Model-Driven Dataset Generation for Data-Driven Battery SOH Models

***

***Abstract*—State of health (SOH) estimation of batteries is essential for ensuring reliable operations of the battery system. Since there is no practical way to instantaneously measure it at run time, a model is required for its estimation. Recently, several data-driven SOH models have been proposed, whose accuracy heavily relies on the quality of the datasets used for their training. Since these datasets are obtained from measurements, they are limited in the variety of the charge/discharge profiles.**

In order to solve this scarcity issue, in this work we propose to generate the datasets by simulating a traditional battery model (e.g., a circuit-equivalent one). The main advantage of this paradigm is that a simulatable battery model can be used to evaluate a potentially infinite number of workload profiles that can be used for the training of the data-driven model. Moreover, this general concept can be applied using any simulatable battery model, and it thus allows a fine spectrum of accuracy/complexity tradeoff. Our approach shows that using the simulated data allows to reach a reasonable accuracy w.r.t. SOH estimation, obtaining a 7.2% error w.r.t. the simulated model, in exchange for a 27x memory reduction and a ≈2000x speedup.

***Index Terms*—Battery modeling, digital twin, automotive**

I. Introduction

The accuracy of on board state of health (SOH) estimation in battery management systems (BMS) is essential for ensuring the safety and reliability of battery systems of a battery-powered device, and in particular for Electric Vehicles (EVs). As there is no practical physical way to instantaneously measure the SOH, such tracking inevitably requires a model.

The literature about SOH models is extremely vast, including electrochemical, equivalent circuits, semi-empirical, analytical, and statistical models [1]. More recently, on the wave of the Machine Learning (ML) hype, a new category of *data-driven models* has emerged, in which a set of instantaneously measurable battery parameters (typically, voltage, current, and temperature) relative to a charge or discharge session of a battery is eventually labeled with a value of SOH, calculated at the end of the session [2]. These labelled measures are then used as a dataset to train appropriate ML models [3], [4].

Data-driven models essentially solve the two main drawbacks of traditional models: (i) they are more general, as models for different battery types can be naturally obtained by training them with measurements on different devices, thus also covering variability aspects; and (ii) they do not require any kind of simulation, thus leading to a significant reduction in time and space complexity if the model needs to be deployed on a device with limited resources (i.e., no need for a large amount of simulation operations to estimate battery dynamics and/or for a simulation engine).

On the other hand, the quality of data-driven models is strongly dependent on the size and the variety of the dataset. As these datasets are obtained by experimental measurements, it is *materially unfeasible to provide an acceptable coverage of the design space*: datasets are generated at specific working conditions, determined by the application domain, and in limited time, thus restricting the variety of explored charge/discharge/rest patterns, discharge current profiles, and load currents. Last but not least, such a large exploration space should possibly be repeated on multiple battery instances in order to account for the intrinsic variability of the devices. The consequence is that datasets obtained by measurements are by definition very accurate, but accuracy is guaranteed only in the few points of the experiment space that have been measured.

The key motivation of this paper is to fundamentally swap this asymmetry, i.e., to *sacrifice some accuracy while extending the coverage of the design space*. We propose *to use measurements* (possibly much fewer than those required to generate the whole dataset) *to build a simulatable battery model* that incorporates the desired effects, and use then this model to generate arbitrarily large datasets, that will be the training set for the construction of more lightweight and flexible data-driven models.

Several are the advantages of this approach:

- **Exploration of virtually unlimited data points:** once the battery model is built, any kind of current and/or temperature workload can be simulated to generate as many data as possible, in a very short time and at lower costs compared to the experimental measurement setup;
- **Tunable accuracy/complexity tradeoff:** depending on the quality of the available measurement data, more or less accurate battery models can be built, thus building tunable datasets for the data-driven model;
- **Possibility of model combination:** multiple types of battery models can be built and possibly integrated to cover different aspects of battery dynamics.
- **The final SOH model is still a data-driven SOH model:** its execution does not require simulation and it is essentially a callable function of "live" parameters that can be deployed on a target resource-constrained device.

Results show that using data obtained from the chosen simulation models [5] we can achieve an error of about 7% with respect to the SOH data, in exchange for a 27x memory reduction and a ≈2000x speedup.

II. Background and Related Work

A. Battery Aging

Battery aging is the effect of (i) *calendar aging* (L_{cal}), reflecting battery intrinsic degradation when in rest conditions as an effect of temperature, SOC and of elapsed time; and (ii) *cycle aging*, representing capacity loss during each charge/discharge cycle (L_{cyc}), depending on average values of current I, SOC, cell temperature T and depth of discharge (DoD, i.e., difference between final and initial SOC) [6]–[8]. Overall capacity loss is thus the sum of a global term for calendar aging plus the sum of the degradation in each cycle [9]:

$$L_C(t, SOC, DoD, I, T) = L_{cal}(t, SOC, T) + \sum_{i=1}^{N} L_{cyc}(I_i, SOC_i, DoD_i, T_i) \quad (1)$$

where SOC and T are average over an interval of length t in L_{cal}, and refer to each individual cycle i in L_{cyc}.

State-of-the-art models for L_{cal} and L_{cyc} leverage either the similarities of fatigue process of materials subjected to cyclic loading [6] or include some electro-chemical property of the charge/discharge process [9]. It is outside the objective of this work to provide further details about the models themselves; an exhaustive overview of these models is available in [10].

B. Data-Driven SOH Models

The relative simplicity of casting the estimation of the SOH as the problem of building a predictive model has spurred a number of datasets available online [11] and a vast literature about models [3]. The approaches essentially differ in two aspects:

- How the SOH is measured, i.e., to which quantity is the SOH put in relation with;
- The ML model used for the estimation.

Concerning the former aspect, SOH is measured either in terms of the loss of capacity (more typical) or in terms of the increase of the internal resistance. For the latter aspect, conversely, the spectrum of options is definitely much wider: models range from various types of neural networks (feed-forward or recurrent neural network), to simpler models like random forests, SVMs, or Bayesian networks. Many of these works claim to estimate capacity or resistance with a high level of accuracy w.r.t. the starting dataset, resulting promising candidates for SOH estimation.

As also stated by the authors of [3], however, it is quite difficult to compare different approaches and to identify a few ones that can serve as a reference. One reason for this is the *quality* of the datasets. As a matter of fact, most of these datasets are too limited in size. Besides the obvious impact that small datasets have on the accuracy of data-driven models (in particular NNs), there is also the problem of the *variety* of the dataset points. As they are obtained from lab measurements, there are some intrinsic limitations in generating some specific data points (e.g., very low load currents, which will require prohibitive runtimes) and they may be affected by measurement errors and noises.

III. Methodology

A. The General Flow

Our idea is to use the datasets mentioned in the former section (or possibly a small portion thereof) to build a full-fledged, simulatable battery model including the SOH *together with the entire battery dynamics*, and then use this simulatable model to generate additional data points, that become the training set for a higher quality SOH data-driven model. Figure 1 sketches the envisioned flow to implement this approach.

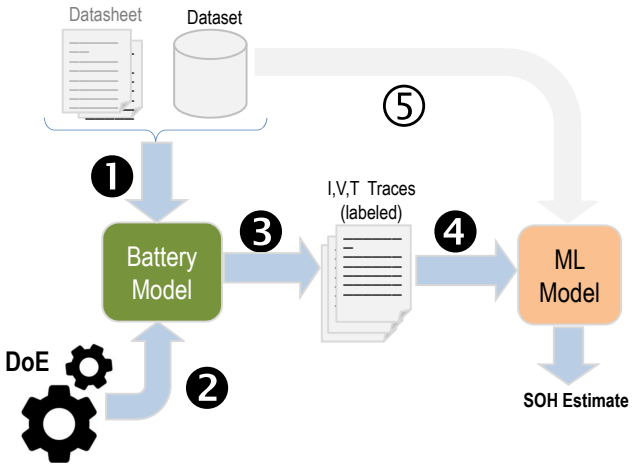


Fig. 1: Conceptual flow of the proposed methodology

The flow starts (❶) from an *available set of battery data*. In the most typical scenario, such data involve measurements collected at various operating conditions and can be freshly generated or already available as public datasets. Note that battery information could be obtained also from datasheets [12]; although feasible, this option would generally result in a poorer accuracy in the construction of the model.

Battery datasets are then used to *identify the parameters of a battery model* (hereafter the *simulation model*), which tracks (at least) the desired target quantity (in our case, SOH). Depending on the type of model, an appropriate procedure is followed for the identification of the model parameters [12]–[14]. Section III-B elaborates the requirements for the simulation model, and Section III-C surveys the main ones available in the literature that comply with such requirements.

We then carry out a design-of-experiment step (❷) in which *an exhaustive set of synthetic traces is generated*, so that as many points as possible in the space of the model inputs are exercised. Then, for each of these design points we run a simulation of the battery model to yield one output trace (❸).

Finally each trace generated by the simulation model is used for training the final ML model (hereafter the *data-driven model*, ❹). Section III-D will describe the various options for the data-driven models and their impact on a BMS. If the format of the traces ❶ and ❸ are the same, we can use a mix of real (measured) and simulated (model-driven) traces to train or test the model (⑤). This step is optional and is not part of the current flow.

B. General Requirements for Battery Simulation Models

The need for a simulatable model brings the issue of identifying a common *interface of the model*, defining the requirements in terms of modeled quantities. The general interface used in our framework is shown in Figure 2.

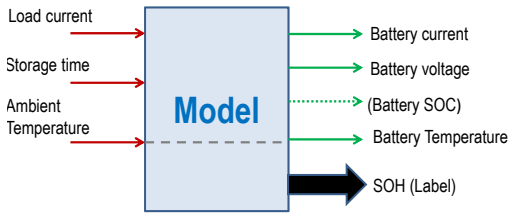


Fig. 2: General Interface of the Battery Simulation Model.

The inputs of the models are the independent variables that represent the external conditions under which the battery is used, and must be measurable at run time. Inputs are load (charge/discharge) current, ambient temperature, and the total storage time (needed for calendar aging). Notice that, although the general aging model (Equation 1) contains the number of cycles N, this is not strictly required as an input of the battery model; N can in fact be simply obtained from the dataset by using the timestamp of each data point: a new cycle is counted anytime we complete a charge/discharge sequence.

The outputs of the model are in theory arbitrary, and are quantities typical of most public datasets [11]; however, there are two constraints to consider: (i) traces generated by the battery model in ❸ **must be labeled**, i.e., include a label that represents SOH, so that the traces can be used to train the ML battery model; and (ii) if possible, the simulation model traces should *follow the same format of the datasets used in* ❶, so that they can potentially be used as an extension of the synthetic dataset to train and/or test the resulting data-driven battery model, thus enabling a wider scenario.

Finally, it is worth emphasizing the distinction between *ambient* and *battery* temperature as they are clearly not the same quantity. If the model is chosen accurately, it will include a thermal model that, based on the electrical quantities **and** the external temperature, will yield the battery's internal temperature. If this is not available, a rough approximation will be to simply use the ambient temperature as a proxy of the internal one (the dashed line). Temperature as an output of the model is essential as SOH is highly sensitive to temperature.

C. Choice of the Battery Simulation Model

The requirements implied by the model interface defined in the previous section might result in a possible difficulty in our approach. Fortunately, models with these characteristics are available in the literature [15], with different ranges of accuracy and complexity. Electro-chemical models represent the chemical reactions as differential equations [16], but they are too complex to be executed on board in real time. Circuit-equivalent models like [17], [18] require a RC network solver or a state-space computation environment and may result in being very heavy; improving their performance on the other hand implies a reduction in accuracy. Kalman filters are a good compromise: they allow to specify an error bound and, being an iterative process, they tend to reduce the estimation error to zero [19].

For the aforementioned reasons, in this work we chose as a reference simulation model the Kalman filter-based one of [5] (Figure 3). This model is pre-defined and populated to describe a 27Ah battery. The model reproduces capacity fading due to thermal cycling, by using Simscape to represent both the thermal and the electrical dynamics of the battery. An Unscented Kalman Filter is then used to estimate battery SOC and internal resistance based on values of voltage, current and internal temperature at runtime. Battery SOC, temperature and internal resistance are then taken in input by a SOH estimation block, built with measurement-based lookup tables.

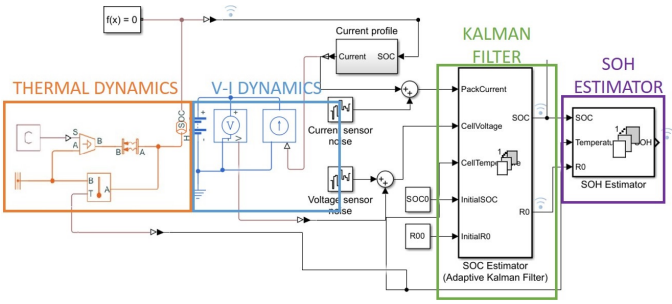


Fig. 3: Adopted battery model based on Kalman filters.

D. Choice of the Battery Data-Driven Model

The last step of the flow is the training of a data-driven model on the dataset generated by the simulatable model. The proposed flow is *independent of the target data-driven model*: in principle, any type of regressor can be embedded in the "ML Model" block of Figure 1, ranging from a simple linear model to a complex deep neural network: this offers a much finer-grain tradeoff between accuracy and computational complexity than traditional physics-based models.

We focus on a scenario in which SOH estimation must be implemented on a low-power Microcontroller (MCU) possibly hosted in the battery management system (BMS) device to enable on-board SOH estimation. We select thus two lightweight data-driven models: (i) a Gradient Boosted Tree (GBT) regressor, i.e., an ensemble of decision trees, selected due to its very fast prediction, consisting of a small number of branching operations [20], and (ii) a Multi-Layer Perceptron (MLP), i.e., a simple fully-connected neural network, appropriate for dealing with pre-aggregated features. These are just examples to prove the flexibility of the proposed approach; the most suitable data-driven model should be selected based on the desired SOH estimation accuracy and on the constraints of the target platform.

1) Model Training: The training dataset is populated from the raw samples of battery current (I), voltage (V) and temperature (T) generated by the simulation model at the

frequency f_s we expect on-board measurements to occur ($f_s = 1$ Hz in our experiments, but this highly depends on the time constants of the considered system). The ML regressors are then trained on 12 statistical features of I, V and T, aggregated over a *time-window* of configurable duration W (2 hours in our experiments): *mean*, *variance*, *min*, and *max* of each quantity. Given that we target a deployment on a highly-constrained, low-performance MCU, we avoid more complex features (e.g., skewness or Kurtosis) since the computational cost for their calculation could outweigh the resulting accuracy improvement. Given that windows have a constant duration, it is not necessary to include also the elapsed time.

ML models are trained to predict $\Delta SOH = SOH_{initial} - SOH_{final}$ in each window, normalized to [0:1] for numerical stability. The training loss function for both models is the Mean Squared Error (MSE) between their prediction and the ΔSOH estimated by the simulation model. Note that the raw SOH prediction can be recovered de-normalizing and accumulating predictions over consecutive windows (1 multiplication and 1 sum per window).

2) Model Space Exploration: A large number of design points can be obtained by (1) exploring model hyper-parameters, and (2) selecting model features. Concerning hyperparameters exploration, we vary the number of estimators (decision trees) of the GBT and their maximum depths in the sets [5, 10, 20, 50, 100, 200] and [1, 2, 3, 4, 5, 10, 30, 50] respectively, for a total of 48 configurations. Similarly, we consider 42 MLP variants, with 1 to 2 hidden layers of sizes in [4, 8, 16, 32, 64, 128]. We explore these options with grid search and 50/20/30% train/validation/test split, to obtain a Pareto-frontier in the MSE vs. execution time and MSE vs. memory occupation spaces. For the MLP, we use the Adam optimizer with a batch size 64 and a learning rate of 0.001, training for 50 epochs. The remaining hyper-parameters are kept at the default values of the respective training libraries (see Section IV-A).

A selection of the best feature set, down to a minimum of 3 features, is applied by using a Recursive Feature Elimination (RFE) algorithm. Thus, the grid search described above is repeated 10 times, once for each feature sub-set, resulting in a total of 480 GBT and 420 MLP models being evaluated.

Since the deployment of such a large number of model variants would be impractical, we use simple mathematical models in the exploration phase, to estimate the time/energy and memory complexity of each point.

For time and energy estimation, we count the operations required for the two steps involved when using the model:

- **Feature Extraction**: Since all considered features can be extracted with $O(N_{samples})$ operations, where $N_{samples} = \frac{W}{f_s}$ is the number of samples in a window, we estimate feature extraction time as $\frac{W}{f_s} \cdot N_f$, where N_f is the number of features.
- **Model Evaluation**: MLP evaluation cost is obtained counting the total number of multiply-and-accumulate (MAC) operations, while GBT cost is obtained by counting the number of branching operations.

Since both phases consist mainly of arithmetic operations (with no division), we use the above two time quantities also as proxies of energy consumption: this is reasonable as the exploration relies on relative comparisons and it is architecture-independent, so we can reasonably assume that the energy cost is roughly equivalent to the execution time times the average power cost of an arithmetic operation. With these models, we note that feature extraction time is one or two orders of magnitudes higher than model evaluation time, especially for GBTs.

Concerning memory, we estimate the cost of a MLP configuration as the number of bytes required to store the network weights and the two largest activation buffers [21]. For the GBT, we count the bytes required to store the ensemble data structure and the input feature buffer [20].

While simplified, these models are effective in preserving complexity rankings among different configurations, especially for simple hardware like a MCU. On the other hand, using them allows us to perform the entire training and hyper-parameters search process in less than 4 hours on a laptop.

IV. Results

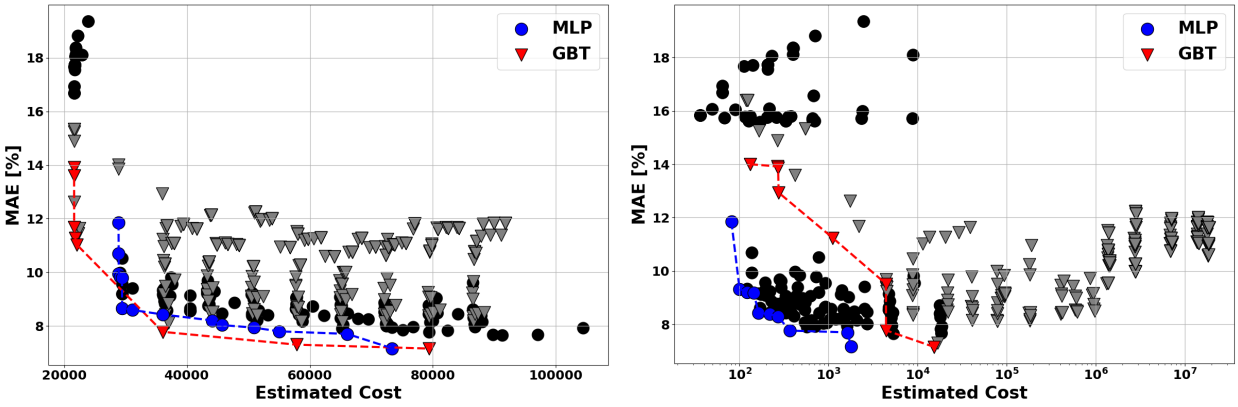
A. Setup

We train data-driven models using the Scikit-Learn and Keras Python libraries for GBT and MLP respectively. As target embedded device we consider the ultra-low-power RISC-V MCU PULPissimo [22], onto which we deploy both GBTs and MLPs using optimized libraries written in C. For GBTs, we leverage an in-house implementation, similar to the one described in [20] for random forests, whereas for the MLPs we use a single-core version of the PULP-NN library [21]. Time and energy results refer to a 22nm realization of PULPissimo working at 205.1 MHz [22].

B. Dataset Generation

Simulation data is generated by configuring the model of [5] with the default parameters. We then run a set of simulations, each lasting a maximum of 1,000 hours, or until battery SOH reaches 0. Each simulation uses either a different temperature or a different current profile. Specifically, we consider ambient temperature values in the [10°C : 40°C] range with a step of 5°C. We stimulate the battery model with constant, square-wave and "random walk" load current profiles for both charge and discharge cycles, with values ranging from $\pm$0.25A to $\pm$2A. Each current pulse in a random walk has a duration of $\approx$1 minute, whereas square waves have a period of $\approx$30 minutes. Combining these conditions, we obtained 110 simulations that, once aggregated in non-overlapping windows of length $W = 2$ hours, gave us 17,842 samples.

We split those data into training, validation and test sets with a 50/20/30% proportions for all our experiments. Importantly, the split is performed *at the simulation level* (i.e., not at the window level), meaning that windows belonging to the same simulation *cannot* be simultaneously present, for instance, in the training and test sets. This is the most realistic scenario, since in order to perform well, the data-driven models must



(a) Time/Energy

(b) Memory

Fig. 4: Pareto-fronts obtained from the hyper-parameters exploration of data-driven models

learn to extrapolate the ΔSOH for different simulation conditions w.r.t. those seen during training.

Notably, this setup would allow us to easily generate more data, for example conducting an error analysis to identify the T and I conditions in which our GBT/MLP models perform worse, and enhancing the dataset accordingly.

C. Pareto Analysis

Figure 4 shows the results of model space exploration for data driven models. The x axes of the two plots report the time/energy and memory estimated cost respectively, according to the models of Sec. III-D, whereas the y axis reports the ΔSOH MAE with respect to the simulation model, in percentage. In both charts, each dot/triangle refers to one MLP/GBT hyper-parameter and input features configuration, and the blue/red points highlight the respective Pareto fronts.

Tuning models configurations, we obtain Pareto-optimal solutions spanning a 4x range in estimated time/energy and more than 2 order of magnitudes in memory occupation, with MAEs ranging between 7% and 14%. GBT models achieve superior results in terms of error vs. time/energy trade-off, but have higher estimated memory than MLPs. This demonstrates the flexibility achievable by selecting from the rich spectrum of data-driven model architectures.

Table I reports the detailed results of the *extremes* of the two Pareto curves. Namely, for each model type, we report the configuration achieving the lowest-error (-E suffix), the lowest estimated time (-T) and the lowest estimated memory (-M). Note that the latter two coincide for the MLP. Besides the precise number of features and hyper-parameter setting, and the MAE, we also report two additional error metrics, i.e., the Mean Squared Error (MSE) and the R^2 score. These results show that both hyper-parameter tuning and feature selection are important to find optimal data-driven model configurations.

D. Deployment Results

Figure 5 reports the results of deploying all Pareto-optimal models from Figure 4 on PULPissimo, thus replacing the cost estimates with actual latency and memory (data plus code)

TABLE I: Extremes of the Pareto-curve.

GBT

Model	# Feat.	# Trees	Max Depth	MAE [%]	MSE [%]	R^2
GBT-E	11	50	5	7.15	0.96	0.729
GBT-T	3	5	2	13.92	2.87	0.182
GBT-M	4	5	1	14.00	3.14	0.107

MLP

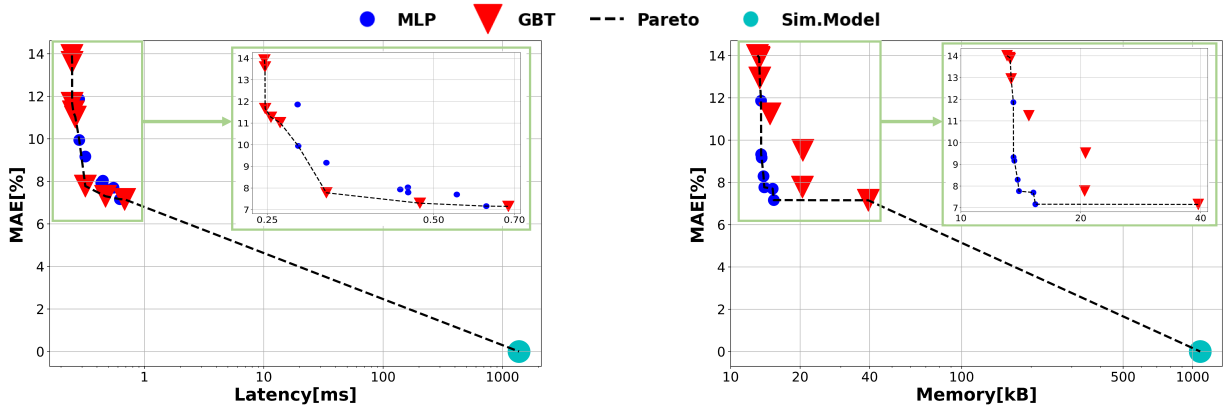
Model	# Feat.	# Layers	Hidden Size	MAE [%]	MSE [%]	R^2
MLP-E	10	3	128	7.16	0.93	0.736
MLP-T	4	3	8	11.86	2.04	0.420
MLP-M	4	3	8	11.86	2.04	0.420

measures on the target. The detailed deployment results for the extremes of the MLP and GBT Pareto fronts (same models of Table I) are also reported in the first six rows of Table II, in terms of number of clock cycles, latency, total memory occupation, and energy consumption per regression.

In order to compare time and memory costs of our data-driven models against those incurred by deploying a simulation-based model on-device, we compiled our ground truth reference from [5] to a binary executable, through the Simulink coder toolbox. We targeted a laptop-class CPU (Apple M1 Pro), since [5] turned out to be impossible to compile for our RISC-V embedded target, as the Simulink coder relies on pre-compiled, x86-only support libraries, and the memory required by this model exceeds the 512kB L2 available on the target.

To perform a fair comparison, we also compiled our lowest-MAE data-driven model (GBT-E) for the M1 Pro, using the C-based library of [20]. The two rows marked with † in Table II report the corresponding results. The last row estimates the figures of the simulation model on PULPissimo by re-scaling the simulation model results on M1 Pro to the RISC-V, using the ratio of the results obtained by GBT-E on the two platforms as proportionality factors. The corresponding latency and memory values are also represented as light-blue dots in Figure 5.

The results demonstrate the flexibility of a data-driven approach: we obtain configurations with latency, energy and memory values that vary approximately by a factor of 3 (e.g., from 25ms/0.94μJ to 68ms/2.6μJ per regression and from 13.1



(a) Time (b) Memory

Fig. 5: Deployed data-driven models and comparison with the simulation-based model.

TABLE II: Deployment Results.

Model	MAE [%]	Cycles	Latency [ms]	Memory [kB]	Energy [μJ]
GBT-E	7.15	$140{\cdot}10^3$	0.68	39.46	2.6
GBT-T	13.92	$51{\cdot}10^3$	0.25	13.26	0.94
GBT-M	14.00	$58{\cdot}10^3$	0.28	13.06	1.04
MLP-E	7.16	$122{\cdot}10^3$	0.6	15.38	2.36
MLP-T	11.86	$58{\cdot}10^3$	0.28	13.54	1.01
MLP-M	11.86	$58{\cdot}10^3$	0.28	13.54	1.01
Simulation-model Comparison					
GBT-E†	7.2	$560{\cdot}10^3$	0.20	49.15	n.a.
Simul.†	0	$127{\cdot}10^7$	395	1343.49	n.a.
Simul.*	0	$31 \cdot 10^6$	1377	1078.57	n.a.

† Results collected on Apple M1 Pro
* Results scaled from those on Apple M1 Pro

to 39.5 kB), with corresponding MAE values ranging from 7.2 to 14%. Considering that we use a window of 2 hours as input to the models, the energy consumption values obtained by the data-driven solutions can be considered completely negligible.

Furthermore, on the M1 Pro CPU, our lowest error model pays a 7.2% MAE in exchange for a striking 2,000x reduction in latency and 27x reduction in memory, with respect to directly executing a simulation-based SOH model. The latter, when scaled to the RISC-V platform, would require more than 1s to execute, implying a much higher energy overhead, and more than 1MB of total memory, which would exceed the available space on most embedded microcontrollers.

V. Conclusions

Data-driven models are the most suitable option for a digital twin of a battery to be hosted on-board a BMS, but their fidelity strongly depends on the quality of the training dataset. We have shown that it is possible to use a simulation model to generate an arbitrarily large dataset in a much smaller time than that required by datasets obtained through measurements. As results showed, this option also allows a tradeoff between model accuracy and model execution time or memory.

References

[1] S. Tamilselvi *et al.*, "A review on battery modelling techniques," *Sustainability*, vol. 13, no. 18, 2021.

[2] S. A. Hasib *et al.*, "A comprehensive review of available battery datasets, rul prediction approaches, and advanced battery management," *IEEE Access*, vol. 9, pp. 86 166–86 193, 2021.

[3] C. Vidal *et al.*, "Machine learning applied to electrified vehicle battery state of charge and state of health estimation: State-of-the-art," *IEEE Access*, vol. 8, pp. 52 796–52 814, 2020.

[4] F. Heinrich *et al.*, "A comprehensive study on battery electric modeling approaches based on machine learning," *Energy Inform*, vol. 4, no. 17, 2021.

[5] MathWorks, "Battery state-of-health estimation," https://it.mathworks.com/help/simscape-battery/ug/battery-state-of-health-estimation.html, 2023.

[6] A. Millner, "Modeling lithium ion battery degradation in electric vehicles," in *IEEE CITRES*, 2010, pp. 349–356.

[7] A. Bocca *et al.*, "An aging-aware battery charge scheme for mobile devices exploiting plug-in time patterns," in *33rd IEEE ICCD*, 2015, pp. 407–410.

[8] J. Vetter *et al.*, "Ageing mechanisms in lithium-ion batteries," *Journal of Power Sources*, vol. 147, no. 1, pp. 269–281, 2005.

[9] B. Xu *et al.*, "Modeling of lithium-ion battery degradation for cell life assessment," *IEEE Trans Smart Grid*, vol. 9, no. 2, pp. 1131–1140, 2018.

[10] G. Vennam *et al.*, "A survey on lithium-ion battery internal and external degradation modeling and state of health estimation," *Journal of Energy Storage*, vol. 52, p. 104720, 2022.

[11] G. dos Reis *et al.*, "Lithium-ion battery data and where to find it," *Energy and AI*, vol. 5, no. 1, 2021.

[12] M. Petricca *et al.*, "An automated framework for generating variable-accuracy battery models from datasheet information," in *IEEE/ACM ISLPED*, 2013, pp. 365–370.

[13] V. Barreras *et al.*, "Datasheet-based modeling of li-ion batteries," in *2012 IEEE Vehicle Power and Propulsion Conference*, 2012.

[14] A. Bocca *et al.*, "Composable battery model templates based on manufacturers' data," *IEEE Design & Test*, vol. 35, no. 3, 2017.

[15] B. Balagopal and M.-Y. Chow, "The state of the art approaches to estimate the state of health (soh) and state of function (sof) of lithium ion batteries," in *13th IEEE INDIN*, 2015, pp. 1302–1307.

[16] K. A. Smith *et al.*, "Model-based electrochemical estimation and constraint management for pulse operation of lithium ion batteries," *IEEE Trans. on Control Systems Technology*, vol. 18, no. 3, pp. 654–663, 2010.

[17] O. Erdinc *et al.*, "A dynamic lithium-ion battery model considering the effects of temperature and capacity fading," in *2009 ICCEP*, 2009, pp. 383–386.

[18] M. Cacciato *et al.*, "Real-time model-based estimation of soc and soh for energy storage systems," *IEEE Trans. Power Electron.*, vol. 32, no. 1, pp. 794–803, 2017.

[19] D. Haifeng *et al.*, "A new soh prediction concept for the power lithium-ion battery used on hevs," in *2009 IEEE VPPC*, 2009, pp. 1649–1653.

[20] F. Daghero *et al.*, "Adaptive random forests for energy-efficient inference on microcontrollers," in *IFIP/IEEE 29th VLSI-SoC*, 2021, pp. 1–6.

[21] A. Garofalo *et al.*, "PULP-NN: accelerating quantized neural networks on parallel ultra-low-power RISC-V processors," *Philos. Trans. Royal Soc. A*, vol. 378, no. 2164, p. 20190155, 2 2020.

[22] P. D. Schiavone *et al.*, "Quentin: an ultra-low-power pulpissimo soc in 22nm fdx," in *IEEE S3S*, 2018, pp. 1–3.