Journal of Energy Storage 25 (2019) 100817

Contents lists available at ScienceDirect

Journal of Energy Storage

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

A deep learning method for online capacity estimation of lithium-ion batteries

Sheng Shen[a], Mohammadkazem Sadoughi[a], Xiangyi Chen[b], Mingyi Hong[b], Chao Hu[a,c,*]

[a] Department of Mechanical Engineering, Iowa State University, Ames, IA 50011, USA
[b] Department of Electrical and Computer Engineering, University of Minnesota Twin Cities, Minneapolis, MN 55455, USA
[c] Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011, USA

ARTICLE INFO

Keywords:
Capacity estimation
Health monitoring
Deep learning
Lithium-ion batteries

ABSTRACT

The past two decades have seen an increasing usage of lithium-ion (Li-ion) rechargeable batteries in diverse applications including consumer electronics, power backup, and grid-scale energy storage. To guarantee safe and reliable operation of a Li-ion battery pack, battery management systems (BMSs) should possess the capability to monitor, in real time, the state of health (SOH) of the individual cells in the pack. This paper presents a deep learning method which utilizes deep convolutional neural network (DCNN) for cell-level capacity estimation based on the voltage, current, and charge capacity measurements during a partial charge cycle. The unique features of DCNN include the local connectivity and shared weights, which enable the model to accurately estimate battery capacity using the measurements during charge. To the best of our knowledge, this is one of the first attempts to apply deep learning to the online capacity estimation of Li-ion batteries. Ten-year daily cycling data from eight implantable Li-ion cells and half-year cycling data from 20 18650 Li-ion cells were utilized to verify the performance of the proposed deep learning method. Compared with traditional machine learning methods such as shallow neural networks and relevance vector machine (RVM), the proposed deep learning method is demonstrated to produce higher accuracy and robustness in the online estimation of Li-ion battery capacity.

1. Introduction

Over the course of past 20 years, lithium-ion (Li-ion) rechargeable batteries have been increasingly applied in consumer electronics, such as mobile and laptops, in power backup applications, such as emergency lighting units and burglar alarms, and in transportation applications, such as electric vehicles and hybrid-electric vehicles. Safety and reliability of the Li-ion batteries in these applications have attracted much attention from both industry and academia. Online state of health (SOH) estimation by battery management systems (BMSs) is of great importance for ensuring the safety and reliability of individual cells operating in the field [1]. The cell capacity and resistance are two parameters that are often used to quantify the SOH of a Li-ion battery cell [2]. When the capacity is utilized to indicate the SOH, the SOH is often defined as the ratio of the cell capacity at the current charge/discharge cycle to its initial capacity or a rated capacity provided by the cell manufacturer. An accurate capacity estimation enables a battery cell to be substituted right before the cell capacity fades to an end of life limit, thus allowing useful life of the cell to be fully utilized while not compromising cell safety.

Recent studies have reported many interesting approaches for online estimating the capacity of a Li-ion battery cell. Generally, these approaches can be divided into two categories: (1) model-based methods considering empirical models [3–10] and electrochemical models [11,12] and (2) data-driven methods using kernel regression approaches [19–21] and neural network (NN) approaches [22–26].

In empirical/electrochemical model-based methods, an empirical/electrochemical model is often used, in conjunction with experimental data, to accomplish the task of online capacity estimation. Plett [3] attempted the combined use of extended Kalman filter (EKF) and an empirical model to online estimate the capacity of a Li-ion cell. The empirical model is used to formulate the state transition and measurement functions in a discrete-time state-space model, where cell state of charge (SOC) is treated as the state and capacity and resistance are treated as the parameters. The use of dual EKF enables the online estimation of both the state and parameters. EKF may show instability for highly nonlinear state-space models due to the linearization of these models via the first-order Taylor series expansion [4]. To overcome this

* Corresponding author at: Department of Mechanical Engineering, Iowa State University, Ames, IA 50011, USA.
E-mail addresses: chaohu@iastate.edu, huchaostu@gmail.com (C. Hu).

https://doi.org/10.1016/j.est.2019.100817
Received 24 March 2019; Received in revised form 19 June 2019; Accepted 30 June 2019
Available online 08 August 2019

theoretical limitation of EKF, Plett later employed an improved variant of Kalman filter, named sigma-point Kalman filter (SPKF), to enable online capacity estimations. Based on a series of sigma points (a minimal set of deterministically-chosen, weighted sample points), SPKF approximates the probability distribution of capacity by utilizing the unscented transformation that allows the sigma points to be scaled to an arbitrary dimension [5]. The use of joint/dual SPKF enables simultaneous estimation of SOC, capacity and resistance. These two earlier studies by Plett stimulated the later development of a number of empirical model-based methods that use EKF [6,7], SPKF [8], H-infinity filter [9], and particle filter [10] to estimate the SOC and capacity or the capacity alone. The estimation accuracy of an empirical model-based method is, however, highly dependent on the fidelity of the underlying empirical model. Additionally, empirical models lack physical significance and the capability to provide physical insight into a battery cell. An alternative model-based approach to the online capacity estimation is to use a physics-based electrochemical model. Moura et al. [11] developed an adaptive observer algorithm to identify, based on voltage and current measurements, the physical parameters of an electrochemical model that are directly related to the capacity of a Li-ion cell. Bartlett et al. [12] estimated the capacity fade of a composite electrode battery by applying a reduced-order electrochemical model in three Bayesian filtering algorithms, namely the EKF, fixed-interval Kalman smoother, and particle filter. Electrochemical model-based methods focus mostly on tracking the health-relevant parameters of a full- or reduced-order electrochemical model. It is often difficult to accurately track these physically meaning parameters based only on voltage and current measurements and with a low computational effort [2,13].

However, constructing an accurate battery model is not an easy task since it often requires a large amount of physical knowledge or experimental data under carefully designed, well-controlled conditions, which is not often available or too expensive to acquire. As an alternative to model-based methods, data-driven methods for the capacity estimation have been gaining popularity in recent years, because 1) these methods are exclusively dependent on experimental data and often do not require extensive domain knowledge about battery working principles and 2) the rapid adoption of onboard sensing, communication, and computing technologies has enabled the collection of sensor data from a large number (e.g., hundreds of millions to billions) of battery cells operating in the field. The publicly available datasets collected from battery cycling tests mainly include: NASA Prognostics Data Repository [14], Oxford battery degradation dataset from the Howey research group [15], datasets from the Centre for Advanced Life Cycle Engineering (CALCE) battery research group [16], and research & development data repository from Sandia National Labs [17]. Most recently, a new battery test dataset consisting of 124 cells has been publicly available [18].

In data-driven methods, online capacity estimations can be carried out by learning the complicated dependency of cell characteristic features extracted from the current and voltage measurements on the capacity of a cell. This dependency can be modeled using different machine learning methods which can be categorized into the kernel regression approaches [19–21] and the neural network (NN) approaches [22–26].

The kernel regression approaches model the non-linear relationship between the measurable features and the cell capacity by way of kernel functions. Several techniques based on kernel regression approaches such as relevance vector machine (RVM) [19] and k-nearest neighbor (kNN) regression [20,21] have been developed to estimate the capacity of Li-ion batteries. Hu et al. [19] manually selected five cell characteristic features that are indicative of the capacity from partial charge curves and then leveraged RVM to approximate the mapping from the charge-related features of a cell to its capacity. The proposed RVM model, after being trained offline, was applied to online estimate the capacity of a battery cell based on the five selected charge-related features. Unlike SVM and RVM, kNN regression is a lazy learning algorithm since it does not involve explicit training steps [20]. Hu et al. [21] developed a data-driven method for online estimation of battery capacity based on charge voltage and current curves. In this approach, the kNN regression technique was adapted to learn the relationship between the capacity and charge-related characteristic features, and then particle swarm optimization was implemented to optimize the feature weights of the kNN regression model with a high accuracy.

The NN approaches build a network structure consisting of a number of interconnected "neurons" that maps the input features (e.g., cell terminal voltage, current and temperature) to the output (i.e., cell capacity). A framework based on a NN was built for the online capacity estimation of electric vehicle batteries, using the measurement data such as current, voltage, and temperature while leveraging their historical distributions [22]. Eddahech et al. [23] employed a recurrent NN to estimate the two SOH-related parameters, the capacity and equivalent series resistance, of a high-power-density Li-ion cell based on the temperature, current, SOC variations, and historical cell behavior. Lin et al. [24] selected three features, namely the constant current charging time, the voltage drop, and the open circuit voltage, as the inputs to a probabilistic NN for the capacity estimation. Bai et al. [25] developed a data-driven method that utilized an artificial NN to estimate the terminal voltage, and the artificial NN was then integrated with a dual EKF for the online capacity estimation. Combined with a cycle life model, a new Radial Basis Function NN model was proposed to eliminate the battery degradation's effect on the capacity estimation accuracy [26]. This approach first builds the cycle life model to predict the practical capacity based on the aging cycle tests and then designs the Radial Basis Function NN model for the capacity estimation of Li-ion batteries based on the terminal voltage, current, and practical capacity outputted from the cycle life model.

Although these traditional machine learning approaches are able to produce satisfactory accuracy in the capacity estimation, they have some inevitable limitations. First, traditional machine learning methods lack the capability to analyze and derive full value from large volumes of data. Another limitation is that the performance of the traditional machine learning methods highly depends on how the undergoing trend of the data could be represented by the extracted characteristic features. In other words, the poorly extracted characteristic features may limit the performance of the traditional machine learning methods [27]. However, it is very hard to identify appropriate characteristic features that carry the most useful information for the capacity estimation [28].

To alleviate the aforementioned issue, this study presents a deep convolutional neural network (DCNN) model based on large volumes of charge data for achieving state-of-the-art accuracy in online estimation of Li-ion battery capacity. The main advantages of the proposed method are summarized as follows:

1 By using a deep learning model, this study automates the feature learning process from the large amounts of charge data. This avoids the manual feature extraction that relied heavily on human labor and the risk of dropping useful information in the charge data.
2 With the aid of deep network architecture (a stack of multi-level layers), the proposed deep learning method can learn highly representative features that carry the most useful information of charge data. It can be obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw charge data) into the representation at a higher, slightly more abstract level [28]. With the composition of enough such transformations, highly representative features can be learned.
3 Compared with traditional machine learning methods, the proposed deep learning method improves the generalization capability in the battery capacity estimation. This capability is to produce reasonable outputs for new inputs that have never been seen during the training process.

The proposed deep learning method involves three main steps, summarized as follows:

1 First, the measured voltage, current, and charge capacity curves of a cell during a partial charge cycle are discretized into n segments (corresponding to n equal time intervals), respectively. The discretized values of voltage, current, and charge capacity are denoted as the inputs to proposed deep learning method. The corresponding output is the discharge capacity that is calculated using the coulomb counting method, which integrates the discharge current over time for the entire discharge cycle.
2 Second, DCNN is used to learn the relationship between the capacity of the cell and its charge-related input variables. A DCNN regression model, after being trained offline, is used to infer the unknown capacity of a cell online from a sample of voltage and current measurements during a partial charge cycle.
3 Third, the effect of the number of convolutional layers and the input size on the accuracy of the DCNN are experimentally investigated. We also quantify the performance benefits of deeper neural networks relative to shallower neural networks by varying the number of convolutional layers and observing the changes in the accuracy of capacity estimation.

To the best of our knowledge, the present study is one of the first to investigate the use of a deep learning method to infer the battery capacity from charge data. We intend to make this DCNN available publicly. This work is built upon our earlier study presented in a conference paper [29].

The remainder of this paper is divided into five sections. Section 2 introduces the principal concept of deep learning and presents the architecture of the proposed DCNN model. Section 3 describes the use of DCNN to construct a predictive model that approximates the non-linear input-output mapping from large-scale datasets. Ten-year daily cycling data from eight implantable Li-ion cells and half-year cycling data from 20 18650 Li-ion cells are utilized to verify the effectiveness of the DCNN model, respectively. Section 4 exhibits and explains the verification results. Some concluding remarks and future directions of research are summarized in Section 5.

2. Methodology

2.1. Input and output structures

The objective of this study is to estimate the capacity of a Li-ion cell based on measured cell voltage, current, and charge capacity curves during a partial charge cycle. It is worth mentioning that the voltage and current can be directly measured from the cell, whereas the charge capacity is required to be calculated using the coulomb counting method, which integrates the charge current with respect to time for a partial charge cycle [30]. To better showing the time-series inputs and output, the voltage, current, and charge capacity curves of a battery cell for one partial charge cycle are plotted in Fig. 1, respectively.

Each curve is discretized into 25 segments and the discretized values of the voltage, current, and charge capacity are referred to as input variables to the model. As such, the input to the model is a matrix with fixed size 25 × 3, of which the first, second, and third columns are associated to the discretized values of the voltage, current, and charge capacity, respectively [29]. The matrix is defined as:

$$\text{Input} = \begin{bmatrix} \hat{V}_1 & \hat{I}_1 & \hat{C}_1 \\ \vdots & \vdots & \vdots \\ \hat{V}_i & \hat{I}_i & \hat{C}_i \\ \vdots & \vdots & \vdots \\ \hat{V}_{25} & \hat{I}_{25} & \hat{C}_{25} \end{bmatrix}_{25\times 3} \quad (1)$$

where $\hat{V}_i$, $\hat{I}_i$, and $\hat{C}_i$ respectively denote the discretized values of the voltage, current, and charge capacity for the ith segment measured at the ith time interval (Δt_i) of a partial charge cycle. Each partial charge cycle has a corresponding discharge capacity that serves as the true output of the DCNN model. The discharge capacity is calculated using the coulomb counting method, which integrates the discharge current over time for the entire full discharge cycle that immediately follows the partial charge cycle.

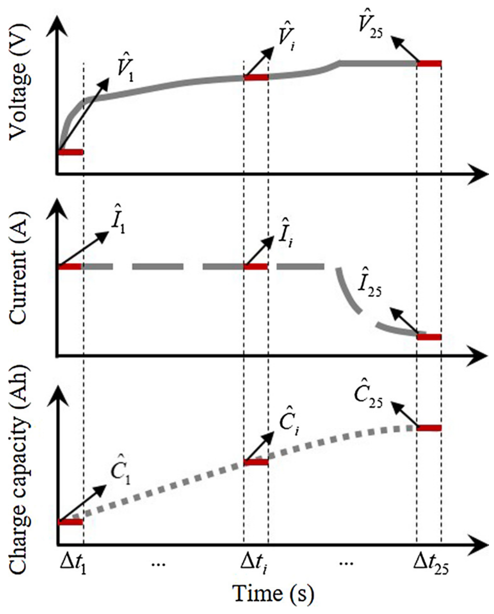


Fig. 1. Voltage, current, and charge capacity curves of a battery cell for one partial charge cycle.

2.2. Review of traditional machine learning and deep learning

Recent progress in deep learning has led to impressive successes in a wide range of practical applications, including computer vision (e.g. facial recognition) [31], natural language processing (e.g. photo tagging) [32], and information retrieval (e.g. search engines) [33]. In these applications, the deep learning methods produced results comparable and in some cases even superior to human experts [34,35]. Although deep learning is considered as a branch of a broader family of machine learning, it has a number of distinctive features that make it different from the traditional machine learning methods.

Fig. 2 shows the main difference between traditional machine learning and deep learning in handling the problem of online capacity estimation. In a traditional machine learning method, characteristic features that are indicative of the cell capacity (e.g., the sample entropy and statistics, initial charge voltage, final charge current, and charge capacity) are manually identified and extracted from the voltage and current curves and then fed as input into a machine learning model. The most commonly used machine learning methods for the capacity estimation are nonlinear least-squares regression [36], Gaussian process regression [37,38], and RVM [19]. In contrast, in deep learning methods, the complete set of raw data during the cell charge process (i.e., the current and voltage measurements from the beginning to the end of the charge process) is employed as the input without the extraction and selection of characteristic features.

2.3. Overall architecture of DCNN

This research will investigate the use of a deep architecture, DCNN, to online assess the capacity of a battery cell based on the voltage, current, and charge capacity measurements during a partial charge cycle. The unique features of DCNN include the sparse interactions (also

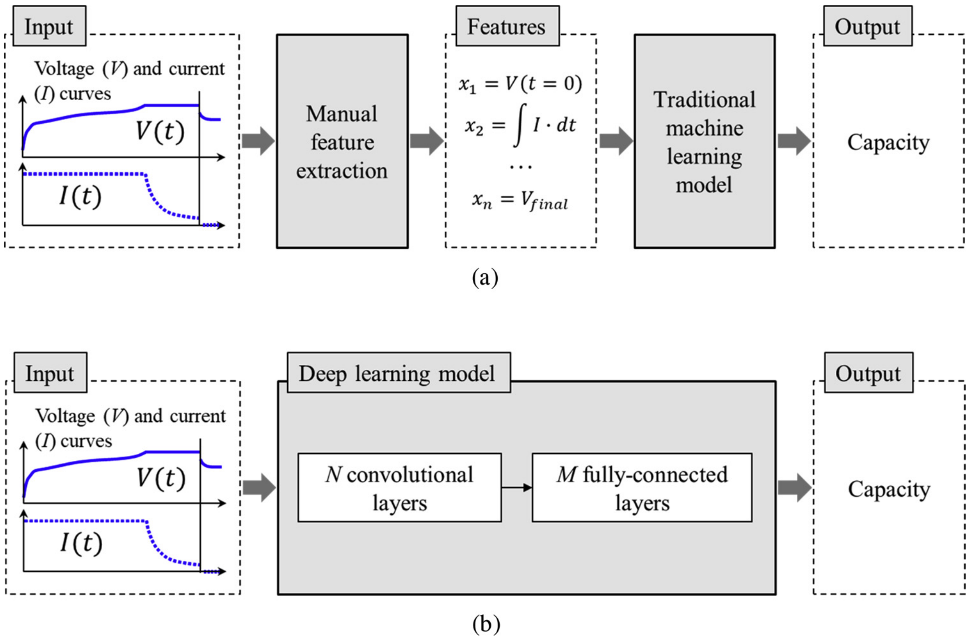


Fig. 2. The principal differences between traditional machine learning and deep learning in capacity estimation: (a) traditional machine learning method; and (b) deep learning method.

referred to as sparse connectivity or sparse weights), parameter sharing, and equivariant representations, which enable the predictive model to automatically extract high-level characteristic features from large volumes of charge data and make full use of these features to provide accurate capacity estimations of Li-ion batteries. The structure of the implemented DCNN in this study is shown in Fig. 3(a). The DCNN mainly consists of two types of layers: convolutional layers (Conv.) and fully-connected layers (FC.). Convolutional layers are used to execute a special kind of linear operation named convolution. In the context of deep learning, convolution is an operation on the inputs and kernels. Specifically, each unit of a convolutional layer is connected to local patches in the feature maps of the previous layer through a set of weights called filter banks. The result of this locally weighted sum is then passed through a variety of layers, such as a rectified linear units (ReLU) [39] and batch normalization (BN) [40], to form the feature maps of the next layer. Fully-connected layers perform matrix multiplication with separate weight matrices describing the pair-wise interactions between all the input and output units. In other words, the value of the ith unit in the 2nd fully connected layer (i.e., n_i^2) is determined by multiplying all the units of the previous layer (i.e., $n_1^1, n_2^1, \ldots, n_{40}^1$) with a weight matrix.

The overall architecture of the proposed DCNN model consists of eight building components, as shown in Fig. 3(b). The first five building components are convolution stages and the remaining three are fully-connected stages. Each convolution stage is composed of a convolutional layer, a BN, and a ReLU except the first convolution stage, which has an additional max-pooling layer applied to the output of the ReLU of this stage. The BN is a widely used technique in deep learning for reducing the distribution of each layer's inputs changes during training. The ReLU is applied to introduce non-linear properties into the neural networks. The remaining three fully-connected stages are followed by the five convolution stages. The output of the last fully-connected stage is received by a regression layer, which outputs a capacity estimation target.

The constructed DCNN model obtain over 28,000 parameters and about 6000 neurons, and consist of five convolutional layers and three fully-connected layers. Table 1 shows the number of neurons and parameters for each layer in DCNN.

2.4. *Training algorithm*

A DCNN model contains a set of unknown parameters (e.g., weights and bias of convolutional and fully-connected layers) that need to be identified during the training process. To properly identify these parameters, a cost function J was defined to measure differences (or generalization errors) between the model predictions and the associated ground truth. The expected generalization error given by the cost function was minimized by a widely used optimization method, stochastic gradient descent (SGD) with momentum [41]. The SGD with momentum updated the parameters, $\boldsymbol{\theta}$ (weights, $\boldsymbol{\omega}$, and biases, $\mathbf{b}$), to minimize the generalization error by taking small steps in the direction of the negative gradient of the cost function. This process was repeated many times, with each iteration being executed on a small set of training samples, until the generalization error was close to zero. The cost function $J(\boldsymbol{\theta})$ with the regularization term was defined as:

$$J_R(\theta) = J(\theta) + \lambda\Omega(\omega) = \frac{1}{2m}\sum_{i=1}^{m} (h_\theta(\mathbf{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2}\omega^T\omega \tag{2}$$

where $J_R(\theta)$ denotes the cost function, $\lambda\Omega(\omega)$ denotes the regularization term, λ denotes the L_2 regularization factor that weighed the relative contribution of the norm penalty term, $\Omega(\omega)$, $h_\theta(\mathbf{x})$ denotes the hypothesis function, m denotes the number of samples used in each iteration, $\mathbf{x}^{(i)}$ denotes the ith input sample matrix, and $y^{(i)}$ denotes the corresponding target value.

The hypothesis function $h_\theta(\mathbf{x})$ could be expressed as:

$$h_\theta(\mathbf{x}) = b_0x_0 + \omega_1x_1 + \ldots + \omega_nx_n \tag{3}$$

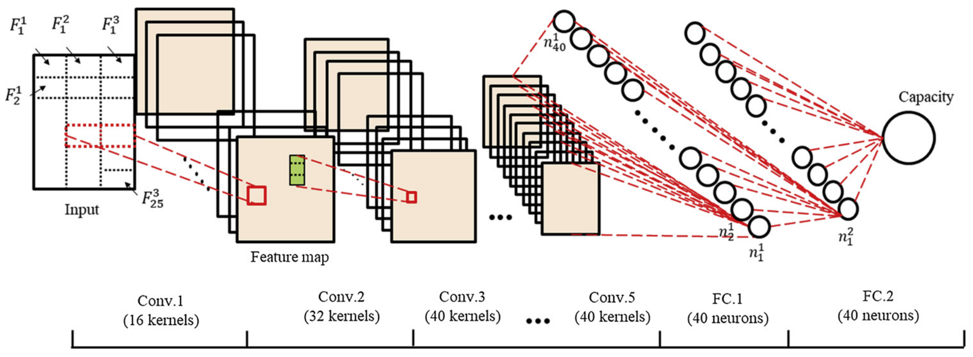
where ω_n and x_n denote the nth unknown parameter and its corresponding input variable, respectively. Note that b_0 was the bias and $x_0 = 1$.

The parameter θ could be iteratively updated in accordance with:

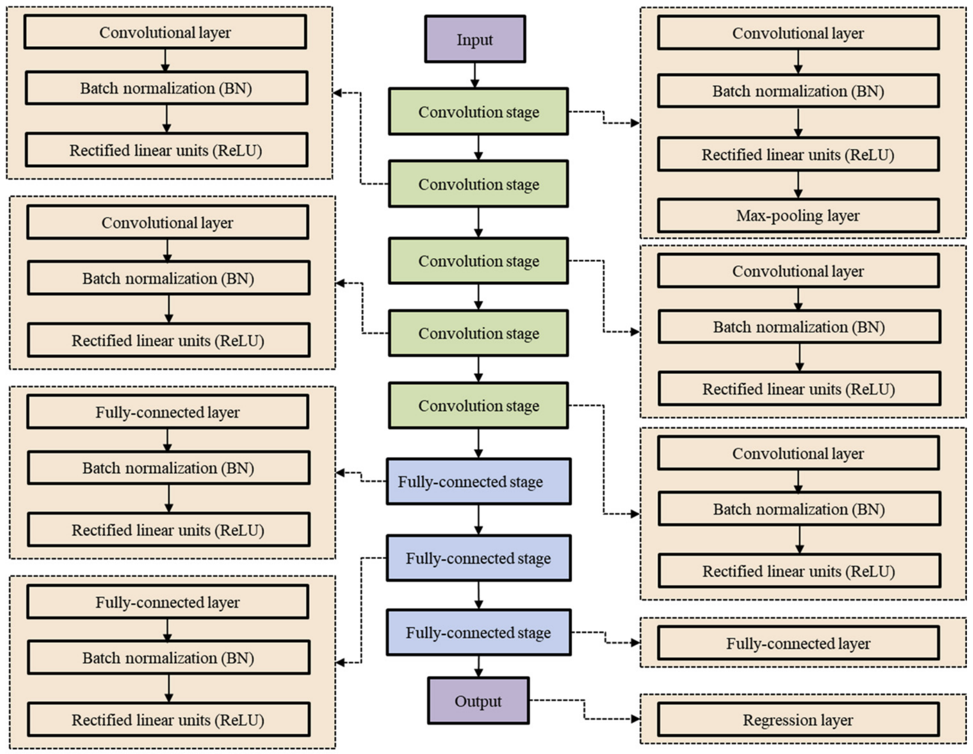
$$\hat{g} = \frac{1}{2q}\sum_{i=1}^{q} (h_\theta(\mathbf{x}_j^{(i)}) - y_j^{(i)})^2 \tag{4}$$

$$\theta_{j+1} = \theta_j - \alpha\hat{g} + \gamma(\theta_j - \theta_{j-1}) - \lambda\alpha\theta_j \tag{5}$$

where $\hat{g}$ denotes the estimator of the exact gradient that samples a mini-batch of q samples, $\mathbf{x}_j^{(i)}$ denotes the ith input matrix of the mini-batch in the jth iteration, and $y_j^{(i)}$ denotes the corresponding target value, α



(a)



(b)

Fig. 3. Overview of the implemented DCNN model in this study: (a) model structure; (b) architecture of the DCNN model.

denotes the step size (or initial learning rate), γ denotes the momentum that determined the contribution of gradients from the previous iteration to the current iteration, and θ_j denotes the parameter estimate in the j^{th} iteration.

3. Experimental data and implementation of deep learning

The performance of the DCNN was determined by two sets of experimental data. The first dataset was from a 10-year cycling test (i.e., repeated full charge/discharge cycles) on eight Li-ion prismatic cells used in implantable applications. The second dataset was from a half-year cycling test on 20 18650 Li-ion cells. This section first presents the procedure of these two cycling tests and the capacity fade performance of the cells during each of these tests, then introduces the dataset construction for the capacity estimation, and finally discusses the DCNN implementation detail in these two experimental verifications.

3.1. Test procedure and cycling data

For the 10-year cycling test, 16 Li-ion cells were constructed in

Table 1
The numbers of neurons and parameters for each layer in DCNN.

Layer/Item	Conv.1	Conv.2	Conv.3	Conv.4	Conv.5	FC.1	FC.2	FC.3
Number of parameters	48	128	160	160	160	25,640	1,640	41
Number of neurons	1,600	1,280	1280	960	640	40	40	1

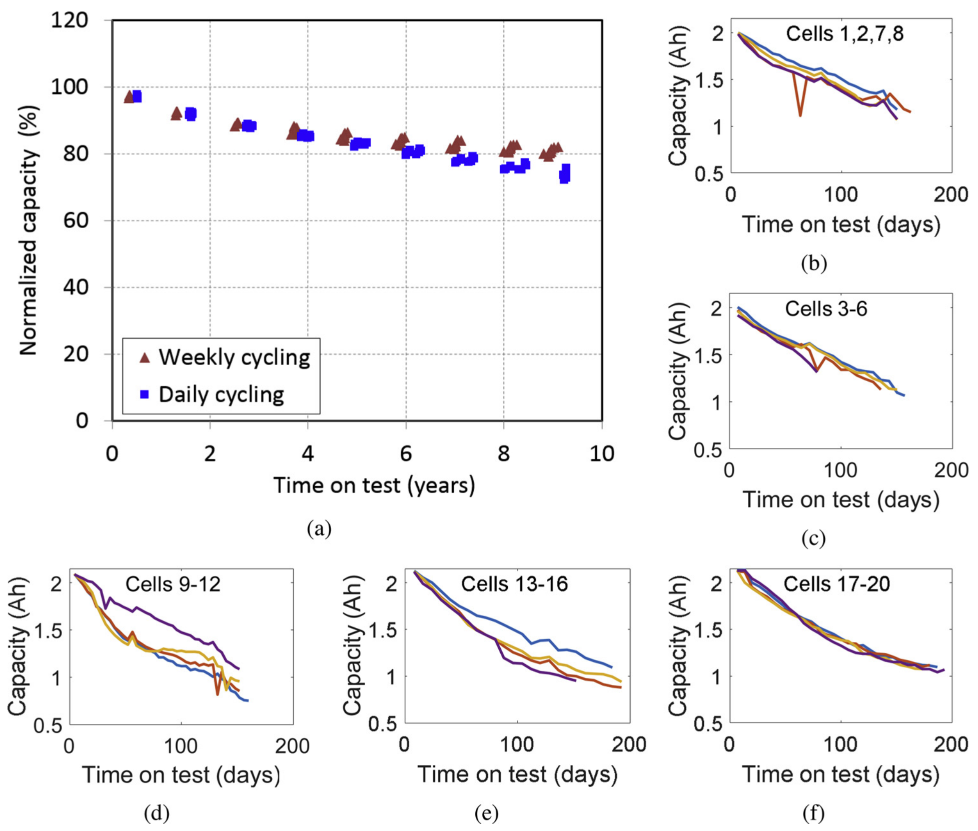


Fig. 4. Experimental data. (a) Normalized capacity vs. time on the 10-year cycling test for cells used in this research. For confidentiality reasons, the discharge capacity of a cell in this figure and in the later discussion is presented after being normalized using the initial discharge capacity of the cell. (b-f) Capacity vs. time on the half-year cycling test for five groups of cells.

hermetically sealed prismatic cases and subjected to the full depth of charge/discharge cycling under 37 °C in a lab [19,30,42]. Among these 16 cells, eight were constructed in 2002 and then cycled using a nominally weekly discharge rate [19], and the other eight constructed in 2006 and cycled using a nominally daily discharge rate [42]. The discharge capacities of these cells against the time on test are plotted in Fig. 4(a). This study employed the daily cycling data to verify the effectiveness of the proposed deep learning method in accurately estimating the capacity of a battery cell. These eight daily cells were charged with the following parameter settings during the 10-year cycling test: (i) the charge rate of the constant current (CC) charge step was C/4; (ii) the constant current charge cutoff voltage was set to 4.075 V; and (iii) the time duration of the constant voltage (CV) charge step was 60 min. The daily cycling data consisted of 2.6 million data points (i.e., the current and voltagemeasurements for both the charge and discharge processes) collected over 3000 charge/discharge cycles.

The experimental data of the half-year cycling test was collected by NASA. This experimental data can be accessed from the Prognostics Center of Excellence (PCoE) at Ames Research Center [14]. In this study, we used the first five groups of cells from the NASA battery usage dataset. The five groups consist of a total of 20 cells, with each group being composed of four cells. These cells were cycled at room temperature throughout the duration of the test with five different cycling protocols as described in Table 2. A time-varying charge/discharge profile was conducted on the cells in each protocol group.

Compared with the 10-year daily cycling dataset, for which only a single protocol and charge/discharch profile were used, the half-year cycling dataset presented a greater challenge for the capacity estimation since it included cells cycled with five different cycling protocols and a time-varying charge/discharge profile within each protocol group. To make this challenge more apparent, the capacity fade for the cells in all five groups are plotted against the time on test as illustrated in Fig. 4(b) - Fig. 4(f). It can be observed from Fig. 4 that the cells in each protocol group exhibited a less homogeneous capacity fade behavior compared with the implantable cells used in 10-year daily cycling dataset. Even for the cells in the same protocol group, a totally different capacity fade behavior throughout the cycling test could also be observed, especially for cells 9–12 (see Fig. 4(d)) and cells 13–16 (see Fig. 4(e)). Hense, one of our goals was to investigate whether the proposed DCNN model could produce accurate capacity estimates for the cells cycled with various cycling protocols and profiles. The results of this investigation are presented in Subsection 4.2.

In this study, both the 10-year daily cycling test and the half-year cycling test followed a constant current-constant voltage (CC-CV) charge and CC discharge protocol. In a CC-CV charge cycle, the charge process is divided into two sequential steps, the CC charge step and the CV charge step. Consequently, the total charge capacity is composed of the CC charge capacity and CV charge capacity. In the CC charge step, a constant current is applied to charge a cell until its voltage rises to a preset maximum charge voltage, V_{max}. Subsequently, the charge process immediately transits from the CC charge step to the CV charge step. In the CV charge step, the charging voltage is held constant at V_{max} for a fixed duration and the charge current continuously declines. Upon the completion of the charge cycle, the cell is subjected to a CC discharge cycle, where a nominally daily discharge rate is used to discharge the cell to a minimum discharge voltage (V_{min}) [19].

3.2. Dataset construction for capacity estimation

It is worth mentioning that in practice a Li-ion battery cell often experiences an incomplete discharge process before the next charging cycle. A user often intends to charge the cell before it has been completely depleted. In an attempt to simulate this real life case where the cell starts to be charged at the partially discharged state and ends up being fully charged, a partial charge curve was generated by truncating the full charge curve below a pre-assigned initial charge voltage ($V_{initial}$) [19]. The basic idea about generating the partial charge curve is illustrated in Fig. 5. The value of $V_{initial}$ was randomly drawn from a uniform distribution between a lower bound V_{low} and an upper bound V_{high}. To investigate the effect of initial charge voltage on the capacity

Table 2
Summary of the five cycling protocols applied to NASA cells [14]. The five cycling protocols were used to investigate the capacity fade under more practical battery usage conditions.

Test routine	CC charge step		CV charge step		Discharge step	
Item	Charge rate	Charge cutoff voltage (V)	Charge voltage (V)	Charge termination condition	Discharge rate	Discharge cutoff voltage (V)
Group 1 (Cells 1, 2, 7, and 8)	2A with a randomly selected duration (i.e., 0.5, 1, 1.5, 2, or 2.5 h)	4.2	4.2	The current drops below a threshold or the charging time exceeds the allotted interval	A value was randomly selected between 0.5A and 4A every 5 min	3.2
Group 2 (Cells 3–6)	2A	4.2	4.2	The current drops below a threshold or the charging time exceeds the allotted interval	A value was randomly selected between 0.5A and 4A every 5 min	3.2
Group 3 (Cells 9-12)	A value was randomly selected from a set (i.e., 0.75A, 1.5A, 2.25A, 3A, 3.75A, and 4.5A) every 5 minutes	4.2	N/A	N/A	A value was randomly selected from a set (i.e., 0.75A, 1.5A, 2.25A, 3A, 3.75A, and 4.5A) every 5 min	3.2
Group 4 (Cells 13–16)	2A	4.2	4.2	The current drops below a threshold or the charging time exceeds the allotted interval	A value was randomly selected between 0.5A and 4A every 5 min	3.2
Group 5 (Cells 17–20)	2A	4.2	4.2	The current drops below a threshold or the charging time exceeds the allotted interval	A value was randomly selected between 0.5A and 4A every 5 min	3.2

estimation, we considered two settings with different voltage ranges that are associated with a low initial SOC and a high initial SOC, respectively. It should be mentioned that the low initial SOC setting refers to that a cell undergoes a deeper discharge, as compared to the high initial SOC setting. Table 3 shows the two initial SOC settings and the associated voltage ranges.

As shown in Fig. 5, three partial charge curves, highlighted by the dashed lines of different colors (orange, green, and purple) and with numbers ①, ②, and ③, were generated from the same full charge curve based on three randomly sampled initial voltage values (i.e., V_1, V_2, and V_3). The charge capacities corresponding to these three partial charge curves are denoted as $\tilde{y}_1$, $\tilde{y}_2$, and $\tilde{y}_3$. It is important to note that the charge capacity $\tilde{y}_i$ ($\tilde{y}_i = C_{CC_i} + C_{CV_i}$, $i = 1,\ 2,\ and\, 3$) is comprised of two parts: the CC charge capacity (C_{CC_i}) and the CV charge capacity (C_{CV_i}). Also note that the CV charge capacities, $C_{CV_1},\ C_{CV_2},\ and\, C_{CV_3}$, are the same for the three partial charge curves because all the truncated curves contain the full CV charge step. On the other hand, the CC charge capacity (C_{CC_i}) may differ from one partial charge curve to another due to the distinct truncated areas (see S_1, S_2, and S_3 in Fig. 5) resulting from different initial voltages. Lastly, the full charge capacity corresponding to a partial charge curves, y_i, is the sum of the partial charge capacity $\tilde{y}_i$, and the charge capacity corresponding to the truncated area, C_{S_i}, i.e., $y_i = C_{S_i} + C_{CC_i} + C_{CV_i}$, $i = 1,\ 2,\ and\, 3$. Since each partial charge curve is derived from the same full charge curve, these full charge capacities are equal to one another, $y_1 = y_2 = y_3$.

In addition to the two initial SOC settings, the effect of current measurement error was also investigated. This investigation took two scenarios into account and were given follow: Scenario I — No bias in current measurement and Scenario II — 2% positive bias in current measurement. In Scenario I, no change was made to the current and charge capacity measurements, while in Scenario II, the current-related features, the current and charge capacity, were artificially increased by 2% to account for the effect of the current sensor error in the capacity calculation [29]. The combination of the two scenarios and the two initial SOC settings presents a total of four different cases, listed as follows:

- Case 1 — Low initial SOC (3%–23%), no bias in current measurement;
- Case 2 — Low initial SOC (3%–23%), 2% positive bias in current measurement;
- Case 3 — High initial SOC (23%–43%), no bias in current measurement;
- Case 4 — High initial SOC (23%–43%), 2% positive bias in current measurement.

Therefore, four different datasets, with the distinctive settings of the initial voltage and current measurement noise, were constructed for the capacity estimation. DCNN models were built for each of these datasets to verify the accuracy of the proposed method in the online capacity estimation as well as its robustness to the initial SOC level and current measurement error.

3.3. *Implementation detail*

The objective of the optimization algorithm SGD with momentum is to lower the expected generalization error given by Eq. (2). To achieve this objective, the DCNN model was trained with 35 epochs and a mini-batch of 128 examples. An initial learning rate α was set to 0.01 for all convolutional and fully-connected layers and this rate dropped by a factor of 5 for every 7 epochs. Table 4 summarizes several important parameters used in training the DCNN model.

The DCNN model mainly consisted of five convolutional layers and three fully-connected layers. We randomly initialized the weights in each layer with the mean of 0 and standard deviation of 0.01, and the biases with 0. The network configurations are outlined in Table 5.

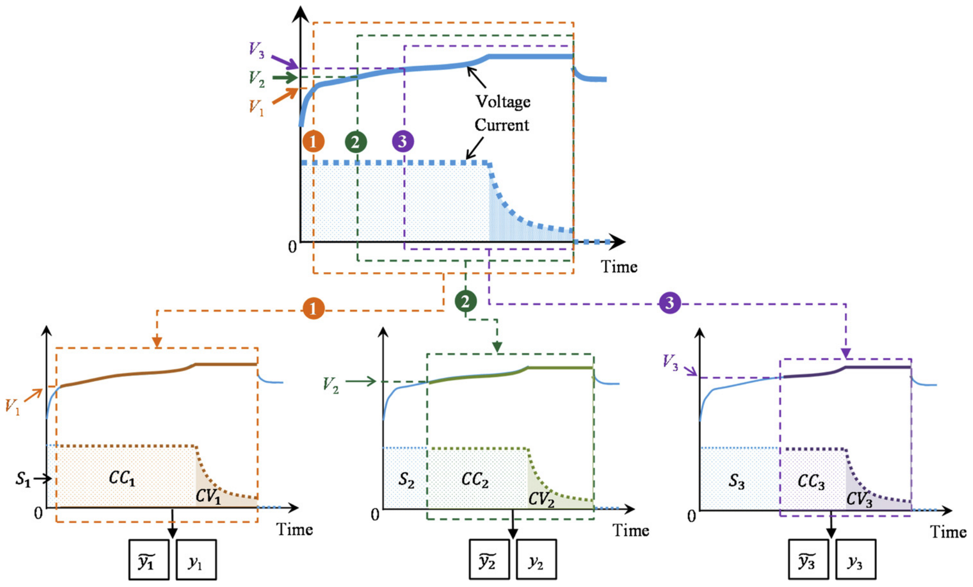


Fig. 5. Illustration of the generation of partial charge curves. It should be mentioned that the implemented truncation strategy ignores the battery polarization in the charge process.

Table 3
The two initial SOC settings considered in generation of partial charge data.

Setting	Setting name	Voltage range	SOC range
Setting I	Low initial SOC	V_{low} = 3.65 V, V_{high} = 3.80 V	Roughly 3%–23%
Setting II	High initial SOC	V_{low} = 3.80 V, V_{high} = 3.85 V	Roughly 23%–43%

Table 4
List of parameter values used in the DCNN training.

Parameter	Value
Number of epochs	35
Momentum, γ	0.9
Initial learning rate, α	0.01
L_2 regularization, λ	0.0001
Mini-batch size	128

Table 5
Summary of the configuration of the DCNN model.

Layer name	Filter size	Number of kernel	Sride size	Number of weights	Number of biases
Input	25 × 3 × 1	–	–	–	–
Conv.1	1 × 2×1	16	(1,1)	32	16
Max-pooling	3 × 1 × 1	16	(2,1)	–	–
Conv.2	3 × 1 × 1	32	(1,1)	96	32
Conv.3	3 × 1 × 1	40	(1,1)	120	40
Conv.4	3 × 1 × 1	40	(1,1)	120	40
Conv.5	3 × 1 × 1	40	(1,1)	120	40
FC.1	40 × 1	–	–	25600	40
FC.2	40 × 1	–	–	1600	40
FC.3	1 × 1	–	–	40	1

Early stopping and dropout are two effective techniques that are often used to mitigate the overfitting issue during the training process. Due to the use of BN, dropout was not considered in the training of our model but early stopping was kept [40]. The results of this study show that removing dropout results in higher validation accuracy. Early stopping was designed to cause the algorithm to halt whenever overfitting begins to occur, the validation RMSE was allowed to be larger than or equal to the previous smallest validation RMSE five times before the network training stops, and the frequency of validation was every one epoch.

Furthermore, a deep learning model often takes a long time to train. This is mainly due to a large number of parameters needed to be determined through the training process. In this study, we used an NVIDIA TITAN XP graphics processing unit (GPU), which possesses a massively parallel architecture with thousands of small, efficient cores designed for handling multiple tasks simultaneously. With support from the GPU, we were able to simultaneously compute the updates that minimize $J_R(\Theta)$ based on multiple mini-batches of 128 samples. In other words, the training process in one epoch was decomposed into a number of iterations. These iterations would be run in parallel, each of which carried out the parameter updating with one mini-batch. As such, training the DCNN model with a large amount of data became computationally affordable and the hyper-parameters that have significant impact on the performance of the networks were rapidly identified.

3.4. Definition of error measures

The accuracy of the proposed DCNN model was first evaluated with the 10-year daily cycling dataset using eight-fold cross validation as illustrated in Fig. 6.

The complete dataset was first divided into eight mutually exclusive folds corresponding to the eight implantable Li-ion cells as described in Subsection 3.1. Eight trials consisting of training, validation, and test were then implemented such that within each trial the data from one cell out of the entire dataset was held out, to be used for testing, while the remaining data from the other seven cells were split into a training set and a validation set in a ratio of 70:30. The performance of the proposed method was evaluated using a test set that consisted of the complete data from one cell and was separated from the training and validation sets. After performing all eight cross validation trials, the overall test error ε_{RMS}^{All} of the DCNN model was estimated by taking the average of the individual test errors ε_{RMS}^{k} across the eight trials.

$$\varepsilon_{\mathrm{RMS}}^{k} = \sqrt{\frac{1}{N_k}\sum_{i=1}^{N_k}(y^k(\mathbf{x}_i^k) - \hat{y}^k(\mathbf{x}_i^k))^2} \tag{6}$$

$$\varepsilon_{\mathrm{RMS}}^{\mathrm{All}} = \sqrt{\frac{1}{\sum_{k=1}^{8}N_k}\sum_{k=1}^{8}\sum_{i=1}^{N_k}(y^k(\mathbf{x}_i^k) - \hat{y}^k(\mathbf{x}_i^k))^2} \tag{7}$$

where N_k denotes the number of samples used for the test in the k^{th}

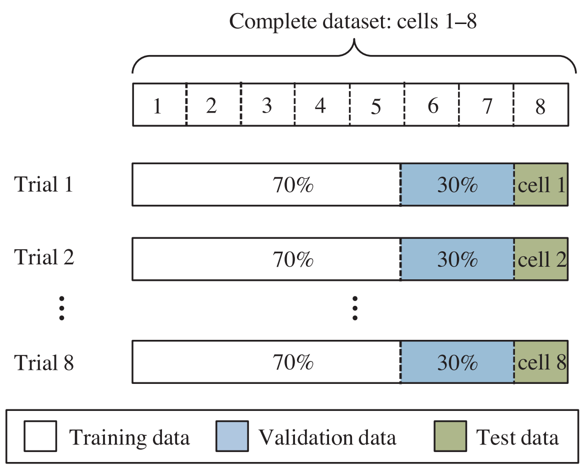


Fig. 6. Procedure of eight-fold cross validation. In each cross validation trial, one cell was used as the test set. The training set in each trial consisted of 70% of all samples randomly drawn from the remaining dataset of seven cells, and the remaining 30% of the samples were used to form the validation set.

trial, $\hat{y}^k(\mathbf{x}_i^k)$ and $y^k(\mathbf{x}_i^k)$ denote the estimated and measured (or true) capacities for the i^{th} sample in the k^{th} trial, respectively, and $\mathbf{x}_i^k$ denotes the i^{th} sample matrix in the k^{th} trial. It is important to note that eight trained DCNN models, each associated with one cell and tested in one trial, were assessed by the eight-fold cross validation (see Fig. 6). The performance of the proposed DCNN model was reflected by the overall test error defined according to Eq. (7).

Next, the performance of the proposed method on the capacity estimation of cells cycled with various cycling protocols and profiles was verified by a five-fold cross-validation. These cells were subjected to the half-year cycling test by NASA, as described in Subsection 3.1. It is noted that the accuracy evaluation process (i.e., cross validation) between the half-year cycling dataset and the 10-year daily cycling dataset was similar except the constitution of each trial. Instead of consisting of one cell at each fold, the half-year cycling dataset had four different cells at each fold.

4. Results and discussion

4.1. Case Study 1: capability estimation of implantable cells

In order to determine if the proposed deep learning method can achieve the state-of-the-art accuracy in capacity estimations, a traditional machine learning method, RVM, was also implemented in the four cases described in Subsection 3.2. In the RVM model, the Gaussian kernel function was employed as the kernel function with a kernel width of 0.8. The maximum number of iterations for training the RVM model was set to 500. The kernel type and the value of the kernel width were empirically determined based on the accuracy of a trained RVM model in capacity estimations, and the chosen kernel type (Gaussian kernel) and width (0.8) resulted in a trained RVM model with the best accuracy in capacity estimations. The empirical study that identified the kernel type and width can be found in [19].

Tables 6–9shows the comparison results for the RVM and DCNN models with regard to both the RMSE and maximum error (MaxE). To minimize the effect of randomness in the capacity estimation, we generated 150 different training and validation datasets with each being executed one time. The capacity estimation results provided by the DCNN model with the lowest validation error out of the 150 runs were displayed in Tables 6–9.

On the basis of these results, three important observations can be made and are listed as follows:

- First, the overall RMSEs and MaxEs of DCNN were apparently lower than that of RVM, indicating that the DCNN could make more accurate capacity estimation results regardless of the initial SOC settings and current measurement errors. While there were several situations where the individual MaxE of DCNN was slightly higher than the corresponding MaxE of RVM, the overall performance of DCNN was evidently superior to RVM in all four cases. Moreover, it could be observed that the performance improvement of DCNN over RVM was more significant in the high SOC than the low SOC.
- Second, the DCNN was capable of making satisfactory capacity estimations when a 2% positive bias was artificially introduced to the current measurement (compare Case 1 versus Case 2 and Case 3 versus Case 4), suggesting that the DCNN could adapt the regression model to changes in the input feature matrixes.
- Third, the RMSE by the proposed deep learning method did not show large increases when the initial SOC range was elevated (10.7% increase from Case 1 to Case 3 and 12.6% increase from Case 2 to Case 4). In contrast, much larger increases were observed in the RMSE by RVM (45.3% increase from Case 1 to Case 3 and 48.8% increase from Case 2 to Case 4). These results thus indicated that neither deep (low initial SOC 3%–23%) nor shallow (high initial SOC 23%–43%) discharge conditions affected the capacity of the DCNN to make an accurate capacity estimation.

Based on the above discussion, it can be concluded that the DCNN model was able to achieve a more accurate capacity estimation than RVM regardless of the initial charge level and the current measurement noise level. This is not surprising, since deep learning methods were able to automate feature extraction process and leverage larger volumes of information as compared to traditional machine learning methods.

The capacity estimation results of cell 1 produced by DCNN for Cases 1–4 are shown in Fig. 7. The measured capacity for a charge/discharge cycle was calculated using the coulomb counting method, which integrated the discharge current over time for the entire discharge cycle. It can be observed from Fig. 7 that the DCNN model could closely track the capacity fade trend throughout the 10-year cycling test for all four cases. In other words, gross changes in the inputs seemed to have negligible effects on the robustness of the DCNN model in making an accurate capacity estimation. Similar observations can be made on the other seven cells.

4.2. Case study 2: capability estimation of NASA cells

In Subsection 4.1, Case Study 1 has shown that the proposed DCNN model is able to accurately estimate the capacity of Li-ion batteries cycled with a single cycling protocol and profile. In this subsection, an open source dataset from NASA was employed to further investigate the performance of the DCNN model on the cells cycled with various

Table 6
Capacity estimation results achieved by RVM and DCNN for Case 1.

Method	Item	Cell #1	Cell #2	Cell #3	Cell #4	Cell #5	Cell #6	Cell #7	Cell #8	Overall
RVM	RMSE (%)	0.338	0.355	0.523	0.442	0.469	0.403	0.420	0.399	0.419
	MaxE (%)	4.531	2.143	3.012	2.603	2.916	2.468	2.488	2.881	4.531
DCNN	RMSE (%)	0.302	0.298	0.309	0.368	0.410	0.312	0.277	0.262	0.317
	MaxE (%)	2.491	2.683	3.329	3.524	2.296	2.534	2.885	2.027	3.524

Table 7
Capacity estimation results achieved by RVM and DCNN for Case 2.

Method	Item	Cell #1	Cell #2	Cell #3	Cell #4	Cell #5	Cell #6	Cell #7	Cell #8	Overall
RVM	RMSE (%)	0.361	0.441	0.384	0.534	0.506	0.364	0.382	0.371	0.418
	MaxE (%)	4.047	3.225	2.991	3.005	3.233	3.489	2.674	2.927	4.047
DCNN	RMSE (%)	0.358	0.251	0.322	0.445	0.392	0.337	0.279	0.271	0.333
	MaxE (%)	2.815	2.347	4.440	4.924	2.130	2.893	4.261	2.059	4.924

cycling protocols and profiles.

Table 10 summarizes the capacity estimation results by the artificial neural network (ANN) and the proposed DCNN model based on the NASA cells. The ANN was a six-layer feedforward network with 75 input neurons and one output neuron (i.e., capacity estimate). The numbers of neurons in the first, second, third, and fourth hidden layers were 500, 500, 250, and 100, respectively. It should be pointed out that the number of neurons in each layer was identified by minimizing the RMSE for the validation dataset. As can be seen from Table 10, the overall RMSEs of the proposed DCNN model were less than 2.00% for all the trials and the MaxEs less than 10.00%. These results suggested that the proposed DCNN model could still achieve reasonable accuracy in capacity estimations when cells were cycled with different cycling protocols and profiles. In addition, a comparison of the RMSEs and MaxEs between ANN and DCNN suggested that the traditional machine learning method (ANN) produced less accurate capacity estimations than the proposed deep learning method (DCNN).

While the results from Case Study 2 were less accurate than those in Case Study 1, the reason might arise from the limited amount of available training data rather than the capability of the proposed method. It is common knowledge that data-driven methods will perform better on the unseen data with more training data. Compared to the 10-year daily cycling data during over 3000 charge/discharge cycles, the half-year cycling data used in Case Study 2 only had roughly 30 charge/discharge cycles. The results therefore further verified that the proposed method was effective in estimating the capacity of Li-ion batteries. Combining the results in Case Study 1, it can be concluded that the proposed method was capable of making an accurate capacity estimation for cells not only cycled with a single protocol and profile but also various protocols and profiles.

To thoroughly investigate how the size of a training dataset affects the performance of the proposed DCNN model, we constructed eight new training datasets of different sizes by increasing and decreasing the number of samples in the original training dataset. The maximum number of samples in a training dataset is denoted as n_{max} and ranged from $N/10$ to $N*3$, i.e., $n_{max} \in \{N/10, N/4, N/3, N/2, N/1.75, N/1.5, N, N*2, N*3\}$. Note that when $n_{max} = N$, the original training dataset was used without any adjustment to the sample size. For each of the nine training datasets, an identical test dataset was used for evaluating the generalization ability of the trained DCNN model. The overall test RMSE is plotted as a function of the number of training samples in Fig. 8. It is well established that training a deep learning model with a small set of training data may cause the overfitting issue. This phenomenon can be easily observed for training datasets whose numbers of samples are no larger than $N/2$ (i.e., $n_{max} \leq N/2$). As shown in Fig. 8, a DCNN model reached convergence to below 2% test RMSE when the number of samples in the training dataset increased from $N/10$ to $N/1.75$ (i.e., approximately 43% less samples than the original training dataset). This observation indicates that the degree of overfitting was not severe for the proposed DCNN model which was trained with $n_{max} = N$.

4.3. Computational efficiency

In this subsection, we compare the proposed DCNN model with the RVM method in terms of computational times and memory usages for each of the eight cells in the 10-year daily cycling data. This comparison was executed on an Intel Core processor i7-8700 CPU 3.2 GHz and NVIDIA Titan Xp GPU with 12 GB GDDR5X memory and 64 GByte RAM. To minimize the effect of randomness during the measurements, we generated 150 different training and validation datasets with each being executed one time. Table 11 summarizes the mean computational times and the memory usages on the best 30 results out of 150 individual runs. It is not surprising that the training of the proposed DCNN model required 7.6 times more training time (508.2 s on average) than the RVM model (66.6 s on average). However, on average, the DCNN model required 9.8 times less test time and 139.1 times less memory compared with the RVM model, suggesting that the proposed DCNN model was more feasible in the online capacity estimation than the RVM model.

4.4. Sensitivity to input size

The input to the DCNN model consisted of a sample matrix with a fixed size $25 \times 3 \times 1$. To investigate the sensitivity of the proposed method to the input size, we created different sizes of input matrices including $12 \times 3 \times 1$, $25 \times 3 \times 1$, $47 \times 3 \times 1$, $83 \times 3 \times 1$, $139 \times 3 \times 1$, and $219 \times 3 \times 1$, inspired by Schroff et al. [43] and Ciresan et al. [44]. Fig. 9(a) shows the performance of the proposed DCNN model with different sizes of input matrices. As can be observed from Fig. 9(a), all the input sizes generated similar levels of RMSE, except the input size $12 \times 3 \times 1$ which slightly had a higher RMSE than other input sizes. On the other hand, it can be observed from Fig. 9(b) that the training time and test time required by the proposed method increased as the input matrix size increased. Considering the overall performance and the required computational time of the proposed method, an option of $25 \times 3 \times 1$ was chosen as the input matrix size in this study.

4.5. Effect of convolutional layers

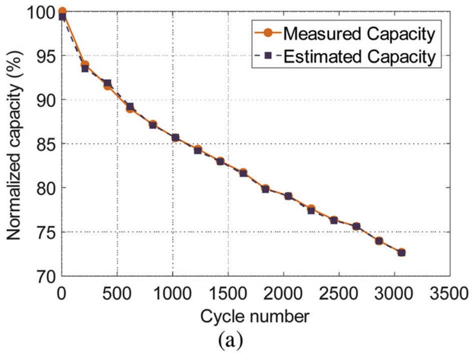
It is commonly known that the optimal number of convolutional layers is closely related to the amount and complexity of the available data. To further explore this concept, we conducted a parametric study

Table 8
Capacity estimation results achieved by RVM and DCNN for Case 3.

Method	Item	Cell #1	Cell #2	Cell #3	Cell #4	Cell #5	Cell #6	Cell #7	Cell #8	Overall
RVM	RMSE (%)	0.473	0.492	0.574	0.558	0.841	0.560	0.761	0.604	0.608
	MaxE (%)	5.408	2.012	6.242	6.226	5.626	7.651	3.624	2.240	7.651
DCNN	RMSE (%)	0.340	0.360	0.338	0.470	0.392	0.278	0.323	0.308	0.351
	MaxE (%)	4.346	2.837	6.286	6.428	4.557	5.860	2.698	2.192	6.428

Table 9
Capacity estimation results achieved by RVM and DCNN for Case 4.

Method	Item	Cell #1	Cell #2	Cell #3	Cell #4	Cell #5	Cell #6	Cell #7	Cell #8	Overall
RVM	RMSE (%)	0.419	0.554	0.569	0.647	0.881	0.639	0.662	0.601	0.622
	MaxE (%)	5.085	2.148	6.169	6.010	3.567	8.612	3.312	2.526	8.612
DCNN	RMSE (%)	0.341	0.371	0.394	0.420	0.492	0.288	0.270	0.425	0.375
	MaxE (%)	4.606	2.790	5.943	6.549	4.783	4.107	3.711	2.497	6.549



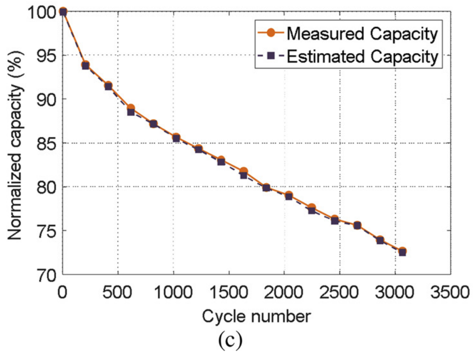


Fig. 7. Capacity estimation results of cell 1 of Case 1 (a), Case 2 (b), Case 3 (c) and Case 4 (d). (Results are plotted every 200 cycles for the ease of visualization.)

Table 10
Capacity estimation results achieved by the ANN and DCNN based on the NASA cells.

Method	Item	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Overall
ANN	RMSE (%)	1.914	1.795	2.175	2.323	1.653	1.972
	MaxE (%)	12.626	15.254	13.549	9.821	4.803	15.254
DCNN	RMSE (%)	1.405	1.305	1.628	1.683	1.365	1.477
	MaxE (%)	8.403	9.479	7.207	6.567	4.973	9.479

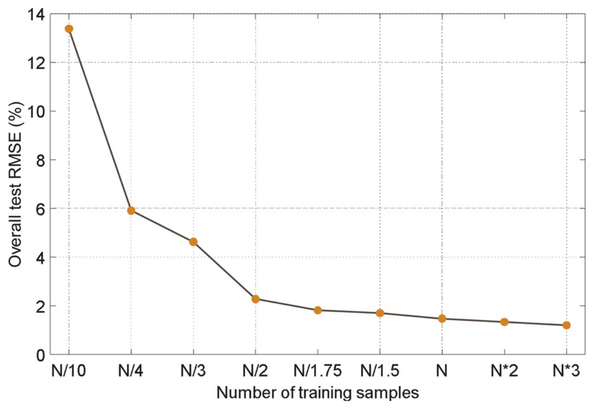


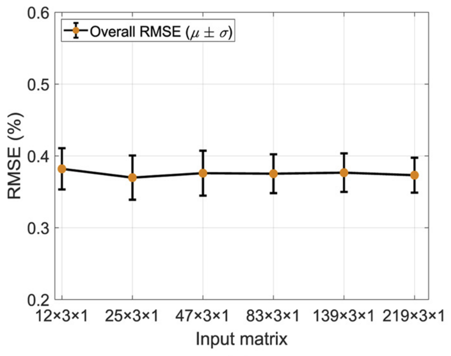
Fig. 8. Overall test RMSE vs. number of training samples.

Table 11
Comparison results of computational time and memory usage with 30 different training and validation datasets.

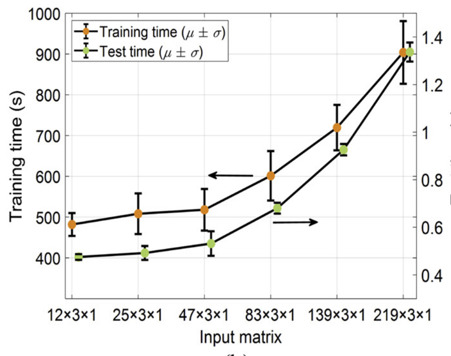
Cell number	Time (s)					
	Training		Test		Memory (MB)	
	DCNN	RVM	DCNN	RVM	DCNN	RVM
Cell #1	531.527	62.152	0.492	4.794	142.800	20019.800
Cell #2	493.660	72.115	0.494	4.793	143.300	19950.800
Cell #3	512.384	63.607	0.487	4.828	137.500	20001.400
Cell #4	461.309	62.453	0.480	4.675	146.300	20045.100
Cell #5	518.215	65.845	0.486	4.870	145.200	19933.200
Cell #6	548.239	69.014	0.497	4.903	146.300	19962.800
Cell #7	502.107	74.600	0.511	4.887	144.400	19974.600
Cell #8	498.160	62.861	0.489	4.956	143.500	19934.400

to empirically investigate the effect of the number of convolutional layers on the capacity estimation accuracy of the DCNN. Two box plots of the overall RMSE against the number of convolutional layers were produced to graphically illustrate the simulation results and are given in Fig. 10. It is important to highlight that a large number of convolutional layers may give rise to a very small size of outputs in the last convolutional layer. Therefore, the maximum number of convolutional layers attempted for the neural networks was set to 6 to prevent the output size of the last layer from becoming too small.

Based on the above plots, three observations can be made that:

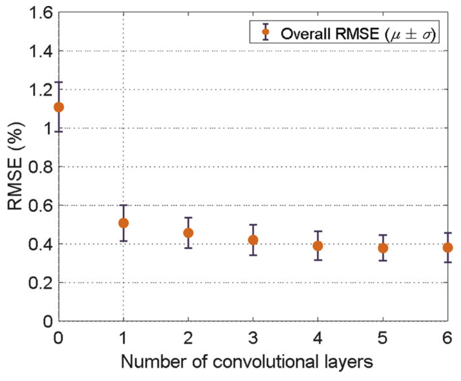


(a)

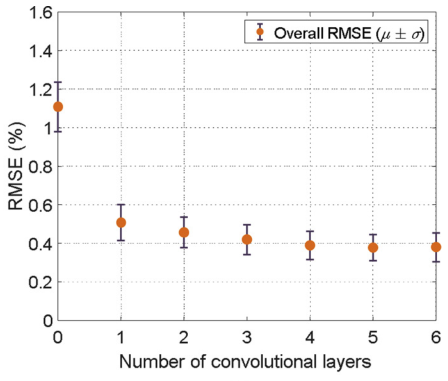


(b)

Fig. 9. Overall RMSE (a), and training time and test time (b) for different input matrices. To quantify the uncertainty estimate of run-to-run variation, 30 different training and validation datasets were generated with each being executed one time and the results were used to plot error bars showing the sensitivity of the proposed method to the input size.



(a)



(b)

Fig. 10. Model performance on different numbers of convolutional layers for Case 1 (a) and Case 3 (b). To quantify the uncertainty estimate of run-to-run variation, 150 different training and validation datasets were generated with each being executed one time and the best 30 results out of the 150 runs were used to plot error bars.

- First, while each convolutional layer in a DCNN model contained less than 1% of the model's parameters (see Tables 1 or 5), the convolutional layers were of importance for the estimation accuracy.
- Second, comparisons of the RMSE between deep neural networks (i.e., DCNN with four or more convolutional layers) and conventional neural networks (zero of convolutional layers) suggested that the deeper neural networks dramatically improved the accuracy in the capacity estimation over the conventional neural networks.
- Third, an inappropriate selection of the number of convolutional layers might give rise to a decrease in estimation accuracy.

As a result of our study, we can conclude that the DCNN model with five convolutional layers produced a slightly lower RMSE as compared with the model with any other number of convolutional layers. Therefore, the final DCNN model was chosen to contain five convolutional layers and this network depth seemed to ensure satisfactory performance of the proposed deep learning method. The results described earlier (Tables 6–9 and Fig. 7) were derived using this chosen value of 5.

5. Conclusion

This paper presents a deep convolutional neural network (DCNN) for online estimation of Li-ion battery capacity based on the voltage, current, and charge capacity measurements during a partial charge cycle. The DCNN model is capable of approximating the complex data-to-health relationship of battery cells from large-scale datasets. To the best of our knowledge, this study is one of the first attempt to apply deep learning to the online capacity estimation of Li-ion batteries. The effectiveness of the proposed deep learning method was verified using 10-year daily cycling data acquired from eight Li-ion battery cells and half-year cycling data acquired from 20 cells. The verification results demonstrate that the proposed deep learning method achieves promising accuracy in the capacity estimation, indicating that our method is an efficient tool for online health management of Li-ion batteries.

Based on this study, the proposed deep learning method has shown plausible benefits for battery capacity estimations. However, the proposed deep learning method still has some limitations. The fixed size input matrix and the insufficient understanding of the inner working mechanism hinder the application of the proposed method in the battery capacity estimation and need further investigation. Another important consideration is the applicability of the proposed method under variable ambient temperature conditions. The present results apply to a single temperature for each case study (i.e., 37 °C for Case Study 1 and 25 °C for Case Study 2). However, variations in temperature can result in significant changes to the capacity and accounting for the effect of temperature will be essential for the proposed method to be applicable when the ambient temperature varies considerably.

These limitations motivate us to explore further improvements of the proposed deep learning method. Our future study will investigate the applicability of the proposed deep learning method to cells cycled with the constant power-constant voltage (CP-CV) charging protocol and more dynamic operation protocols, for example, the urban dynamic driving schedule (UDDS) and highway fuel economy driving schedule (HWFET). Another interesting future study will be to combine the DCNN model with a recurrent neural network (RNN) model that is scalable to any size of the input [45]. These future studies will allow us to more thoroughly study the effectiveness of deep learning in the online capacity estimation of Li-ion batteries.

Acknowledgement

This research was in part supported by the US National Science Foundation (NSF) Grant Nos. CNS-1566579 and ECCS-1611333, the NSF I/UCRC Center for e-Design, and the U.S. Department of Transportation, Office of the Assistant Secretary for Research and

Technology (USDOT/OST-R) through the Midwest Transportation Center (MTC). Any opinions, findings or conclusions in this paper are those of the authors and do not necessarily reflect the views of the sponsoring agencies. The authors would also like to express special thanks to Dr. Gaurav Jain and Dr. Hui Ye at Medtronic, Inc. for sharing the long-term cycling data for this study. Finally, the authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

[1] Davide Andrea, Battery Management Systems for Large Lithium Ion Battery Packs, Artech House, 2010.
[2] Languang Lu, Xuebing Han, Jianqiu Li, Jianfeng Hua, Minggao Ouyang, A review on the key issues for lithium-ion battery management in electric vehicles, J. Power Sources 226 (2013) 272–288.
[3] G.L. Plett, Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs: part 3. state and parameter estimation, J. Power Sources 134 (2) (2004) 277–292.
[4] Rudolph Van Der Merwe, Eric Wan, Simon Julier, Sigma-point Kalman filters for nonlinear estimation and sensor-fusion: applications to integrated navigation, AIAA Guidance, Navigation, and Control Conference and Exhibit, (2004), p. 5120.
[5] Gregory L. Plett, Sigma-point Kalman filtering for battery management systems of LiPB-based HEV battery packs: part 2: simultaneous state and parameter estimation, J. Power Sources 161 (2) (2006) 1369–1384.
[6] C. Hu, B.D. Youn, J. Chung, A multiscale framework with extended Kalman filter for lithium-ion battery SOC and capacity estimation, Appl. Energy 92 (2012) 694–704.
[7] Yuan Zou, Xiaosong Hu, Hongmin Ma, Shengbo Eben Li, Combined state of charge and state of health estimation over lithium-ion battery cell cycle lifespan for electric vehicles, J. Power Sources 273 (2015) 793–803.
[8] Fei Zhang, Guangjun Liu, Lijin Fang, Battery state estimation using unscented Kalman filter, IEEE International Conference on Robotics and Automation, 2009. ICRA'09 (2009) 1863–1868.
[9] Cheng Chen, Rui Xiong, Weixiang Shen, A lithium-ion battery-in-the-loop approach to test and validate multiscale dual h infinity filters for state-of-charge and capacity estimation, IEEE Trans. Power Electron. 33 (1) (2018) 332–342.
[10] Benjamín E. Olivares, Matias A. Cerda Munoz, Marcos E. Orchard, Jorge F. Silva, Particle-filtering-based prognosis framework for energy storage devices with a statistical characterization of state-of-health regeneration phenomena, IEEE Trans. Instrum. Meas. 62 (2) (2013) 364–376.
[11] Scott J. Moura, Nalin A. Chaturvedi, Miroslav Krstić, Adaptive partial differential equation observer for battery state-of-charge/state-of-health estimation via an electrochemical model, J. Dyn. Syst. Meas. Control 136 (1) (2014) 011015.
[12] Alexander Bartlett, James Marcicki, Simona Onori, Giorgio Rizzoni, Xiao Guang Yang, Ted Miller, Electrochemical model-based state of charge and capacity estimation for a composite electrode lithium-ion battery, IEEE Trans. Control. Syst. Technol. 24 (2) (2016) 384–399.
[13] Rui Xiong, Linlin Li, Jinpeng Tian, Towards a smarter battery management system: a critical review on battery state of health monitoring methods, J. Power Sources 405 (2018) 18–29.
[14] Brian Bole, Chetan S. Kulkarni, Matthew Daigle, Adaptation of an Electrochemistry-based Li-ion Battery Model to Account for Deterioration Observed Under Randomized Use, SGT, Inc. Moffett Field United States, 2014.
[15] Robert R. Richardson, Christoph R. Birkl, Michael A. Osborne, David A. Howey, Gaussian process regression for in situ capacity estimation of lithium-ion batteries, IEEE Trans. Ind. Inform. 15 (1) (2018) 127–138.
[16] CALCE Battery Research Group. (2017). Retrieved June 18, 2019, from https://web.calce.umd.edu/batteries/data.htm.
[17] Heather M. Barkholtz, Armando Fresquez, Babu R. Chalamala, Summer R. Ferreira, A database for comparative electrochemical performance of commercial 18650-format lithium-ion cells, J. Electrochem. Soc. 164 (12) (2017) A2697–A2706.
[18] Kristen A. Severson, Peter M. Attia, Norman Jin, Nicholas Perkins, Benben Jiang, Zi Yang, Michael H. Chen, et al., Data-driven prediction of battery cycle life before capacity degradation, Nat. Energy (2019) 1.
[19] Chao Hu, Gaurav Jain, Craig Schmidt, Carrie Strief, Melani Sullivan, Online estimation of lithium-ion battery capacity using sparse Bayesian learning, J. Power Sources 289 (2015) 105–113.
[20] Amir Navot, Lavi Shpigelman, Naftali Tishby, Eilon Vaadia, Nearest neighbor based feature selection for regression and its application to neural activity, Advances in Neural Information Processing Systems, (2006), pp. 996–1002.
[21] Chao Hu, Gaurav Jain, Puqiang Zhang, Craig Schmidt, Parthasarathy Gomadam, Tom Gorka, Data-driven method based on particle swarm optimization and k-nearest neighbor regression for estimating capacity of lithium-ion battery, Appl. Energy 129 (2014) 49–55.
[22] Gae-won You, Sangdo Park, Oh Dukjin, Real-time state-of-health estimation for electric vehicle batteries: a data-driven approach, Appl. Energy 176 (2016) 92–103.
[23] Akram Eddahech, Olivier Briat, Nicolas Bertrand, Jean-Yves Delétage, Jean-Michel Vinassa, Behavior and state-of-health monitoring of Li-ion batteries using impedance spectroscopy and recurrent neural networks, Int. J. Electr. Power Energy Syst. 42 (1) (2012) 487–494.
[24] Ho-Ta Lin, Tsorng-Juu Liang, Shih-Ming Chen, Estimation of battery state of health using probabilistic neural network, IEEE Trans. Ind. Inform. 2 (9) (2013) 679–685.
[25] Guangxing Bai, Pingfeng Wang, Chao Hu, Michael Pecht, A generic model-free approach for lithium-ion battery health management, Appl. Energy 135 (2014) 247–260.
[26] Liu Wang Kang, Xuan Zhao, Ma. Jian, A new neural network model for the state-of-charge estimation in the battery degradation process, Appl. Energy 121 (2014) 20–27.
[27] Yoshua Bengio, Aaron Courville, Pascal Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[28] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, Deep learning, nature 521 (7553) (2015) 436.
[29] Sheng Shen, M.K. Sadoughi, Xiangyi Chen, Mingyi Hong, Hu. Chao, Online estimation of lithium-ion battery capacity using deep convolutional neural networks, ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (2018) pp. V02AT03A058–V02AT03A058.
[30] C. Hu, G. Jain, P. Tamirisa, T. Gorka, Method for estimating capacity and predicting remaining useful life of lithium-ion battery, Appl. Energy 126 (2014) 182–189.
[31] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, Zbigniew Wojna, Rethinking the inception architecture for computer vision, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2016), pp. 2818–2826.
[32] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, David McClosky, The stanford CoreNLP natural language processing toolkit, Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, (2014), pp. 55–60.
[33] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, Ye-Yi Wang, Representation learning using multi-task deep neural networks for semantic classification and information retrieval, (2015).
[34] Dan Ciregan, Ueli Meier, Jürgen Schmidhuber, Multi-column deep neural networks for image classification, 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012).
[35] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (NIPS 2012) (2012) 1097–1105.
[36] Xiaosong Hu, Shengbo Eben Li, Zhenzhong Jia, Egardt Bo, Enhanced sample entropy-based health management of Li-ion battery for electrified vehicles, Energy 64 (2014) 953–960.
[37] Duo Yang, Xu Zhang, Rui Pan, Yujie Wang, Zonghai Chen, A novel Gaussian process regression model for state-of-health estimation of lithium-ion battery using charging curve, J. Power Sources 384 (2018) 387–395.
[38] Robert R. Richardson, Michael A. Osborne, David A. Howey, Gaussian process regression for forecasting battery state of health, J. Power Sources 357 (2017) 209–219.
[39] Vinod Nair, Geoffrey E. Hinton, Rectified linear units improve restricted boltzmann machines, Proceedings of the 27th International Conference on Machine Learning (ICML-10), (2010).
[40] Sergey Ioffe, Christian Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, International Conference on Machine Learning, (2015).
[41] Boris T. Polyak, Some methods of speeding up the convergence of iteration methods, USSR Comput. Math. Math. Phys. 4 (5) (1964) 1–17.
[42] C. Hu, Y. Hui, G. Jain, C. Schmidt, Remaining useful life assessment of lithium-ion batteries in implantable medical devices, J. Power Sources 375 (2018) 118–130.
[43] Florian Schroff, Dmitry Kalenichenko, James Philbin, Facenet: a unified embedding for face recognition and clustering, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2015), pp. 815–823.
[44] Dan C. Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, Jürgen Schmidhuber, Flexible, high performance convolutional neural networks for image classification, IJCAI Proceedings-International Joint Conference on Artificial Intelligence vol. 22 (1, (2011) 1237.
[45] Hao Wu, Saurabh Prasad, Convolutional recurrent neural networks for hyperspectral data classification, Remote Sens. 9 (3) (2017) 298.