foxBMS - free and open BMS platform focused on functional safety and AI

Stefan Waldhör, Steffen Bockrath, Martin Wenger, Radu Schwarz, Dr. Vincent R. H. Lorentz

Fraunhofer Institute for Integrated Systems and Device Technology IISB, Germany

Corresponding author: Radu Schwarz, radu.schwarz@iisb.fraunhofer.de

The Power Point Presentation will be made available after the conference.

Abstract

The last years have shown a strong market demand for lithium-ion battery systems with higher energy densities, longer lifetimes, and lower costs, but at the same time without compromising safety. To help developers, engineers and researchers worldwide, Fraunhofer IISB has established the free and open source Battery Management System (BMS) development platform ”foxBMS”. The foxBMS platform consists of a modular hardware and software architecture and a complete software development toolchain. Based on the experience providing foxBMS-based solutions to customers and the research community, the next generation of foxBMS is strongly focused on functional safety standards. The hardware architecture and the hardware components themselves help to ensure that functional safety standards are met. Additionally, foxBMS supports a workflow for implementing Artificial Intelligence (AI)-based battery state estimators for the BMS. Using foxBMS as a data generator within this workflow a Neural Network (NN) based on a Long Short-Term Memory (LSTM) is trained offline to estimate the state of charge (SOC) based on the current and voltage measurement input. The simulation results, obtained with the trained NN running online on the device, are shown in this paper.

1 Introduction

Worldwide the markets are demanding battery systems with higher energy densities, longer lifetimes, and lower costs, but without compromising safety. Consequently, the efforts to design suitable battery management systems according to these requirements are getting higher. To help developers, engineers and researchers worldwide, Fraunhofer IISB has established a free and open Battery Management System (BMS) platform called foxBMS[1] in 2016 [1]. This BMS platform consists of a modular hardware and software architecture and includes a full-featured toolchain for software development. The second generation of the foxBMS platform will keep up the success of the foxBMS project by adding important aspects of functional safety to the hardware and software and their respective development processes [2]. By providing certification ready hardware and software function blocks, the time to market including the certification process according to specific norms (e.g., ISO 26262, IEC 61508, DO-254, DO-178) can be reduced. The other challenging domain in the field of batteries is bringing advanced algorithms rapidly from development environments into the embedded domain of the BMS. foxBMS enables developers to easily share battery related data from the battery system with an algorithm running on a connected device. This paper shows how to use foxBMS to generate battery usage data, preprocess this data and use these results to train a Long Short-Term Memory (LSTM) Neural Network (NN). The resulting NN is then transferred to a Processing Unit (PU), which can be directly integrated on the edge of the foxBMS hardware or reside in the cloud. The PU is connected to foxBMS with a communication interface (Controller Area Network (CAN), Serial Peripheral Interface (SPI) or Ethernet) and continuously fed with the data stream of battery usage data. The model is able to precisely predict the state of charge (SOC) of the battery. To illustrate

[1] https://foxbms.org

the simplicity in the workflow, a Raspberry Pi has been used, but any other CPU can be connected to the BMS and thus algorithms can be easily developed and tested.
In addition, the OSI approved open source license of the software (BSD 3-Clause) and the open license of the hardware and the documentation (both CC BY 4.0) allows developers and users to freely adapt and enhance the existing source base to their specific requirements and applications.
The paper is structured into two main parts: first an overview of the hardware architecture and function safety considerations are given, followed by the description of the workflow of integrating an AI based battery state estimator into the BMS.

2 BMS for Automotive Applications

The foxBMS platform serves to developers and engineers of battery systems and battery management systems, not only in the automotive domain, by providing a bleeding edge BMS development platform. The following design resources are provided free of charge and free for commercial use through adapted open source licenses:

- Hardware design files including complete Altium Designer source files (schematics and board layout) and bill of materials
- Embedded software source code including operating system, low level drivers and high level configurable BMS application software
- Toolchain for the development, debugging and deployment of the embedded software
- Battery management system users manual and documentation

As foxBMS is meant to be a development and evaluation platform for BMS applications, it is not focused on costs or size, but optimized for features, modularity, configurability and compatibility. The foxBMS development was driven by the experience gained over 2 decades in the field of BMS development in various domains (e.g., automotive, aviation, railway, marine) and is designed to cover most state-of-the-art use-cases in these domains. This design philosophy and the implementation makes the foxBMS platform versatile starting from the first concept and proof of concept phase, to the first prototype and industrialized product. By adding high data rate communication interfaces, the foxBMS platform not only helps BMS hardware and embedded software designers, but also enables faster algorithm and battery state estimator development. Through its Ethernet and CAN) interfaces it serves as a data generator to be used in the development of AI based state estimators. The next sections describe the hardware architecture and the according functional safety approach more detailed.

2.1 Hardware Architecture

The architecture of foxBMS is designed similar to automotive state-of-the-art systems: BMS Slave Unit are mounted on each battery module. The BMS Slave Units are then connected to a central BMS Master Unit via a proprietary communication interface depending on the used analog frontend. The following list gives a short overview of the functions the BMS Master Unit of foxBMS incorporates:

- Data acquisition from battery sensors: BMS Slave Unit (e.g., battery cell voltages and temperatures), global pack current sensor, and other sensors
- Processing of the sensor data for monitoring the Safe Operating Area (SOA) of the battery and state estimation of the battery (e.g., state of charge, state of health, state of function, state of safety)
- Communication with higher level vehicle control units (e.g., VCU)
- Control of actors (e.g., contactors, pumps, fans) and additional safety components

In addition, various communication interfaces and components (e.g., large non-volatile data storage) are added, meant to be used during BMS and algorithm development and may not be part of a industrialized automotive BMS solution. Further, a flexible selection of monitoring solutions provided by various battery monitoring IC vendors is covered

by a modular approach: the interface electronics between the microcontroller and the proprietary communication interface of the IC vendor is kept as an addon board. Complementary, the embedded software interface is kept lean to enable easy selection and integration of one of the many supported monitoring solutions.

2.2 Functional Safety

At first sight the idea of having a development platform covering multiple domains and various use-cases in these domains might seem contradictory to any functional safety approach. It is clear that any certification is strongly use-case dependent and has to take the surrounding system, environmental and regulatory conditions into consideration. However, after analyzing many use-cases it becomes obvious that certain functional blocks are present in the majority of battery systems and it would make sense to re-use these blocks and the documentation required in the certification process. Over the course of the foxBMS II project, several battery systems for use-cases in different domains will be developed in a manner that they can be certified. From a high level perspective all these projects differ, however, following a top-down approach, similar functional blocks will be required towards the lower system levels. The system architecture, as well as the according documentation and processes will be set up to support this modular approach, in order to end up with reusable units of hardware, software and documentation suitable for certification.
In a next step, this idea can also be transferred to certification processes in other domains, e.g., in industrial and aerospace applications. In fact, both, ISO 26262 in the automotive sector DO-254 in the aerospace sector are derived from IEC 61508, defining the functional safety development process in the industrial domain. In order to facilitate cross domain use, foxBMS II is based on a Texas Instruments TMS570 microcontroller, which is a well-established safety controller in automotive domain as well as in aerospace applications. Moreover, in foxBMS II only fully automotive qualified electronic components were selected. This is of relevance, as high component costs and certification efforts, especially in the aerospace sector are the driver for the increased use of commercial off-the-shelf (COTS) components, focusing on automotive qualified components. Although the acceptance of automotive reliability figures and qualification processes in the aerospace industry is currently not straightforward, the need for safe and affordable battery systems will eventually leverage this cross domain approach.

3 Artificial Intelligence for State Estimation

For the purpose of safe and economically viable usage of an energy storage system based on lithium-ion batteries, a precise state estimation is needed. Based on these state estimations foxBMS is able to optimize the control of battery systems and its cells. However, due to dynamically changing environment conditions, traditionally state estimation approaches like equivalent circuit models are frequently limited due to their poor stability. Data-driven methods, like NN, are able to overcome these problems through their high adaptability and self-learning ability. In addition, the computing power of embedded devices such as microcontrollers has increased dramatically over the last few years. This evolution, in parallel with the emerging AI technologies and popular frameworks for AI development, makes it possible to compile offline generated NN (i.e., on a dedicated High Performance Cluster (HPC)) for an embedded device such as the Battery Management System foxBMS. By using highly adaptable and real-time capable data-driven approaches like NN, foxBMS is able to accurately model the eminently non-linear behavior of lithium-ion batteries.

3.1 Data Generation and Preprocessing

In order to obtain a NN which is able to model the lithium-ion cell even for dynamic changing environment conditions, reliable training is essential. For this purpose, a big amount of data representing the diverse environment and operating conditions has to be collected. The measurement of the data was done by the foxBMS platform. The experiment setup consists of a foxBMS Master Unit and one Slave Unit per battery module. The hardware is integrated into a commercially available battery electric vehicle with an approved roadworthiness certification by TÜV Süd. Multiple foxBMS Slave

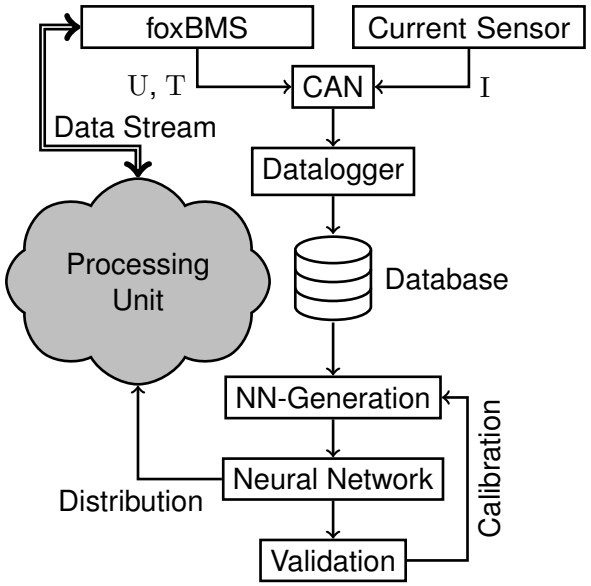

Fig. 1: foxBMS provides detailed battery usage data (e.g., U, T, I) to a database . An AI framework is used to train a NN on this data sets, like in this example the SOC. The trained NN is transferred to the PU and continuously fed with the current battery usage. The NN is able to accurately predict battery states and provide feedback to foxBMS to support e.g., extended lifetime through optimized operational strategies.

Units measure the cell voltages of each individual cell in the battery system and an additional high precision current sensor is integrated into the battery system to measure the battery current accurately. The precise measurement of the current is used to derive a reference SOC. Both, the voltages and the current, are measured with a frequency of 50 Hz. The measured data from foxBMS and the current sensor along with others such as vehicle speed, acceleration, and vehicle position are transmitted over CAN to a data logger integrated into the vehicle. The raw data is regularly uploaded to a server, converted to a readable format and subsequently shared with a HPC. With this setup, real-world driving data has been accumulated for more than one year. During this period, more than one terabyte of driving data has been logged. For a better insight into the data, clustering algorithms are used, which partition and divide the driving profiles into most relevant clusters. Based on the clustering, the most relevant information is extracted, and used for optimized training regarding computational efficiency and generalizability of the NN. The same procedure has already been applied by the team in [3].

3.2 Long Short-Term Memory

LSTM networks are a type of Recurrent Neural Network (RNN) [4]. RNNs generally and LSTMs in particular, are both specially designed for processing, detecting and memorizing patterns of sequence problems [5]. The core idea behind LSTMs is to have feedback loops inside the network architecture whereby results from previous computation steps are memorized instead of getting forgotten [6]. This mechanism allows the network to learn and store temporal contextual information. To train a RNN for long sequences, many computation steps are required. As a result, the memory of the first input gradually fades. With its sophisticated memory management through separate cell states which model short-term and long-term memory effects, the LSTM is able to overcome this drawback. In the following the architecture of the LSTM, provided by the machine learning framework TensorFlow, is explained. The general architecture of a LSTM cell is provided in Fig. 2, where x_t describes the input and y_t the output. The long-term state is taken into account by c_t while h_t describes the short-term state. The current input x_t and the state of the short-term memory h_{t-1} are passed to four gates (i_t, f_t, o_t, g_t, see Eqs. (1) to (4)) [6]. The input modulation gate g_t analyzes the current input x_t and the previous state h_{t-1}. The output of g_t is partially stored in the long-term memory. The other three gates are the gate controllers. The forget gate f_t controls which parts of the long-term memory should be no longer taken into account. The input gate i_t controls which parts of g_t should go into the long-term memory. Finally, the output gate o_t controls which parts of the information are considered for the short-term state h_t.

$$i_t = \sigma\left(W_{xi}^T \cdot x_t + W_{hi}^T \cdot h_{t-1} + b_i\right) \quad (1)$$

$$f_t = \sigma\left(W_{xf}^T \cdot x_t + W_{hf}^T \cdot h_{t-1} + b_f\right) \quad (2)$$

$$o_t = \sigma\left(W_{xo}^T \cdot x_t + W_{ho}^T \cdot h_{t-1} + b_o\right) \quad (3)$$

$$g_t = \tanh\left(W_{xg}^T \cdot x_t + W_{hg}^T \cdot h_{t-1} + b_g\right) \quad (4)$$

As the long-term state c_{t-1} traverses the network from left to right, it first passes the forget gate f_t which is responsible that irrelevant information is not considered anymore. Subsequently, new inputs are

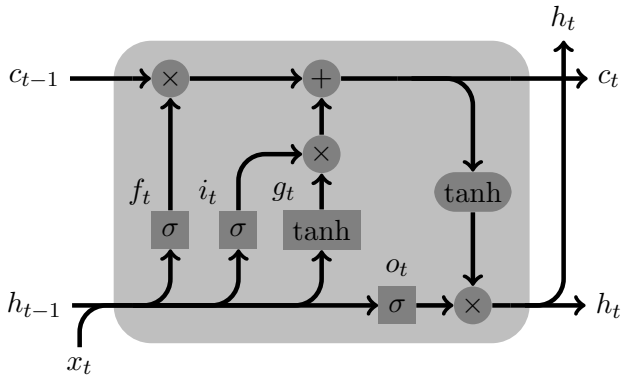

Fig. 2: Basic structure of a Long Short-Term Memory (LSTM) cell [7].

added via an addition operation. The new long-term state c_t is output without further transformations. In addition, the long-term state c_t is copied, applied to the $\tanh$-function and filtered by the output gate. The result is the state of the short-term memory h_t. c_t and h_t are described in Eqs. (5) and (6).

$$c_t = f_t \times c_{t-1} + i_t \times g_t \tag{5}$$

$$h_t = o_t \times \tanh(c_t) \tag{6}$$

The LSTM with the described architecture is able to describe the behavior of the lithium-ion battery over its entire life-time. The training of the LSTM NN is done offline on a HPC. As input data, the cell voltage, current, and temperature are used. The input is mapped to the corresponding SOC. In order to receive a precise prediction of the battery's SOC, an appropriate reference is needed. The reference SOC is computed using the Coulomb Counting (CC) approach. This is a straightforward method for determining the SOC by using current integration. The charge or discharge current is summed over time and then subtracted or added to the current SOC. However, the integrated value has to be initialized and the computed SOC shifts over time due to small integration errors. Therefore, time-consuming re-calibrations were done periodically. The deviation between the LSTM estimated SOC_t and the reference SOC_t^* is defined by the loss function L in Eq. (7).

$$L = \sqrt{\sum_{t=1}^{n} \frac{1}{n} (\mathrm{SOC}_t - \mathrm{SOC}_t^*)^2} \tag{7}$$

Furthermore, a hyper-parameter optimization approach is applied in order to determine the optimal architecture of the NN. As a result, the NN obtains very good generalizability, whereby a precise and stable SOC estimation is ensured for dynamically changing environment and working conditions.

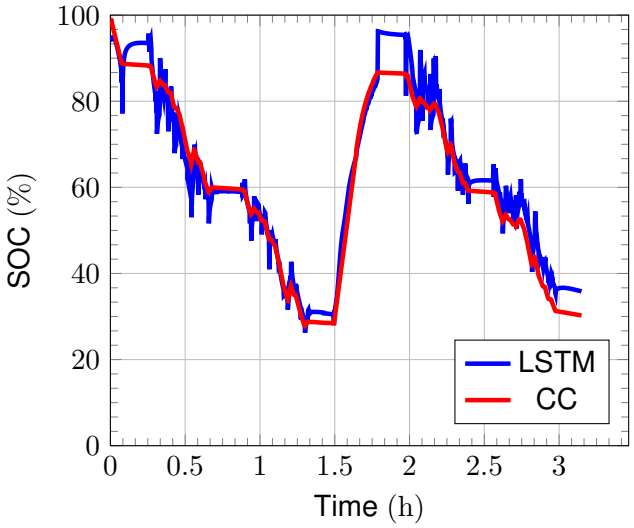

Fig. 3: Simulation of the SOC during a three hours drive with the trained NN compared to the CC reference. The cycle started with a fully charged vehicle and performed two highway drives with varying speeds and an intermediate fast charging stop.

3.3 NN Implementation and Results on Embedded System

The NN was implemented using TensorFlow [8]. The model was generated on a HPC and transferred on the embedded device. For demonstration purposes a Raspberry has been chosen. The trained NN is continuously fed with a data stream of battery usage data in real time as shown in Fig. 1. Figure 3 shows the simulation result of a three hour road test drive. It is possible to transfer battery usage data from a battery system from foxBMS to any PU that supports common communication interfaces such as CAN, SPI or Ethernet. These PUs are able to overcome computational limits that classic microcontrollers suffer from. With the additional computational power advanced algorithms are possible to find optimal operation strategies online and enable e.g., longer battery system lifetimes.

4 Conclusion

In the next steps, the workflow, described in this paper, and used for bringing AI-based algorithms on edge devices, has to be further developed. Besides that, additional research should focus on analyzing and optimizing the architecture of the LSTM in order to achieve a more accurate SOC model of the lithium-ion battery.
Based on the improvements from foxBMS, foxBMS II provides researchers and engineers with a well documented, configurable, already considering functional safety aspects, hardware platform including a full-featured software development framework. foxBMS II enables to effortlessly bring advanced algorithms, well-known physics based as the 2D Single Particle Model or innovative AI based ones, into the application stage at an early stage of the development process and thus enabling fast time-to-market cycles.

Acknowledgment

Part of the research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 769900 (DEMOBASE). Part of the research leading to these results has received funding from the ECSEL Joint Undertaking under grant agreement No. 826060 (AI4DI). This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation program and the ECSEL member states (i.e., BMBF for Germany).

References

[1] M. Giegerich, M. Akdere, C. Freund, T. Fuhner, J. Grosch, *et al.*, "Open, flexible and extensible battery management system for lithium-ion batteries in mobile and stationary applications," in *2016 IEEE 25th International Symposium on Industrial Electronics (ISIE)*, IEEE, Jun. 2016. DOI: 10.1109/isie.2016.7745026.

[2] M. Akdere, M. Giegerich, M. Wenger, R. Schwarz, S. Koffel, *et al.*, "Hardware and software framework for an open battery management system in safety-critical applications," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, IEEE, Oct. 2016. DOI: 10.1109/iecon.2016.7793001.

[3] S. Bockrath, A. Rosskopf, S. Koffel, S. Waldhor, K. Srivastava, and V. Lorentz, "State of Charge Estimation using Recurrent Neural Networks with Long Short-Term Memory for Lithium-Ion Batteries," in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, IEEE, Oct. 2019. DOI: 10.1109/iecon.2019.8926815.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning (Adaptive Computation and Machine Learning series)*. MIT Press, 2016, http://www.deeplearningbook.org.

[5] C. Loukas, F. Fioranelli, J. L. Kernec, and S. Yang, "Activity Classification Using Raw Range and I & Q Radar Data with Long Short Term Memory Layers," in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, IEEE, Aug. 2018. DOI: 10.1109/dasc/picom/datacom/cyberscitec.2018.00088.

[6] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[7] B. Moons, D. Bankman, and M. Verhelst, *Embedded Deep Learning - Algorithms, Architectures and Circuits for Always-on Neural Network Processing*. Berlin, Heidelberg: Springer, 2018.

[8] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, *et al.*, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, Software available from tensorflow.org, 2015.