# Comprehensive Comparative Analysis of Deep-Learning-Based State-of-Charge Estimation Algorithms for Cloud-Based Lithium-Ion Battery Management Systems

Dominic Karnehm, *Graduate Student Member, IEEE*, Akash Samanta, *Student Member, IEEE*, Latha Anekal, *Graduate Student Member, IEEE*, Sebastian Pohlmann, *Student Member, IEEE*, Antje Neve, and Sheldon Williamson, *Fellow, IEEE*

***Abstract*—A modern battery management system in electric vehicles plays a crucial role in enhancing battery pack safety, reliability, and performance, particularly in E-transportation applications. To achieve more accurate estimation methods, combining battery digital twinning with cloud computing for computational power and data storage capabilities proves beneficial. Over the last decade, various data-driven state-of-charge (SOC) estimation methods, such as machine learning and deep learning approaches, have been introduced to provide highly precise estimations. The widely used SOC estimation method in the industry is the extended Kalman filter (EKF). To explore and analyze the potential use of SOC estimation in a cloud platform, this article develops and conducts a comparative analysis of four SOC estimation methods: EKF, feedforward neural network, gated recurrent unit, and long short-term memory. These models are deployed in two cloud computing infrastructures, and their accuracy and computing time are thoroughly examined in this study. This study concludes that the EKF method is the fastest and most accurate among all considered methods. It boasts an average execution time of 54.8 ms and a mean absolute error of $2 \times 10^{-4}$ when measured over a physical distance of approximately 450 km via the mobile network long-term evolution.**

***Index Terms*—Artificial intelligence, battery management system (BMS), cloud computing, data-driven techniques, digital twining, electric vehicles (EVs), lithium-ion batteries, machine learning, state estimation.**

Manuscript received 8 August 2023; revised 17 November 2023 and 2 February 2024; accepted 23 February 2024. Date of publication 5 March 2024; date of current version 18 April 2024. This work was supported by dtec.bw—Digitalization and Technology Research Center of the Bundeswehr, which is funded by the European Union—NextGenerationEU. *(Corresponding author: Dominic Karnehm.)*

Dominic Karnehm, Sebastian Pohlmann, and Antje Neve are with the Electrical Engineering and Technical Informatics Department, University of Bundeswehr Munich, 85577 Neubiberg, Germany (e-mail: dominic.karnehm@unibw.de; sebastian.pohlmann@unibw.de; antje.neve@unibw.de).

Akash Samanta, Latha Anekal, and Sheldon Williamson are with the Electrical, Computer, and Software Engineering Department, Ontario Tech University, ON L1G 0C5, Canada (e-mail: akash.samanta@ontariotechu.net; latha.anekal@ontariotechu.net; sheldon.williamson@ontariotechu.net).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JESTIE.2024.3373267.

Digital Object Identifier 10.1109/JESTIE.2024.3373267

## I. Introduction

TOWARD decarbonization of societies, transportation electrification is one of the main steps [1]. To ensure optimal capacity utilization, safety, and longer battery life, a battery management system (BMS) plays a crucial role in electric vehicles (EVs). Over the last decade, the functionality of the BMS has raised from battery safety and protection to enable higher battery output and more secure battery systems [2]. Furthermore, an accurate and precious estimation of state-of-charge (SOC), state-of-health (SOH), and temperature is essential for the effective operation of the BMS. The state-of-the-art onboard BMS has limited computing resources and low local storage. Therefore, due to limited resources, simple models, such as equivalent circuit models (ECMs) and Kalman filter (KF), are widely used to estimate battery states, such as SOC, SOH, and temperature, in real-world applications. Here, an increasing interest in cloud computing and digital-twin-based BMS is noticed recently.

Besides the classical communication techniques, such as controller area network (CAN) bus and wireless technologies, such as long-term evolution (LTE), 5G, ZigBee, and Bluetooth, can be used in the BMS [3]. Wireless communication in the BMS is typically used in combination with Internet of Things technologies, such as cloud computing and Big Data, to enable cloud-based BMS. The path toward developing a fully functional cloud-based BMS includes multiple levels and development stages. Hossain Lipu et al. [4] mentioned computing and storage in the cloud as an enabler for precise data-driven estimation techniques. Contributing toward the cloud-based BMS, Karnehm et al. [5] proposed a framework to store, visualize, and analyze historical data and high-resolution real-time data of EVs on the cloud platform that provides flexibility and data interoperability. Tran et al. [2] described the concept of a hybrid architecture of a cloud-supported BMS where basic safety mechanisms are implemented onboard and additional and computational resource-intensive functionalities are implemented on a cloud platform. The hybrid approach circumvented the limitation of low data storage and the limited computing power of an onboard BMS.

Multiple cloud computing approaches for state estimations such as SOC, terminal voltage, and SOH have been proposed [6],

[7], [8]. Merkle et al. [9] proposed an architecture for a cloud-based digital twin deployed at Amazon Web Services (AWS) to estimate multiple states of a battery pack of a VW Golf-e, such as SOC, SOH, and the internal resistance of the cell. Furthermore, an architecture for the cloud-based BMS based on the cyber hierarchy and interactional network (CHAIN) framework is proposed in [10]. The authors illustrated the necessity of each component of the architecture and the location sensitivity of the components to execute distributed computing. The study also differentiated and discussed in detail end-computing, edge-computing, and cloud computing. Li et al. [11] implemented a cloud-based BMS for SOC and SOH estimation based on an extended Thevenin model with an adaptive extended H-infinity filter and particle swarm optimization. There, the authors have not discussed the effect of the usage of cloud technologies on accuracy and computing time. This missing inspection is closed with the presented work. Tran et al. [2] proposed a cloud-side model for highly accurate SOH estimation. The study mainly focused on the accuracy of the model. Evaluation of computational cost, details of technical architectural frameworks, and fault detection and prevention for the cloud-based BMS were overlooked.

Yassin et al. [12] saw a research gap in the connectivity speed and loss of distributed computing for digital twins of power systems. The primary contributions of this article are as follows.

1) Typically, automotive OEMs and battery pack manufacturers use models with nonlinear filters for SOC estimation. Currently, neural-networks-based SOC estimations are also popular. Therefore, in this article, a comprehensive comparative analysis is conducted to highlight the suitability of the aforementioned techniques for SOC in a cloud environment.
2) An experimental evaluation of the usability of SOC estimation in the cloud as a function of a cloud-based BMS is also presented.
3) The impact of data loss on prediction accuracy is also closely examined to emulate network interruption in real-world conditions.

## II. Methods

### A. SOC Estimation

As mentioned, different estimation methods are established for SOC estimation. This article compares different estimation methods in the case of a cloud computing implementation. To do so, an extended Kalman Filter (EKF), as an adaptive filter algorithm and classical approach, and three different neural networks are implemented and compared. The neural network architectures are feedforward neural network (FNN), long short-term memory (LSTM), and gated recurrent unit (GRU). Guo and Ma [13] conducted a comparison between the neural network architectures FNN, LSTM, and GRU, and temporal convolutional network. This work highlights that multiple algorithms for SOC estimation based on neural networks have been proposed but a lack of comparison between them is noticed. Therefore, Guo and Ma [13] compared the models regarding estimation accuracy and computational cost for on-board implementation.

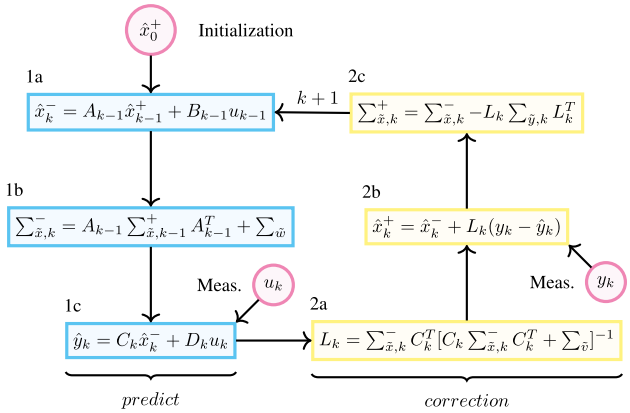


Fig. 1. Schematic layout of the considered EKF [18].

The GRU model showed the highest precision among the models considered based on the experimental evaluation. One of the primary concerns related to the usability of the cloud-based BMS is the computational cost. Therefore, in this study, the GRU is considered since it demonstrated the highest prediction accuracy in terms of root mean square error (RMSE) and mean absolute error (MAE). Furthermore, FNN and LSTM are also considered in this study as these models showed the lowest and highest computational cost.

*1) Extended Kalman Filter (EKF):* Due to the nonlinear characteristics of the lithium-ion battery cell, the KF is not best suited for accurate battery state estimation. Therefore, the EKF has been used frequently for SOC estimation [14], [15] instead of the basic KF. The EKF uses partial derivatives and resistor-capacitor (RC)-model expansions to linearize the battery model [16]. It relies on a set of observations of battery voltage $y_k$ and current $u_k$ for adaptive and accurate state estimation $\hat{x}_k^-$ [17]. Fig. 1 shows the detailed steps of the EKF [18], where $u_k$ and $y_k$ are the inputs for the prediction and correction operation stages of the filter, respectively. Hannan et al. [16] highlighted the EKF as a method with high prediction accuracy but limited robustness and linearization of errors, which could occur in a highly nonlinear system.

*2) Machine Learning:* This section gives an overview of the three deep learning algorithms considered here, namely, FNN, LSTM, and GRU, with mathematical descriptions and characteristics.

*a) Feedforward neural network (FNN):* In Fig. 2, the structure of an FNN is shown. It includes an input layer, multiple hidden layers, and an output layer [13]. Each layer is fully connected through the next layer, so the information flows from the input nodes through the hidden layers to the output layer. The relation between the neuron $i$ of the current and the neuron $j$ of the previous layer is defined as

$$o_j = \sigma\left(\sum_{i=1}^{N}(w_{j,i}x_i + b_{j,i})\right) \tag{1}$$

$$\sigma(x) = x^+ = \max(0,x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{2}$$

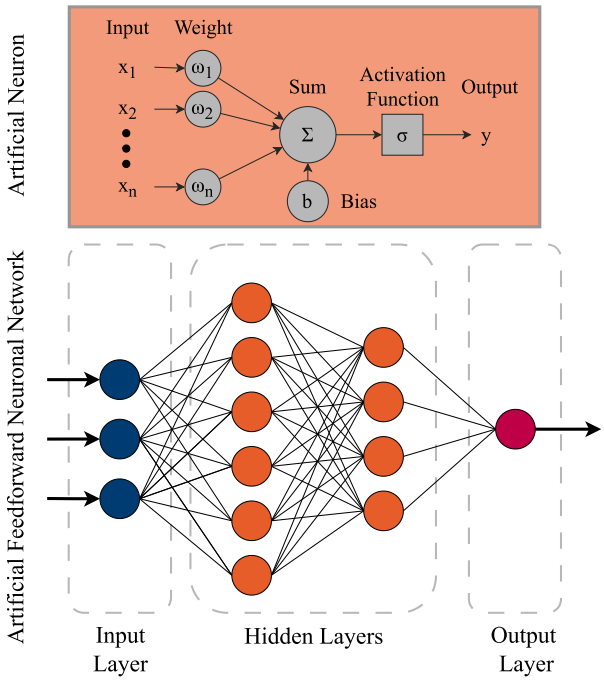


Fig. 2. FNN architecture.

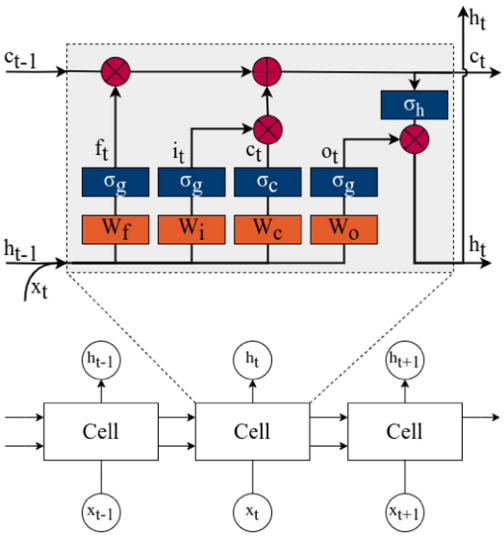


Fig. 3. LSTM architecture.

where $\sigma(x)$ describes the activation function rectified linear unit, $N$ the number of inputs, $w_{j,i}$ the weight of the input $i$ at neuron $j$, $x_i$ the input value, and $b_{j,i}$ the bias.

*b) Long short-term memory (LSTM):* Unlike the FNN, LSTM has a feedback connection, making it a type of recurrent neural network (RNN). It is ideal for processing and predicting time-series data. The cell memory unit is the key component of an LSTM [19]. Fig. 3 shows the structure of an LSTM model and a typical LSTM cell.

The LSTM cell utilizes an input gate $i$, forget gate $f$, and the output gate $o$. The forward pass at time $t$ of an LSTM cell is processed as follows [19]:

$$
\begin{aligned}
f_t &= \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \\
i_t &= \sigma_g\left(W_i x_t + U_i h_{t-1} + b_i\right) \\
o_t &= \sigma_g\left(W_o x_t + U_o h_{t-1} + b_o\right) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c\left(W_c x_t + U_c h_{t-1} + b_c\right) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned} \tag{3}
$$

where $x_t$ and $h_t$ are the cell input and output at time $t$, and $c_t$ denotes hidden cell memory. $W$ and $U$ are weight matrices, and $b$ bias vector learned during training. $i_t$, $f_t$, and $o_t$ represent the gates, while $\sigma_g$, $\sigma_c$, and $\sigma_h$ are the corresponding activation functions.

LSTM networks are especially good at detecting contextual anomalies by learning temporal relationships and capturing them in a compact state representation. They are particularly effective in modeling multivariate time-series and time-variant systems, accommodating both stationary and nonstationary dynamics, as well as short- and long-term dependencies [20].

*c) Gated recurrent unit (GRU):* Like LSTM, a GRU is a kind of RNN. Compared to LSTM, it also addresses long-term memory problems, is cheaper in the sense of computational cost, and has comparable accuracy [21]. The structure of a GRU cell is shown in Fig. 4.

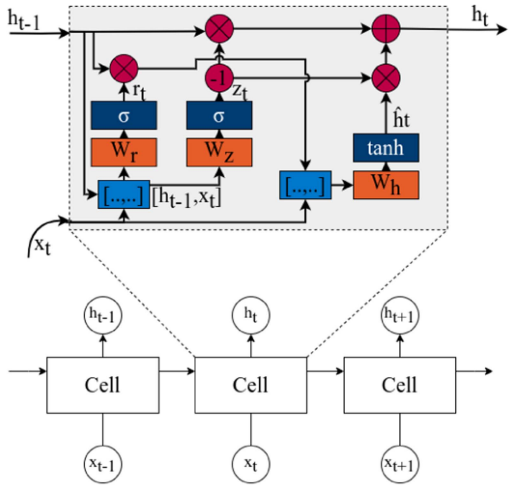


Fig. 4. GRU architecture.

The GRU cell utilizes an update gate $z$ and a reset gate $r$. The forward pass at time $t$ of a GRU cell is processed as follows [22]:

$$
\begin{aligned}
r_t &= \sigma\left(W_r\left[h_{t-1}, x_t\right] + b_r\right) \\
z_t &= \sigma\left(W_z\left[h_{h-1}, x_t\right] + b_z\right) \\
\hat{h}_t &= \tanh\left(W\left[r_t \circ h_{t-1}, x_t\right] + b_{\hat{h}}\right) \\
h_t &= z_t \circ \tilde{h}_t + (1 - z_t) \circ h_{t-1}
\end{aligned} \tag{4}
$$

where $x_t$ represents the input at time $t$, $\tilde{h}_t$ and $h_t$ are information vectors, representing the output, $W_r$, $W_z$, and $W$ the weight matrices of the gates and the output, and $b_r$, $b_z$, and $b_{\hat{h}}$ represent the different biases corresponding to the different weight matrices.

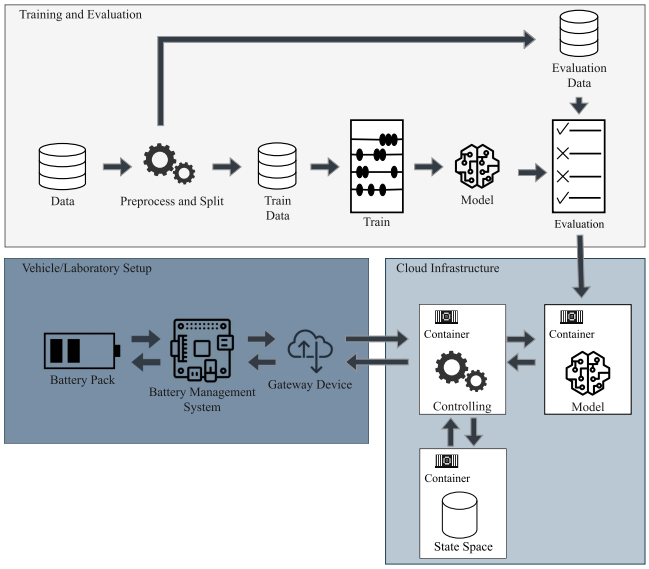


Fig. 5. Cloud computing architecture.

### B. Cloud Computing

To enable a cloud-computing-based SOC estimation, it is necessary to define the framework that needs to be implemented. The focus of the proposed framework is a cloud-ready software architecture based on open-source standards and software, to enable a cloud as well as an on-premise deployment. The framework is shown in Fig. 5. Such a focus enables the usage of modern technologies and decreases the risk of dependence on a single service provider. The framework mainly consists of three components: a training and evaluation component, the cloud computing component, and the vehicle/laboratory setup. Training and evaluation are two main stages in machine learning. These steps also take place to implement the EKF model. Therefore, during training, the battery parameters have to be determined. Furthermore, the performance of the implemented model also needs to be examined based on the evaluation data. Typically, the data are preprocessed and split into train and test data. Then, the train data are used to fit the machine learning model and the test data are used to evaluate the accuracy during the fit process. The trained models are deployed for usage on the cloud infrastructure. This contains a controlling component, machine learning models, and a state space. The components are implemented in a containerized way, so each of these components is implemented as a single Docker [23] container. An introduction to containerized software engineering for cloud computing using Docker is given by Bartlett [24]. To serve the machine learning models implemented in Keras [25], the Docker image of TensorFlow Serving [26] is used. For control, a self-developed solution is implemented. Controlling includes data preprocessing and parsing for machine learning models, the EKF implementation, and state handling for time dependence of machine learning models and EKF. The state space itself is implemented as a containerized Redis [27] database. Separation of single components from each other enables individual scaling, which is also a separate derivative of each component. For this study, the laboratory setup consists of a physical battery pack, a BMS to control the battery pack, and a gateway device for establishing a communication channel between the cloud infrastructure, and the classical BMS. Therefore, it converts the messages transferred over the bus systems, such as the CAN, to data formats such as JavaScript Object Notation. Furthermore, it also ensures security features such as encrypted data transmission or access management.

The two architectural components, training and evaluation, and production are independent of each other and can be deployed on different infrastructures. Thus, due to the chosen components, the architecture is independent of the service provider or hardware used, resulting in the production architecture being able to be deployed as an on-premise and a cloud computing solution. Here, in this study, the cloud computing service provider AWS in Frankfurt, Germany, and Montreal, QC, Canada, has been chosen as an example only.

TABLE I
ARCHITECTURE OF THE MODELS

| Layers | FNN | LSTM | GRU |
|---|---|---|---|
| Layer 1 | Flatten | LSTM(128) | GRU(128) |
| Layer 2 | Dense(512) | Dropout(0.2) | Dropout(0.2) |
| Layer 3 | Dense(128) | Dense(8) | Dense(8) |
| Layer 4 | Dense(64) | Dense(1) | Dense(1) |
| Layer 5 | Dense(32) | | |
| Layer 6 | Dense(1) | | |

## III. Description of Data and Model Training

### A. Dataset

The dataset used for the estimation of SOC is taken from Kollmeyer et al. [28]. The data include charge, discharge, and driving cycle measurements of current, voltage, capacity, and temperature with a measurement frequency of 10 Hz. The researchers used a new 5-A·h Turnigy cell (Turnigy Graphene 5-Ah 65-C cell) and described the experimental procedure in detail. The tests have been carried out at a wide range of temperatures starting from 40 °C, 25 °C, 10 °C, 0 °C, −10 °C to −20 °C. The temperature-depending $2RC$ ECM parameters $R_0$, $R_1$, $C_1$, $R_2$, and $C_2$ of this dataset have been given by Khanum et al. [29]. In this study, data from the urban dynamometer driving schedule (UDDS) drive cycle at 0 °C are used for experimental validation of the research concept. The remaining data are split randomly into train and test datain a 70:30 ratio for training and testing purposes of the considered neural network models.

### B. Model Implementation and Data Preprocessing

The main focus of this work is to evaluate and compare methods to estimate SOC in a cloud environment. Therefore, neural network models already proposed for SOC estimation are used. Guo and Ma [13] proposed multiple models. Among them, FNN, LSTM, and GRU models are implemented in this study as the models with the lowest and highest computational cost and the lowest error with their considered dataset. The architecture of these models is shown in Table I. For implementation, the open-source deep learning library Keras 2.12.0 and the programming

TABLE II
MACHINE LEARNING HYPERPARAMETERS

| Parameter | Value |
|---|---|
| Batch Size | 64 |
| Epochs | 150 |
| Learning Rate | $10^{-4}$ |

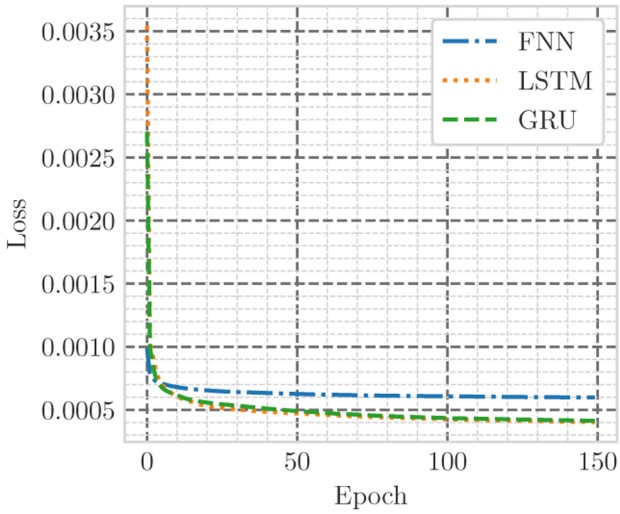


Fig. 6. Training loss of FNN, LSTM, and GRU models.

language Python 3.8 are used. Furthermore, the implementation of the EKF has been done in Python.

As data are preprocessed for the neural networks, feature scaling takes place. Each feature, voltage ($u$), current ($i$), and temperature ($T$), is scaled by a Min–Max scaler, defined as follows.

$$x_{\text{scaled}}(k) = \frac{x(k) - \min(x)}{\max(x) - \min(x)} \tag{5}$$

where $x$ is the set of elements of a single feature, and $x(k)$ is an element of this set. Such scaling is necessary to mitigate the impact of scale discrepancies among the different features. The hyperparameters, batch size, number of epochs, and learning rate, of the model training are listed in Table II. Each training used the same hyperparameters. The training loss of each epoch during training of the models FNN, LSTM, and GRU can be seen in Fig. 6.

## IV. COMPARATIVE ANALYSIS AND DISCUSSION

To discuss the usage of SOC estimation in the cloud, the following different scenarios are considered:

1) distance by request time;
2) network depending on request time;
3) model depending on accuracy;
4) model accuracy depending on the network stability.

The three artificial intelligence models are trained with the dataset described in Section III-A.

TABLE III
MAE AND RMSE OF FNN, LSTM, GRU, AND EKF OF 0.75 H (DISCHARGE OF SOC 100% − 91.81%) UDDS DRIVING CYCLE AT 0 °C AND A MEASUREMENT FREQUENCY OF 10 HZ

| *Model* | *MAE* | *RMSE* |
|---|---|---|
| EKF | 0.0002 | 0.0002 |
| FNN | 0.0067 | 0.01 |
| LSTM | 0.0744 | 0.0763 |
| GRU | 0.0455 | 0.0493 |

### A. Accuracy of the Models

To evaluate the precision of the different methods, the models are evaluated on the basis of the first 10% measurement data of the UDDS driving cycle at 0 °C [28]. These data include a discharge of SOC 100%−91.81% in 2 760 s. Table III shows the MAE and RMSE of this validation.

From Table III, it can be seen that the highest accuracy is demonstrated by the EKF, with MAE and RMSE of 0.0002. With an MAE of 0.0067, the FNN model showed the best results compared to the two other neural network models in this experiment. This result deviates from the results of Guo and Ma [13]. Significantly worse results were observed for the FNN there. Therefore, a possible reason is the difference in the dataset used. The high precision of the EKF compared to the neural network models is primarily due to the small amount of training data used for neural networks compared to the highly accurate battery parameters used by the EKF. However, if the model parameters are not adaptive to aging and other variabilities in real practice, the accuracy of the EKF will tend to degrade over time. Moreover, the primary focus of this work is not to propose highly accurate neural networks for SOC but rather the possible implementation of different SOC estimation methods in the cloud-based BMS. Therefore, the results of this experiment can mainly be seen as a baseline for discussion in terms of computational cost, response time, and the impact of network connection losses on the performance of the could-based BMS.

### B. Location Dependence on Execution Time

In this scenario, the basic feasibility of a cloud BMS in terms of computing time and the need to establish a global infrastructure in the case of global fleet services must be analyzed. For these experiments, two elastic cloud computing (EC2) instances have been set up at two different locations. EC2 is a virtual machine service from the cloud service provider AWS. The chosen instance type is t2.2xlarge and Amazon Linux is the operating system of the instances. The instances are deployed in two geographic regions: *Canada (Central)*, located in Montreal, QC, Canada, and *Europa (Frankfurt)*, located in Frankfurt, Germany, and the execution request is sent from Ontario Tech University, Oshawa, ON, Canada. The direct distance between the different locations can be seen in Table IV. To reduce the likelihood of network problems, a wired Internet connection is used for this experiment.

Fig. 7 shows the request time in milliseconds of the four different implementations by the location of the servers. As seen in Fig. 7(a), the average request time from Oshawa to Montreal is

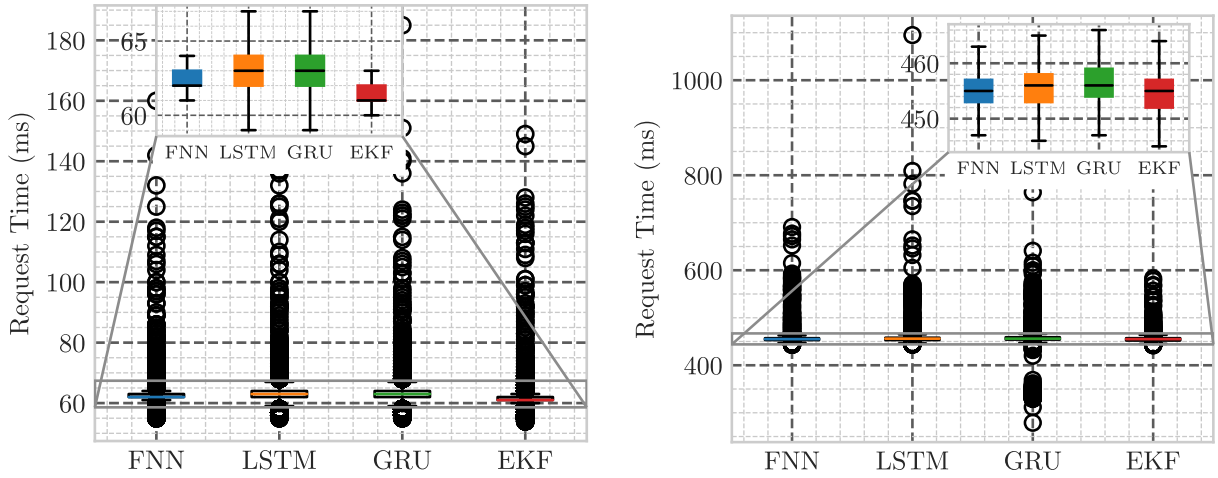


Fig. 7. Request time depending on the model between (a) Oshawa (ON) and Montreal (QC), and (b) Oshawa (ON) and Frankfurt (DE).

TABLE IV
APPROXIMATE DISTANCE BETWEEN THE CLOUD DEPLOYMENT LOCATIONS AND THE CLIENT, LOCATED IN OSHAWA

| *Client location* | *Cloud location* | *Direct distance* |
|---|---|---|
| Oshawa, ON | Montreal, QC | ≈450 km |
| Oshawa, ON | Frankfurt, DE | ≈6300 km |

around 62.5 ms. The mean request time to Frankfurt, Germany, is approximately 455 ms, as can be seen from Fig. 7(b). Taking into account that all models located in Montreal easily reach a measurement time of 100 ms, which means that a measurement frequency of 10 Hz can be achieved, excluding a few out risers as shown in the box plot. To the best of our knowledge, a standard for SOC estimation frequency is not yet set. Wei et al. [30] highlighted 1–10 Hz as a typical sampling frequency according to current practice. Furthermore, Rosewater [31] has mentioned these ranges as a time window of estimation. In addition, they have also demonstrated that the faster the measurement, the better the accuracy. Thus, for this study, a measurement frequency of 10 Hz is defined as the baseline to compare execution time.

Under the condition of a computing time below 100 ms, none of the solutions deployed in Frankfurt can be utilized. If cloud-based BMS solutions need to be used for a global fleet operation, the choice of different locations of cloud servers and data orchestration of cloud infrastructure needs to be considered carefully. Because of the usage of the same instance type, the rise in computational time can be explained by the distance, the resulting latency, and package loss. LSTM and GRU are showing a higher variability, located in Montreal, compared to the variability of FNN and EKF. This could be explained by the higher complexity of the models, the necessary state handling, and the resulting computational cost.

### C. Network Dependence on Request Time

In practice, LTE networks are widely used for mobile devices. Thus, to evaluate the possible use of the cloud-based BMS in real applications, measurements have also been taken with LTE as a network connection. The experiments are conducted with the EKF and FNN, the methods with the highest accuracy, as seen in Table III. During the experiment, the device under test (DUT) was located at the campus of Ontario Tech University and was static. The average request time for the EKF is 66.1 ms ± 29.4 ms and for the FNN is 54.8 ms ± 10.3 ms as obtained from the experiments. A better request time with LTE in the case of the FNN is noticed compared to the wired request time. The higher request time for the EKF can be explained by the measurements in Fig. 8. At 632.5 s, a network interruption of 1.1 s can be seen, as highlighted in the figure. After the interruption, a lower mean request time of 53.5 ms is observed. Possible reasons could be an issue on the network provider side, network reconfiguration, or a switch on the radio cell. Still, both measurement series showed an acceptable result, and the maximum time could be 0.1 s. For the FNN, and after disconnecting also for the EKF, a better request time can be seen compared to the wired connection. This could be explained by bad network configurations of the wired network. In summary, it can be inferred from this series of experiments that cloud-based SOC estimation is possible in real-world scenarios but a careful consideration of the execution time needs to be taken.

### D. Accuracy Depending on Loss Rate

The impact of connectivity losses needs to be considered while analyzing the feasibility of the cloud-based BMS implementation. Based on the usage of mobile networks and the fact that the complete network coverage cannot be ensured, the impact of the connection loss rate on the prediction accuracy over time is assessed. For this only, the FNN and EKF models are chosen, as these demonstrated the highest accuracy level among all the considered models in Table III. A synthetic loss of packages with a specific loss rate is randomly distributed over all sent packages to emulate the network connection loss rate.

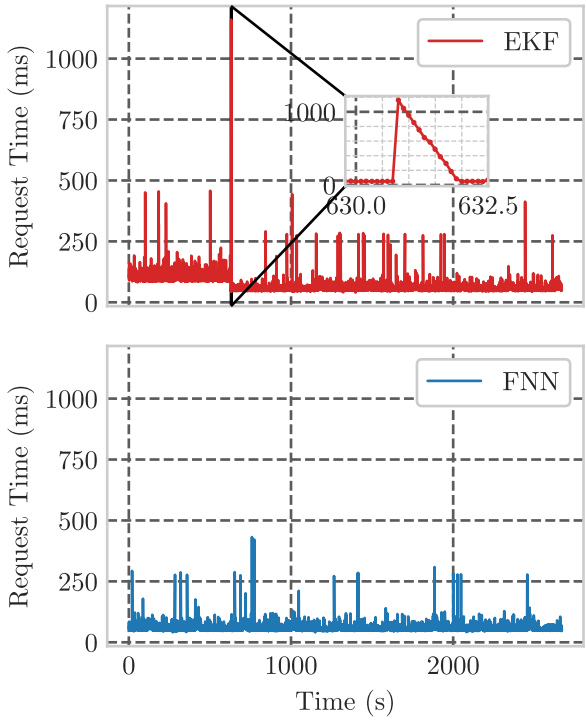


Fig. 8. Request time measurement using LTE as network.

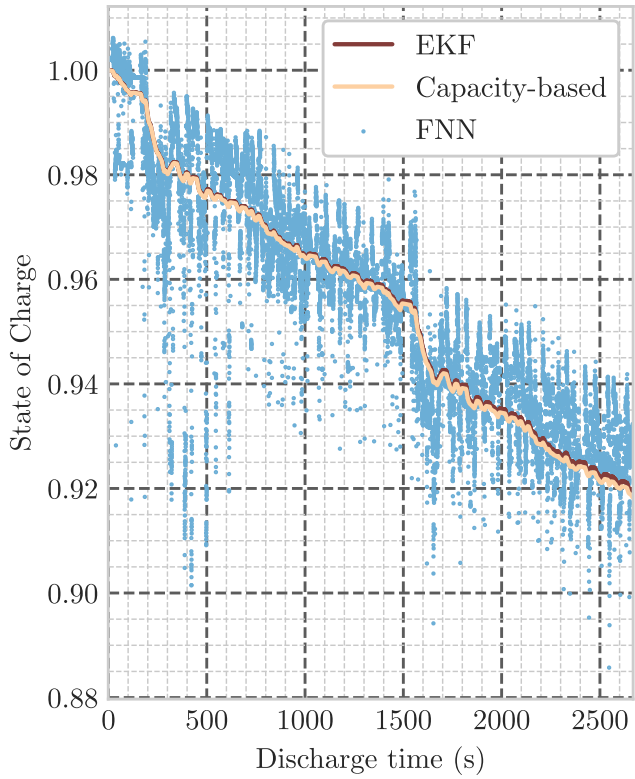


Fig. 9. SOC of the EKF and FNN compared to the capacity-based SOC.

TABLE V
MAE OF ESTIMATED SOC BY FNN, AND EKF COMPARED TO THE CAPACITY-BASED SOC AT DIFFERENT LOSS RATES

| Loss Rate | MAE FNN | MAE EKF |
|---|---|---|
| 0 % | 0.0062 | 0.0002 |
| 1 % | 0.0062 | 0.0002 |
| 2.5 % | 0.0062 | 0.0002 |
| 5 % | 0.0062 | 0.0004 |
| 7.5 % | 0.0062 | 0.0005 |
| 10 % | 0.0062 | 0.0005 |

In Fig. 9, the estimated SOC over time with a loss rate of 10% for the FNN, EKF, and the capacity-based SOC is shown. Here, the capacity-based SOC estimation is considered the baseline for comparison purpose. The EKF depends on state observations. Cipral and Romera [32] proposed that the missing observations due to connection loss can be replaced by zero values to handle incomplete data. In Table V, the MAE depending on the connection loss rate in % is shown. An increase in the MAE of EKF is noticed with the rise in connection loss. On the other hand, no change in the error of the FNN is noticed. Worth noting in all cases, the mean accuracy of the EKF is significantly higher compared to the FNN model. It can be concluded that for cloud-based SOC estimation, a fusion of the EKF and a data-driven method, such as FNN, GRU, or LSTM could be used where the EKF can be used to calibrate the model parameters of the neural networks to improve and maintain the estimation accuracy.

## V. CONCLUSION

In this article, a comparative analysis among EKF as classical approach and the neural network models FNN, LSTM, and GRU as data-driven methods for understanding the possible implementation of cloud-based SOC estimation is presented. These methods have been implemented and deployed on a cloud-based platform towards the practical application of cloud- and digital-twin-based BMS for e-mobility applications. For this purpose, the proposed framework has been deployed using AWS as a cloud service provider. To evaluate the effect of the geographical location of the cloud server, all the models are deployed and validated in two different geographical regions namely, Frankfurt, Germany, and Montreal, Canada.

Some of the major findings include the following.

1) A minimal average computing time of 54.8 ms is noticed in the FNN with LTE mobile network connection, as shown in Fig. 8.
2) No significant differences in computational cost is observed among the considered methods EKF, FNN, LSTM, and GRU, as shown in Fig. 7. Thus, for the considered estimation methods, the difference in local time, as Guo and Ma [13] discussed, is negligible.
3) The main impact on computing time is given by the distance between the DTU, and the cloud infrastructure, as can be seen from Fig. 7. These results indicate the clear necessity of using global cloud infrastructure in the case of using cloud-based BMS fleet operation and management.
4) The highest accuracy among the considered methods is noticed in the case of the EKF with an mean absolute error (MAE) of 0.0002 as seen in Table III.

To the best of our knowledge, there is no existing standard to define a required estimation time for SOC. According to Wei et al. [30], the typical sampling frequency for state estimation

purposes is 1–10 Hz, as reported in the literature for battery estimation. This frequency was also mentioned by Rosewater [31]. Due to this, the computational time of 66.1 ms ± 29.4 ms for the EKF and 54.8 ms ± 10.3 ms for the FNN under LTE as network connection is considered reasonably brief. However, further research and development are required to establish a more robust method to handle network issues in the cloud-based SOC estimation method for practical application.

The EKF demonstrated the best precision among the methods considered. How et al. [33] mentioned that the performance of KFs is highly dependent on the battery model parameters, noise level, physical parameters, and initial conditions. Neural networks are effective under dynamic operating conditions including drive cycles, battery aging, and operating temperature, however, the performance is constrained by the training data. The usage of cloud computing infrastructure can increase the performance of neural networks, because of the possibility of centralized data collection, which can be used for training of these models. Also, a fleet-wide comparison of aging and update of estimation methods can be greatly simplified. In future work, the proposed method of state estimation needs to be tested on pack level to evaluate the real-world usage of state estimation in the cloud.

In addition, a fusion of the EKF and a data-driven method, such as FNN, GRU, or LSTM, could be used, where the EKF can be used to calibrate the model parameters of the neural networks to improve and maintain the estimation accuracy over time in cloud-based SOC estimation.

## REFERENCES


[1] J. Buberger, A. Kersten, M. Kuder, R. Eckerle, T. Weyh, and T. Thiringer, "Total CO2-equivalent life-cycle emissions from commercially available passenger cars," *Renewable Sustain. Energy Rev.*, vol. 159, 2022, Art. no. 112158.

[2] M.-K. Tran, S. Panchal, T. D. Khang, K. Panchal, R. Fraser, and M. Fowler, "Concept review of a cloud-based smart battery management system for lithium-ion batteries: Feasibility, logistics, and functionality," *Batteries*, vol. 8, no. 2, p. 19, 2022, Art. no. 19.

[3] M. S. H. Lipu et al., "Battery management, key technologies, methods, issues, and future trends of electric vehicles: A pathway toward achieving sustainable development goals," *Batteries*, vol. 8, no. 9, p. 119, 2022, Art. no. 119.

[4] M. Hossain Lipu et al., "Data-driven state of charge estimation of lithium-ion batteries: Algorithms, implementation factors, limitations and future trends," *J. Cleaner Prod.*, vol. 277, 2020, Art. no. 124110.

[5] D. Karnehm, S. Pohlmann, A. Wiedenmann, M. Kuder, and A. Neve, "Introduction of a cloud computing architecture for the condition monitoring of a reconfigurable battery system for electric vehicles," in *Proc. 6th Conf. Cloud Internet Things*, 2023, pp. 29–33.

[6] D. Shi et al., "Cloud-based deep learning for co-estimation of battery state of charge and state of health," *Energies*, vol. 16, no. 9, 2023, Art. no. 3855.

[7] R. Di Rienzo et al., "Cloud-based optimization of a battery model parameter identification algorithm for battery state-of-health estimation in electric vehicles," *Batteries*, vol. 9, no. 10, p. 486, 2023, Art. no. 486.

[8] S. Li, H. He, Z. Wei, and P. Zhao, "Edge computing for vehicle battery management: Cloud-based online state estimation," *J. Energy Storage*, vol. 55, 2022, Art. no. 105502.

[9] L. Merkle, M. Pöthig, and F. Schmid, "Estimate e-golf battery state using diagnostic data and a digital twin," *Batteries*, vol. 7, no. 1, p. 15, 2021, Art. no. 15.

[10] S. Yang et al., "Implementation for a cloud battery management system based on the chain framework," *Energy AI*, vol. 5, 2021, Art. no. 100088.

[11] W. Li, M. Rentemeister, J. Badeda, D. Jöst, D. Schulte, and D. U. Sauer, "Digital twin for battery systems: Cloud battery management system with online state-of-charge and state-of-health estimation," *J. Energy Storage*, vol. 30, 2020, Art. no. 101557.

[12] M. A. Yassin, A. Shrestha, and S. Rabie, "Digital twin in power system research and development: Principle, scope, and challenges," *Energy Rev.*, 2023, Art. no. 100039.

[13] S. Guo and L. Ma, "A comparative study of different deep learning algorithms for lithium-ion batteries on state-of-charge estimation," *Energy*, vol. 263, 2023, Art. no. 125872.

[14] G. Pérez, M. Garmendia, J. F. Reynaud, J. Crego, and U. Viscarret, "Enhanced closed loop state of charge estimator for lithium-ion batteries based on extended Kalman filter," *Appl. Energy*, vol. 155, pp. 834–845, 2015.

[15] Z. Chen, Y. Fu, and C. C. Mi, "State of charge estimation of lithium-ion batteries in electric drive vehicles using extended Kalman filtering," *IEEE Trans. Veh. Technol.*, vol. 62, no. 3, pp. 1020–1030, Mar. 2013.

[16] M. Hannan, M. Lipu, A. Hussain, and A. Mohamed, "A review of lithium-ion battery state of charge estimation and management system in electric vehicle applications: Challenges and recommendations," *Renewable Sustain. Energy Rev.*, vol. 78, pp. 834–854, 2017.

[17] A. Wadi, M. Abdel-Hafez, H. A. Hashim, and A. A. Hussein, "An invariant method for electric vehicle battery state-of-charge estimation under dynamic drive cycles," *IEEE Access*, vol. 11, pp. 8663–8673, 2023.

[18] G. L. Plett, *Battery Management Systems, Volume II: Equivalent-Circuit Methods*. Norwood, MA, USA: Artech House, 2015.

[19] F. Yang, S. Zhang, W. Li, and Q. Miao, "State-of-charge estimation of lithium-ion batteries using LSTM and UKF," *Energy*, vol. 201, 2020, Art. no. 117664.

[20] B. Lindemann, B. Maschler, N. Sahlab, and M. Weyrich, "A survey on anomaly detection for technical systems using LSTM networks," *Comput. Ind.*, vol. 131, 2021, Art. no. 103498.

[21] Z. Zhang et al., "An improved bidirectional gated recurrent unit method for accurate state-of-charge estimation," *IEEE Access*, vol. 9, pp. 11252–11263, 2021.

[22] M. Jiao, D. Wang, and J. Qiu, "A GRU-RNN based momentum optimized algorithm for SOC estimation," *J. Power Sources*, vol. 459, 2020, Art. no. 228051.

[23] *Docker: Accelerated Container Application Development*. Accessed: Aug. 8, 2023. [Online]. Available: https://www.docker.com/

[24] J. Bartlett, *Cloud Native Applications With Docker and Kubernetes: Design and Build Cloud Architecture and Applications With Microservices, EMQ, and Multi-Site Configurations*. Berkeley, CA, USA: Apress, 2023.

[25] *Keras: Deep Learning for Humans*. Accessed: Aug. 8, 2023. [Online]. Available: https://keras.io/

[26] *Serving Models TFX*. Accessed: Aug. 8, 2023. [Online]. Available: https://www.tensorflow.org/tfx/guide/serving

[27] *Redis*. Accessed: Aug. 8, 2023. [Online]. Available: https://redis.io/

[28] P. Kollmeyer and M. Skells, "Turnigy graphene 5000mah 65c Li-ion battery data," *Mendeley Data*, vol. 1, pp. 10–17632, 2020.

[29] F. Khanum, E. Louback, F. Duperly, C. Jenkins, P. J. Kollmeyer, and A. Emadi, "A Kalman filter based battery state of charge estimation Matlab function," in *Proc. IEEE Transp. Electrific. Conf. Expo.*, 2021, pp. 484–489.

[30] Z. Wei, C. Zou, F. Leng, B. H. Soong, and K.-J. Tseng, "Online model identification and state-of-charge estimate for lithium-ion battery with a recursive total least squares-based observer," *IEEE Trans. Ind. Electron.*, vol. 65, no. 2, pp. 1336–1346, Feb. 2018.

[31] D. Rosewater, "Battery management system standard—IEEE p2686 recommended practice for battery management systems in energy storage applications," *Energy Storage Syst. Saf. Rel. Forum*. Accessed: Jun. 19, 2023. [Online]. Available: https://www.osti.gov/biblio/2002547

[32] T. Cipra and R. Romera, "Kalman filter with outliers and missing observations," *TEST: An Official J. Spanish Soc. Statist. Operations Res.*, vol. 6, no. 2, pp. 379–395, 1997.

[33] D. N. How, M. Hannan, M. H. Lipu, and P. J. Ker, "State of charge estimation for lithium-ion batteries using model-based and data-driven methods: A review," *IEEE Access*, vol. 7, pp. 136116–136136, 2019.