

Preprint Version Notice

This document is a preprint of the following publication:

Faryar, D., Berta, R., Fresta, M., Saad, A., Lazzaroni, L., Ballout, H., Srouf, O., and Bellotti, F. (2026). Lightweight Deep Learning for SOC Estimation of Various Lithium-Ion Batteries on Xilinx ZCU104 FPGA. In: Ruo Roch, M., et al. Applications in Electronics Pervading Industry, Environment and Society. ApplePies 2025. Lecture Notes in Electrical Engineering, vol 1553. Springer, Cham. https://doi.org/10.1007/978-3-032-17174-0_4 .

Please cite the published version of this work as indicated above.

Lightweight Deep Learning for SOC Estimation of Various Lithium-Ion Batteries on Xilinx ZCU104 FPGA

Danoush Faryar, Riccardo Berta, Matteo Fresta, Ammar Saad, Luca Lazzaroni,
Hadi Ballout, Ossama Srour, and Francesco Bellotti

Department of Electrical, Electronic and Telecommunication Engineering (DITEN),
University of Genoa, Via Opera Pia 11a, 16145 Genoa, Italy
matteo.fresta@edu.unige.it

Abstract. Accurate estimation of the State of Charge (SOC) in Lithium-ion batteries is critical for the safe and efficient operation of electric vehicles and energy storage systems. This paper proposes a lightweight deep learning framework for SOC estimation, targeting deployment on resource-constrained edge AI hardware. We evaluate several recurrent and convolutional neural network architectures using the UNIBO Powertools Dataset, with a GRU-based model achieving the best balance between accuracy and memory footprint. The GRU model, featuring a single recurrent layer followed by three dense layers, is reimplemented in C++ and deployed on a Xilinx ZCU104 FPGA. The deployment achieves a latency of 38.76 ms and an energy consumption of 59.70 mJ per inference, with minimal resource usage. These results demonstrate the suitability of our approach for real-time, low-power battery management applications and confirm the benefits of FPGA acceleration for embedded SOC estimation tasks.

Keywords: State of Charge Estimation, Lithium-Ion Batteries, Deep Learning, GRU, Edge AI, FPGA Deployment, Real-Time Inference, Embedded Systems.

1 Introduction

The transition toward sustainable energy systems has accelerated the adoption of electric vehicles [1, 2] and energy storage solutions [3], driving the demand for high-performance and reliable battery technologies. Lithium-ion batteries, as the prevailing energy source for these applications, require advanced Battery Management Systems (BMS) to ensure safety, efficiency, and longevity. A central component of any BMS is the estimation of the State of Charge (SOC), which reflects the remaining usable energy in the battery as a percentage of its total capacity [4]. Accurate SOC estimation is essential for preventing over-charge and over-discharge conditions, both of which can degrade battery performance or cause safety risks [5].

Moreover, reliable real-time SOC prediction enables improved performance, enhanced safety, and extended battery lifespan across a variety of applications, from consumer electronics to industrial automation. Traditional model-based estimation techniques, used in collaboration with adaptive filters such as Kalman filter, H-infinity filter, and Particle filter[6], while effective, often require precise modeling and calibration, which may not generalize well across different battery types and usage conditions.

Recent advancements in deep learning methods have enabled high-accuracy estimation of the SOC in Lithium-ion batteries, as demonstrated in several studies that

employed advanced neural network architectures. Dubey et al. [7] applied an LSTM with Bayesian optimization to reach an RMSE of 0.87% and MAE of 0.64% at 40°C; Chemali et al. [8] utilized an LSTM-RNN to achieve an RMSE of 0.69% and MAE of 0.57% at 10°C. Wang et al. [9] used a residual Convolutional Neural Networks (CNN) to obtain an RMSE of 1.26% and MAE of 1.00% across various temperatures. Hannan et al. [10] employed a self-supervised Transformer model to obtain an RMSE of 0.90% and MAE of 0.44% at room temperature.

However, deploying such estimators in embedded and resource-constrained environments, such as electric vehicles or portable tools, requires solutions that are not only accurate, but also energy-efficient and capable of real-time operation. In this context, Field-Programmable Gate Arrays (FPGAs) provide a powerful hardware platform, enabling parallel execution, low latency, and reduced power consumption [11].

In this work, we conduct a comprehensive evaluation of several deep learning architectures for SOC estimation, including LSTM, Bidirectional LSTM, GRU, Bidirectional GRU, and 1D-CNN. To support this study, we utilize the UNIBO Powertool Dataset [12], which includes measurements from multiple batteries of different manufacturers, nominal capacities, and aging stages, providing a rich benchmark for state-of-the-art SOC prediction that spans multiple discharge cycles and battery degradation states. We propose a sliding windows preprocessing approach, which segments the data into shorter, fixed-length sequences and allows SOC estimation at any point in the cycle, not only when starting from 100 %. This confirms maximum flexibility, making our method suitable for dynamic real-world applications.

We evaluate each model in terms of the trade-off between predictive accuracy and memory footprint, with the goal of identifying lightweight models suitable for real-time deployment on edge hardware such as FPGAs. The best-performing model is synthesized on a Xilinx ZCU104 FPGA to demonstrate its feasibility for real-time, low-power deployment on edge hardware. Our results highlight the potential of lightweight neural networks to provide accurate and reliable SOC estimation with limited resource and energy constraints.

2 Methodology

2.1 Dataset

This study utilizes the UNIBO Powertools Dataset [12], collected by the University of Bologna to analyze the behavior of Lithium-ion batteries during repeated charge and discharge cycles. The dataset was specifically designed to evaluate a variety of battery cells intended for use in cleaning equipment such as vacuum and automated floor cleaners. Our analysis focuses on the standard test configuration, and the dataset comprises 16 batteries from different manufacturers, with nominal capacities ranging from 2.0Ah to 4.0Ah. The experiments were designed to capture battery behavior across various stages of usage, from beginning of life to end of life.

In the standard test protocol, batteries were discharged at a constant current of 5A. The cycling procedure followed a structured approach: the charge cycle was conducted using a Constant Current-Constant Voltage method at 1.8A and 4.2V with a 100mA cut-off, while the discharge cycle applied a constant current until the cut-off voltage of

2.5V was reached. This cycle was repeated 100 times. During each discharge cycle, data was recorded every 10 seconds, providing high-resolution temporal information on battery dynamics. The input features included voltage (V), current (I), and temperature (T) measurements, with the corresponding SOC percentage as the target output for model training and evaluation.

2.2 Preprocessing

Unlike the original dataset configuration [12], which uses the full sequence of 287 time steps as model input, and thus requires each cycle to start from a fixed full-charge reference at 100 %, our approach adopts a flexible windowing strategy to segment the data into shorter sequences of fixed length. This means that it can be applied at any SOC point, and we are not constrained to begin from 100 %, allowing the model to operate during partial cycles, mid-discharge, or real-world varied usage conditions, making our method far more advantageous in practical scenarios.

Also, the shorter windowing design choice aims to reduce the amount of input data required for each prediction, searching for smaller models suitable for edge deployment. However, this also increases the difficulty of the task, as the model SOC prediction has a more limited temporal context. We evaluated four different window lengths, in particular 6, 12, 18, and 30 time steps, corresponding to 1, 2, 3, and 5 minutes, to assess the impact of sequence length on model performance. To augment the dataset and increase the number of training samples, we applied a sliding window with a stride of 1. The original dataset already includes a predefined training-test split, and we further partitioned 20% of the training set to serve as a validation set for model selection during hyperparameter optimization. For each model architecture, we conducted 50 Bayesian optimization trials, with a maximum of 100 training epochs per trial. An early stopping criterion with a patience of 15 epochs was applied based on validation loss. The results reported in this paper correspond to the best-performing configuration identified on the validation set.

2.3 Deep Learning Models

We implemented and evaluated five deep learning architectures for the task of SOC estimation, each selected for its ability to model temporal dependencies and extract relevant patterns from sequential data. To optimize model performance, we employed Bayesian optimization to explore the best configuration of hyperparameters within predefined ranges for each architecture:

- LSTM and Bidirectional LSTM models: Both architectures were configured with 1 to 3 layers, each containing 32 to 256 units (step size of 32), using the tanh activation function. The `return_sequences` parameter was enabled for all layers except the final one to maintain compatibility with subsequent dense layer. The bidirectional variant enhances context modeling by processing sequences in both directions.
- GRU and Bidirectional GRU models: Like the LSTM-based architectures, they featured 1 to 3 layers with 32 to 256 units per layer. GRUs provide a more lightweight alternative to LSTMs, with faster training and fewer parameters. The bidirectional GRU extends this by adding context from both directions of the input sequence.

- 1D-CNN model: The 1D-CNN model included between 1 and 3 convolutional layers, each with 32 to 256 filters with step 32 and kernel sizes ranging from 2 to 5. The SELU activation function was used, and padding 'same' to preserve sequence length.

Bayesian optimization also tuned the number of dense layers (1-3), units per layer (64-512), dropout rate (0.0-0.5), and selected the learning rate from a discrete set of values: 1×10^{-5} , 5×10^{-5} , 1×10^{-4} , 5×10^{-4} , 1×10^{-3} . All models were trained using the Huber loss function, which ensures robustness to outliers and stable convergence.

3 Experimental Results

To evaluate the effectiveness of the proposed approach, we conducted different experiments, comparing different architectures and window size (Table 1). Among all tested models, the 30 window size GRU demonstrated the best trade-off between accuracy and size, achieving near the lowest error values (RMSE = 3.379%, MAE = 2.693%) while maintaining a compact .tflite size of just 741 KB. This configuration was selected for further deployment and analysis. The selected model configuration consists of a single GRU layer with 192 units, followed by three fully connected dense layers. A dropout layer with a rate of 0.1 was applied after the GRU. The dense layers were configured with 160, 64, and 512 units respectively, interleaved with dropout of 0.2 and 0.1 after the first and second dense layers. The learning rate was set to 1×10^{-3} .

Table 1. Performance comparison of models for different window sizes.

Model	Window Size	RMSE (%)	MSE (%)	MAE (%)	Size (KB)
LSTM	6	4.663	0.217	3.730	1,199
	12	4.430	0.196	3.544	1,628
	18	4.103	0.168	3.282	790
	30	4.096	0.168	3.277	1,118
Bi-LSTM	6	4.519	0.204	3.615	4,760
	12	4.282	0.183	3.426	1,290
	18	4.157	0.173	3.325	2,820
	30	3.827	0.147	3.061	1,370
GRU	6	3.864	0.149	3.091	1,760
	12	3.699	0.137	2.959	980
	18	3.566	0.127	2.853	1,020
	30	3.379	0.114	2.693	741
Bi-GRU	6	4.718	0.223	3.774	5,520
	12	3.619	0.131	2.895	3,440
	18	3.483	0.121	2.807	1,670
	30	3.338	0.111	2.610	4,450
1D-CNN	6	4.411	0.195	3.529	588
	12	3.965	0.157	3.112	2,101
	18	3.844	0.148	3.075	2,831
	30	3.825	0.146	2.968	1,340

To validate our choice of smaller windows techniques, we compared in Table 2 our selected model (GRU, window size 30) against a state-of-the-art LSTM-based solution

from the literature, which requires a fixed starting reference. While our model achieves slightly worse accuracy (RMSE = 3.38% vs 1.81%), it significantly outperforms in terms of size and number of parameters. Our GRU model requires less than one-fifth of the memory (741 KB vs 3,923 KB) and over five times fewer parameters (188K vs nearly 1M), making it better suitable for edge deployment than the state-of-the-art.

Table 2. Comparison with state-of-the-art.

Fixed reference	Model	Input	RMSE (%)	MAE (%)	Size (KB)	Parameters
Yes ([12])	LSTM	278, 3	1.81	0.96	3,923	996,993
No (our)	GRU	30, 3	3.38	2.69	741	188,449

To validate the feasibility of deployment on edge AI hardware, we synthesized the GRU (window 30) model on a Xilinx ZCU104 FPGA board. The model was reimplemented from scratch in C++, without relying on any high-level synthesis from Python-based frameworks. We then used Xilinx Vivado to simulate the deployment of the C++ design onto the FPGA to extract accurate measurements of latency, power usage, and energy consumption. To ensure a fair comparison, both GRU and LSTM models were implemented without applying any architecture-specific optimizations. This allows for an unbiased evaluation of their inherent efficiency under the same development conditions. Table 3 reports the results in terms of latency, power consumption, energy usage, and resource utilization. Our GRU-based system achieved a latency of 38.76 ms and consumed only 59.70 mJ per inference, confirming its suitability for low-power real-time applications. Notably, the model utilized just 18.73% of Flip-Flops and 37.51% of Look-Up Tables, leaving room for further integration or multi-model deployment. By contrast, the state-of-the-art LSTM, using full window size as input, exhibits 15× higher latency, 16× greater energy consumption, and increased resource demand, making it less suitable for edge deployment.

Table 3. FPGA deployment performance on ZCU104.

Model	Latency (ms)	Power (W)	Energy (mJ)	FF (%)	LUT (%)	BRAM (%)	DSP (%)
LSTM ([12])	585.45	1.61	942.58	37.91	46.13	69.74	29.41
GRU (our)	38.76	1.54	59.70	18.73	37.51	73.72	20.72

4 Conclusion

We presented a lightweight and energy-efficient deep learning approach for SOC estimation of Lithium-ion batteries, with a focus on real-time deployment on edge AI hardware. Among several deep learning architectures, a GRU-based model with a window size of 30 time steps demonstrated the best trade-off between accuracy and memory footprint, achieving an RMSE of 3.38% and a size of only 741 KB.

Deployability was assessed with a Xilinx ZCU104 FPGA hardware synthesis. The FPGA deployment achieved a latency of 38.76 ms and an energy consumption of 59.70 mJ per inference, confirming its suitability for low-power, real-time applications. The system also showed limited hardware resource utilization, offering potential for additional integration and highlights the effectiveness of direct low-level implementation.

Future work may concern dynamic quantization and mixed-precision techniques to further reduce model size and power, as well as integration with adaptive battery management systems for online operation [13].

References

1. Zhang, D., Zhong, C., Xu, P., Tian, Y.: Deep Learning in the State of Charge Estimation for Li-Ion Batteries of Electric Vehicles: A Review. *Machines*. 10, 912 (2022). <https://doi.org/10.3390/machines10100912>.
2. Damodarin, U.M., Cardarilli, G.C., Di Nunzio, L., Re, M., Spanò, S.: Smart Electric Vehicle Charging Management Using Reinforcement Learning on FPGA Platforms. *Sensors*. 25, 2585 (2025). <https://doi.org/10.3390/s25082585>.
3. Dhungana, H., Bellotti, F., Fresta, M., Dhungana, P., Berta, R.: Assessing a Measurement-Oriented Data Management Framework in Energy IoT Applications. *Energies*. 18, 3347 (2025). <https://doi.org/10.3390/en18133347>.
4. Lima, A.B. de, Salles, M.B.C., Cardoso, J.R.: State-of-Charge Estimation of a Li-Ion Battery using Deep Forward Neural Networks, <http://arxiv.org/abs/2009.09543>, (2020). <https://doi.org/10.48550/arXiv.2009.09543>.
5. Ouyang, D., Chen, M., Liu, J., Wei, R., Weng, J., Wang, J.: Investigation of a commercial lithium-ion battery under overcharge/over-discharge failure conditions. *RSC Adv*. 8, 33414–33424 (2018). <https://doi.org/10.1039/C8RA05564E>.
6. Omiloli, K., Awelewa, A., Samuel, I., Obiazi, O., Katende, J.: State of charge estimation based on a modified extended Kalman filter. *International Journal of Electrical and Computer Engineering (IJECE)*. (2023). <https://doi.org/10.11591/ijece.v13i5.pp5054-5065>.
7. Dubey, A., Zaidi, A., Kulshreshtha, A.: State-of-Charge Estimation Algorithm for Li-ion Batteries using Long Short-Term Memory Network with Bayesian Optimization. In: 2022 Second International Conference on Interdisciplinary Cyber Physical Systems (ICPS). pp. 68–73 (2022). <https://doi.org/10.1109/ICPS55917.2022.00021>.
8. Chemali, E., Kollmeyer, P.J., Preindl, M., Ahmed, R., Emadi, A.: Long Short-Term Memory Networks for Accurate State-of-Charge Estimation of Li-ion Batteries. *IEEE Transactions on Industrial Electronics*. 65, 6730–6739 (2018). <https://doi.org/10.1109/TIE.2017.2787586>.
9. Wang, Y.-C., et al.: State-of-Charge Estimation for Lithium-Ion Batteries Using Residual Convolutional Neural Networks. *Sensors*. (2022). <https://doi.org/10.3390/s22166303>.
10. Hannan, M.A., et al.: Deep learning approach towards accurate state of charge estimation for lithium-ion batteries using self-supervised transformer model. *Sci Rep*. 11, 19541 (2021). <https://doi.org/10.1038/s41598-021-98915-8>.
11. Spagnolo, F., Corsonello, P., Frustaci, F., Perri, S.: Efficient implementation of signed multipliers on FPGAs. *Computers and Electrical Engineering*. 116, 109217 (2024). <https://doi.org/10.1016/j.compeleceng.2024.109217>.
12. Wong, K.L., Bosello, M., Tse, R., Falcomer, C., Rossi, C., Pau, G.: Li-Ion Batteries State-of-Charge Estimation Using Deep LSTM at Various Battery Specifications and Discharge Cycles. *Conference on Information Technology for Social Good. Association for Computing Machinery, New York, USA* (2021). <https://doi.org/10.1145/3462203.3475878>.
13. Gianoglio, C., Ragusa, E., Gastaldo, P., Gallesi, F., Guastavino, F.: Online Predictive Maintenance Monitoring Adopting Convolutional Neural Networks. *Energies*. 14, 4711 (2021). <https://doi.org/10.3390/en14154711>.