Journal of Energy Storage 152 (2026) 120673

Contents lists available at ScienceDirect

Journal of Energy Storage

journal homepage: www.elsevier.com/locate/est

Research papers

Embedded strategy for battery module states estimation using tiny machine learning models

Spyridon Giazitzis [a,*], Jack Ferrari [b], Antoni J. Woss [c], Susheel Badha [d], Filippo Rosetti [d], Emanuele Ogliari [a]

[a] Department of Energy, Politecnico di Milano, Via Raffaele Lambruschini 8, Milano, MI 20156, Italy
[b] MathWorks, 1 Apple Hill Dr, Natick, Massachusetts, MA 01760, USA
[c] MathWorks, 1 Cambridge Science Park Milton Rd, Cambridge, CB4 0JL, UK
[d] Infineon Technologies, Siemensstraße 2, Villach, 9500, Austria

ARTICLE INFO

Keywords:
State of charge
State of health
TinyML
Embedded machine learning
Cell-to-module
Model optimization

ABSTRACT

The growing adoption of battery powered systems demands advanced Battery Management System (BMS) solutions for accurate State of Charge (SoC) and State of Health (SoH) estimation, ensuring safe and efficient operation. To address this need, the authors propose a practical and scalable algorithm based on state-of-the-art TinyML models capable of handling variable-length input sequences without padding, for the SoH estimation. These models enable embedded SoC and SoH estimation at the module level, effectively functioning as smart sensors within a BMS. Four deep learning architectures were developed for battery state estimation, based on CNN, GRU, LSTM, and Dense layers. The models were converted into TinyML compatible formats using an innovative compression technique based on neural network projection. After projection, the models were fine-tuned for a few epochs to restore their accuracy, and then evaluated using metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), model size, and inference latency. Latency measurements were obtained on an RTX 3060 GPU and are reported only for relative model comparison. For SoC estimation, the models utilized a time window of 60 samples, each containing voltage, current, and temperature measurements, to produce accurate predictions throughout the first life phase of the battery (up to 70% SoH). The projected Pr_LSTM TinyML model was selected for deployment due to its minimal memory footprint (4.0 kB), low latency (0.391 ms), and low error rates (MAE: 0.0114, RMSE: 0.0286). For SoH estimation, the models were trained and tested using partially charging data of varying lengths, reflecting practical real-world conditions. Two input feature sets were considered: the Incremental Capacity Analysis (ICA) curve alone, and ICA combined with voltage data. The projected Pr_CNN_GRU TinyML model, using only ICA features, was selected for its high accuracy and compact size (3.4 kB), achieving an MAE of 0.0060, RMSE of 0.0130, and a latency of 1.607 ms. Finally, an embedded SoC/SoH estimation strategy was proposed to scale predictions from the cell to the module level, enabling potential real-time implementation on the Infineon Mobile Robot (IMR) BMS.

1. Introduction

The rapid growth of Artificial Intelligence (AI) across various fields, including electrical grids [1], robotics [2], and electric vehicles [3], is driven by significant advancements in hardware [4] (AI accelerators), software [5] (deep learning frameworks), and the increasing availability of massive datasets. These factors collectively enable the development of more efficient, accurate, robust, and powerful AI models. In the context of electric vehicles and other battery-powered applications, the development of a sophisticated Battery Management System (BMS) that can accurately estimate various states of battery cells, is essential for ensuring safety and optimal performance [6,7]. Since many critical battery states can not be directly measured using standard sensors, AI models can be deployed to provide precise estimations, working as smart sensors [8], enhancing the reliability and efficiency of such systems [9].

Several model-based and data-driven approaches have been developed for battery states estimation [10–12]. Data-driven models are particularly suitable for this transformation, as they do not require the development of complex partial differential equations or designing Equivalent Circuit Models (ECM) to describe internal electrochemical processes [13]. Moreover, by leveraging transfer learning techniques,

* Corresponding author.
E-mail address: spyridon.giazitzis@polimi.it (S. Giazitzis).

https://doi.org/10.1016/j.est.2026.120673
Received 1 September 2025; Received in revised form 29 December 2025; Accepted 17 January 2026
Available online 23 January 2026

they can maintain high accuracy across various battery cell types, thereby enhancing their robustness [14]. Furthermore, data-driven architectures are well suited for compression and hardware-oriented optimization, making them well-suited candidates for embedded or resource-constrained battery management applications.

To enable these benefits on microcontroller based platforms, Tiny Machine Learning (TinyML) models are utilized, providing data-driven solutions tailored for ultra low power and memory constrained hardware [15]. These models represent a specialized class of light weight neural networks designed to run efficiently on microcontrollers through several optimization techniques such as model compression and hardware–software co-design [16]. With recent advancements in embedded ML toolchains and optimization frameworks [17–19], TinyML has matured into a practical and widely adopted approach for deploying intelligent algorithms directly on edge devices. In particular, its ability to perform accurate real-time estimations within only a few kilobytes of memory, while remaining compatible with diverse microcontroller architectures, makes it highly suitable for applications requiring low latency and minimal energy consumption [20]. By leveraging these capabilities, TinyML enables battery state estimation algorithms to operate directly within BMS hardware, reducing dependence on external computation and enhancing system autonomy and efficiency.

In [21], the authors developed simple Convolutional Neural Network (CNN) and Artificial Neural Network (ANN) models for battery SoC estimation, achieving efficient performance with minimal memory and computational demands. They used MATLAB and Infineon software for model development, compression and deployment to the Infineon-Cypress microcontroller. The compressed ANN, int16 $\times$ 16 (16-bit weights with 16-bit activations), achieved the best Mean Absolute Error (MAE) of 2.81% in early cycles while occupying less than 3% of the available flash memory. The PoliMi-TUB single battery cell dataset [22] was used for training and testing the models. Similarly, in [23], the authors developed several data-driven models, including Feed-Forward Networks (FFN), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), Temporal Convolutional Networks (TCN), and Legendre Memory Units (LMU), for lithium-ion battery SoC estimation. These models were deployed on STM32H5731-DK microcontrollers (MCUs) and STM32MP257F-EV1 microprocessors (MPUs) using the STM32Cube.AI software. The parallel TCN architecture achieved the highest accuracy combined with low latency when deployed to a MCU. For this study, IEEE open-source datasets [24] were used for model development and validation. Lastly, in [25], the authors developed a TinyML based framework for battery SoC estimation in redox flow batteries (RFBs) using an LSTM-based deep learning model optimized for edge deployment. Post-training quantization was applied to compress the model and enable deployment on an ESP32 microcontroller. The dynamically quantized LSTM model achieved an RMSE of 0.043 and an MAE of 0.035 for all vanadium RFB, with a model size of 97.8 kB and an inference time of 1.72 s, outperforming other quantized variants in efficiency. Latency and memory consumption were measured directly on the ESP32 edge device, confirming its real-time applicability.

For the deployment of data-driven models for SoH estimation, a similar procedure was followed, using identical software tools to compress the models and convert them into TinyML compatible versions. In [26], a simple CNN and ANN were compared for SoH estimation, utilizing Incremental Capacity Analysis (ICA) [27] and voltage charging data. Various quantization techniques were evaluated, with testing conducted at two distinct degradation stages: one in the early life of the battery and the other near its 'knee-point' [28]. Except for the int8 $\times$ 8 quantized models, all approaches achieved satisfactory results. Likewise, in [29], the authors developed TinyML models for SoH estimation, focusing on the deployment of lightweight deep learning architectures for efficient edge computing. They implemented CNN, LSTM, GRU, CNN-LSTM, and CNN-GRU models, trained on NASA's Li-Ion battery dataset, using voltage, current, temperature, and time-series discharge data as input features. Among the tested models, CNN-GRU achieved the best performance, with an RMSE of 4.88% and MAE of 4.14%. Additionally, a comparative analysis of the TinyML models' characteristics, including memory footprint and computational requirements, was conducted using TensorFlow Lite for Microcontrollers (TFLM) [30] for model conversion.

Lastly, in [31], the authors developed several deep learning models, including CNN, CNN-GRU, and TCN, for Remaining Useful Life (RuL) estimation, which is based on predicting the trajectory of SoH values. They utilized TFLM for model compression and conversion into TinyML models, which were subsequently deployed on an Arduino board. The paper provides a detailed analysis of TinyML model errors, latency, and size reduction compared to the original models. Among the tested models, the TCN model demonstrated the best performance, achieving an error of less than 1% and an inference latency of under 5 ms.

Currently, there is a lack of research on optimizing and converting data-driven methods, especially deep learning models, into TinyML models suitable for deployment on microcontrollers. Most researchers focus on developing highly accurate models for states estimation without considering the computational and memory limitations, necessary for integration into such resource constrained devices. In the case of SoH estimation, where the length of input data can vary significantly, many studies rely on fixed-length inputs, which is impractical in real-world scenarios, especially when users, in most cases, partially charge their devices. Additionally, many state estimation algorithms are designed for single battery cells and do not offer a comprehensive strategy for scaling to the module level, which is more relevant for practical applications. Lastly, although RNN/LSTM layer support has only recently been added on some open-source libraries [30], their current optimization and compression implementations still assume fixed-length inputs, making variable-length sequence processing (without padding) unsupported. To address these challenges, the authors introduced several key innovations such as:

- Create state-of-the-art TinyML models, by combining CNN, LSTM, GRU and Dense layers, and converting them to their projected version, utilizing an innovative compression technique based on neural network projection [32], enabling them to act as smart sensors for embedded SoC and SoH estimation.
- Optimize and convert variable-length input models for SoH estimation into a deployment-ready format, leveraging ICA curves and voltage data obtained from partial battery-charging data, to improve accuracy and real-world usability.
- Develop an end-to-end algorithmic strategy, implementable in the BMS for Infineon's Mobile Robot (IMR), that upscales SoC and SoH estimation from the cell to the module level.

The remaining sections are organized as follows: Section 2 describes the most commonly used methods for battery states estimation, as reported in the literature. Section 3 provides a brief overview of up-scaling techniques for cell-to-module states estimation. Section 4 presents an analysis of a novel neural network compression technique that was utilized. Sections 5 and 6 detail the dataset used and the methods and results of the proposed models for states estimation, respectively. Finally, Section 7 concludes the work and discusses potential directions for future research.

2. State estimation methods for BMS

The primary responsibility of a BMS is to ensure the safety, reliability, and optimal performance of the battery pack. To achieve this, the BMS manages several key functions, including cell balancing, thermal management, states estimation, and fault detection [33]. Among these functions, accurate estimation of SoC and SoH is critical, as

they provide essential information to both user and control systems. This information is vital for making intelligent decisions about battery usage, predicting remaining range or capacity, and detecting potential safety risks, such as overcharging, deep discharging, or unexpected degradation [34].

SoC refers to the percentage of the remaining usable capacity of a battery cell, and its accurate estimation is necessary for the control of most BMS operations. It can not be directly measured by conventional sensors and instead, it must be estimated. The SoC value is influenced by various environmental factors (such as temperature, pressure, and humidity) and operational conditions (including C-rates and the cell's degradation state) [35]. For this reason, an accurate and robust SoC estimation model is essential for effective and reliable battery management.

Considering SoH estimation, there are two main definitions commonly used. The first defines SoH based on the loss of battery capacity over time, while the second relies on the increase in internal resistance [36]. The first definition is the most widely accepted, as it is more straightforward and does not require complex or time-consuming experimental procedures, unlike the second approach. Accurate SoH estimation plays a crucial role in evaluating the current condition of the battery. In addition to providing detailed insights into the historical operating conditions of the battery, it also facilitates the prediction of its future behavior and performance. In conclusion, accurate SoH estimation supports both diagnostic and prognostic functions in BMS.

According to the existing literature [37], numerous methods have been developed to accomplish this task, which can be broadly classified into three main categories: model-based, data-driven, and hybrid approaches, which will be presented more in detail in the following subsections. However, to develop an accurate and robust model, it is first necessary to conduct several experiments, under known and controlled conditions and to acquire precise measurements. In Fig. 1, a general framework is depicted that can be followed for developing and deploying both model-based and data-driven methods for SoC and SoH estimation. In this paper, the authors follow the data-driven approach for reasons that will be discussed in the following sections.

2.1. Laboratory methods for SoC/SoH calculation

This section provides a brief overview of methodologies and experiments that can be carried out in laboratory settings, where environmental and operational conditions are precisely known and controlled. Under these conditions, it is possible to accurately determine several battery states, such as SoC and SoH. These calculations rely on direct measurements rather than the development of predictive models, ensuring a high level of data accuracy. Such laboratory-based techniques are particularly important for model training and validation, where data quality is critical. However, due to their time-consuming nature and the fact that such controlled conditions are rarely achievable outside the laboratory, these methods are generally limited to laboratory environments and are not suitable for real-world applications.

For extracting the SoC information of the battery cells, initially cells should be fully charged according to the instructions of the datasheet that the battery manufacturing company provides, typically by using CC-CV as charging profile at specific temperature and C-rate current. In the CC stage, there is a constant current that is applied to the battery cell, often with 1C-rate, until the battery cell voltage reaches maximum voltage. Afterwards, in the CV phase, the voltage is kept in a steady stage, and the current starts decreasing until it reaches a threshold. Then, for the discharging process, the Eq. (7) is applied at specific time frame, to update the SoC value, since the initial SoC is 100%. The battery cell is fully discharged, 0% SoC, once it reaches the minimum voltage. The current maximum capacity $Q_{\text{cur\_max}}$ also should be updated to provide accurate measurements [38].

To extract the true maximum battery cell capacity and calculate its real SoH, a capacity characterization test should be performed, by fully discharging the battery cell, after a certain amount of full cycles. It must be mentioned, that the battery cell must be initially fully charged. The capacity tests provide the current maximum capacity of the cells, at specific environmental conditions. This process can provide the real value SoH, since the calculation of the capacity is direct, but the process is time consuming and once it is started should not be stopped. Furthermore, the environmental conditions should be steady, especially the temperature, since high temperatures lead to higher capacity values, and applied pressure at cell affects directly its capacity [39].

2.2. Model-based approach

Model-based approaches are currently used widely in many applications due to their proven reliability and satisfactory accuracy under specific environmental and operational conditions [40]. These methods typically rely on ECMs such as the Rint model, Thevenin model, or higher-order Thevenin models, chosen based on the desired balance between accuracy and computational cost [41]. ECM, depending on their complexity, can simulate several internal electrochemical behavior of battery cell through their parameters, such as charge transfer resistance, double-layer capacitance and diffusion processes, so their accurate estimation is necessary [42]. To achieve this, some characterization tests must be conducted, such as Hybrid Power Pulse Characterization (HPPC) [43] or Electrochemical Impedance Spectroscopy (EIS) test, and then fit the data into the ECM through an optimization algorithm, based on Particle Swarm Optimization (PSO) or Genetic Algorithm (GA), to extract parameters [44].

2.2.1. State of Charge estimation

In a model-based approach for SoC estimation, the initial SoC is determined either from the Open-Circuit Voltage (OCV)–SoC curve if the battery is at rest, or from the last known SoC value if the battery is already in operation. Once initialized, SoC is updated using Coulomb Counting (CC), by integrating the measured current over time Eq. (7). This updated SoC is then used to estimate the OCV through the OCV–SoC lookup table. With the estimated OCV and input current, the terminal voltage is estimated, using a battery cell ECM, by solving its governing differential equations. Finally, a Kalman filter or observer can be applied to compare the estimated terminal voltage with the measured one, to correct the internal states, reducing estimation error and compensating for sensor drift and model inaccuracies, thereby refining the SoC estimation [45,46].

SoC estimation generally relies on the OCV–SoC curve, which maps the relationship between SoC and the cell's voltage at rest. However, the OCV curve is itself affected by external factors like temperature and aging, making its accurate characterization both important and challenging [47]. Extracting a reliable OCV curve typically requires time-consuming characterization tests, such as low-current dis/charge tests (e.g., at C/20, where C is the C-rate of the battery), or HPPC/EIS tests, conducted at every 10% SoC increment. In those tests, voltage is measured after a resting period, to estimate the equilibrium voltage [48].

In general, model-based SoC estimation methods can offer accurate and robust performance, but they also come with several drawbacks:

- **OCV Curve Requirement**: Accurate SoC estimation depends heavily on the OCV–SoC curve, which requires time-consuming testing to obtain. Moreover, the OCV curve is sensitive to temperature, cell aging, and operating conditions.
- **Initial SoC Sensitivity**: A large error in the initial SoC estimate can lead to significant deviations in SoC prediction. However, this error can be mitigated using appropriate filters (e.g., Kalman) or observers.

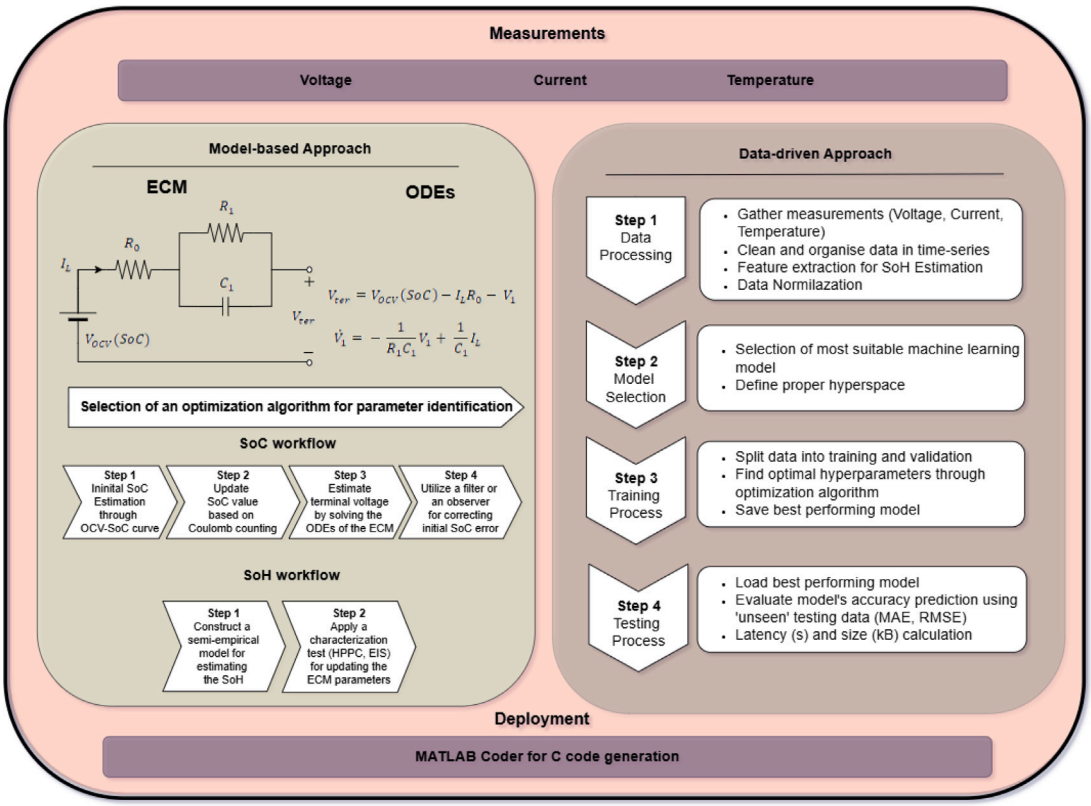


Fig. 1. Basic overall framework for SoC and SoH estimation of battery cell.

- **Accuracy vs. Complexity Trade-off**: More accurate ECMs, such as second-order Thevenin models, provide better voltage prediction but come with increased computational complexity, which may not be suitable for real-time embedded systems. Lastly, there should be an additional algorithm that will regularly update the model's parameters, considering different degradation stages and operation conditions.

2.2.2. State of Health estimation

The SoH estimation approach with model-based methods, differs significantly from the SoC estimation, since it is directly dependent on the values of ECM's parameters. By measuring and tracking the changes of the parameters, the degradation stage of the battery cell can be identify, as those parameters provide insights into internal aging mechanisms formed in the cell (e.g. growth of Solid Electrolyte Interphase (SEI) layer and lithium plating). By the formation of those battery's aging mechanisms, degradation modes are enabled, such as Loss of Lithium Inventory (LLI), Loss of Active Material (LAM), and Loss of Electrolyte (LE), which lead to cell's capacity and power fade [49].

After the accurate estimation of the ECM's parameters, by a testing process, such as HPPC or EIS, a semi-empirical model can be formed and be utilized for providing an estimation of the SoH value. According to the literature, there are plenty implementation of this process, and most of the researchers select to form polynomial and logarithmic equations, containing several parameters such as ECM's parameters, temperature, and SoC value, for accurate SoH information [50,51]. Lastly, there are some implementations of this algorithm that depending on the degradation stage of the battery cell, before and after 'knee-point', a different equation is utilized [52].

In conclusion, SoH estimation using model-based approaches can be an easy and straightforward method for gaining insights into battery degradation levels. However, several issues remain that prevent them from being ideal, such as:

- **Data Requirement**: Most semi-empirical models are based on the development of polynomial and logarithmic equations, which rely on data fitting to identify the model parameters. However, acquiring the necessary data is a time-consuming process.
- **Accuracy and Robustness**: The aforementioned described methodology is easy to develop and implement to a BMS but the accuracy and its provided robustness is not high, considering other techniques. This is because, the degradation features, that the ECM parameters describe, do not fully exploited by developing semi-empirical models.

Overall, model-based methods can provide a simple and easy-to-use solution for specific applications where high accuracy and robustness are not critical. Both SoC and SoH estimation rely on continuous updates of the ECM parameters to achieve satisfactory performance, which increases the overall complexity of the algorithm. To perform these updates, certain experiments, such as HPPC or EIS tests, must be conducted under specific environmental conditions. Lastly, it should be noted that the accuracy and robustness of model-based methods are generally inferior to those of data-driven approaches, which will be analyzed in the following subsection.

2.3. Data-driven approach

Nowadays, the industry is increasingly shifting towards data-driven solutions, driven by the significant growth in data availability, continuous improvements in hardware computational power, and the development of models that achieve high performance while requiring relatively low memory. In this section, authors present the most commonly used techniques and models for SoC and SoH estimation. As mentioned earlier, both states depend on several factors that directly influence the behavior of a battery cell. For this reason, features such as voltage, current, and temperature should be used, as they provide valuable insights into the environmental and operational conditions

under which the battery operates. Furthermore, deep learning models capable of simultaneously learning both local and global patterns are essential for achieving accurate SoC and SoH estimation [53]. In this section, there will not be separate subsections for SoC and SoH estimation, since the only difference is the input features that are utilized for their estimation. The overall, training and testing process is similar to both scenarios.

A general methodology for estimating the battery states involves three main stages: data processing, model training, and testing. Proper data processing begins with collecting relevant measurements from battery cells. The data must then be organized into an appropriate format, typically by splitting it into training and testing sets, often using 80% and 20% of the total data, respectively. Measurement errors and offsets should be corrected, and time series should be constructed using a sliding time window to capture meaningful behavioral patterns of the cell. Finally, the data should be normalized to ensure compatibility with deep learning model inputs [54]. SoH estimation algorithm, requires one more step in data processing which is the feature extraction, since the input features that the model requires, such as the curves of ICA, Differential Voltage Analysis (DVA) and Differential Temperature Analysis (DTA), must be formed from the current, voltage and temperature measurements by applying a smoothing algorithm [55].

During the training phase, models capable of capturing both local features, such as CNNs, and long-term dependencies, such as LSTM or GRU layer-based networks, should be selected. Once the model architecture is chosen, the corresponding hyperparameters, such as the number of units, number of layers, and activation functions, must be defined. This is typically achieved by utilizing hyperparameter optimization algorithms such as Grid Search (GS), Random Search (RS), or Bayesian Hyperparameter Optimization (BHO), to find the optimal set of hyperparameters within a defined hyperspace [56]. The trained model should then be evaluated based on its accuracy, inference latency, and memory consumption.

According to the literature, data-driven models can achieve very high accuracy while requiring relatively low memory and computational resources. Moreover, they can be easily re-trained using transfer learning techniques to estimate the battery cells states, of different battery types [57,58]. However, despite these advantages, the performance of data-driven models depends heavily on the quality of the data [59]. To obtain accurate and meaningful results, a well-designed experimental campaign is essential. Such campaign requires appropriate equipment, significant time investment, and expert knowledge in both hardware and software. Additionally, it should cover a wide range of operational and environmental conditions to ensure the model's robustness and generalizability.

2.4. Hybrid approach

Apart from model-based and data-driven approaches, hybrid models can also be developed. The hybrid method refers to the fusion of both approaches, or to the estimation of battery cell states by considering additional states [60].

For the fusion hybrid method, usually there is a development of an ECM, such as second order Thevenin, and once its parameters are estimated, by performing a characterization test like HPPC or EIS, they can be utilized as input features to a machine learning model for providing an estimation of battery state [61]. This process can achieve satisfactory results, and reduce significantly the overall complexity and computational requirements since there is no need for solving the systems differential equations nor is there a need to an identification of the OCV–SoC curve.

Considering the joint estimation of cell's states, the most common practice is to first estimate the cell's SoH and then utilizing this information. For example using the estimated value of SoH as an input feature to a deep learning model [62]. But also, there are some other cases such as using SoC information as a factor inside a semi-empirical model for SoH estimation [51].

Generally, hybrid models can reduce significantly the computational requirements of the overall algorithm, achieving better accuracy and increase model's robustness since the state estimation depends on both physics and data. A great disadvantage of the hybrid models is their complexity not only in the architecture design, but also in the implementation process, due to the co-existence of many methods-models, make them not preferable in real-time estimations.

In this paper, the authors chose to utilize deep learning models for separately estimating SoC and SoH. This approach provides an easy-to-use solution that can achieve satisfactory accuracy without the need to develop any ECM. Moreover, as it will be explained Section 6, these models can be easily scale up from cell-level to pack-level estimation.

3. Cell to pack state estimation

Accurate state estimation of battery modules is a challenging topic due to electrochemical inconsistencies among individual cells, even when cells are of the same battery type. These inconsistencies arise primarily from variations in manufacturing and welding processes and significantly impact the overall performance of battery modules. Cell inconsistencies can be observed as differences in internal resistance, initial SoC, capacity, and SoH. Such variations can lead to cell imbalance, uneven current distribution, and non-uniform temperature profiles, significantly accelerating battery module degradation and potentially causing safety hazards [63].

According to the literature, various strategies have been proposed to estimate battery module SoC and SoH [64]. These methods include lumped cell, reference cell, and mean and difference cell estimation approaches [65], which can be implemented using either model-based or data-driven techniques. The lumped cell estimation method does not consider cell-to-cell inconsistencies and utilizes overall module measurements, treating the module as a single equivalent cell. The reference cell method uses the most degraded cell as a representative for the entire module. Selecting an appropriate reference cell is challenging, as it requires detailed characterization of each cell or simulations of temperature distribution, given that higher temperatures accelerate degradation. Finally, the mean cell and difference SoC estimation method accounts for variations among cells by averaging state estimations. However, using average values can reduce accuracy, and computational costs are not significantly lowered because each cell's state must still be estimated but by using simpler models.

Overall, these methods primarily aim to reduce computational complexity, but their accuracy often suffers due to simplifying assumptions and cell selection challenges, without any significant improvement of the computational cost.

In this paper, the authors propose a straightforward strategy for battery module states estimation by using individual cell voltage, current and temperature measurements to estimate each cell's state. During discharging, the module's SoC is represented by the lowest SoC, while during charging the highest SoC is taken as the module SoC Eq. (1). This approach not only ensures accurate state estimation but also prevents cell overcharging and over-discharging, thereby reducing the risk of significant damage and thermal runaway. For SoH estimation, the most degraded cell, which has the lowest value, represents the module's overall SoH, Eq. (2). Conducting individual cell characterization tests ensures high accuracy and supports effective health management, providing crucial data for prognostics and diagnostics. Moreover, this strategy can extend the battery module's lifespan by allowing the replacement of cells that fall below approximately 70% SoH in electric vehicle applications. The primary drawback is the potentially higher computational cost; however, the authors address this by employing TinyML models, which eliminate the need for developing complex

ECM or solving differential equations, thus mitigating computational concerns.

$$SoC_{module} = \begin{cases} \min\limits_{i\in\{1,\dots,n\}} SoC_{cell_i}, & \text{during Discharging} \\ \max\limits_{i\in\{1,\dots,n\}} SoC_{cell_i}, & \text{during Charging} \end{cases} \quad (1)$$

$$SoH_{module} = \min_{i\in\{1,\dots,n\}} SoH_{cell_i} \quad (2)$$

The Eqs. (1) and (2) describe the strategy for the module states estimation, where n is the number of the overall cells connected in series.

4. Neural network compression

Deploying deep learning models on microcontrollers or other Internet of Things (IoT) devices requires aggressive model compression to reduce memory footprint, computational load, and energy consumption; factors that are critical for real-time inference in embedded systems. Standard compression techniques such as quantization [66], pruning [67], and weight clustering [68] have proven effective for convolutional and feed-forward architectures, where input shapes are typically static and well-defined [69]. However, applying these techniques to recurrent architectures, especially for LSTM and GRU layers, presents distinct challenges, particularly when the input sequence length varies across inferences. Current embedded inference frameworks such as TensorFlow Lite Micro and Apache TVM fundamentally require static input tensor shapes, leading to deployment pipelines that enforce fixed-length sequences via padding or truncation.

While these static-shape limitations constrain the applicability of conventional compression methods to variable-length recurrent models, an alternative approach has emerged that can reduce model complexity in such models. This method, known as network projection [32], replaces selected network layers with projected counterparts that operate in a significantly lower-dimensional subspace. This reduces the number of learnable parameters while maintaining satisfactory performance, and can be easily applied using MATLAB. Unlike traditional low-rank decomposition approaches used for neural network compression, such as low-rank matrix factorization and related tensor decomposition methods [70,71], the projection framework is data-driven, using in-distribution samples to characterize the activation patterns of intermediate layers and to construct projection operators that preserve the most informative subspaces. In Table 1 there is a summary of the most important deep learning compression techniques designed to reduce model size and computational complexity for deployment in TinyML and embedded environments.

To create the projection layers, first the matrix covariance of the layers' neurons is formed, and its eigenvalues and eigenvectors are computed. Then the neurons are projected to a subspace ($\mathbb{R}^{b_i} \subset \mathbb{R}^{a_i}$, with $b_i \leq a_i$ where a_i is the number of neurons in ith layer), by applying the projection operator $\mathbb{P}$, Eq. (3), which eliminates the eigenneurons (neurons defined by eigenvectors) with small eigenvalues.

$$\mathbb{P} = \sum_{p=1}^{b_i} v_p v_p^T \quad (3)$$

where v_i is the corresponding eigenvector of the neuron's covariance matrix. The b_i is selected depending on the preferred retained variance. If $b_i \to a_i$, the projection operator tends to the identity matrix.

The projection operator, can be applied to each neuron of a layer, for defining a new projected layer, $f_i^{\mathbb{P}}$, which basically maps a layer to its projected equivalent. The projected layer is described by Eqs. (4) and (5), assuming there are no bias terms for simplicity purposes.

$$f_i^{\mathbb{P}} : \mathbb{R}^{a_{i-1}} \to \mathbb{R}^{a_i} \quad (4)$$

$$\begin{aligned} f_i^{\mathbb{P}}(x) &= \mathbb{P}_i f_i(\mathbb{P}_{i-1}x) = \left(v_i v_i^T\right) W \left(v_{i-1} v_{i-1}^T\right) x \\ &= \widetilde{W} \left(v_{i-1}^T x\right) \end{aligned} \quad (5)$$

Table 1
Summary of deep learning compression methods for embedded and TinyML deployment.

Method	Description
Quantization [66,72]	Reduces precision of weights and activations (e.g., FP32 to INT8 or smaller), lowering memory and compute requirements with minimal accuracy loss.
Pruning [67,73,74]	Removes low-importance weights or neurons, reducing parameter count and enabling sparse computations. Includes unstructured pruning (individual weights) and structured pruning (entire neurons or filters), often followed by fine-tuning.
Weight clustering [68,75]	Groups similar weights and shares representative values, decreasing storage while preserving model behavior.
Low-Rank factorization [71,76]	Decomposing large weight matrices/tensors into smaller low-rank matrices, capturing essential information with fewer parameters.
Knowledge distillation [77–79]	Trains a compact student to mimic a larger teacher model, preserving task accuracy while significantly reducing size.
Network projection [32]	Projects layer activations into data-driven subspaces, reducing parameters while maintaining functionality.

The projection operator $\mathbb{P}_{i-1} = \left(v_{i-1} v_{i-1}^T\right)$ projects the input into a lower-dimensional subspace (keeping the "important" eigenneurons), f_i apply the normal layer mapping (without the bias term for simplicity) and $\mathbb{P}_i$ applies projection to output neurons to keep the output restricted to the correct subspace. Lastly, the term $\widetilde{W} = \left(v_i v_i^T\right) W(v_{i-1})$ is the projected weight matrix that is stored, instead of W, which contains a reduced number of parameters.

In Fig. 2, there is a depiction of the original fully connected layer, the projected layer with the input and output projection operations, and lastly the final compressed form of the projected layer.

Fig. 2(a), presents a simple fully connected layer that takes 3 inputs and produces 2 outputs ($n_i^{(1)}(x)$, $n_i^{(2)}(x)$). In Fig. 2(b), there is the projected fully connected layer, with the projection operations applied to the input and output. The original fully connected layer can be seen with the black neurons, while the single neurons with blue and orange color comprising the eigenneurons, that were formed by applying the projection operation to the input and output neurons. The weights on the connections between layer input eigenneurons and layer output eigenneurons correspond to the projected weight values in the subspace defined by the eigenvectors of the neuron covariance matrices.

In Fig. 2(c), there is the depiction of the final compressed form of the projected layer, where W and $\widetilde{W}$ represent the weight matrices of the fully connected layer and the projected one, containing 6 and 5 learnable parameters, respectively. Thus, the projected weight matrix $\widetilde{W}$ is stored, instead of W, thereby reducing memory requirements.

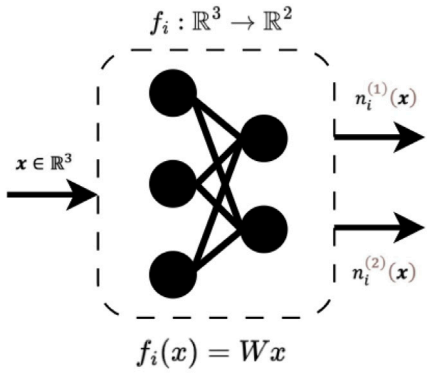
Finally, a projected network is constructed by linking multiple projected layers, as shown in Eq. (6).

$$F(x) = f_N^{\mathbb{P}}(f_{N-1}^{\mathbb{P}}(\cdots f_1^{\mathbb{P}}(x)\cdots)) \quad (6)$$

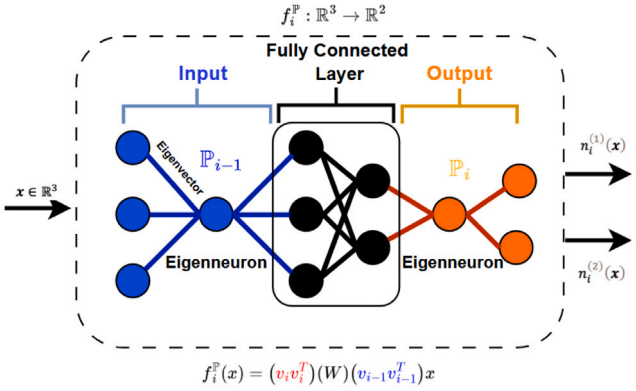
In this process, each layer maps its input onto eigenneurons, lowers the rank by discarding eigenneurons with low variance, and then converts the result back into the original basis before passing it to the following layer. It should be noted that the final projected models should be retrained for a few epochs, since performance may initially drop, due to loss of some information. More technical information about the neural network projection can be found in [32].

5. Dataset

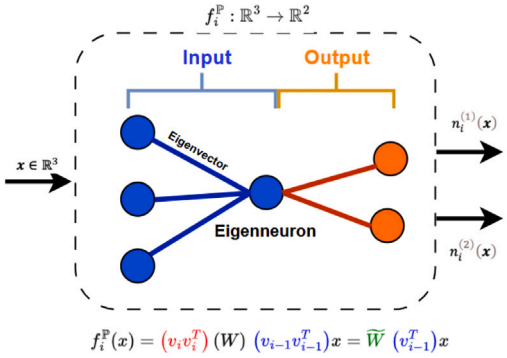
For this study, an online battery dataset made available by Polimi [22] was utilized. The dataset includes voltage, current, and temperature measurements for six LG18650HE4 battery cells (data from cell



(a) An example of a simple fully connected layer.



(b) Projected layer with input and output projection operations.



(c) Final compressed form of the projected layer.

Fig. 2. In Fig. 2(a) there is simple fully connected layer, while in Fig. 2(b) there is a depiction of the projected layer with the input and output projection operations. Fig. 2(c) composes the final structure of the projected layer with lower number of learnable parameters. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4 are not available), each with a nominal capacity of 2.5 Ah. These measurements were recorded at various ambient temperatures ranging from −20°C to 35°C. Following the temperature variation experiments, a degradation campaign was conducted at 35°C to monitor the aging behavior and degradation process of each battery cell. During the discharging phase, several real-world driving profiles were used, including UDDS, US06, LA92, as well as six randomly combined drive cycles labeled Mixed 1 through Mixed 7. For the SoC estimation, the models were trained on these dynamic driving profiles, which contain both discharging and charging (through regenerative braking) data. Data

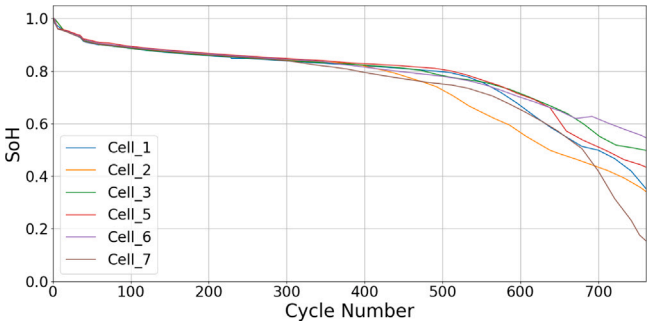


Fig. 3. Battery cells SoH true values, for different degradation phases.

from cells 1–6 were utilized for training the models (80% for pure training and 20% for validation), while the developed models' accuracy was evaluated on the "unseen" dataset of cell 7 (testing data). The selection of the training–testing cells, came from the selection of the worst performing models for both SoC and SoH estimation, on a cross-validation experiment. It should be noted that the LA92 and Mixed 5–7 cycles were used exclusively for testing, and not for training, to further enhance the models' robustness. Charging was performed using the Constant Current Constant Voltage (CCCV) protocol recommended in the manufacturer's datasheet to ensure that the cell is fully charged.

From the recorded data, additional features were derived, including SoC, SoH, and the ICA curve, all of which are essential for training and testing machine learning models. For the SoC calculation, the Coulomb Counting (CC) method, as described in Eq. (7), was utilized. It was assumed that 100% SoC and 0% SoC are reached when the battery cell voltage reaches its maximum and minimum values, respectively. A definition of the SoC can be seen in equation Eq. (8), where $Q_{\text{cur\_max}}(cycle)$ is the maximum capacity of the battery cell at specific cycle (degradation stage), while $Q_{(t)}$ is the cell's available capacity at specific time.

$$SoC = SoC(t_0) - \frac{1}{Q_{\text{cur\_max}}(cycle)} \int_{t_0}^{t_0+\tau} I_{\text{bat}}\, dt \cdot 100\% \tag{7}$$

$$SoC = \frac{Q_{(t)}}{Q_{\text{cur\_max}}(cycle)} \cdot 100\% \tag{8}$$

For SoH estimation, according to the literature there are two main definitions [80], as it was mentioned above, one based on capacity degradation, Eq. (9) and the other based on resistance increase, Eq. (10). In general, to determine the current maximum capacity $Q_{\text{cur\_max}(cycle)}$ or the current internal resistance of the cells $R_{\text{cur}}(cycle)$, specific characterization tests, such as full charge–discharge cycles for capacity measurement or HPPC tests for resistance measurement, must be conducted every few cycles (e.g. 20–40 cycles in this dataset) in order to update the SoH value. These procedures are typically performed under controlled temperature and current conditions to ensure consistency and repeatability, as both capacity and resistance are sensitive to operating conditions. In Fig. 3, there is a depiction of the SoH true values of the battery cells, for different degradation phases. Data from cell 4 were excluded due to channel errors. For data extraction in this work, authors utilized Eq. (9), where $Q_{\text{ini\_max}}$ is the initial maximum capacity of the battery cell, instead of Eq. (10). In Eq. (10), R_{EOL} and R_{BOL} represent the End of Life (EOL) and Beginning of Life (BOL) resistances, respectively.

$$SoH_Q = \frac{Q_{\text{cur\_max}}(cycle)}{Q_{\text{ini\_max}}} \cdot 100\% \tag{9}$$

$$SoH_R = \frac{R_{\text{EOL}} - R_{\text{cur}}(cycle)}{R_{\text{EOL}} - R_{\text{BOL}}} \cdot 100\% \tag{10}$$

By generating the ICA curve [dQ/dV] and the DVA curve [dV/dQ] [81], valuable insights about internal electrochemical reactions, degradation mechanisms, capacity and power fade of battery cells can be

Table 2
Performance of SoC models.

Metrics	CNN_GRU	Pr_CNN_GRU	CNN_LSTM	Pr_CNN_LSTM	GRU	Pr_GRU	LSTM	Pr_LSTM
MAE	0.0100	0.0129	**0.0093**	0.0124	0.0099	0.0119	0.0129	0.0114
RMSE	0.0276	0.0296	**0.0269**	0.0291	0.0274	0.0289	0.0283	0.0286
Size (kB)	155.5	5.3	390	8.8	200.5	7.5	**68.8**	4.0
Latency (ms)	**0.265**	0.194	0.386	0.242	0.476	0.411	0.417	0.391
Size Red. (%)		96.59		97.74		96.26		94.19
Latency Red. (%)		26.79		37.31		13.66		6.24

Table 3
BHO searching space for both state estimation models.

Parameter	Model's Hyperspace	
	SoC	SoH
BHO max tuner	40	
Epochs	1000	
Early stopping	50	
Input shape	(None,60,3)	(None,None,1-2)
Batch size	256	1
Kernel size	(2,5)	
Number of units orfilters for all layers	range(4, 32, 4)	range(16, 128, 16)
Learning rate	(0.001, 0.0001, 0.00001)	
Additional LSTM or GRU layers	(0,1)	
Additional dense layers	(0,1)	
Layers' activation function	(leaky_relu, gelu, swish, selu, linear)	

obtained. These insights are derived by tracking the shifts in the peaks and valleys of the curves, which are typically associated with degradation phenomena such as LLI, LAM, and LE [82].

To extract meaningful features from these curves, it is essential to apply a noise-reduction filter [83], such as a Gaussian filter, Savitzky–Golay filter, or rolling window (Moving Average) filter, since differentiation tends to amplify measurement noise. In this study, only voltage and ICA curve data were utilized as input features for the deep learning models used in SoH estimation. A moving average filter was applied to smooth the data and enhance the visibility of peaks and valleys, due to its simplicity and ease of implementation. More information about the dataset can also be found in [62]. Similarly to the SoC case, the SoH estimation models were trained on the datasets of cells 1–6 and tested on the unseen data from cell 7 to further improve model robustness and generalization capability. Since the charging time varies significantly, not only among the same cell, due to degradation, but also across different cells, authors selected layers capable of handling variable-length input data without the usage of the padding or truncation technique.

6. Methods and results

In this paper, authors conducted an analysis of several deep learning models, such as GRU, LSTM, CNN_GRU and CNN_LSTM, for both SoC and SoH estimation. Afterwards, by utilizing MATLAB software, they compressed the models by using neural network projection, and generate the proper C code to be ready for deployment into a microcontroller of a battery management system.

For both cases the BHO algorithm was utilized for finding the optimal hyperparameters of the models during the training procedure. The BHO algorithm [84], is an iterative technique designed to efficiently identify the global optimum of complex functions. It uses a probabilistic surrogate model, typically a Gaussian process, to approximate the target function and quantify uncertainty. This model guides the optimization by balancing two key strategies: exploration, which investigates less-tested regions of the hyperparameter space, and exploitation, which focuses on areas expected to yield high performance. At each step, BHO selects the next set of parameters to test based on an acquisition function that captures this trade-off. After evaluating these parameters, the surrogate model is updated, and the cycle continues until convergence or a specified stopping condition is reached. In Table 3, there is a depiction of the hyperspace in which the BHO will search in order to find the optimal architecture of the models. The total number of optimization trials is determined by the max tuner value. Additional training related parameters, such as the number of epochs, batch size, and learning rate, are also included in the table.

To evaluate the performance of the models, considering the accuracy, the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are utilized, calculated as in Eqs. (11) and (12), respectively:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_j - \hat{y}_j)^2} \tag{11}$$

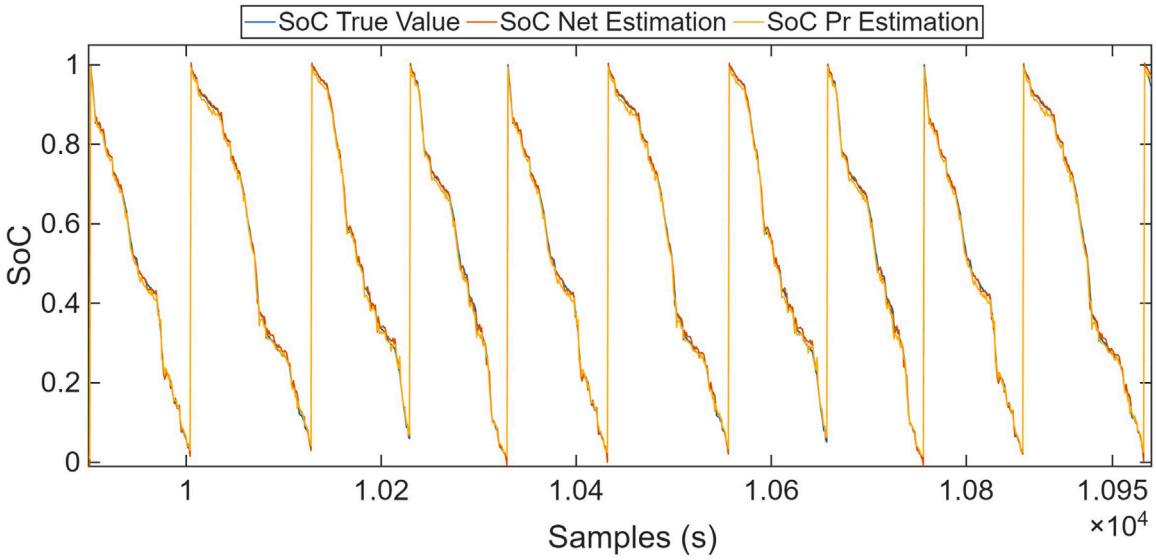
$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}\left|y_j - \hat{y}_j\right| \tag{12}$$

where y_j is the real value of the SoH and $\hat{y}_j$ is the predicted value generated by the model.

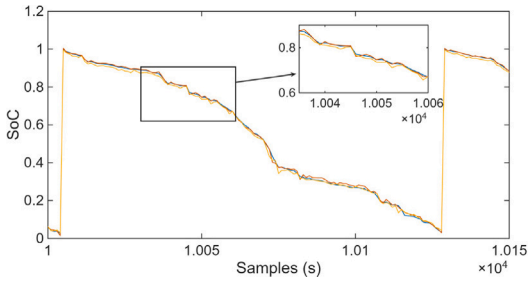
After training and evaluating the "simple" models, using TensorFlow environment, they were imported into MATLAB, where their layers were compressed using neural network projection technique to form the projected models. However, this projection led to a significant drop in performance. To address this, the models were fine-tuned for few epochs, using a small batch of the original training dataset. Once the performance was restored, dedicated functions for SoC and SoH estimation were developed. Finally, using MATLAB's C code generation capabilities, the corresponding C source files were generated, preparing the models for deployment on a microcontroller.

The latency of the models were assessed on a GPU (NVIDIA GeForce RTX 3060) using dlCodegenBench, an add-on created by developers at MathWorks for benchmarking the runtime performance of code generated from deep learning models in MATLAB. For each function defined for SoC and SoH estimation, MATLAB Executable (MEX) functions were first generated and then benchmarked using dlCodegenBench, which executed two warm-up calls followed by 100 inference calls of the generated code, reporting latency as the 90th percentile of the measured times [85].

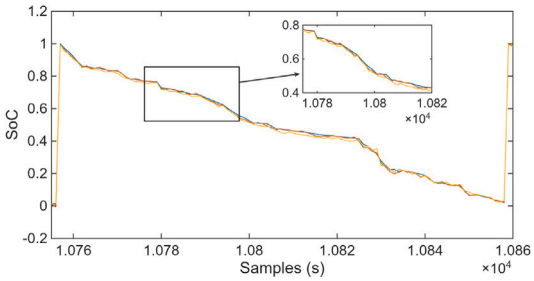
In the upcoming subsections, there will be an analysis of the SoC and SoH algorithms, that were developed and lastly, there will be a presentation of the method that can be utilized for a real BMS.



(a) Sample of estimation results of the deep learning models for 10 cycles.



(b) Enhanced results of the LA92 driving profile.



(c) Enhanced results of the Mixed-6 driving profile.

Fig. 4. SoC estimations results from CNN_LSTM model before compression (Net) and fine-tuned projected model Pr_LSTM for mid-degradation stage of the battery cell.

6.1. State of Charge

To achieve accurate SoC estimation, the models utilized a time window of 60 samples, with each sample representing one second and containing three input features: current, voltage, and temperature measurements. As previously mentioned, all input data were normalized before being fed into the models. Importantly, the dataset includes SoC information recorded under varying temperatures, C-rates, and degradation stages, which enhances the robustness of the models across diverse operational and environmental conditions.

In Table 2, the results of the SoC models are presented, highlighting the "simple" models that achieved the best performance and underlining the projected models with the lowest error. The CNN_LSTM model achieved the highest accuracy, with an MAE of 0.0093 and an RMSE of 0.0269. Among the projected models, Pr_LSTM demonstrated the best accuracy, with an MAE of 0.0114 and an RMSE of 0.0286.

Considering both memory and computational requirements, the size and latency of the models were also evaluated. As observed, the "simple" models require more memory, which in some cases can cause significant challenges when deployed on a microcontroller. On the other hand, their latency remains, in both cases, within a satisfactory range. Since the latency of both "simple" and projected models is similar, the focus is placed on model size, as it directly affects energy consumption. Fewer parameters lead to fewer memory accesses and multiplications, which significantly reduces energy consumption.

The LSTM model achieved the smallest size among the "simple" models, requiring just 68.8 kB, while still delivering acceptable accuracy. After applying compression and fine-tuning, the Pr_LSTM model not only achieved the highest accuracy but also reached the lowest size of 4.0 kB, with a 94.19% reduction. Additionally, its latency improved by 6.26%, resulting in an estimation time of 0.391 ms per batch. This combination of competitive accuracy and minimal memory usage makes the Pr_LSTM model particularly well-suited for embedded TinyML applications. Lastly, as it can be deduced from the results, almost all models achieved satisfactory results throughout different temperature, C-rates and degradation stages which are very important for real BMS.

In Fig. 5, a boxplot is presented to compare the per-sample MAE of different SoC estimation models. The plot illustrates the distribution of errors for each model: the light-blue boxes represent the interquartile range (25th to 75th percentile), the black whiskers extend to the minimum and maximum non-outlier values, and the black dots indicate the mean MAE for each model, as detailed in Table 2. Additionally, the blue median line represents the typical error for a single sample without the consideration of extreme offset values. Notably, the median often differs from the mean MAE due to a small number of high-error instances (offsets), which may result from faulty measurements, extreme environmental conditions, or the battery operating near its degradation knee-point.

Fig. 4 shows the SoC estimation results for both the 'simple' (Net) CNN_LSTM and the fine-tuned projected Pr_LSTM (TinyML) model, compared with the actual SoC values. Fig. 4(a), presents 10 cycles of a mid-degradation stage of the battery cell, approximately 90% SoH, within the full degradation range analyzed (100% to 70% SoH). As it was mentioned above, those 10 cycles are not the same since for the testing process only the 4 driving profiles were utilized (LA92 and

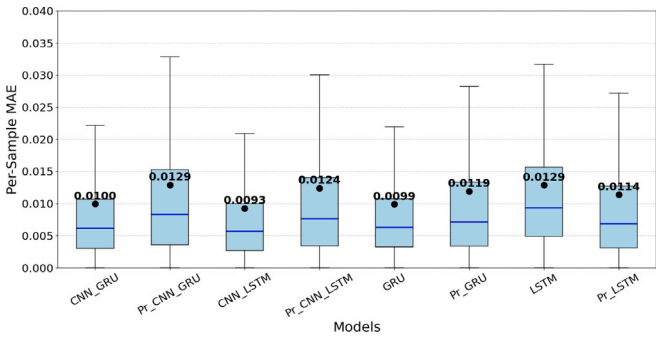


Fig. 5. Per-sample MAE results for battery cell SoC estimation considering both projected and non-projected models. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

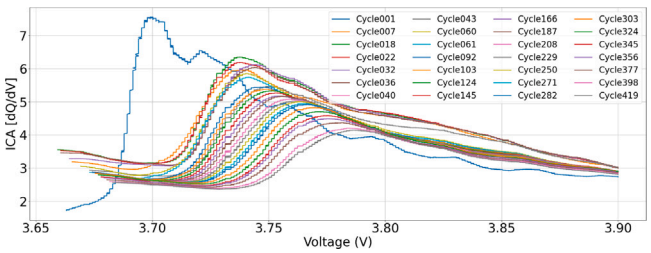


Fig. 6. Incremental capacity analysis curve, depending on the voltage level, for different degradation phases.

Mixed 5–7). The utilization of unseen driving profiles in the testing procedure strengthens the generalization and robustness of the models. Finally, Figs. 4(b) and 4(c) present a closer look at the CNN_LSTM and Pr_LSTM model results for the LA92 and Mixed-6 cycles, respectively. Notably, these models not only accurately predict the battery's SoC during its first life, but also maintain satisfactory estimation performance for steep profiles and initial SoC estimation, after charging scenarios, in which model-based methods fail to achieve good accuracy.

6.2. State of Health

Similarly, for SoH estimation, the same architecture deep learning models were evaluated, followed by a neural network projection step to identify the most suitable model for deployment on a microcontroller. Two scenarios were considered: one using only ICA data as input feature, and the other combining ICA and voltage data. In both cases, the models were designed to handle variable-length input sequences, enabling SoH estimation from partial battery charging within the voltage range of 3.6 V to 3.9 V, thus avoiding the need for full charge–discharge cycles. This approach significantly reduces processing time and enhances applicability in real-world conditions, since in practice users do not always fully charge or discharge their batteries.

Fig. 6 displays the ICA curves of battery cell number 7 across various degradation stages, up to cycle 419, for clarity. To generate these curves, capacity and voltage differences (ΔQ and ΔV) were extracted at one-minute intervals (every 60 samples) to compute the ICA profiles. Although this data pre-processing helped reduce noise, some spikes remained, so a rolling average with a 300-sample window was applied to further smooth the data. Notably, the ICA curve from cycle 1 is significantly different from those of later cycles.

The aforementioned phenomenon can be explained by the different internal electrochemical behavior of the cell during its early cycles, such as the SEI layer formation, which starts to form when charging the cells for the first time and differs from that of aged cells. Additionally, until cycle 60, the cell experienced extreme temperature variations, ranging from −20°C to 35°C. After that, it was cycled at a constant temperature of 35°C to promote gradual degradation until reaching the knee-point. These ICA signals, which vary in length, were used as the primary input for SoH estimation.

Tables 4 and 5 present the performance of the deep learning models under both input scenarios. The "simple" models without compression which achieved the best accuracy are shown in bold text, while the top-performing projected (TinyML) models are marked with underscores. In both input scenarios, the "simple" GRU model delivered the best performance, achieving an MAE of 0.0031 and RMSE of 0.0057 using only ICA data. Among the TinyML models, the projected GRU model (Pr_GRU) yielded the lowest error, 0.0058 MAE and 0.0087 RMSE, when using both ICA and voltage data.

It must be noted that, in principle, adding more input features can provide richer information that may improve model performance. To address this uncertainty, the authors conducted this feature sensitivity analysis, which showed no consistent or significant improvement when voltage was added alongside ICA data. In several cases, performance even decreased (GRU), although accuracy remained within an acceptable range. This is likely because the ICA curve already captures the key degradation patterns derived from the voltage–capacity relationship, so including voltage adds little to no new information. Consequently, the accuracy differences between the ICA and ICA_V models are small and not systematically in favor of either configuration.

Similarly, the model size and latency were calculated to assess memory usage and computational requirements. On average, the size reduction across both input cases was 93.79%, while latency was reduced by 17.58%. For embedded deployment, the TinyML model selected by the authors is the Pr_CNN_GRU, using just the ICA data as input feature, which achieves competitive accuracy with an MAE of 0.0060 and RMSE of 0.0130, while also offering the smallest size and lowest latency, just 3.4kB and 1.607 ms, respectively.

In embedded applications, model size is critical, as many microcontrollers have limited memory capacity. Additionally, larger models tend to consume more energy. Although model size can influence latency, it is important to note that latency also heavily depends on the hardware on which the model is deployed, which in this case is a GPU whose characteristics are described above.

Fig. 7 illustrates the SoH predictions of both the "simple" CNN_GRU and the fine-tuned projected Pr_CNN_GRU model, compared to the actual SoH values. While both models show reduced accuracy in the early cycles, where extreme conditions were also applied, and again near cycle 400, where degradation accelerates, their overall predictions are highly satisfactory throughout all the first life cycle of battery, which is until 70% of SoH.

To further analyze models' performance, Fig. 8 plots the MAE at each cycle. As expected, estimation errors are highest during early cycles (1–61), likely due to the extreme environmental conditions the batteries experienced, as discussed previously. These included temperatures ranging from −20°C, to 35°C, which can trigger lithium plating and other processes that can degrade significantly the battery cell. Error increases again as the battery nears 80% SoH, marking the beginning of the "knee-point", where degradation becomes more rapid and battery cell becomes more unpredictable. This instability is often connected to manufacturing variability in the construction phase, such as differences in internal resistance and capacitance, even among identical cells. As degradation progresses, these differences become more noticeable, which is why cells falling below 80%–70% SoH are generally declared as unsuitable for high-stress applications such as electric vehicles. It should be noted that, since the LSTM based models did not achieve satisfactory results, they were not included in Fig. 8, to improve clarity and allow the errors of the remaining models to be more easily compared.

In conclusion, the developed SoH estimation method is more complex compared to SoC estimation, primarily due to the additional step

Table 4
Comparison of models with ICA input features.

Metrics	CNN_GRU	Pr_CNN_GRU	CNN_LSTM	Pr_CNN_LSTM	GRU	Pr_GRU	LSTM	Pr_LSTM
MAE	0.0060	0.0060	0.0095	0.0079	**0.0031**	0.0135	0.0227	0.0558
RMSE	0.0089	0.0130	0.0114	0.0148	**0.0057**	0.0156	0.0265	0.0698
Size (kB)	**23.1**	3.4	64.1	4.8	99.8	6.6	444.1	12.2
Latency (ms)	**1.614**	1.607	2.042	2.040	15.113	13.477	29.567	19.192
Size Red. (%)		85.28		92.51		93.39		97.25
Latency Red. (%)		0.43		0.10		10.83		35.09

Table 5
Comparison of models with ICA_V input features.

Metrics	CNN_GRU	Pr_CNN_GRU	CNN_LSTM	Pr_CNN_LSTM	GRU	Pr_GRU	LSTM	Pr_LSTM
MAE	0.0064	0.0094	0.0071	0.0060	**0.0055**	0.0058	0.0116	0.0549
RMSE	0.0079	0.0112	0.0086	0.0093	**0.0071**	0.0087	0.0134	0.0687
Size (kB)	819.1	17.5	178.8	7.9	380.5	12.8	**59.0**	4.8
Latency (ms)	5.31	2.67	**2.86**	2.66	23.37	15.86	17.37	16.44
Size Red. (%)		97.86		95.58		96.64		91.86
Latency Red. (%)		49.72		6.99		32.14		5.35

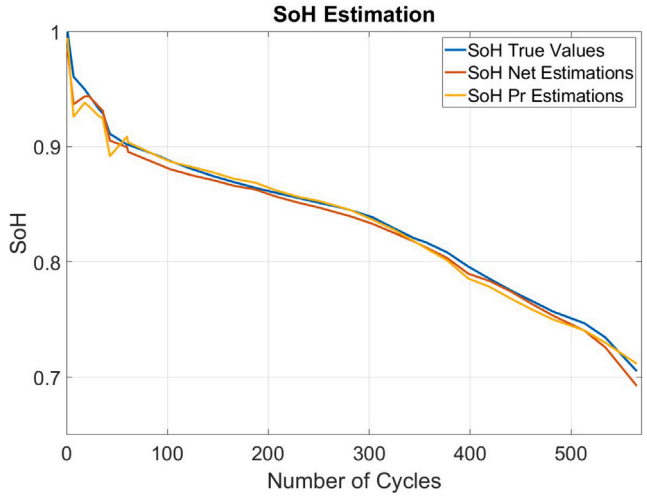


Fig. 7. SoH estimation results for CNN_GRU (Net) and Pr_CNN_GRU (Pr) models using just ICA data.

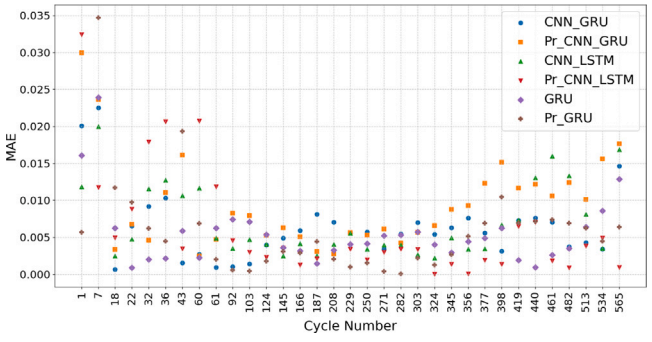


Fig. 8. Comparison of SoH estimation models considering MAE per cycle (excluding LSTM based models).

of constructing the ICA data using a rolling window. Once the user stops the charging process, a few current and voltage measurements are collected within the voltage range of 3.6 V to 3.9 V. These measurements are then used to generate the ICA curve, as previously described. The resulting data is normalized and subsequently fed into the deep learning model. As demonstrated, this approach achieves high accuracy in SoH estimation across most stages of battery cell degradation.

6.3. Deployment into Infineon battery module

The target application of the developed models is to estimate the SoC and SoH of a real battery system. To achieve this, the models must first be compressed to reduce memory and computational requirements without compromising accuracy. After compression, models should be converted into a suitable format (C code) to enable deployment on microcontrollers. Infineon Technologies has recently developed a BMS for a robotic application, Infineon Mobile Robot (IMR), which consists of several subsystems designed to ensure optimal performance and safety.

The IMR is classified as an autonomous mobile robot due to its integration of intelligent motor control, a BMS, and a robust power distribution system, all of which interact with precise Infineon's built-in sensors. It operates independently without the need for external guidance, as it can generate maps from its environment, accurately identify its own location, and perform obstacle avoidance. The system architecture of the IMR reveals that it is powered by motors on each of its four wheels and includes two hot-swappable BMS units along with a pair of power distributors. These distributors use an interleaved buck converter design to reduce current ripple within the system. For its autonomous navigation capabilities, it utilizes a hybrid Time-of-Flight camera.

The IMR's BMS, serves as a fully integrated, independent battery-pack solution consists of both control and power components, as illustrated in Fig. 9. At the core of the power board is the highly dependable Infineon's BMS Integrated Circuit (IC) TLE9012DQU, which supports balancing of up to 12 Li-Ion cells connected in series. This IC also enables precise voltage monitoring for each individual cell. It includes five temperature sensing channels, which contribute to accurate assessments of the battery's SoC and SoH.

Furthermore, the system incorporates diagnostic functionality for detecting open-wire and open-load conditions, as well as identifying overcurrent and undercurrent issues during the balancing process. To ensure stable communication with the main controller, a differential UART interface is employed. Moreover, IMR's BMS contains several other devices such as the TLI4971 25 A, current sensor for the accurate measurement of the overall module current, as well as a 2ED4820-EM MOSFET Gate driver that provides protection against overvoltage, undervoltage, and overcurrent conditions.

The current BMS includes algorithms for SoC and SoH estimation, relying on a coulomb-counting approach for SoC and a full discharge/charge cycle for SoH assessment. Additionally, the current state estimation strategy is based on aggregated system-level measurements, overlooking potential variations between individual battery cells, as discussed earlier. In this context, the authors propose an algorithm

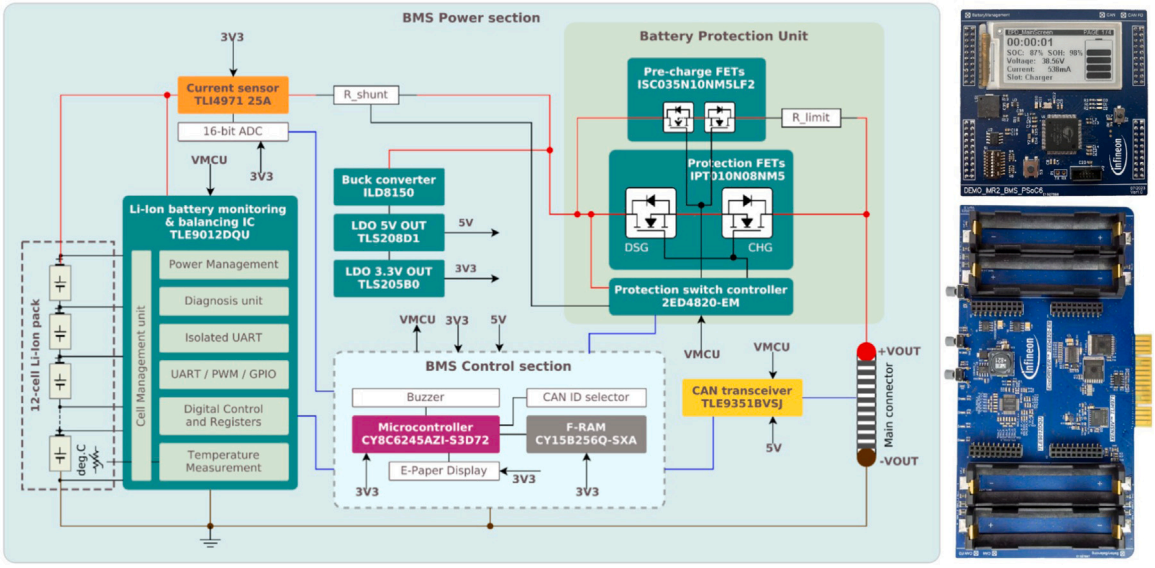


Fig. 9. IMR's BMS architecture and components for optimal control and power management.

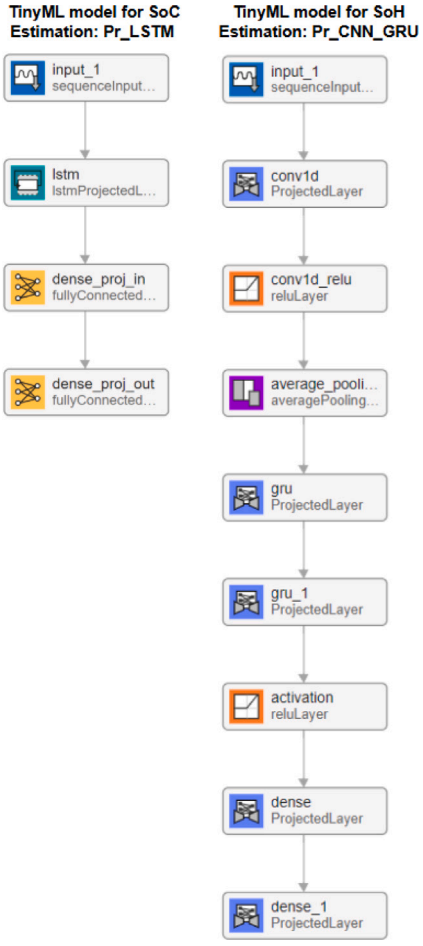


Fig. 10. Architecture of the fine-tuned TinyML models for SoC and SoH estimation.

suitable for integration into the existing BMS, capable of delivering accurate SoC and SoH estimations with minimal memory footprint, thus meeting the constraints of embedded deployment.

Table 6
Hyperparameter results of the BHO search of the top performing models for SoC (LSTM) and SoH (CNN_GRU).

SoC (LSTM)	SoH (CNN_GRU)
Input: (None, 60, 3)	Input:(None, None, 1)
LSTM_1_Units: 64	Kernels: 3
Dense_1_Units: 1	Filters: 32
Activation: relu	GRU_1_units: 16
–	GRU_2_units: 16
–	Dense_1_units: 96
–	Dense_2_units: 1
–	Activation: relu
Parameters: 17,473	Parameters: 5889
Pr_Parameters: 7653	Pr_Parameters: 1160

For this task, the authors suggest using the Pr_LSTM model for SoC estimation, as it achieved the lowest memory requirements, of 4.0kB, with satisfactory accuracy of 0.0114 and 0.0286 MAE and RMSE respectively and a satisfactory latency of 0.391 ms. Considering the SoH estimation, the Pr_CNN_GRU model, by utilizing just ICA data, achieved simultaneously, very low errors, with 0.0060 and 0.0130 MAE and RMSE respectively, and size of just 3.4 kB with 1.607 ms latency. The architecture of the projected models that authors recommend for implementation are presented in Fig. 10, where hyperparameters of those models, such as activation function, number of layers and units, were selected by the BHO algorithm and are presented explicitly in Table 6. The model hyperparameters, such as the number of hidden units and the filter size, remain the same in both the 'simple' and the projected models. The only aspect that changes during projection is the structure of the layers, where the convolutional, LSTM/GRU, and dense layers are replaced by their projected counterparts. Further details on the formulation of these projected layers can be found in [32].

It must be also noted that the aforementioned models were trained and tested for estimating the states of a single battery cell, not the entire battery module. Therefore, an algorithm should be developed to scale up the models' cell-level estimations to the module level. The proposed algorithm is described in Section 3 and illustrated in Fig. 11, where, depending on the state to be estimated, different types of measurements must be extracted and converted into a specific form.

These measurements are obtained from Infineon's IC TLE9012DQU, which provides voltage readings from each cell and includes five temperature channels for measuring ambient temperature within the device. Additionally, the current of the entire battery module is measured

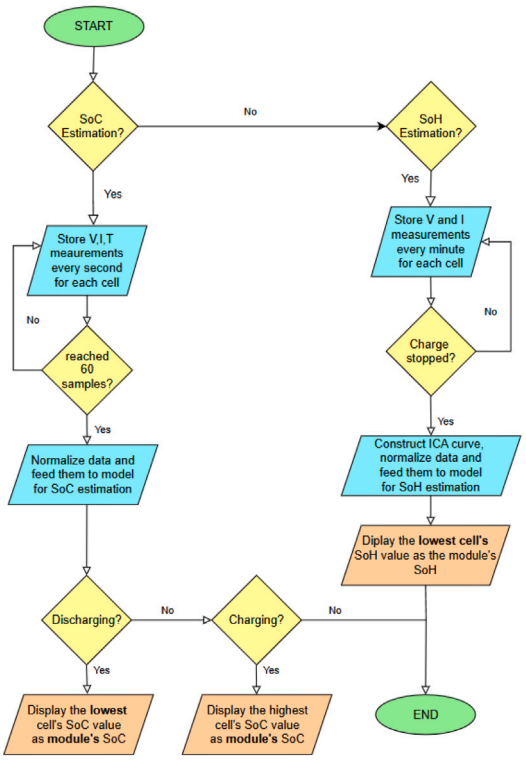


Fig. 11. Suggested diagram for displaying battery module's SoC and SoH estimation.

using the high-precision, coreless current sensor TLI4971 25 A, as it was mentioned above.

Subsequently, based on the BMS mode (charging or discharging), the SoC value of the 'worst-performing' cell is selected to represent the state of the battery module. This approach is adopted due to the battery module's architecture, 12 battery cells connected in series, where over-charging or over-discharging even a single cell, can compromise the safety and functionality of the entire module, especially in lithium-ion batteries.

For the SoH value, the most aged cell is selected due to its high unpredictability, which can lead to localized temperature increases caused by its elevated internal resistance, further amplifying cell imbalances and potentially resulting in reduced performance or even thermal runaway. It should also be noted that the aged cells are typically the 'bad' ones, limiting the overall energy and power capabilities of the module. This SoH algorithm enables the identification of such cells, and their replacement can significantly enhance the lifespan of the battery module, thereby contributing to more accurate and effective health prognostics and diagnostics.

The overall objective of this study is to demonstrate a novel method for compressing and optimizing deep learning models using an innovative neural network projection technique developed in MATLAB, and to convert them into state-of-the-art TinyML models that can be efficiently deployed on microcontrollers designed for time-series processing. These models are capable of capturing both short- and long-term patterns, handling variable-length inputs for the SoH estimation task, and enabling accurate battery state estimation while significantly reducing memory usage, computational load, and energy consumption. The resulting TinyML models can be scaled from individual cell-level estimations to module-level estimations for Infineon's BMS, which manages 12 battery cells connected in series. To support this, the authors propose a strategy that is straightforward to implement while ensuring safety, control, and effective battery health monitoring (both prognostics and diagnostics). In this study, however, no module-level validation experiments for SoC or SoH were performed, as the chosen approach focuses on the performance of single-cell state estimations, which were conducted offline for the ready-to-deploy TinyML models analyzed in Section 6. Module-level verification will be examined in future research, integrating it with the overall BMS control strategy, including cell balancing and temperature management.

7. Conclusion

In conclusion, this study developed four state-of-the-art TinyML models capable of processing variable-length inputs without padding for SoH estimation and fix length input (60 samples) for SoC estimation. The models, based on CNN, LSTM, and GRU architectures, were comprehensively evaluated using multiple performance metrics, including MAE, RMSE, model size (kB), and latency (ms). For utilizing those models into a microcontroller, to perform real-time estimations, the computational and memory requirements of the models, must be minimized. For this task, authors utilized an innovative compression technique based on the projection of neural networks. This projection technique compresses model size up to 97.86% and in some cases reduced latency up to 49.72%, while their accuracy remained almost the same after fine-tuning the models. The top-performing TinyML variants were the Pr_LSTM for SoC and the Pr_CNN_GRU for SoH, each trained solely on ICA features. Pr_LSTM reached MAE = 0.0114 and RMSE = 0.0286 with a 4.0 kB footprint and 0.391 ms latency, while Pr_CNN_GRU achieved MAE = 0.0060 and RMSE = 0.0130, occupying just 3.4 kB and executing in 1.607 ms. Lastly, authors proposed an algorithm for up-scaling the estimations from cell to pack level, considering safe operation without cells' over or under dis/charge, and with the ability to perform optimal health management of the battery pack, enhancing its overall lifespan.

Considering future applications, these models could support the development of more holistic BMSs, which utilize the models' states estimation for balancing battery cells and thermal management. Moreover, a simpler architecture can be developed for SoH estimation, without the need of filters or features extraction from measurements, which can lower the complexity and computational requirements of the BMS. To further enhance inference efficiency, joint model optimization techniques can be explored, such as projection-quantization or pruning-quantization, when compatibility allows. These combined approaches can potentially reduce latency and energy consumption during embedded deployment while maintaining high accuracy. Lastly, a comparison analysis can be conducted for comparing the energy consumption of the models for state estimation, between cloud and on-edge solution.

CRediT authorship contribution statement

Spyridon Giazitzis: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Data curation, Conceptualization. **Jack Ferrari:** Writing – review & editing, Visualization, Software, Methodology, Investigation, Data curation, Conceptualization. **Antoni J. Woss:** Writing – review & editing, Methodology, Conceptualization. **Susheel Badha:** Writing – review & editing, Supervision. **Filippo Rosetti:** Writing – review & editing, Supervision. **Emanuele Ogliari:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Seyed Mahdi Miraftabzdeh, Mohammed Ali Khan, Navid Bayati, Dario Zaninelli, Deep learning methods and evaluation of the extensive carbon emission predictive solution for danish grid, Sustain. Energy Technol. Assessments 75 (2025) 104242.

[2] Mohsen Soori, Behrooz Arezoo, Roza Dastres, Artificial intelligence, machine learning and deep learning in advanced robotics, a review, Cogn. Robot. 3 (2023) 54–70.

[3] Carlos Vidal, Pawel Malysz, Phillip Kollmeyer, Ali Emadi, Machine learning applied to electrified vehicle battery state of charge and state of health estimation: State-of-the-art, Ieee Access 8 (2020) 52796–52814.

[4] Pudi Dhilleswararao, Srinivas Boppu, M Sabarimalai Manikandan, Linga Reddy Cenkeramaddi, Efficient hardware architectures for accelerating deep neural networks: Survey, IEEE Access 10 (2022) 131788–131828.

[5] Yanming Yang, Xin Xia, David Lo, John Grundy, A survey on deep learning for software engineering, ACM Comput. Surv. 54 (10s) (2022) 1–73.

[6] Kailong Liu, Kang Li, Qiao Peng, Cheng Zhang, A brief review on key technologies in the battery management system of electric vehicles, Front. Mech. Eng. 14 (2019) 47–64.

[7] Simin Peng, Yujian Wang, Aihua Tang, Yuxia Jiang, Jiarong Kan, Michael Pecht, State of health estimation joint improved grey wolf optimization algorithm and LSTM using partial discharging health features for lithium-ion batteries, Energy 315 (2025) 134293.

[8] Jamile Mohammadi Moradian, Amjad Ali, Xuehua Yan, Gang Pei, Shu Zhang, Ahmad Naveed, Khurram Shehzad, Zohreh Shahnavaz, Farooq Ahmad, Balal Yousaf, Sensors innovations for smart lithium-based batteries: Advancements, opportunities, and potential challenges, Nano-Micro Lett. 17 (1) (2025) 279.

[9] Chollet Nicolas, Bouchemal Naila, Ramdane-Cherif Amar, Tinyml smart sensor for energy saving in internet of things precision agriculture platform, in: 2022 Thirteenth International Conference on Ubiquitous and Future Networks, ICUFN, IEEE, 2022, pp. 256–259.

[10] Dickson NT How, MA Hannan, MS Hossain Lipu, Pin Jern Ker, State of charge estimation for lithium-ion batteries using model-based and data-driven methods: A review, Ieee Access 7 (2019) 136116–136136.

[11] Panagiotis Eleftheriadis, Spyridon Giazitzis, Cemay Ozmalatyalilar, Sonia Leva, Ricardo Zich, An overview of data-driven methods for the state of health estimation, in: 2023 IEEE International Conference on Environment and Electrical Engineering and 2023 IEEE Industrial and Commercial Power Systems Europe, EEEIC/I&CPS Europe, IEEE, 2023, pp. 1–6.

[12] Panagiotis Eleftheriadis, Spyridon Giazitzis, Sonia Leva, Emanuele Ogliari, Data-driven methods for the state of charge estimation of lithium-ion batteries: An overview, Forecasting 5 (3) (2023) 576–599.

[13] Lili Ma, Yonghong Xu, Hongguang Zhang, Fubin Yang, Xu Wang, Cheng Li, Co-estimation of state of charge and state of health for lithium-ion batteries based on fractional-order model with multi-innovations unscented Kalman filter method, J. Energy Storage 52 (2022) 104904.

[14] Panagiotis Eleftheriadis, Spyridon Giazitzis, Sonia Leva, Emanuele Ogliari, Transfer learning techniques for the lithium-ion battery state of charge estimation, IEEE Access 12 (2023) 993–1004.

[15] Marek Żyliński, Amir Nassibi, Ildar Rakhmatulin, Adil Malik, Christos Michael Papavassiliou, Danilo P Mandic, Deployment of artificial intelligence models on edge devices: A tutorial brief, IEEE Trans. Circuits Syst. II: Express Briefs 71 (3) (2023) 1738–1743.

[16] Partha Pratim Ray, A review on TinyML:: State-of-the-art and prospects, 2022.

[17] Liangzhen Lai, Naveen Suda, Vikas Chandra, Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus, 2018, arXiv preprint arXiv:1801.06601.

[18] Apache Software Foundation, Apache TVM: Open Deep Learning Compiler Stack, 2023, (Accessed 11 November 2025) https://tvm.apache.org/.

[19] Delia Velasco-Montero, Bart Goossens, Jorge Fernandez-Berni, Ángel Rodríguez-Vázquez, Wilfried Philips, A pipelining-based heterogeneous scheduling and energy-throughput optimization scheme for cnns leveraging apache tvm, IEEe Access 11 (2023) 35007–35021.

[20] Vasileios Tsoukas, Anargyros Gkogkidis, Eleni Boumpa, Athanasios Kakarountas, A review on the emerging technology of TinyML, ACM Comput. Surv. 56 (10) (2024) 1–37.

[21] Spyridon Giazitzis, Maciej Sakwa, Sonia Leva, Emanuele Ogliari, Susheel Badha, Filippo Rosetti, A case study of a tiny machine learning application for battery state-of-charge estimation, Electronics 13 (10) (2024) 1964.

[22] Panagiotis Eleftheriadis, PoliMi-TUB dataset - LG 18650HE4 li-ion battery, 2024.

[23] Danilo Pietro Pau, Alberto Aniballi, Tiny machine learning battery state-of-charge estimation hardware accelerated, Appl. Sci. 14 (14) (2024) 6240.

[24] Matthias Steinstraeter, Johannes Buberger, Dimitar Trifonov, Battery and heating data in real driving cycles, 2020.

[25] Anshul Kumar Yadav, Rajesh Kumar, Anil Kumar Saini, P Ragupathy, Aashish Ranjan, Anand Abhishek, et al., Edge computing enabled battery state of charge (SoC) estimation using TinyML techniques, in: 2024 IEEE International Conference on Power Electronics, Drives and Energy Systems, PEDES, IEEE, 2024, pp. 1–6.

[26] Spyridon Giazitzis, Maciej Sakwa, Emanuele Ogliari, Susheel Badha, Filippo Rosetti, Tiny machine learning for li-ion battery state of health estimation, in: 2024 IEEE 22nd Mediterranean Electrotechnical Conference, MELECON, IEEE, 2024, pp. 1019–1024.

[27] Matthieu Dubarry, Vojtech Svoboda, Ruey Hwu, Bor Yann Liaw, Incremental capacity analysis and close-to-equilibrium OCV measurements to quantify capacity fade in commercial rechargeable lithium batteries, Electrochem. Solid-State Lett. 9 (10) (2006) A454.

[28] Weiping Diao, Saurabh Saxena, Bongtae Han, Michael Pecht, Algorithm to determine the knee point on capacity fade curves of lithium-ion cells, Energies 12 (15) (2019) 2910.

[29] Giulia Crocioni, Danilo Pau, Jean-Michel Delorme, Giambattista Gruosso, Li-ion batteries parameter estimation with tiny neural networks embedded on intelligent IoT microcontrollers, IEEE Access 8 (2020) 122135–122146.

[30] GitHub - tensorflow/tflite-micro: Infrastructure to enable deployment of ML models to low-power resource-constrained embedded targets.

[31] Yuqin Weng, Wenkai Guan, Cristinel Ababei, Prediction of remaining useful life and cell temperature for li-ion batteries using TinyML, in: 2024 IEEE 67th International Midwest Symposium on Circuits and Systems, MWSCAS, IEEE, 2024, pp. 562–566.

[32] Antoni Woss, Compressing neural networks using network projection Published: May 2023 https://it.mathworks.com/company/technical-articles/compressing-neural-networks-using-network-projection.html.

[33] Yuqing Yang, Stephen Bremner, Chris Menictas, Merlinde Kay, Modelling and optimal energy management for battery energy storage systems in renewable energy systems: A review, Renew. Sustain. Energy Rev. 167 (2022) 112671.

[34] Osman Demirci, Sezai Taskin, Erik Schaltz, Burcu Acar Demirci, Review of battery state estimation methods for electric vehicles-part I: SOC estimation, J. Energy Storage 87 (2024) 111435.

[35] Eyyup Aslan, Yusuf Yasa, C-Rate-and temperature-dependent state-of-charge estimation method for li-ion batteries in electric vehicles, Energies 17 (13) (2024) 3187.

[36] Mohamed Elmahallawy, Tarek Elfouly, Ali Alouani, Ahmed M Massoud, A comprehensive review of lithium-ion batteries modeling, and state of health and remaining useful lifetime prediction, Ieee Access 10 (2022) 119040–119070.

[37] Bo Yang, Yucun Qian, Qiang Li, Qian Chen, Jiyang Wu, Enbo Luo, Rui Xie, Ruyi Zheng, Yunfeng Yan, Shi Su, et al., Critical summary and perspectives on state-of-health of lithium-ion battery, Renew. Sustain. Energy Rev. 190 (2024) 114077.

[38] Ashikur Rahman, Xianke Lin, Li-ion battery individual electrode state of charge and degradation monitoring using battery casing through auto curve matching for standard cccv charging profile, Appl. Energy 321 (2022) 119367.

[39] Ines Baccouche, Sabeur Jemmali, Bilal Manai, Alexandros Nikolian, Noshin Omar, Najoua Essoukri Ben Amara, Li-ion battery modeling and characterization: an experimental overview on NMC battery, Int. J. Energy Res. 46 (4) (2022) 3843–3859.

[40] Hossam M Hussein, Ahmed Aghmadi, Mahmoud S Abdelrahman, SM Sajjad Hossain Rafin, Osama Mohammed, A review of battery state of charge estimation and management systems: Models and future prospective, Wiley Interdiscip. Rev.: Energy Environ. 13 (1) (2024) e507.

[41] Merve Tekin, M. İhsan Karamangil, Comparative analysis of equivalent circuit battery models for electric vehicle battery management systems, J. Energy Storage 86 (2024) 111327.

[42] Maheshwari Adaikkappan, Nageswari Sathiyamoorthy, Modeling, state of charge estimation, and charging of lithium-ion battery in electric vehicle: a review, Int. J. Energy Res. 46 (3) (2022) 2141–2165.

[43] Islam Md Monirul, Li Qiu, Rukhsana Ruby, Accurate SOC estimation of ternary lithium-ion batteries by hppc test-based extended Kalman filter, J. Energy Storage 92 (2024) 112304.

[44] Spyridon Giazitzis, Abdisamad Ahmed Isse, Nicola Blasuttigh, Alessandro Massi Pavan, Davide M Raimondo, Susheel Badha, Filippo Rosetti, Emanuele Ogliari, TinyML models for SoH estimation of lithium-ion batteries based on electrochemical impedance spectroscopy, J. Power Sources 653 (2025) 237568.

[45] Fatma Ahmed, Khaled Abulsaud, Ahmed M. Massoud, On equivalent circuit model-based state-of-charge estimation for lithium-ion batteries in electric vehicles, IEEE Access (2025).

[46] Lin He, Xingwen Hu, Guangwei Yin, Guoqiang Wang, Xingguo Shao, Jichao Liu, A current dynamics model and proportional–integral observer for state-of-charge estimation of lithium-ion battery, Energy 288 (2024) 129701.

[47] Xuemei Wang, Ruiyun Gong, Zhao Yang, Longyun Kang, State of charge estimation of lithium-ion batteries based on online ocv curve construction, Batteries 10 (6) (2024) 208.

[48] Kevin Moy, Muhammad Aadil Khan, Simone Fasolato, Gabriele Pozzato, Anirudh Allam, Simona Onori, Second-life lithium-ion battery aging dataset based on grid storage cycling, Data Brief 57 (2024) 111046.

[49] M.M. Kabir, Dervis Emre Demirocak, Degradation mechanisms in li-ion batteries: a state-of-the-art review, Int. J. Energy Res. 41 (14) (2017) 1963–1986.

[50] Zheyuan Pang, Kun Yang, Zhengxiang Song, Pengcheng Niu, Guangyang Chen, Jinhao Meng, A new method for determining SOH of lithium batteries using the real-part ratio of EIS specific frequency impedance, J. Energy Storage 72 (2023) 108693.

[51] Qunming Zhang, Cheng-Geng Huang, He Li, Guodong Feng, Weiwen Peng, Electrochemical impedance spectroscopy based state-of-health estimation for lithium-ion battery considering temperature and state-of-charge effect, IEEE Trans. Transp. Electrification 8 (4) (2022) 4633–4645.
[52] Pietro Iurilli, Claudio Brivio, Rafael E Carrillo, Vanessa Wood, Physics-based SoH estimation for li-ion cells, Batteries 8 (11) (2022) 204.
[53] Mei Zhang, Zhihui Wang, SOC estimation for lithium batteries using a CNN-attention-LSTM model, J. Energy Storage 130 (2025) 117479.
[54] Wenxiang Li, K. L. Eddie Law, Deep learning models for time series forecasting: A review, IEEE Access 12 (2024) 92306–92327.
[55] Yunhong Che, Søren Byg Vilsen, Jinhao Meng, Xin Sui, Remus Teodorescu, Battery health prognostic with sensor-free differential temperature voltammetry reconstruction and capacity estimation based on multi-domain adaptation, ETransportation 17 (2023) 100245.
[56] Shahram Hanifi, Andrea Cammarono, Hossein Zare-Behtash, Advanced hyperparameter optimization of deep learning models for wind power prediction, Renew. Energy 221 (2024) 119700.
[57] Qingkai Xing, Ming Zhang, Yaping Fu, Kai Wang, Transfer learning to estimate lithium-ion battery state of health with electrochemical impedance spectroscopy, J. Energy Storage 110 (2025) 115345.
[58] Yongsong Yang, Yuchen Xu, Yuwei Nie, Jianming Li, Shizhuo Liu, Lijun Zhao, Quanqing Yu, Chengming Zhang, Deep transfer learning enables battery state of charge and state of health estimation, Energy 294 (2024) 130779.
[59] Sesidhar D.V.S.R., Chandrashekhar Badachi, Robert C. Green II, A review on data-driven SOC estimation with li-ion batteries: Implementation methods & future aspirations, J. Energy Storage 72 (2023) 108420.
[60] Seunghyeon Oh, Jiyong Kim, Il Moon, Hybrid data-driven deep learning model for state of charge estimation of li-ion battery in an electric vehicle, J. Energy Storage 97 (2024) 112887.
[61] Francesco Santoni, Alessio De Angelis, Antonio Moschitta, Paolo Carbone, Matteo Galeotti, Lucio Cinà, Corrado Giammanco, Aldo Di Carlo, A guide to equivalent circuit fitting for impedance analysis and battery state estimation, J. Energy Storage 82 (2024) 110389.
[62] Panagiotis Eleftheriadis, Spyridon Giazitzis, Julia Kowal, Sonia Leva, Emanuele Ogliari, Joint state of charge and state of health estimation using bidirectional lstm and bayesian hyperparameter optimization, IEEE Access (2024).
[63] Jing Hou, Tian Gao, Yan Yang, Xin Wang, Yuenan Yang, Siying Meng, Battery inconsistency evaluation based on hierarchical weight fusion and fuzzy comprehensive evaluation method, J. Energy Storage 84 (2024) 110878.
[64] Shiyun Liu, Kang Li, James Yu, Battery pack condition monitoring and characteristic state estimation: Challenges, techniques, and future prospectives, J. Energy Storage 105 (2025) 114446.
[65] Mina Naguib, Phillip Kollmeyer, Ali Emadi, Lithium-ion battery pack robust state of charge estimation, cell inconsistency, and balancing: Review, IEEE Access 9 (2021) 50570–50582.
[66] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, Kurt Keutzer, A survey of quantization methods for efficient neural network inference, in: Low-Power Computer Vision, Chapman and Hall/CRC, 2022, pp. 291–326.
[67] Sunil Vadera, Salem Ameen, Methods for pruning deep neural networks, Ieee Access 10 (2022) 63280–63300.
[68] Tushar Choudhary, Ashish Mishra, A comprehensive survey on model compression and acceleration, Artif. Intell. Rev. 53 (2020) 5113–5155.
[69] Colby R. Banbury, et al., Benchmarking TinyML systems: Challenges and directions, in: Proceedings of MLSys, 2021.
[70] Xiyu Yu, Tongliang Liu, Xinchao Wang, Dacheng Tao, On compressing deep models by low rank and sparse decomposition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7370–7379.
[71] Wei Dai, Jicong Fan, Yiming Miao, Kai Hwang, Deep learning model compression with rank reduction in tensor decomposition, IEEE Trans. Neural Netw. Learn. Syst. (2023).
[72] Defu Liu, Yixiao Zhu, Zhe Liu, Yi Liu, Changlin Han, Jinkai Tian, Ruihao Li, Wei Yi, A survey of model compression techniques: Past, present, and future, Front. Robot. AI 12 (2025) 1518965.
[73] Pierre Vilar Dantas, Waldir Sabino Da Silva, Lucas Carvalho Cordeiro, Celso Barbosa Carvalho, A comprehensive review of model compression techniques in machine learning, Appl. Intell. 54 (22) (2024) 11804–11844.
[74] Bo Zhao, Weige Zhang, Yanru Zhang, Caiping Zhang, Chi Zhang, Junwei Zhang, Lithium-ion battery remaining useful life prediction based on interpretable deep learning and network parameter optimization, Appl. Energy 379 (2025) 124713.
[75] Vasileios Tsouvalas, Aaqib Saeed, Tanir Ozcelebi, Nirvana Meratnia, Communication-efficient federated learning through adaptive weight clustering and server-side distillation, in: ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2024, pp. 5805–5809.
[76] Vishwanath Saragadam, Randall Balestriero, Ashok Veeraraghavan, Richard G Baraniuk, DeepTensor: Low-rank tensor decomposition with deep network priors, IEEE Trans. Pattern Anal. Mach. Intell. (2024).
[77] Jianping Gou, Baosheng Yu, Stephen J Maybank, Dacheng Tao, Knowledge distillation: A survey, Int. J. Comput. Vis. 129 (6) (2021) 1789–1819.
[78] Jianping Gou, Yu Chen, Baosheng Yu, Jinhua Liu, Lan Du, Shaohua Wan, Zhang Yi, Reciprocal teacher-student learning via forward and feedback knowledge distillation, IEEE Trans. Multimed. 26 (2024) 7901–7916.
[79] Rufan Yang, Hung Dinh Nguyen, Temperature distribution learning of li-ion batteries using knowledge distillation and self-adaptive models, Appl. Energy 382 (2025) 125196.
[80] Lei Yao, Shiming Xu, Aihua Tang, Fang Zhou, Junjian Hou, Yanqiu Xiao, Zhijun Fu, A review of lithium-ion battery state of health estimation and prediction methods, World Electr. Veh. J. 12 (3) (2021) 113.
[81] Linfeng Zheng, Jianguo Zhu, Dylan Dah-Chuan Lu, Guoxiu Wang, Tingting He, Incremental capacity analysis and differential voltage analysis based state of charge and capacity estimation for lithium-ion batteries, Energy 150 (2018) 759–769.
[82] Gyuwon Seo, Jaeyun Ha, Moonsu Kim, Jihyeon Park, Jaewon Lee, Eunoak Park, Sungyool Bong, Kiyoung Lee, Soon Jong Kwon, Seung-pil Moon, et al., Rapid determination of lithium-ion battery degradation: High C-rate lam and calculated limiting LLI, J. Energy Chem. 67 (2022) 663–671.
[83] Jiangtao He, Xiaolei Bian, Longcheng Liu, Zhongbao Wei, Fengjun Yan, Comparative study of curve determination methods for incremental capacity analysis and state of health estimation of lithium-ion battery, J. Energy Storage 29 (2020) 101400.
[84] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, Nando de Freitas, Taking the human out of the loop: A review of Bayesian optimization, Proc. IEEE 104 (1) (2016) 148–175.
[85] Suze Zhang, DlCodegenBench, 2025, Published: April 2025 https://www.mathworks.com/matlabcentral/fileexchange/180806-dlcodegenbench.