Enhancing Intelligent Battery Management on Automotive Microcontrollers Using LightGBM

# Enhancing Intelligent Battery Management on Automotive Microcontrollers Using LightGBM

Khaled Alamin, Chao Song, Sara Vinco, Francesco Daghero
Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy
Email: {firstname.lastname}@polito.it

***Abstract*—State-of-Health (SoH) is a core metric of any Battery Management System (BMS), impacting operational safety and performance optimization. However, its on-board estimation remains challenging due to the dynamic nature of battery data and the tight memory and energy constraints of embedded systems, requiring frequent yet impractical model updates. We present a framework for training and deploying ensembles of decision trees on BMS hardware, delivering accurate SoH estimates across the entire battery lifespan without model updates, while also being resource-inexpensive on on-board Microcontrollers (MCUs). On two benchmark datasets, our method achieves up to 39% lower MAE than state-of-the-art deep-learning baselines. Moreover, when deployed on an industry-grade automotive MCU, it delivers inference speedups of 11–30x while occupying under 3% of the available memory.**

***Index Terms*—Battery management system, Lithium-ion batteries, State-of-health estimation, Edge computing, Light gradient boosting machine, Automotive microcontrollers**

## I. Introduction

Accurate on-vehicle estimation of State of Health (SoH) and State of Charge (SoC) is crucial for electric vehicles (EVs): it enables optimized charging strategies, better range prediction, and enhanced overall safety and reliability [1], while its uncertainty is a primary obstacle to fast-charging and residual-value guarantee [2].

Conventional battery modeling techniques generally rely on Equivalent Circuit Models (ECMs), computationally efficient but with limited accuracy [3], or ElectroChemical Frameworks (ECFs), highly precise yet too heavy for real-time embedded deployment [4]. Both strategies lose accuracy and/or increase complexity outside their calibrated temperature/load envelope and require labor-intensive parameter tuning [5].

Recent Machine Learning (ML) approaches learn battery dynamics directly from data and can reach minimal Mean Absolute Error (MAE) on public benchmarks, yet most remain off-board [6]. Cloud-centric solutions introduce unacceptable latency, bandwidth demands, security risks, and require continuous connectivity [7]. On the other hand, edge-based ML addresses such issues by offloading inference to the BMS controller itself; however, with strict constraints on model size, memory footprint, and computational load [8]. Moreover, updating an edge-deployed model in the field is logistically challenging and resource-intensive. This update challenge is particularly critical for SoH estimation, because battery aging gradually changes the duration and profile of charge/discharge cycles, rendering a fixed inference time window increasingly suboptimal [9].

To address this challenge, we introduce a framework for training and deploying LightGBM ensembles on edge automotive-grade MCUs. These ensembles can accommodate a wide range of input window lengths, eliminating the need for frequent model updates. Specifically, by segmenting training data into variable-length windows, our models remain robust to evolving operating conditions.

We applied the proposed strategy to two public NASA datasets [10], [11], and achieved up to 39% MAE reduction versus fixed-window deep-learning baselines. When deployed on an automotive-grade microcontroller, our LightGBM solution runs up to 11× faster while occupying under 3% of the available memory.

## II. Related Work

Data-driven approaches mostly focus on the modeling challenge, while the online deployment on actual hardware, either on edge or on the cloud or both, for real-time BMS applications has not yet been fully addressed in the literature [6].

One of the few works covering all aspects, from modeling to deployment on BMS hardware, is [12]. This paper discusses bidirectional LSTM (BiLSTM) model development from idea to deployment to a 120 MHz SPC58 MCU, specifically designed for automotive tasks, as is thus the baseline for this work. Their approach relies on several limiting assumptions. It assumes that discharge cycles start with a fully charged battery (SoC equal to 100%), which is unrealistic in real-world driving, where trips can start at arbitrary SoC levels. It also fixes the window length to 20 samples, meaning a new model must be trained whenever the sampling rate or trip duration changes, as dynamic-length inputs for DL models are not supported on embedded HW targets. Finally, training, validation, and test sets are often drawn from the same battery cell, overlooking the cell-to-cell and chemistry variations common in practical deployments.

## III. Methodology

This work proposes, to the best of our knowledge, the first framework to train accurate input length agnostic models for on-device SoH estimation. In this Section, we detail the design choices behind our framework.

### A. Framework flow

Figure 1 shows the training and deployment workflow that we propose in our framework. Input data reflects typical

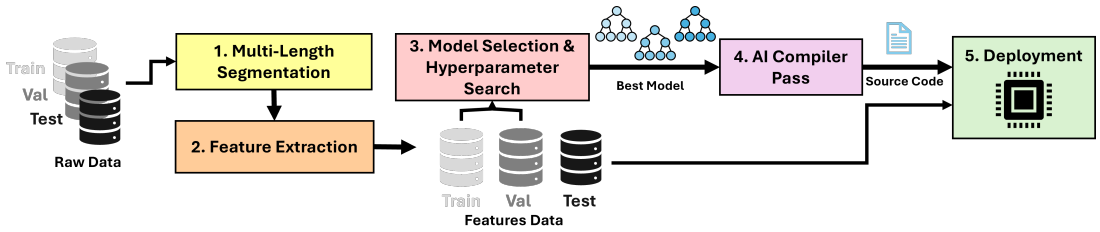


Figure 1. High-Level overview of the proposed framework

datasets, and includes charge-discharge logs featuring temperature ($T$), voltage ($V$), current ($i$), and sampling timestamp ($t$) collected with variable or fixed frequency.

*1. Multi-Length Segmentation:* In the first phase, we segment the input signals in windows of length $wlen \in [20, 30, 40, 50, 60]$, without overlap. While the number of windows and the overlap can vary depending on the user's needs or the sampling rate of the dataset, we select the previous values as they are the most commonly employed in the literature.

*2. Feature Extraction:* In this step, we take inspiration from [13] and extract a set of relevant features for batteries' age modeling from the available raw data. Specifically, we compute the derivative of voltage ($dV$) and of temperature ($dT$), power ($P$), capacity ($Q$), and the window-relative timestamp (i.e., the first element of the window always starts from 0). To represent battery dynamics within each window, we compute the discharge rate for both voltage and temperature to capture how quickly the battery loses energy, which is indicative of cell behavior under load and is affected by both operating voltage and thermal conditions. We also compute the derivative of voltage with respect to capacity ($dV/dQ$), a signal informative for characterizing internal electrochemical processes [14].

As a final step, we extract a fixed-size feature vector by averaging the values of these signals over the window. In addition, since $dV/dQ$ carries strong diagnostic value, we include its minimum and maximum values over the window to capture its range and potential nonlinearities. The resulting feature vector consists of 10 elements per window: the mean discharge voltage rate, the mean voltage, the mean discharge temperature rate, the mean power, the mean timestep, the mean, maximum, and minimum of $dV/dQ$, the total duration of the window, and the mean temperature.

*3. Model selection & Hyperparameter Search:* The choice of data-driven algorithms can significantly impact the performance and resource allocation of the model. Gradient-boosted decision trees, and LightGBM in particular, recently emerged as a flexible solution, as they match neural networks in SoH accuracy while offering millisecond-level inference, trivial model compression, and input-length agnosticism. Additionally, the same ensemble can consume feature vectors engineered from any window length, making it resilient to variable sampling frequencies, missing data, and partial discharge events. For this reason, our work adopts LightGBM regressors as the baseline. Then, to find the best hyperparameter set for these models, we employ a Bayesian search, whose goal is to minimize the Mean Absolute Error (MAE) on the validation set. We report in Table I the search space. Notably, we restrain the maximum depth and the $\#Leaves$ upper bounds in the search space to avoid models that are too large to be deployed at the edge. This search is performed with the Optuna framework, which efficiently explores the parameter space by balancing exploration and exploitation.

Table I
HYPERPARAMETER SEARCH SPACE

| Parameter | Search Space |
|---|---|
| Learning Rate (LR) | $[10^{-5}, 10^{-1}]$ |
| Max Depth | $[3, 8]$ |
| #Leaves | $[2, 25]$ |
| $\alpha/\beta$-reg | $[10^{-6}, 1]$ |
| #Estimators | $[50, 450]$ |
| Min. Child Samples (MCS) | $[2, 25]$ |
| Col.Samples By Tree (CST) | $[0.1, 1]$ |

*4. AI Compiler Pass:* The top-performing model from the previous step is converted to lightweight static C source code that can execute efficiently on the hardware target. Since such conversion is not directly supported by [15] for LightGBM models, we extend an open-source AI compiler named EDEN [16], automatically generating source code for efficient inference for tree-based ensembles. The model is stored using three arrays, each with one entry per node: the splitting threshold, the splitting feature index, and the index of the right child. Nodes are in pre-order, so the left child is implicitly the next node in memory. For leaf nodes, the feature index is set to $\#\text{Features} + 1$, and the threshold array holds the predicted value. A fourth array stores the root indices of each tree in the ensemble for fast iteration.

*5. Deployment:* The source code generated by the AI compiler is deployed on the target hardware, together with a hand-written custom implementation of the feature extraction step. The model is finally evaluated to collect metrics including memory usage, inference time, cycle count, and MAE.

## IV. RESULT

### A. Experimental Setup

We evaluated our proposed framework on the SPC58EC80E5 [15] microcontroller by STMicroelectronics, based on a dual-core e200z420n3 architecture specifically designed for automotive applications.

To benchmark our approach, we used two publicly available and widely recognized datasets. The *NASA Battery Dataset (NBD)* [10] consists of discharge cycle data from Lithium-Ion (Li-ion) batteries operating at room temperature. For this dataset, we follow the preprocessing and data partitioning protocol presented in [12], estimating directly the battery capacity. The second dataset, the *NASA Randomized Dataset*

Table II
BEST HYPERPARAMETERS SETS

| Dataset | LR | Max Depth | #Leaves | #Estimators | $\alpha$-reg | $\beta$-reg | MCS | CST |
|---|---|---|---|---|---|---|---|---|
| NBD | 0.147 | 8 | 25 | 384 | 6.5e-2 | 6.7e-5 | 6 | 0.46 |
| NRD | 0.085 | 8 | 18 | 443 | 1.9e-6 | 3.6e-4 | 17 | 0.61 |

*(NRD)* [11], contains data collected from 28 Lithium Cobalt Oxide (LCO) 18650 batteries subjected to randomized charge and discharge cycles. The goal is to estimate the SoH value of each window. Both datasets include measurements of voltage, current, temperature, sampling timestamp, and SoH, sampled at frequencies ranging from 0.09 Hz (NBD) to 0.33 Hz (NRD).

All deployment results refer to floating-point models obtained either through our optimized flow for LightGBM or via SPC5Studio for LSTM models. Performance profiling in terms of execution cycles, flash memory usage, and RAM occupation was conducted using SPC5Studio. Table II reports the best hyperparameter sets found with the search detailed in III for LightGBMs. BiLSTMs instead have been trained for 100 epochs with a learning rate of 0.01 and a batch size of 16.

### B. SOTA Comparison

Table III shows a comparison between our LightGBM model (Ours) and the BiLSTM model introduced in [12]. Specifically, we report the MAE on the First Window (FW) per discharge cycle of the test data (as proposed in [12]), on the full test set (Full), the inference latency, and energy to perform an inference on a single input window. Notably, the model of [12] has been re-trained and deployed on our hardware target for consistent latency and energy measurements, while keeping the data split identical (FW) for a fair comparison.

Table III
STATE OF THE ART COMPARISON

| Work | MAE (FW) | MAE (Full) | Latency [ms] | Energy [uJ] |
|---|---|---|---|---|
| [12] | 0.0202 | 0.0194 | 4.656 | 279.36 |
| **Ours** | **0.0167** | **0.0188** | **0.403** | **24.18** |

Our LightGBM model outperforms the BiLSTM model in terms of MAE both on the FW benchmark and on the full test data, respectively reducing it by 17% and 3%. In terms of latency, tree ensembles are significantly faster, reducing the average inference time by an order of magnitude (11.55×) w.r.t. BiLSTMs. This translates to lower impact on the operational capabilities of the BMS, together with a reduction in terms of energy per inference of the model, going from 279.36 uJ to 24.18 uJ. Considering the prediction frequency, i.e., once every 20 samples, our model leads to significant energy savings during its lifetime on the vehicle, together with a negligible impact on the operational capabilities of the BMS.

### C. LightGBM vs BiLSTM

Figure 2 compares the MAE of a single LightGBM model with that of a BiLSTM trained for specific window lengths. Across both datasets, our approach consistently achieves lower

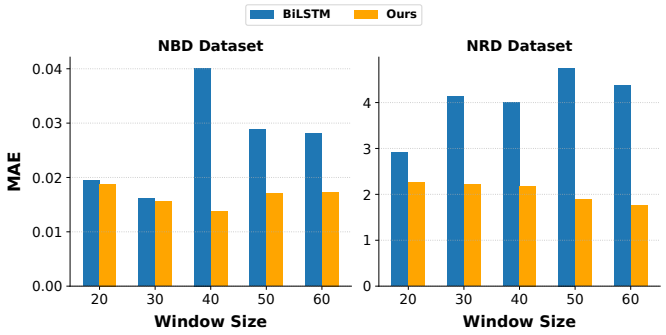

Figure 2. SoH prediction error vs. window length ($wlen$)

MAE regardless of the window length. On the NBD dataset, the performance gap is modest (less than 0.006), as BiLSTM performs well with shorter windows, while our feature set benefits from longer temporal contexts. This advantage becomes more evident at window lengths $\geq 40$, where the BiLSTM's MAE increases notably, rising by 0.0264 at $wlen = 40$ and 0.0108 at $wlen = 60$, whereas our model remains stable.

In the NRD dataset, which reflects a more complex and realistic deployment scenario involving multiple cells, our method shows a clear and consistent advantage. Even at $wlen = 20$, we achieve a 0.6517 lower MAE, and at longer windows, our model yields up to a 2.5× reduction in error. This highlights the robustness of our approach under varied operating conditions.

Table IV
DEPLOYMENT RESULTS. ABBREVIATIONS: L. (LIGHTGBM), B. (BILSTM), WL (WINDOW LENGTH)

| | WL | Model | MAE | MSE | Cyc. | RAM% | Flash% |
|---|---|---|---|---|---|---|---|
| NBD | 20 | L | **0.0188** | 0.0008 | **72.6k** | 0.49 | 2.71 |
| | 20 | B | 0.0194 | **0.0007** | 838k | **0.13** | **0.21** |
| | 30 | L | **0.0156** | 0.0006 | **74.7k** | 0.49 | 2.71 |
| | 30 | B | 0.0162 | **0.0005** | 1.25M | **0.13** | **0.21** |
| | 40 | L | **0.0137** | **0.0004** | **77.1k** | 0.49 | 2.71 |
| | 40 | B | 0.0401 | 0.0026 | 1.67M | **0.13** | **0.21** |
| | 50 | L | **0.0171** | **0.0008** | **79.0k** | 0.49 | 2.71 |
| | 50 | B | 0.0288 | 0.0014 | 2.08M | **0.13** | **0.21** |
| | 60 | L | **0.0173** | **0.0006** | **81.3k** | 0.49 | 2.71 |
| | 60 | B | 0.0281 | 0.0013 | 2.50M | **0.13** | **0.21** |
| NRD | 20 | L | **2.2561** | **12.622** | **73.3k** | 0.49 | 2.23 |
| | 20 | B | 2.9078 | 14.897 | 838k | **0.13** | **0.21** |
| | 30 | L | **2.2078** | **11.788** | **75.4k** | 0.49 | 2.23 |
| | 30 | B | 4.1269 | 27.588 | 1.25M | **0.13** | **0.21** |
| | 40 | L | **2.1831** | **10.498** | **77.8k** | 0.49 | 2.23 |
| | 40 | B | 4.0019 | 28.180 | 1.67M | **0.13** | **0.21** |
| | 50 | L | **1.8812** | **7.137** | **79.6k** | 0.49 | 2.23 |
| | 50 | B | 4.7511 | 40.870 | 2.08M | **0.13** | **0.21** |
| | 60 | L | **1.7509** | **7.357** | **82.0k** | 0.49 | 2.23 |
| | 60 | B | 4.3676 | 30.323 | 2.50M | **0.13** | **0.21** |

Table IV presents detailed deployment metrics, including MAE, MSE, inference cycles (Cyc.), RAM usage, and Flash usage. Our LightGBM-based model (L) consistently outperforms BiLSTMs (B) across all window lengths on the NRD dataset, achieving MSE reductions ranging from 1.18× ($wlen = 20$) to 5.72× (wlen=50). On the NBD dataset, our model yields lower MAE and MSE for all window lengths $wlen \geq 40$,

with MSE improvements of up to 6.5×. For shorter windows ($wlen < 40$), the MSE is only marginally higher, by less than $10^{-4}$, compared to the BiLSTM.

In terms of inference cycles, tree ensembles offer significant advantages. Even when accounting for feature extraction (which constitutes 6.37%–15.99% of the total inference time), our model achieves speedups ranging from 11.5× to 30.7× over BiLSTM. While BiLSTM inference cost scales sharply with window length, the cost for LightGBM remains nearly constant, growing by fewer than 10k cycles from $wlen = 20$ to 60, due solely to the feature extraction step.

However, LightGBM models are more memory-intensive. Both RAM and Flash usage are higher than for BiLSTM models. The increased RAM footprint arises from the feature extraction step, which is less optimized than the deep learning primitives provided by the hardware vendor for BiLSTMs. Flash usage grows with model size, following $O(\#\text{Trees} \cdot 2^{\text{depth}})$, as each tree node requires 64 bits. Nevertheless, LightGBM models remain lightweight in absolute terms, using less than 3% of available Flash and under 0.5% of RAM, well within acceptable limits for BMS deployment, especially considering the substantial gains in accuracy and inference speed.

### D. Generalization to Unseen Windows

Real-world deployment scenarios may require adapting the window size dynamically, without the overhead of retraining or updating models. In this section, we evaluate the robustness of our solution against BiLSTMs when tested on window lengths unseen during training. Since embedded hardware does not support variable-length inputs for BiLSTMs, we simulate unseen window lengths by trimming the oldest timesteps for larger-than-trained inputs, and by padding shorter sequences. In contrast, our *single* LightGBM model naturally accommodates arbitrary window lengths without modification.

Figure 3 reports MAE heatmaps for our model (L) and BiLSTMs trained on window lengths $wlen \in [20, 30, 40, 50, 60]$ (denoted as $B_{\text{wlen}}$) and evaluated on a different set ($wlen \in [10, 35, 55, 75]$). Our method consistently outperforms BiLSTMs, matching performance at $wlen = 10$ and improving MAE by up to 26.72× (NBD) and 12.26× (NRD).

On NBD, BiLSTM results are erratic, with no correlation between training and testing window proximity and MAE. In contrast, our model achieves a stable average MAE of 0.046.

Similarly, on the more complex NRD dataset, BiLSTMs exhibit large fluctuations in MAE, peaking sharply (28.2) when evaluated at $wlen = 75$ after training on $wlen = 60$. Our model remains robust, maintaining a low average MAE of 2.4, confirming its reliability under varying operating conditions.

**(a) NBD**

| **M** | 10 | 35 | 55 | 75 |
|---|---|---|---|---|
| L | 0.041 | 0.023 | 0.025 | 0.095 |
| $B_{20}$ | 0.106 | 0.491 | 0.492 | 0.274 |
| $B_{30}$ | 0.059 | 0.042 | 0.504 | 0.497 |
| $B_{40}$ | 0.102 | 0.073 | 0.668 | 0.660 |
| $B_{50}$ | 0.097 | 0.068 | 0.033 | 0.643 |
| $B_{60}$ | 0.041 | 0.023 | 0.025 | 0.095 |

**(b) NRD**

| **M** | 10 | 35 | 55 | 75 |
|---|---|---|---|---|
| L | 3.7 | 2.3 | 1.7 | 2.3 |
| $B_{20}$ | 5.3 | 15.3 | 7.5 | 10.8 |
| $B_{30}$ | 5.5 | 19.8 | 19.9 | 12.6 |
| $B_{40}$ | 6.8 | 4.2 | 24.0 | 24.5 |
| $B_{50}$ | 7.5 | 4.8 | 21.7 | 22.2 |
| $B_{60}$ | 8.9 | 4.8 | 4.2 | 28.2 |

Figure 3. MAE on unseen window lengths ((10,35,55,75)). **M** = Model, **L** = LightGBM, $\mathbf{B_{XX}}$ = BiLSTM trained on window of length $XX$.

## V. Conclusions

In this work, we presented a lightweight and deployment-friendly approach for State-of-Health (SoH) prediction in battery management systems. By combining a compact feature extraction pipeline with a single LightGBM model, our method achieves competitive accuracy with significantly lower computational cost compared to BiLSTM-based solutions.

Experimental results on two public datasets demonstrate that our approach not only generalizes well across different window lengths but also provides substantial improvements in inference efficiency, achieving up to 30× faster inference and reducing MAE by up to 12× in complex scenarios. Despite a modest increase in memory usage, our model remains well within the resource constraints of typical embedded targets, using less than 3% of Flash and 0.5% of RAM.

Moreover, we show that our model exhibits strong robustness to unseen window lengths, a critical feature for real-world deployment where operating conditions may vary dynamically. These results suggest that our method offers a practical and scalable solution for SoH estimation on constrained hardware.

## References


[1] M. El Marghichi *et al.*, "Improving accuracy in state of health estimation for lithium batteries using gradient-based optimization: Case study in electric vehicle applications," *Plos one*, 2023.

[2] G. L. Plett, "Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs: Part 1. Background," *Journal of Power sources*, 2004.

[3] X. Lai *et al.*, "A comparative study of different equivalent circuit models for estimating state-of-charge of lithium-ion batteries," *Electrochimica Acta*, vol. 259, pp. 566–577, 2018.

[4] M. Waseem *et al.*, "Battery technologies and functionality of battery management system for evs: Current status, key challenges, and future prospectives," *Journal of Power Sources*, vol. 580, p. 233349, 2023.

[5] H. He *et al.*, "Evaluation of lithium-ion battery equivalent circuit models for state of charge estimation by an experimental approach," *Energies*, vol. 4, no. 4, pp. 582–598, 2011.

[6] Z. Nozarijouybari *et al.*, "Machine learning for battery systems applications: Progress, challenges, and opportunities," *Journal of Power Sources*, vol. 601, 2024.

[7] M. Ismail *et al.*, "Understanding and overcoming the challenges of building high voltage automotive battery management systems," *Infineon*, 2022.

[8] A. Wheeldon *et al.*, "Low-latency asynchronous logic design for inference at the edge," in *Proc. of DATE*, 2021, pp. 370–373.

[9] T. Kim *et al.*, "Online SOC and SOH estimation for multicell lithium-ion batteries based on an adaptive hybrid battery model and sliding-mode observer," in *Proc. of IEEE ECCE*. IEEE, 2013, pp. 292–298.

[10] B. Saha *et al.*, "Battery data set," *NASA AMES prognostics data repository*, 2007.

[11] B. Bole *et al.*, "Adaptation of an electrochemistry-based li-ion battery model to account for deterioration observed under randomized use," in *PHM*, 2014.

[12] D. Sammartino *et al.*, "Intelligent resource constrained battery management on automotive microcontrollers," in *IEEE GCCE*, 2021.

[13] K. S. S. Alamin *et al.*, "Model-driven feature engineering for data-driven battery soh model," in *Proc. of DATE*, 2024, pp. 1–6.

[14] R. Fathi *et al.*, "Ultra high-precision studies of degradation mechanisms in aged LiCoO2/Graphite Li-Ion cells," *Journal of The Electrochemical Society*, vol. 161, 2014.

[15] STMicroelectronics. [Online]. Available: https://www.st.com/en/automotive-microcontrollers/spc58ec80e5.html

[16] F. Daghero *et al.*, "Dynamic decision tree ensembles for energy-efficient inference on iot edge nodes," *IEEE Internet of Things Journal*, 2023.