evaluation strategies to systematically study cross-model performance.

1) *B/B (Retraining)*: Train on dataset B, and test on dataset B, evaluating in-distribution generalization.
2) *A/B (Zero-Shot Prediction)*: Train on dataset A, and test on dataset B, evaluating out-of-distribution generalization.
3) *A → B/B (Fine-Tuning)*: Train on dataset A, fine-tune on dataset B, and test on dataset B, assessing generalization performance via fine-tuning.
4) *A+B/B (Joint Training)*: Train on the combined dataset (A + B), and test on dataset B, assessing joint distribution generalization.

[1]Due to commercial confidentiality, specifics regarding the cell chemistry and electrical specifications cannot be disclosed.

TABLE V
Comparison of Testing Loss Under Different Training Strategies and Scalers

Strategy	Scaler	AQL ↓	Change ↓
B/B	Old	7.86×10^{-3}	+0.0%
	New	8.38×10^{-3}	+6.2%
A/B	Old	4.06×10^{-2}	+415.9%
	New	6.49×10^{-2}	+726.0%
A→B/B	Old	8.62×10^{-3}	+9.7%
	New	8.23×10^{-3}	+4.7%
A+B/B	Old	$\mathbf{7.60 \times 10^{-3}}$	–3.3%
	New	8.82×10^{-3}	+12.2%

TABLE VI
Key Python Libraries and Their Respective Version

Library	Version
scikit-learn	1.4.2
tsfresh	0.20.2
xgboost	2.0.3
torch	2.3.1+cu121
ray	2.34.0

Moreover, we compare the impact of the data scaler across different evaluation strategies. As the cell chemistries in vehicle models A and B differ, their corresponding voltage ranges also vary, resulting in different scaling behaviors when using a min–max scaler. For retraining (B/B) and zero-shot prediction (A/B) scenarios, we evaluate the effect of applying the old scaler fit on training data from dataset A versus refitting a new scaler using training data from dataset B. For fine-tuning (A → B/B) or joint training (A + B/B) scenarios, the new scaler is refitted using the combined training data, and thus reflects the minimum and maximum values from both datasets. Note that in all cases, model hyperparameters are optimized using the respective validation sets.

Table V presents the resulting testing losses under each scenario. Training exclusively on dataset A and testing on dataset B results in the highest testing loss, with 415.9% higher loss than training solely on dataset B with the old scaler. Fine-tuning on B or combining A and B for training offers improvements compared to solely training on dataset A.

Additionally, using a new scaler fit on dataset B often underperforms compared to the old scaler solely fit on dataset A. This trend can be attributed to the fact that dataset A is significantly larger, and the derived statistics provide a more

Authorized licensed use limited to: TU Delft Library. Downloaded on August 07,2026 at 06:23:01 UTC from IEEE Xplore. Restrictions apply.

representative feature scaling. Poor prediction performance may also result from the differences in cell chemistry between vehicle project B and the training data from project A. If a new scaler is fit using dataset B, but the model parameters remain unchanged, the transformed input distribution becomes misaligned with the model's learned representation. This distribution shift can hinder the model's ability to interpret the inputs correctly, leading to degraded performance. In the fine-tuning scenario (A → B/B), where the model is not trained from scratch but adapted from pretrained weights on dataset A, the new scaler can offer modest improvements. This suggests that when the model parameters are partially aligned with dataset B, adjusting the feature scaling to match the target distribution becomes beneficial. Moreover, the best loss solely with dataset B (8.38×10^{-3}) is significantly higher than the loss achieved for dataset A (2.68×10^{-3}). The discrepancy could be attributed to the availability of data, as less than 70% of the data for vehicle model A is available for vehicle model B, as mentioned in Fig. 2. Therefore, if more data for vehicle model B are available, the predictive performance would also improve.

VII. Conclusion

This work addressed the challenge of modeling voltage hysteresis in silicon–graphite anode-based batteries. We introduced a data harmonization framework to process heterogeneous driving cycles under varying operating conditions and developed a probabilistic approach for hysteresis factor prediction. Unlike prior methods that often overlook predictive uncertainty and computational efficiency, our method explicitly accounts for uncertainty and computational constraints, making it well-suited for deployment in real-world EV applications.

Experimental results demonstrate that QGRU is the most suitable model when predictive performance is the primary concern. QXGB with F-Reg is recommended as a practical tradeoff between accuracy and computation. In contrast, LQR offers minimal computational requirements but performs significantly worse in terms of AQL, making it suitable only for extremely constrained environments. The generalization assessment demonstrates that zero-shot prediction performs significantly worse than transfer learning strategies and is therefore not recommended in practice, particularly when cell chemistries and electrical specifications differ.

VIII. Limitations and Future Work

A key limitation of this work lies in its dependency on the labeling algorithm used to construct learning targets, influencing the predictive performance of QGRU. Moreover, although QGRU achieves the best predictive performance, its computational cost is higher compared to simpler models, which may limit its applicability under highly constrained hardware conditions. Furthermore, the common issue of data imbalance affects the performance of transfer learning, as models trained on skewed distributions may not generalize well to new vehicle chemistries. Future work should focus on improving labeling algorithms within BMS and on constructing chemistry-specific, cell-level datasets to enable a more rigorous evaluation of labeling quality. In addition, exploring advanced yet computationally efficient deep learning models is worthwhile to further improve predictive accuracy while reducing computational demand. At last, as hardware capabilities in electric vehicles continue to improve, it will become feasible and beneficial to increase both the (sub)sequence length and the resampling rate, thereby enhancing model expressiveness and accuracy while still meeting deployment requirements.

Appendix

A. Package Version

The specific Python libraries are listed in Table VI.

B. Hardware Description

All experiments are conducted on a 64-bit Windows workstation equipped with an Intel[2] Core[3] i9-10920X CPU operating at a base frequency of 2.5 GHz. The machine has 128 GB of DIMM RAM and two NVIDIA RTX A5000 GPUs, each with 87.55 GB of GPU memory.

C. Confidentiality of Labeling Algorithm and Data

In the labeling algorithm developed by the Porsche Engineering Group, a parameter is used that must be calibrated for each vehicle project, making the process time-consuming. This algorithm is highly commercially confidential; therefore, we are unable to disclose further technical details.

The fleet data used in this work are subject to commercial confidentiality in the competitive automotive industry. To provide maximum transparency, we have established a project website. If the dataset becomes commercially nonsensitive in the future, we will update the website to release more detailed information.

References

[1] P. Petersen, T. Rudolf, and E. Sax, "A data-driven energy estimation based on the mixture of experts method for battery electric vehicles," in *Proc. 8th Int. Conf. Vehicle Technol. Intell. Transp. Syst.*, 2022, pp. 384–390.

[2] P. Petersen and E. Sax, "A fully automated methodology for the selection and extraction of energy-relevant features for the energy consumption of battery electric vehicles," *Social Netw. Comput. Sci.*, vol. 3, no. 5, p. 342, Jun. 2022.

[3] M. Samant, A. Khatri, G. Batra, and A. Goel, "EY mobility consumer index 2022 study," Ernst & Young Global Ltd. (EY), London, U.K., Tech. Rep., 2022. [Online]. Available: https://www.ey.com/content/dam/eyunified-site/ey-com/en-gl/insights/automotive/documents/ey-mobility-consumer-index-2022-study.pdf

[4] H. Zhang, Y. Yang, D. Ren, L. Wang, and X. He, "Graphite as anode materials: Fundamental mechanism, recent progress and advances," *Energy Storage Mater.*, vol. 36, pp. 147–170, Apr. 2021.

[5] Y. Ding, Z. P. Cano, A. Yu, J. Lu, and Z. Chen, "Automotive Li-ion batteries: Current status and future perspectives," *Electrochemical Energy Rev.*, vol. 2, no. 1, pp. 1–28, Mar. 2019, doi: 10.1007/s41918-018-0022-z.

[6] *Global EV Outlook 2023: Catching Up With Climate Ambitions*, International Energy Agency (IEA), Paris, France, 2023.

[7] A. Evers, "Why porsche, mercedes and GM are betting on silicon-anode batteries," CNBC, Englewood Cliffs, NJ, USA, Tech. Rep., 2022. [Online]. Available: https://www.cnbc.com/2023/04/22/whyporsche-mercedes-and-gm-are-betting-on-silicon-anode-batteries.html

[8] G. L. Plett, "Extended Kalman filtering for battery management systems of LiPB-based hev battery packs: Part 3. State and parameter estimation," *J. Power Sources*, vol. 134, no. 2, pp. 277–292, 2004.

[2]Registered trademark.

[3]Trademarked.

Authorized licensed use limited to: TU Delft Library. Downloaded on August 07,2026 at 06:23:01 UTC from IEEE Xplore. Restrictions apply.

[9] A. Barai, W. D. Widanage, J. Marco, A. McGordon, and P. Jennings, "A study of the open circuit voltage characterization technique and hysteresis assessment of lithium-ion cells," *J. Power Sources*, vol. 295, pp. 99–107, Nov. 2015.

[10] R. Rolt, R. Douglas, P. Nockemann, and R. Best, "*Full battery pack modelling: An electrical sub-model using an EECM for HEV applications*," SAE Tech. Paper 2019-01-1203, 2019.

[11] Y. Xie, S. Wang, G. Zhang, Y. Fan, C. Fernandez, and J. M. Guerrero, "Improved lumped electrical characteristic modeling and adaptive forgetting factor recursive least squares-linearized particle swarm optimization full-parameter identification strategy for lithium-ion batteries considering the hysteresis component effect," *J. Energy Storage*, vol. 67, Sep. 2023, Art. no. 107597.

[12] S. Xie, X. Zhang, W. Bai, A. Guo, W. Li, and R. Wang, "State-of-charge estimation of lithium-ion battery based on an improved dual-polarization model," *Energy Technol.*, vol. 11, no. 4, Apr. 2023, Art. no. 2201364.

[13] J. Xie, J. Ma, Y. Sun, and Z. Li, "Estimating the state-of-charge of lithium-ion batteries using an H-infinity observer with consideration of the hysteresis characteristic," *J. Power Electron.*, vol. 16, no. 2, pp. 643–653, Mar. 2016, doi: 10.6113/jpe.2016.16.2.643.

[14] M. Kwak, B. Lkhagvasuren, J. Park, and J.-H. You, "Parameter identification and SOC estimation of a battery under the hysteresis effect," *IEEE Trans. Ind. Electron.*, vol. 67, no. 11, pp. 9758–9767, Nov. 2020, doi: 10.1109/TIE.2019.2956394.

[15] Y. Ko and W. Choi, "A new SOC estimation for LFP batteries: Application in a 10 ah cell (HW 38120 L/S) as a hysteresis case study," *Electronics*, vol. 10, no. 6, p. 705, Mar. 2021, doi: 10.3390/electronics10060705.

[16] F. Baronti, W. Zamboni, N. Femia, R. Roncella, and R. Saletti, "Experimental analysis of open-circuit voltage hysteresis in lithium-iron-phosphate batteries," in *Proc. 39th Annu. Conf. IEEE Ind. Electron. Soc.*, Nov. 2013, pp. 6728–6733, doi: 10.1109/IECON.2013.6700246.

[17] G. Dong, J. Wei, C. Zhang, and Z. Chen, "Online state of charge estimation and open circuit voltage hysteresis modeling of $LiFePO_4$ battery using invariant imbedding method," *Appl. Energy*, vol. 162, pp. 163–171, Jan. 2016, doi: 10.1016/j.apenergy.2015.10.092.

[18] A. J. Antony and K. Selvajyothi, "A comparative performance analysis of electrical equivalent circuit models with the hysteresis effect of lithium iron phosphate batteries," *Int. J. Green Energy*, vol. 21, no. 7, pp. 1476–1499, May 2024, doi: 10.1080/15435075.2023.2258216.

[19] J. Kim, G.-S. Seo, C. Chun, B.-H. Cho, and S. Lee, "OCV hysteresis effect-based SOC estimation in extended Kalman filter algorithm for a $LiFePO_4$/C cell," in *Proc. IEEE Int. Electr. Vehicle Conf.*, Mar. 2012, pp. 1–5, doi: 10.1109/IEVC.2012.6183174.

[20] M. A. Roscher and D. U. Sauer, "Dynamic electric behavior and open-circuit-voltage modeling of LiFePO4-based lithium ion secondary batteries," *J. Power Sources*, vol. 196, no. 1, pp. 331–336, Jan. 2011, doi: 10.1016/j.jpowsour.2010.06.098.

[21] W. Zhou, X. Ma, H. Wang, and Y. Zheng, "SOC estimation based on hysteresis characteristics of lithium iron phosphate battery," *Machines*, vol. 10, no. 8, p. 658, Aug. 2022, doi: 10.3390/machines10080658.

[22] P. Yu, S. Wang, C. Yu, W. Shi, and B. Li, "Study of hysteresis voltage state dependence in lithium-ion battery and a novel asymmetric hysteresis modeling," *J. Energy Storage*, vol. 51, Jul. 2022, Art. no. 104492, doi: 10.1016/j.est.2022.104492.

[23] Y. Gao, Z. Sun, D. Zhang, D. Shi, and X. Zhang, "Determination of half-cell open-circuit potential curve of silicon-graphite in a physics-based model for lithium-ion batteries," *Appl. Energy*, vol. 349, Nov. 2023, Art. no. 121621, doi: 10.1016/j.apenergy.2023.121621.

[24] Z. Xu, J. Wang, Q. Fan, P. D. Lund, and J. Hong, "Improving the state of charge estimation of reused lithium-ion batteries by abating hysteresis using machine learning technique," *J. Energy Storage*, vol. 32, Dec. 2020, Art. no. 101678, doi: 10.1016/j.est.2020.101678.

[25] W. Li, S. Ruan, A. Bahitbek, Z. Gao, N. Turak, and H. Li, "Exploring the hysteresis effect in SOC estimation of Li-ion batteries," *J. Phys., Conf. Ser.*, vol. 2456, no. 1, Mar. 2023, Art. no. 012023, doi: 10.1088/1742-6596/2456/1/012023.

[26] G. L. Plett, "Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs–Part 2. Modeling and identification," *J. Power Sources*, vol. 134, no. 2, pp. 262–276, doi: :10.1016/j.jpowsour.2004.02.032.

[27] F. Baronti, W. Zamboni, R. Roncella, R. Saletti, and G. Spagnuolo, "Open-circuit voltage measurement of lithium-iron-phosphate batteries," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, May 2015, pp. 1711–1716.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*.

[29] J. Knödler et al., "*The potential of data-driven engineering models: An analysis across domains in the automotive development process*," SAE Tech. Paper 2023-01-0087, 2023.

[30] R. Koenker and G. Bassett, "Regression quantiles," *Econometrica*, vol. 46, no. 1, pp. 33–50, Jan. 1978.

[31] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.

[32] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.

[33] R. Koenker, *Quantile Regression*, vol. 38. Cambridge, U.K.: Cambridge Univ. Press, 2005.

Authorized licensed use limited to: TU Delft Library. Downloaded on August 07,2026 at 06:23:01 UTC from IEEE Xplore. Restrictions apply.