Lithium-ion Battery Capacity Estimation - A Pruned Convolutional Neural Network Approach Assisted with Transfer Learning

Yihuan Li[a], Kang Li[a,*], Xuan Liu[a], Yanxia Wang[b], Li Zhang[c]

[a]*School of Electronic and Electrical Engineering, University of Leeds, LS2 9JT, UK*
[b]*College of Metropolitan Transportation, Beijing University of Technology, Beijing, 100124, China*
[c]*School of Mechatronics and Automation, Shanghai University, Shanghai 200072, China*

Abstract

Online battery capacity estimation is a critical task for battery management system to maintain the battery performance and cycling life in electric vehicles and grid energy storage applications. Convolutional Neural Networks, which have shown great potentials in battery capacity estimation, have thousands of parameters to be optimized and demand a substantial number of battery aging data for training. However, these parameters require massive memory storage while collecting a large volume of aging data is time-consuming and costly in real-world applications. To tackle these challenges, this paper proposes a novel framework incorporating the concepts of transfer learning and network pruning to build compact Convolutional Neural Network models on a relatively small dataset with improved estimation performance. First, through the transfer learning technique, the Convolutional Neural Network model pre-trained on a large battery dataset is transferred to a small dataset of the targeted battery to improve the estimation accuracy. Then a contribution-based neuron selection method is proposed to prune the transferred model using a fast recursive algorithm, which reduces the size and computational complexity of the model while maintaining its performance. The proposed model is capable of achieving fast online capacity estimation at any time, and its effectiveness is verified on a target dataset collected from four Lithium iron phosphate battery cells, and the performance is compared with other Convolutional Neural Network models. The test results confirm that the proposed model outperforms other models in terms of accuracy and computational efficiency, achieving up to 68.34% model size reduction and 80.97% computation savings.

*Corresponding author
Email addresses: elyli2@leeds.ac.uk (Yihuan Li), K.Li1@leeds.ac.uk (Kang Li), elxl@leeds.ac.uk (Xuan Liu), yanxiawang@bjut.edu.cn (Yanxia Wang), zl_qee@163.com (Li Zhang)

Keywords: Capacity estimation, Convolutional neural networks, Lithium-ion batteries, Network Pruning, Transfer learning

---

Nomenclature

Acronyms

BMS	battery management system
CC	constant current
CNN	convolutional neural network
$CNN(S)$	convolutional neural network (pre-trained on the source dataset)
$CNN(S)-TL$	convolutional neural network (pre-trained on the source dataset) with transfer learning
$CNN(T)$	convolutional neural network (directly trained on the target dataset)
CV	constant voltage
EV	electric vehicle
FC	fully connected
$FLOPs$	floating point operations
FRA	fast recursive algorithm
GRP	Gaussian process regression
LFP	lithium iron phosphate
MAE	mean absolute error
NEE	normalized estimation error
NN	neural network

$PCNN$	pruned convolutional neural network
$PCNN(S)-TL$	pruned convolutional neural network (pre-trained on the source dataset) with transfer learning
$PCNN(T)$	pruned convolutional neural network (directly trained on the target dataset)
$RMSE$	root mean-square error
RVM	relevance vector machine
SOC	state of charge
SOH	state of health
SVM	support vector machine
TL	transfer learning

Symbols

$\mathbf{\Theta}$	parameter matrix
φ	candidate model term
E_k	cost function when k model terms are selected
ΔE_k	the net contribution of the k th model term
$\hat{\theta}_j$	parameter of the j th selected model term
$\mathbf{\Psi}$	regression matrix
$\mathbf{\Xi}$	model residual vector
$\mathbf{M}_k$	a recursive information matrix
$\mathbf{R}_k$	a recursive residual matrix
$\mathbf{y}$	system output
I	identity matrix

L	number of neurons inputted to the first fully connected layer
M	number of data chunks segmented from one charge cycle
N	the number of data samples
n	data segmentation length
S	total number of candidate model terms
S_1	number of neurons inputted to the second fully connected layer
W	weights and bias matrix of the second fully connected layer
X	input matrix to the first fully-connected layer
Y	model output

1. Introduction

The battery energy storage systems, which capture energy and store it using electrochemical solutions, are playing vital roles in mass roll-out of electric vehicles (EVs) [1] and acceptance of significant penetration of renewable power worldwide [2], and can provide clear socio-economic benefits in the transition to low carbon economies [3]. Particularly, the lithium-ion battery has been widely used in EVs and grid energy storage systems due to a number of desirable features [1], and it is crucial to maintain its functional performance, cycling life, and safe and reliable operation. The inevitable battery degradation during its utilization, which may lead to potential system failures and safety issues, is often described as the battery state of health (SOH). The capacity and internal resistance are two most commonly used indicators that quantitatively characterize the SOH and assess the battery aging degree [1]. Among the two SOH indicators, the capacity is more intuitive to reflect the driving range that users care about [4]. Thus, a variety of approaches haven been reported to provide accurate capacity estimation and they can be roughly grouped into two categories, namely model-based and machine learning-based approaches.

In model-based battery capacity estimation approaches, physical models [5], empirical models [6], thermal models [7] and fusion models [8] are often used in conjunction with observers or adaptive filtering algorithms to achieve online capacity estimation. For example, Yu et al. [9] used the least squares algorithm to estimate the battery capacity based on the equivalent circuit model. Zheng et al. [10] estimated the capacity using proportional-integral observers based on an electrochemical model. Xiong et al. [11] extracted five parameters that are strongly correlated with battery aging from the electrochemical model to estimate the capacity. Zheng et al. [12] applied sequential extended Kalman filters to estimate capacity based on the empirical model. Different types of battery models were compared and analysed in [13]. And three widely used filtering algorithms for estimating the capacity, i.e. extended Kalman filter, particle filter and recursive least squares, were investigated in [14], and their performances are compared and analysed in terms of accuracy and convergence speed. However, the performance of these methods is highly dependent on the model quality, while to build an accurate battery model is difficult to achieve as it requires a large amount of priori knowledge and experimental data, which are not always available. Therefore, although model-based methods have merits of offering clear physical insights, their dependence on precise battery model limits their broad applications.

As an alternative approach, machine learning-based battery capacity estimation methods have attracted substantial interests in recent years. These methods do not require priori knowledge on the complex physical principles of the battery and are easy to implement. Methods such as Gaussian process regression (GPR) [15], kernel ridge regression [16], support vector machine (SVM) [17], and support vector regression [18], just to name a few, have been successfully applied in battery capacity estimation. Among these methods, Liu et al. [19] employed a GPR-based framework to simultaneously predict capacity and quantify the uncertainty of predicted values. Li et al. [20] proposed a GPR-based method to estimate the battery SOH using the health features extracted from partial incremental capacity curves. The accuracy, robustness and effectiveness of this approach are verified on four batteries from NASA battery degradation dataset. Feng et al. [21] developed a SVM-based online SOH estimation method. The SVM model is first established offline using features extracted from the battery charging curves of cells at different SOHs. Then the model is employed for online SOH estimation by comparing the features of the measured charging voltage segment and the stored models. Guo et al. [22] extracted health features from charging measurements, and eight features which are most relevant to the capacity degradation are selected as the inputs to a relevance vector machine (RVM) model. It is worth noting that the performance of these methods heavily relies on the manually extracted features, which requires significant manual and computational efforts, and the resultant model is often application specific and may not be generalized.

To address this bottleneck, some other machine learning-based techniques such as Elman neural network [23], long short-term memory (LSTM) [24], random forest [25] and Convolutional Neural Network (CNN) [26] have been applied to automatically extract features from the measurements. The Elman neural network was applied in [23] to estimate the battery capacities of the following cycles until its end-of-life in real-time, only using the charged capacities in the past cycles as inputs of the network. A LSTM network was designed in [24] for capacity estimation using direct measurable data, i.e. current and voltage, and the model's robustness and flexibility in dynamic environments has been extensively verified with data collected from more than seventy Lithium-ion batteries cycled with more than ten driving profiles. In [25], the charge capacity recorded in a specific voltage region was used as the input to a random forest regression model, the trained model can directly use online recorded data for capacity estimation. And [26] directly used the voltage, current and charge capacity of each cycle as inputs to train a CNN model for capacity estimation. The performance of three different neural network models for capacity estimation, i.e. feed-forward neural network, LSTM and CNN, were compared in [27], and the test results revealed the difficulty of the resultant models in dealing well with limited available battery data. It is clear that these machine learning-based methods have shown great potentials in battery capacity estimation, yet their performance is heavily dependent on the size of the training dataset. Only models trained with sufficient data are capable of achieving satisfactory accuracy. However, to collect a large battery degradation dataset requires a substantial number of cycling tests, which is extremely time-consuming and costly. To this end, transfer learning techniques can be incorporated into these methods to improve the estimation performance on small dataset. In [28], the battery health was estimated by combining the kernel ridge regression and transfer learning to improve the prediction accuracy. In [29], transfer learning was applied to achieve accurate, quick and steady prediction based on a LSTM model. A CNN model combining the concepts of transfer learning and ensemble learning was used for capacity estimation in [30] with voltage, current and charge capacity as the inputs of the network. Although the inputs to the model were extracted from a partial charge cycle, which can start from any partially discharged state, it must end up being fully charged. However, it does not always have the opportunity to fully charge/discharge an EV in practical applications [21]. Similar work has been reported in [31].

In view of the limited computational capability of the current battery management systems (BMSs) and the difficulty in collecting long-term battery aging data, though the CNN has shown to be a powerful tool to automatically extract features from direct measurements and estimate the capacity with satisfactory accuracy [32], its large model size and significant computational cost together with its limitations in handling small amounts of data hinder its practical applications. This paper aims to address two research questions in order to tackle the aforementioned challenges, 1) how to leverage the knowledge embedded in the existing often huge amount of battery data to predict the capacity degradation of the same type of batteries in new applications where available data are however quite limited; 2) how to significantly reduce the CNN model complexity such that it can be implemented in embedded systems like battery management systems (BMS) used in electric vehicles where memory storage is however quite limited. To answer these research questions, a pruned CNN (PCNN) in combination with transfer learning is developed for battery capacity estimation in this paper, which can greatly compress the model size and reduce the computational complexity, while improve the estimation accuracy on small aging dataset. In this proposed method, measurements from flexible partial charging curves with a fixed length of 225 data points and a flexible starting point are directly used as the model inputs and no additional feature extraction procedure is required. The main contributions of this paper are elaborated as follows:

- Firstly, a new CNN-based battery capacity estimation framework incorporating the concepts of transfer learning and network pruning is proposed. The transfer learning techniques are applied to improve the model performance on a small battery degradation dataset by leveraging the knowledge learned from a large but different dataset. While the network pruning is used to reduce the model size and computational cost, which makes it possible to implement the CNN model in on-board BMSs. The performance improvements of the proposed method in terms of estimation accuracy, model size and computation cost are evaluated on a small battery degradation dataset.

- Secondly, a novel fast recursive algorithm (FRA)-based network pruning approach is proposed to select the neurons one by one that contribute most to the model output and re-assign the weight for each selected neuron, while redundant neurons are removed. This lead to a compact CNN with much fewer neurons and parameters, while the accuracy loss due to nodes pruning is negligible, and the pruned CNN model even performs better than the unpruned one.

- Finally, the CNN model inputs are generated by a signal-to-image transformation and data segmentation method proposed in this paper, which is an enabling stage to fit the CNN model for time series signals and to estimate the battery capacity using only partial charging segment.

With these features, once the model parameters are updated offline using the small historical dataset collected from the target battery, the resultant model can then be used to estimate the unknown capacity of a battery using online measured variables during a partial charging cycle. The remainder of this paper is organized as follows. In Section 2, a brief introduction of the transfer learning, network pruning and FRA algorithm are presented. Section 3 details the proposed battery capacity estimation method, including the signal-to-image transformation method enabling the application of CNNs for time series signals, and construction of the CNN model with knowledge transfer and redundant neurons pruning. Section 4 introduces the experimental data and implementation details. Section 5 presents and discusses the experimental results. Finally, Section 6 concludes the paper.

2. Preliminary work

To improve the readability, this section presents the concepts of transfer learning, network pruning and fast recursive algorithm, which are the key elements of the proposed battery capacity estimation method.

2.1. Transfer learning

Most machine learning technologies are developed under the assumption that the data used for training and testing should be drawn from the same feature space and have the same distribution

[33]. In other words, the statistical model trained on a dataset can not be directly applied to another dataset with different distribution, and the model has to be reconstructed and retrained from scratch on the new dataset. However, collecting enough data to retrain a new model is often time-consuming and costly in real-world applications, and on the other hand, the training process often takes a long time. Transfer learning was proposed to handle such cases, aiming to reduce the need and effort for data recollection and model re-training while leveraging the knowledge learned from a source task to a different but related task though the data of these two tasks may have different distributions [34]. Based on the transfer learning, the latter task requires much less data to retrain the model.

Generally speaking, the transfer learning can be achieved in two ways for neural network models: one is to utilize the original pre-trained network except for its last fully-connected (FC) layer as a fixed feature extractor for the new task, and the other approach is to fine-tune the parameters of the pre-trained network using data of the new task. Choosing the suitable transfer learning technique mainly depends on the size and similarity of the target dataset to the original one used in pre-training [34]. It is widely accepted that the first several layers of a network are used to extract low-level features while the following few layers are used to extract high-level features [35], the FC layers are used to learn non-linear combinations of these features. Therefore, the first transfer learning technique is suitable for cases when the target dataset is quite similar to the source dataset. On the other hand, fine-tuning the pre-trained network is more suitable for a target dataset which has different distribution statistics from the original dataset.

When the CNN is applied for battery capacity estimation, it is obvious that the model trained on one battery degradation dataset can not be directly applied to another type of batteries with different specifications and charging/discharging policies. When the battery specification changes or the operation condition varies significantly, the CNN model needs to be retrained or even rebuilt from scratch on the new battery degradation data. To this end, this paper combines the CNN with transfer learning, aiming to eliminate the need to recollect a complete battery cycling data and build a completely new CNN model.

2.2. Network Pruning

When CNNs are used for battery capacity estimation, the deep and complex network structure and a large number of parameters to tune demand huge memory storage and high computational cost, which hinder its implementation in embedded devices [36]. To acquire a smaller and much more compact model, network pruning, which can remove redundant network connections based on a predefined criteria, has proven to be an effective approach. After pruning, the model will have much fewer parameters with little or no impact on performance, while the computational complexity and storage space can be significantly reduced. According to the granularity level, the pruning can be applied at four levels, namely weight-level, filter-level, channel-level and layer-level [37]. Though a number of methods have been proposed so far for network pruning at different granularity level, they are all based on the similar framework, that is, to evaluate the importance of each weight/ kernel/ channel/ layer and remove those insignificant ones. The core of these methods is to choose a suitable importance indicator that determines which element should be removed. Different criteria have been proposed in the literature, for example, the absolute values of weights were directly used as the importance indicator in [38], the second order derivatives of a layer-wise loss with respect to the corresponding weights were used to identified the importance of each weight in [39], while [40] calculated the percentage of zero activation of a neuron after the ReLU mapping to determine which neuron should be removed, based on the assumptions that neurons with higher percentage values provide less power to the results.

Considering that the CNN structure used in capacity estimation often has less number of filters and layers, weight-level and channel-level pruning are more suitable. Further, neuron pruning simplifies the network structure and also reduces the number of weights, therefore it is often more effective than the weight pruning approach only [41]. As a consequence, a channel-level pruning method is proposed in this paper to remove unimportant neurons, where each neuron is viewed as one channel. Considering that the FC layers are less sensitive to pruning than the convolutional layers [38], and the FC layers have the most number of parameters, we choose the FC layers as the pruning object. In this way, more parameters can be pruned with minimal impact on the output. The final compressed model will use much less memory and require less calculations, which makes is possible to implement the resultant model in the on-board BMS. The detailed pruning method will be elaborated in the following sections.

2.3. Fast recursive algorithm

In the field of system identification, the linear-in-the-parameter model is a popular model structure for approximating a large class of nonlinear systems. These models linearly combine a set of model terms that are nonlinear functions of the system variables. Such models often have an excessive number of candidate terms which may cause overfitting and high computational complexity, therefore, model selection algorithms have been proposed to generate parsimonious models with a much smaller number of terms. In particular, a fast recursive algorithm (FRA) [42] was proposed to simultaneously select most significant model terms and estimate model parameters.

Consider a nonlinear discrete-time dynamic system represented by a linear-in-the-parameters model, which is identified by N data samples $\{x(i), y(i)\}_{i=1}^{N}$

$$\mathbf{y} = \mathbf{\Psi}\mathbf{\Theta} + \mathbf{\Xi} \tag{1}$$

where $\mathbf{y} = [y(1), ..., y(N)]^T \in \Re^N$ denotes the system output, $\mathbf{\Psi} = [\varphi_1, ..., \varphi_j, ..., \varphi_S] \in \Re^{N\times S}$ is the regression matrix that contains all candidate model terms, each term $\varphi_j \in \Re^{N\times 1}, \varphi_j = [\varphi_j(x(1)), ..., \varphi_j(x(N))]^T (j = 1, ..., S)$ represents a nonlinear function of N input samples, $\mathbf{\Theta} = [\theta_1, ..., \theta_S]^T$ are the unknown parameters to be identified, and $\mathbf{\Xi} = [\varepsilon_1, ..., \varepsilon_N]^T$ is the model residual vector.

Two recursive matrices, namely information matrix $\mathbf{M}_k$ and residual matrix $\mathbf{R}_k$, are predefined in FRA to fulfill the forward model selection procedure as:

$$\mathbf{M}_k = \mathbf{\Psi}_k^T \mathbf{\Psi}_k \tag{2}$$

$$\mathbf{R}_k = \mathbf{I} - \mathbf{\Psi}_k \mathbf{M}_k^{-1} \mathbf{\Psi}_k^T \tag{3}$$

where $\mathbf{\Psi_k} \in \Re^{N\times k}$ contains the first k columns of the full regression matrix $\mathbf{\Psi}$, additionally, $k = 1, ..., S$, and $\mathbf{R}_0 = \mathbf{I}$. The distinguished properties of $\mathbf{R}_k$ are given in Appendix A.

Thus, when the first k columns in $\mathbf{\Psi}$ are select, the estimation of parameters that minimizes the cost function and the associated minimal cost function can be formulated as

$$\hat{\mathbf{\Theta}}_k = \mathbf{M}_k^{-1}\mathbf{\Psi}_k^T\mathbf{y} \tag{4}$$

$$E_k = \mathbf{y}^T\mathbf{y} - \hat{\mathbf{\Theta}}_k^T\mathbf{\Psi}_k^T\mathbf{y} = \mathbf{y}^T\mathbf{R}_k\mathbf{y} \tag{5}$$

To simplify the formulas and decrease the computational complexity, three quantities are consequently defined as

$$\begin{cases} \varphi_j^{(k)} \triangleq \mathbf{R}_k\varphi_j, \quad \varphi_j^{(0)} \triangleq \mathbf{R}_0\varphi_j = \varphi_j \\ a_{k,j} \triangleq \left(\varphi_k^{(k-1)}\right)^T \varphi_j^{(k-1)}, \quad a_{1,j} \triangleq \varphi_1^T\varphi_j \\ b_k \triangleq \left(\varphi_k^{(k-1)}\right)^T \mathbf{y}, \quad b_1 \triangleq \left(\varphi_1^{(0)}\right)^T \mathbf{y} = \varphi_1^T\mathbf{y} \end{cases} \tag{6}$$

where $j = 1, ..., S$, and $k = 0, 1, ..., S$. According to the properties of $\mathbf{R}_k$, the net contribution of a new model term φ_{k+1} to the cost function can be explicitly calculated as

$$\Delta E_{k+1} = \frac{\left(\mathbf{y}^T\varphi_{k+1}^{(k)}\right)^2}{\left(\left(\varphi_{k+1}^{(k)}\right)^T \varphi_{k+1}^{(k)}\right)} = \frac{\left(b_{k+1}^T\right)^2}{a_{k+1,k+1}}, \quad k = 0, 1, ..., S-1 \tag{7}$$

By calculating the net contribution of each term, the model terms with maximum contributions will be selected one by one. Finally, after all important model terms have been selected, the parameter for each selected term is calculated as

$$\hat{\theta}_j = \frac{b_j - \sum_{i=j+1}^{k}\hat{\theta}_i a_{j,i}}{a_{j,j}}, \quad j = k, k-1, ..., 1 \tag{8}$$

Note: the summation term in Equation (8) is zero for $j = k$, i.e. $\sum_{i=k+1}^{k}\hat{\theta}_i a_{k,i} = 0$. Equations (7) and (8) constitute the main steps of the FRA, which selects model terms one by one based on (7) and calculates the model parameters for the resultant model based on (8).

3. Methodology

In this section, the proposed battery capacity estimation method is described in detail. First, the input generation method including data normalization, segmentation and time series-to-image conversion is introduced, and the target output of the model is clarified. Then the construction of the proposed CNN model is described step by step, which incorporates both the transfer learning and network pruning.

3.1. Input Generation and Model Output

The input variables used for battery capacity estimation in this paper are current, terminal voltage and charge capacity, where the charge capacity is calculated by integrating the current with respect to time for a partial charging segment. These variables are highly correlated, embedding lots of information. To make the most of the embedded information in these measurements and to automatically extract features, a CNN model is built and a time series-to-image conversion method is proposed to convert the measured time series signals to 3 dimensional images, as the CNN usually takes images as the input. Given that there is little chance to fully charge/discharge a battery from a fixed initial state in practical applications [21], partial charging curves based capacity estimation methods are more practical. Therefore, data segmentation is performed before the conversion stage to generate partial charging curves with flexible start/end point.
Figure 1 illustrates the steps of the proposed input generation method. Note that the raw data, i.e. current, voltage and charge capacity, have different scales, therefore directly converting these variables to 3D inputs will slow the training process and significantly degrade the model performance. Thus, data normalization is applied first. Then M data chunks are segmented from one charge cycle, each data chunk is a $n \times 3$ matrix that represents a partial charging segment, where 3 is the number of involved variables, and n denotes n continuous data points for each variable. Two adjacent data chunks have c overlapping data points. Finally, each partial charging segment is converted into a 3 dimensional image with the size of $\sqrt{n} \times \sqrt{n} \times 3$.
This input generation method can increase the number of samples used for CNN training, and allow fast online capacity estimation only using flexible partial charging curves, thus it is an enabling block in order to apply the CNNs for time series signals. Each input sample is associated with an output sample, which is the full discharge capacity that is calculated by integrating the discharge current over time for the entire full discharge cycle which immediately follows the charge cycle that generated the input sample.

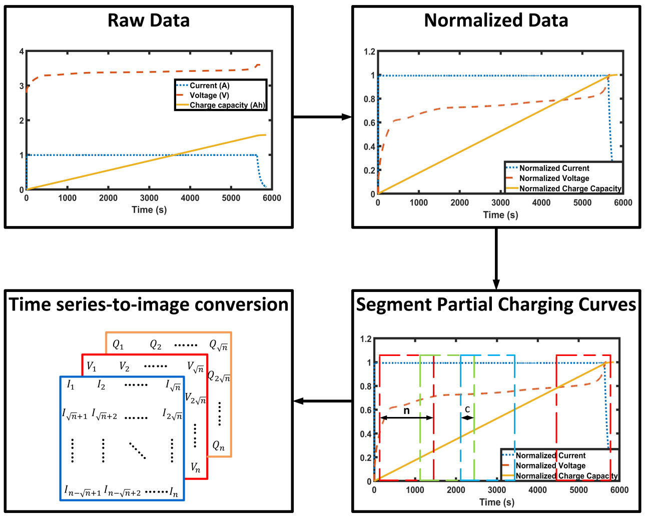


Figure 1: Input generation steps: data normalization, charging curves segmentation, and time series-to-image conversion

3.2. *Model Construction*

As shown in Figure 2, the CNN model proposed in this paper is constructed in three consecutive steps. Firstly, a CNN architecture is designed based on a classical and effective CNN structure named LeNet-5, and the CNN model is pre-trained using the source dataset. Secondly, the knowledge learned from the source dataset is transferred to a new task, and the trainable parameters are fine-tuned using the target dataset to produce the target model. Finally, a pruned CNN model is constructed by pruning insignificant neurons of the target model using the proposed pruning algorithm. The detailed process of each stage will be further elaborated below.

3.2.1. *Pre-training*

The designed CNN architecture is illustrated in Figure 3 and Table 1, and it has 4 convolutional layers, 2 max pooling layers, a flatten layer and 2 fully-connected layers. As aforementioned, the inputs are $\sqrt{n} \times \sqrt{n} \times 3$ matrices, their weight and height dimensions both are $\sqrt{n}$, which are determined by the length n of the data chunk, and 3 is the depth of the input, which is determined by the number of variables. In this paper, n is set to be 225, thus the size of the input is $15 \times 15 \times 3$. The output size of each layer is given at the top of each layer. $w \times h \times c$ refers to the size of the feature maps, where w, h, c denote the weight, height and channel, respectively. For each convolutional layer, $N@k \times m \times d$ denotes the filter design, which implies that N filters

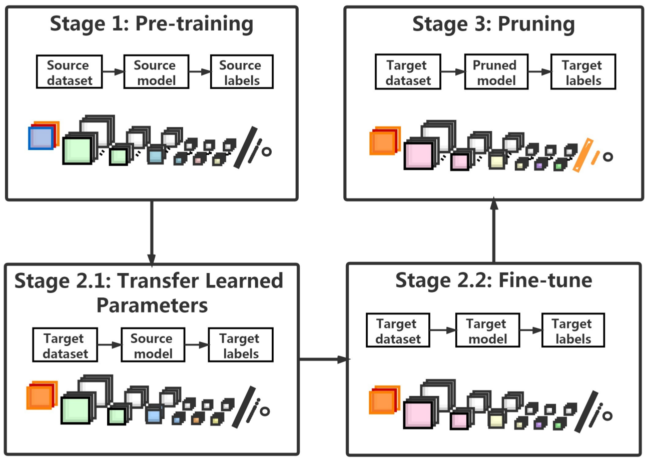


Figure 2: Model construction stages: (1) pre-training, (2) transfer learned parameters and fine-tuning, (3) pruning

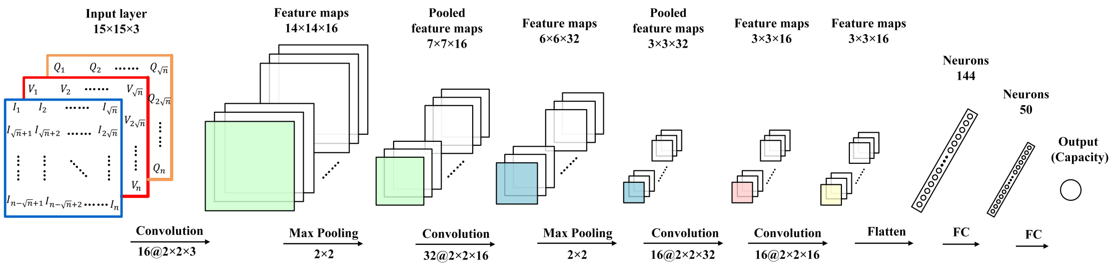


Figure 3: Model structure of the pre-trained CNN model

with the size of $k \times m \times d$ are convolved with the input of this layer. The depth of the input is d, and the number of the output feature maps must be N. The model pre-trained on the source dataset is denoted as CNN(S), and there are totally 12693 parameters in this model, which are trained using the source dataset.

Table 1: Layer configurations and number of parameters of the pre-trained CNN model

Layer	Layer name	Stride	Output size	Parameters	Activation
L1	Conv_1(16@2 × 2 × 3)	(1,1)	(14 × 14 × 16)	208	ReLU
L2	Maxpool_1(2 × 2)	(2,2)	(7 × 7 × 16)	0	ReLU
L3	Conv_2(32@2 × 2 × 16)	(1,1)	(6 × 6 × 32)	2080	ReLU
L4	Maxpool_2(2 × 2)	(2,2)	(3 × 3 × 32)	0	ReLU
L5	Conv_3(16@2 × 2 × 32)	(1,1)	(3 × 3 × 16)	2064	ReLU
L6	Conv_4(16@2 × 2 × 16)	(1,1)	(3 × 3 × 16)	1040	ReLU
L7	Flatten	-	144	0	ReLU
L8	FC1	-	50	7250	ReLU
L9	FC2	-	1	51	ReLU

The source dataset was split into three sets: training, validation and testing. The CNN model is trained from scratch using the training and validation sets, and then tested on the testing set. All the 12693 parameters are trained and then used in the following steps.

3.2.2. Transfer learning and fine-tuning

In real-life applications, it is not a trivial task to collect and annotate battery long-term cycling data. While the CNN architectures used for battery capacity estimation usually contain tens of thousands of free parameters to train, requiring a large number of labeled training data and long training time. To circumvent this problem for practical applications, transfer learning is adopted to transfer the knowledge learned from the source dataset to a new task with a much smaller dataset. In this paper, transfer learning is achieved through the following two steps (Fig. 2): first, the structure and parameters of the CNN model pre-trained on the source dataset are transferred to the target dataset, then specific layers are fine-tuned using the target dataset. The resultant CNN model after the transfer learning is denoted as CNN(S)-TL in the remainder of the paper.

3.2.3. Pruning

As aforementioned, the CNN(S)-TL model used for capacity estimation fine-tuned using the target dataset has 4 convolutional layers, 2 max pooling layers, a flatten layer and 2 fully-connected layers with a total of 12693 parameters. In real-world applications, such a CNN model will require large computation and memory resources, which limits its implementation in on-board BMS. An intuitive way to solve this problem is to reduce the model complexity and the number of parameters. Inspired by the work presented in [43] that selects the best hidden neurons and avoids redundant structure, a FRA-based network pruning method is proposed in this paper to remove all redundant neurons based on measuring the contributions of all neurons to the outputs. Compared with other pruning methods, the proposed method identifies and preserves the most important neurons of networks and re-assign weights to the retained neurons. In other words, the important features are retained, which may help to improve the accuracy of capacity estimation for the CNN model refined on a rather small task dataset. Further, the ability of the proposed method to simultaneously update weights and select neurons implies that there is no need to fine-tune the network after pruning. As shown in Table 1, more than half of the total parameters of the CNN model are from the last two fully-connected layers, the proposed method is therefore deployed to these two layers. The redundant neurons and all the incoming and outgoing connections associated with these neurons are removed, leading to significantly reduced memory usage and computational complexity for online capacity estimation.

Supposing $X \in \Re^{N\times(L+1)}$ is the input matrix to the first fully-connected (FC1) layer, where N is the number of samples and L is the length of the vector formed in the flatten layer and inputted to the FC1 layer, and the input matrix X includes a bias vector with all elements being set to 1. $Y \in \Re^{N\times 1}$ is the output of the second fully-connected (FC2) layer, i.e. the output of the CNN(S)-TL model, which can be written as

$$\mathbf{Y} = f(\mathbf{X\Theta})\mathbf{W} + \mathbf{\Xi} \tag{9}$$

where $\Theta \in \Re^{(L+1)\times S_1}$ represents the parameter matrix of the FC1 layer, which consists of the $L \times S_1$ weight matrix and $1 \times S_1$ bias vector. S_1 is the number of neurons inputted to the FC2 layer, namely, the number of latent features obtained by the CNN-TL model to estimate capacity. W denotes the parameter matrix of weights and bias of the FC2 layer, and Ξ is the residual vector.

To remove redundant neurons in FC1 and FC2 layers without sacrificing the overall performance of the network, the detailed procedure is illustrated in Figure 4, and the neuron selection process is elaborated as follows.

Take the FC1 as an example, the pruning procedure consists of 3 steps:

Step 1 - Calculate the contribution of each neuron in the FC1 layer. Let $\mathbf{Y}^f = f(\mathbf{X\Theta})$, $\mathbf{Y}^f = [\mathbf{y}_1^f, \mathbf{y}_2^f, ..., \mathbf{y}_N^f]^T \in \Re^{N\times S_1}$ is the output of the FC1 layer. According to Equation (7), the net contribution of each neuron in this layer can be calculated as

$$\Delta E_{k+1} = \frac{\left(\left(\mathbf{Y}^f\right)^T x_{k+1}^{(k)}\right)^2}{\left(\left(x_{k+1}^{(k)}\right)^T x_{k+1}^{(k)}\right)}, \quad k = 0, 1, ..., L-1 \tag{10}$$

Step 2 - Rank the neurons based on their contributions, and then select the highest ranked neuron. With the selected neurons, a new output of the FC1 layer is obtained, denoted by $\hat{\mathbf{Y}}^f$.

The root mean square error between $\mathbf{Y}^f$ and $\hat{\mathbf{Y}}^f$ over all the training samples is given as

$$\begin{aligned} rmse^f &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left((\hat{\mathbf{y}}_i^f - \mathbf{y}_i^f)(\hat{\mathbf{y}}_i^f - \mathbf{y}_i^f)^T\right)} \\ &= \frac{1}{\sqrt{N}}\left\|\hat{\mathbf{Y}}^f - \mathbf{Y}^f\right\|_F \end{aligned} \tag{11}$$

where $\|\cdot\|_F$ is the Euclidean norm.

Step 3 - Repeat Step 2 until the maximum number of neurons in the FC1 layer is reached or the $rmse^f$ is smaller than the predefined error bound. The parameters for the selected terms are calculated by

$$\hat{\theta}_j = \frac{\left(x_j^{(j-1)}\right)^T \mathbf{Y}^f - \sum_{i=j+1}^{k} \hat{\theta}_i \left(x_j^{(j-1)}\right)^T x_i^{(j-1)}}{\left(x_j^{(j-1)}\right)^T x_j^{(j-1)}} \tag{12}$$

where $j = k, k-1, ..., 1$, and k is the total number of the selected terms.

The neuron pruning process for FC2 is the same as for the FC1 layer. Repeat step 1-3, and until the equivalent or even better result with fewer neurons is achieved.

The proposed method using FRA for neuron pruning can simultaneously reduce the model size and re-tune the weights, which eliminates the need to fine-tune the network after neuron pruning, and reduces the number of computing operations without sacrificing the accuracy. After pruning, the new network is more compact than the original CNN(S)-TL model in terms of the model size, and hence the computational complexity is significantly reduced. The resultant model is denoted as pruned CNN(S)-TL (PCNN(S)-TL) in the rest of the paper.

4. Datasets and Implementation Details

In the following, the source dataset used to pre-train the CNN model and the target dataset used to validate the performance of the proposed PCNN(S)-TL model are introduced. Then the detailed parameters for training, transfer learning and pruning a CNN-based battery capacity estimation model are given, and the criteria for measuring model performance are determined.

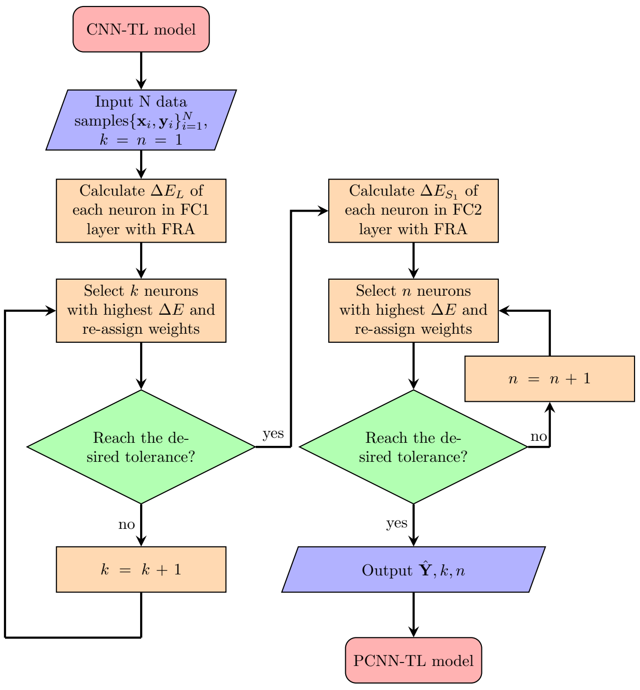

CNN-TL model
Input N data samples$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$, $k = n = 1$
Calculate ΔE_L of each neuron in FC1 layer with FRA
Select k neurons with highest ΔE and re-assign weights
Reach the desired tolerance?
no
$k = k + 1$
yes
Calculate ΔE_{S_1} of each neuron in FC2 layer with FRA
Select n neurons with highest ΔE and re-assign weights
$n = n + 1$
Reach the desired tolerance?
no
yes
Output $\hat{\mathbf{Y}}, k, n$
PCNN-TL model

Figure 4: Flowchart of the FRA-based neuron pruning process

4.1. Datasets

The detailed information of the source and target battery degradation datasets are summarized in Table 2. These include one large-scale test dataset for 16 cells and one small-scale dataset for 4 cells, and each cell in the first dataset has roughly gone through 1000 charging/discharging cycles, while cells in the small dataset were only tested for 30 reference cycles. Therefore, the large-scale dataset is employed as the source dataset to pre-train the CNN model, and then the learned knowledge will be transferred to the small-scale dataset. Note that these two sets of cells are all the same type, but have different specifications and are tested under different cycling scenarios.

4.1.1. Source dataset

The source dataset is collected from 124 commercial lithium iron phosphate (LFP)/graphite batteries that are manufactured by A123 System (APR18650M1A), with a nominal voltage of 3.3 V and nominal capacity of 1.1 Ah [44]. They are cycled to failure under fast-charging policy at a constant temperature of 30°C. Under the fast-charging policy, denoted as "C1(Q1)-C2",

Table 2: Summary of the two battery degradation datasets

	Source dataset	Target dataset
Available cycles for each cell	roughly 1000 charge/discharge cycles	30 reference charge/discharge cycles
Number of cells	16	4
Nominal capacity	1.1 Ah	1.6 Ah
Nominal voltage	3.3 V	3.2 V
Battery type	$LiFePO_4$	$LiFePO_4$

the cell is first charged from 0% to Q1 state of charge (SOC) at a constant current (CC) C1, then charged from Q1 to 80% SOC at constant current C2. After reaching 80% SOC, all cells are charged at 1C until the voltage reaches its upper cutoff potential 3.6 V. Finally a constant voltage (CV) mode continues until the charge current falls to 22 mA. The whole charging policy is illustrated in Figure 5. All cells are discharged under a CC-CV protocol, discharging at CC of 4C until the cell voltage falls to 2.0 V with a current cutoff of 22 mA.

The first 16 cells in dataset 'batch3' are used in this work and they are charged with different

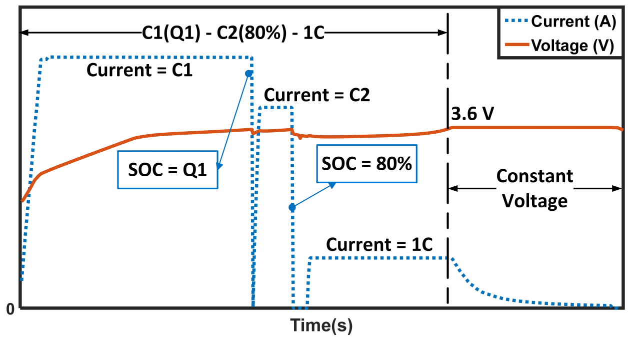


Figure 5: Charging policy of the source dataset

policies. These 16 cells are divided into 2 groups, where the first group consists of 12 cells and is used to train the network, while the second group consists of the remaining 4 cells and is used as the testing set. The detailed policies applied to charge these 16 cells from 0% to 80% SOC are summarized in Table 3, and the groups used for training and testing are also given in the table. During the pre-training process, samples generated from the Group 1 are first shuffled and randomly split into a training set and a validation set with the ratio of 7:3, which are then used to pre-train the CNN model. The Group 2 is then used to test the performance of the trained CNN model.

Table 3: Summary of the policies for charging the cells from 0% to 80% SOC

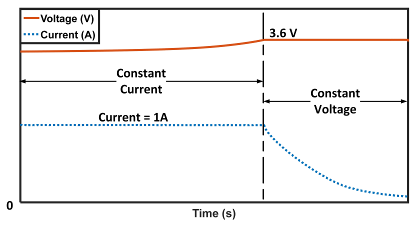
	Battery	Charging policies	Cycles
	1	'5C(67%)-4C'	1008
	2	'5.3C(54%)-4C'	1062
	3	'5.6C(19%)-4.6C'	1266
	4	'5.6C(36%)-4.3C'	1114
	5	'5.6C(19%)-4.6C'	1047
Group 1	6	'5.6C(36%)-4.3C'	827
(Training set)	7	'3.7C(31%)-5.9C'	666
	8	'4.8C(80%)-4.8C'	1835
	9	'5C(67%)-4C'	827
	10	'5.3C(54%)-4C'	1038
	11	'4.8C(80%)-4.8C'	1077
	12	'5.6C(19%)-4.6C'	816
	13	'5.6C(36%)-4.3C'	931
Group 2	14	'5.6C(19%)-4.6C'	815
(Testing Set)	15	'5.6C(36%)-4.3C'	857
	16	'5.9C(15%)-4.6C'	875

4.1.2. Target dataset

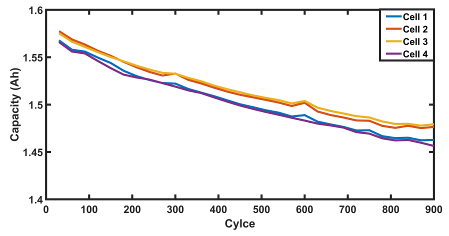
To validate the proposed PCNN(S)-TL based capacity estimation algorithm, data of 4 commercial cylindrical LFP cells with nominal voltage of 3.2 V and nominal capacity of 1.6 Ah are used as target dataset in this work. These cells used in the target dataset are different from the cells used in the source dataset as shown in Table 2, and they are tested in parallel using a BTS 4000 battery test system made by NEWARE. Thermocouples with measurement error less than 0.1 °C are attached to measure the cell surface temperature.

High charging current rates are used to accelerate the aging speed of these cells, and a reference cycle is tested under the temperature of 25 °C every 30 cycles. The protocol of the reference cycle recommended by the manufacturer includes a CC-CV charging and a CC discharging process. As shown in Figure 6(a), under the CC-CV charging policy, the cells were charged at a uniform CC of 1 A until the voltage reaches the upper cutoff voltage 3.6 V, and then charged with 3.6V CV to a current cutoff of 75mA. All cells were subsequently discharged with a 1 A CC to the lower cutoff voltage of 2.0 V. The sampling frequency for all the equipment used in this experiment was set as 1Hz. The resting time between the charge and discharge process was set to 1 hour.

The CNN(S)-TL model is fine-tuned using the data collected from the reference cycles of the 3 cells, while the data from the remaining cell is used for testing. The capacity degradation trends for these 4 cells are shown in Figure 6(b).



(a) CC-CV charging strategy



(b) Capacity degradation trends for cell 1 - cell 4

Figure 6: Charging strategy (a) and actual capacity degradation (b) of the four cells in target dataset

4.2. Implementation Details

4.2.1. Input structure

Training samples generation process is illustrated in Figure 7, and M_i partial charging segments with the length of 225 consecutive data points were extracted from the i th charging cycle, $i = 1, 2, ..., N_t$, where N_t is the total number of charging cycles involved in the training dataset. After a series of trial-and-error tests, it is found that the length of 225 data points is the best trade-off between the number of generated training samples and the information embedded in each sample. Each segment was formulated to a $15 \times 15 \times 3$ matrix. This leads to $\sum_{i=1}^{N_t} M_i$ data segments fed into the network to train the model. For a given overlap size c, the number of samples generated from the i th charge cycle containing L_i data points in total is calculated as follows

$$M_i = floor(\frac{L_i - 225}{225 - c}) + 1 \tag{13}$$

where the function $floor(\cdot)$ gives the largest integer less than or equal to the input value. The overlap size was set as $c = floor(225 \times 0.8)$ in this work.

For the test dataset, only one partial charging segment was extracted per cycle, and this segment has a fixed length of 225 data points and a random starting point. The training data of the source and target datasets are used to pre-train the CNN(S) and fine-tune the CNN(S)-TL model, respectively, and they are also used to prune the input neurons of the fully-connected layers. The testing data from the source dataset is used to verify the effectiveness of the CNN-based capacity estimation method on cells with different charging policies, while the testing data of the target dataset is used to verify the effectiveness of the transfer learning strategy. Further, the testing data of the target dataset are also used to validate the performance of the pruned model.

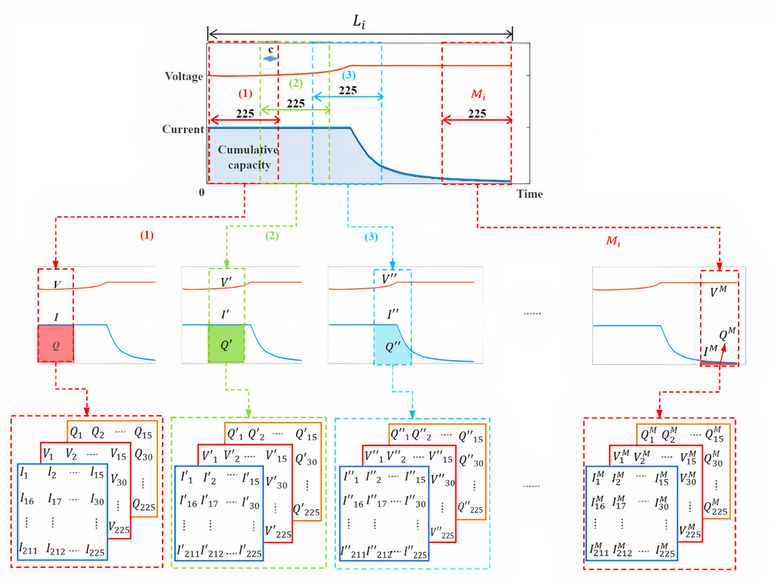


Figure 7: Illustration of training samples generation. M_i samples are segmented from one charge cycle, each sample refers to a partial charge cycle with a length of 225 data points. Three variables, i.e. voltage, current and cumulative charge capacity, of each partial charge segment are then together form a $15 \times 15 \times 3$ input matrix. Note that the charge capacity is calculated by integrating the current with respect to time for a partial charging segment.

4.2.2. Training Protocols

Important hyperparameters are pre-defined before training and fine-tuning the model to maximize the performance of the model. When the CNN model is trained from scratch, all the kernels are initialized based on Xavier uniform initializer and the bias are initialized to zero [45]. The maximum training epochs are set to 80 with the mini-batch size of 128 samples. To avoid overfitting problem, early stopping method with patience set to 5 is applied to stop training once the model performance has not improved for 5 consecutive epochs on the validation dataset. The Adam algorithm with learning rate of 0.001 is used to update the parameters.

The CNN(S)-TL model is fine-tuned from the pre-trained CNN(S) model on the target dataset. First, all layers of the pre-trained CNN(S) are copied to the CNN(S)-TL model. Considering that the target dataset is much smaller than the source dataset, and the features extracted from the first few layers are universal, the parameters of layers prior to the third convolutional layers are retained unchanged. Then the subsequent layers are fine-tuned to learn the specific features on target dataset with a learning rate of 0.0001, which is ten times smaller than the one used in the scratch training.

In the pruning stage, all the layers copied from the CNN(S)-TL model are fixed except for the last two fully-connected layers. When the desired performance of the model is achieved with the minimum number of neurons using the proposed FRA-based network pruning method, it eliminates the necessity of fine-tuning the network after removing redundant neurons, due to that new weights have already been assigned to remaining neurons using the proposed algorithm.

4.2.3. Evaluation Criteria

In the experiment, the total mean absolute error (MAE), root mean-square error (RMSE) and normalized estimation error (NEE) are used to assess the model performance. They are effective measures to assess the deviations in distances between the estimated capacities $\hat{\mathbf{Y}}$ and the reference values $\mathbf{Y}$. The formulas are defined as follows:

$$E_{mae} = \frac{1}{N}\left\|\hat{\mathbf{Y}} - \mathbf{Y}\right\|_1 \tag{14}$$

$$E_{rmse} = \frac{1}{\sqrt{N}}\left\|\hat{\mathbf{Y}} - \mathbf{Y}\right\|_F \tag{15}$$

$$E_{nee} = \frac{1}{\sqrt{N}\mathbf{Q}}\left\|\hat{\mathbf{Y}} - \mathbf{Y}\right\|_F \times 100\% \tag{16}$$

where N is the sample size of test data, Q is the nominal capacity. $\|\cdot\|_1$ and $\|\cdot\|_F$ are the L1 norm and Euclidean norm, respectively. $\mathbf{Y} = \left[\{y_i\}_{i=1}^N\right]^T$ and $\hat{\mathbf{Y}} = \left[\{\hat{y}_i\}_{i=1}^N\right]^T \in \Re^{N\times 1}$, y_i is the reference capacity for the i th sample, while $\hat{y}_i$ is the corresponding estimated value.

In addition, the floating point operations (FLOPs) are considered to measure the computational cost in this study. As the pruning algorithm is applied to the fully-connected layers in this study, only the number of FLOPs for a fully-connected layer is considered, which is calculated as $FLOPs_{fc} = (2\times I)\times O$, where I and O are the number of input and output neurons, respectively.

5. Results and Discussions

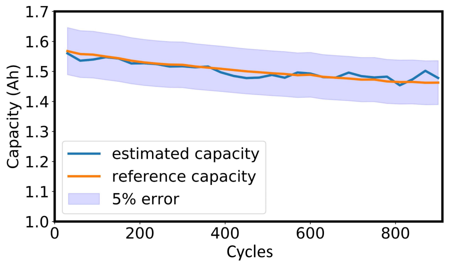
In this section, the datasets introduced in section 4 are used to validate the performance of the proposed method. The capacity estimation results of the proposed method on different cells are given. The performance of the proposed model is compared with the CNN model, CNN with network pruning, and CNN with transfer learning. The performance improvements of the proposed framework in terms of accuracy and model size are analysed. Moreover, the effect of the number of layers fine-tuned during the transfer learning stage is investigated to explain the reasons why we choose to fine-tune 4 layers.

5.1. Capacity estimation results

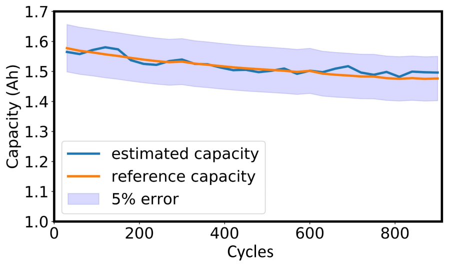
The capacity estimation results of the proposed PCNN(S)-TL model on target dataset are shown in Figure 8 and Figure 9. It can be observed from Figure 8 that the proposed model can produce accurate estimation on the test cells, and the estimated capacity can well track the degradation trend. Here the four-fold cross validation approach is used, when one cell is used for testing, data of the other three cells are used to generate the training samples according to the input generation method introduced in Section 3.1 to fine-tune and prune the pre-trained CNN model. The experimental results have revealed that the parameters of the developed model can be updated using the data from one cell, and then the resultant model can be directly applied to other cells who have the same specifications and similar charging policies. Figure 9 illustrates that updating the proposed model parameters using the data of the first several cycles, the degradation trend for the following cycles can be well captured. Here data of the first 2/3 cycles of all cells in the target dataset are used to fine-tune and prune the model pre-trained on the source dataset, and the remaining 1/3 cycles are used for testing. Both figures demonstrate the performance of the PCNN(S)-TL model for battery capacity estimation.

5.2. Algorithm verification

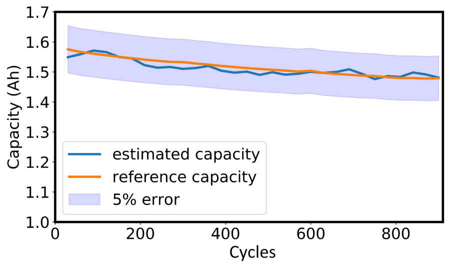
The proposed battery capacity estimation method has been further verified in this section. The CNN model directly trained using the target dataset is denoted as CNN(T), while CNN(S) refers to the model trained using the source dataset. The models acquired using network pruning, transfer learning and both of them are denoted as PCNN(T), CNN(S)-TL and PCNN(S)-TL,



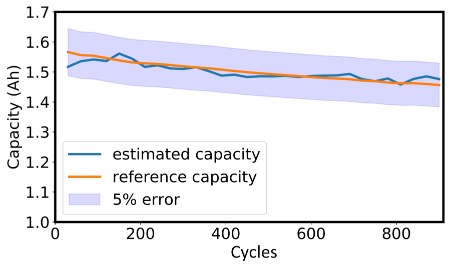
(a) Cell 1



(b) Cell 2



(c) Cell 3

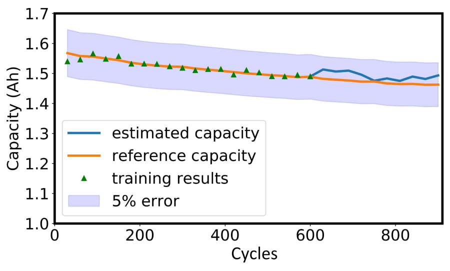


(d) Cell 4

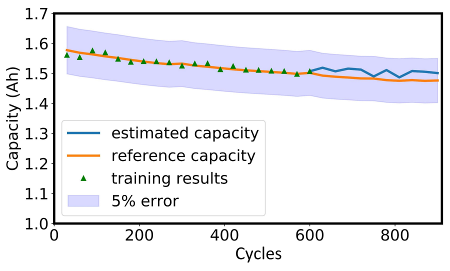
Figure 8: Capacity estimation results produced by the PCNN(S)-TL model for the four cells in target dataset. For testing the performance on each cell, data of other three cells are used to generate training samples and fine-tune and prune the model.

respectively. During the knowledge transfer stage, the first 4 layers (2 convolutional layers and 2 max pooling layers) are fixed, and the remainder layers are fine-tuned. The values of the aforementioned evaluation criteria for these models are averaged based on 100 repetitions of the experiments, and the test results of 4 cells are summarized in Table 4 in terms of MAE, RMSE and NEE. Further, the values of the model size, total number of parameters (of the entire model), and pruned FLOPs (of the last two fully-connected layers) of the CNN(T), PCNN(T), CNN(S)-TL and PCNN(S)-TL models are averaged for the 4 cells, which are listed in Table 5. The results presented in these tables confirm the effectiveness of the proposed capacity estimation methods.

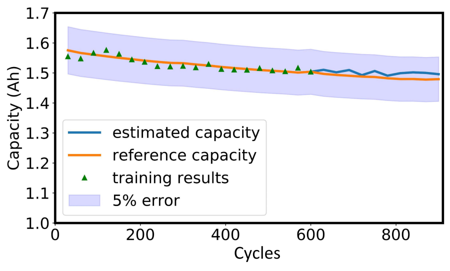
Table 4 reveals that, firstly, the proposed network pruning method does not degrade the estimation accuracy. The average NEE of PCNN(T) is 1.00%, which is 9.91% smaller than that of the CNN(T) model. The NEE of PCNN(S)-TL is 2.33% smaller than that of the non-pruned CNN(S)-TL model. Secondly, due to the transfer learning technique, the estimation performance



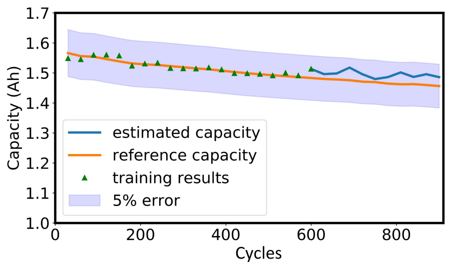
(a) Cell 1



(b) Cell 2



(c) Cell 3



(d) Cell 4

Figure 9: Capacity estimation results produced by the PCNN(S)-TL model for the four cells in target dataset. For testing the performance on each cell, the first 2/3 cycles of all cells in the target dataset are used to generate training samples and fine-tune and prune the model, the remaining 1/3 cycles of each cell are used for testing.

of CNN(S)-TL is better than CNN(T), and the NEE is decreased by 22.52% (from 1.11% to 0.86%). Thirdly, the PCNN(S)-TL model achieves the best result, of which the NEE is 24.32% smaller than the CNN(T) model directly trained on the target dataset.

More statistics are listed in Table 5. Compared with the non-pruned network, the model size of the PCNN(T) and PCNN(S)-TL are dramatically reduced, being only 27.64% and 31.66% of the original model size, leading to the size reduction of 72.36% and 68.34% respectively. By removing the redundant neurons in the fully-connected layers, almost 50% of the parameters of the entire network are removed and more than 80% of the FLOPs in the last two fully-connected layers are reduced, which makes the network more compact and reduces the computational cost. Comparing the PCNN(T) with the PCNN(S)-TL model, it can be found that the model size and the number of parameters of the PCNN(S)-TL model are slightly higher than that of the PCNN(T) model. Since the CNN(S)-TL model was trained on a larger dataset, the specific features extracted in the fully-connected layers for the capacity estimation are expected to be much richer than that

Table 4: Capacity estimation results produced by CNN(T), PCNN(T), CNN(S)-TL, and PCNN(S)-TL on target dataset

Cell No.	Assess	CNN(T)	PCNN(T)	CNN(S)-TL	PCNN(S)-TL
Cell1	MAE (Ah)	0.0137	0.0122	0.0112	0.0109
	RMSE (Ah)	0.0170	0.0152	0.0135	0.0132
	NEE (%)	1.06	0.95	0.84	0.83
Cell2	MAE (Ah)	0.0127	0.0111	0.0114	0.0104
	RMSE (Ah)	0.0164	0.0145	0.0141	0.0130
	NEE (%)	1.03	0.91	0.88	0.81
Cell3	MAE(Ah)	0.0129	0.0130	0.0114	0.0112
	RMSE (Ah)	0.0175	0.0174	0.0139	0.0135
	NEE (%)	1.09	1.09	0.87	0.84
Cell4	MAE (Ah)	0.0170	0.0133	0.0113	0.0111
	RMSE (Ah)	0.0199	0.0167	0.0138	0.0137
	NEE (%)	1.24	1.04	0.86	0.86
Average	MAE (Ah)	0.0137	0.0120	0.0113	0.0109
	RMSE (Ah)	0.0177	0.0160	0.0138	0.0134
	NEE (%)	1.11	1.00	0.86	0.84

Table 5: The model size, total parameters and FLOPs of the CNN(T), CNN(S)-TL, PCNN(T) and PCNN(S)-TL models.

Model	Model size (KB)	Reduced size (%)	Parameter numbers	Parameters pruned (%)	Pruned FLOPs (%)
CNN(T)/CNN(S)-TL	199	-	12693	-	-
PCNN(T)	55	72.36	5711	55.01	95.70
PCNN(S)-TL	63	68.34	6793	46.48	80.97

of the CNN(T) model. That is to say, the CNN(S)-TL model requires to retain more neurons to capture these features during the network pruning process. Consequently, the PCNN(S)-TL model will be slightly bigger than the PCNN(T) model.

In summary, the CNN model with transfer learning and network pruning techniques can achieve more accurate estimation results with much less neurons than a model trained on the target dataset from scratch.

5.3. Factors affecting the model performance

To investigate the influence of the number of layers fine-tuned during the transfer learning stage, the identical training and testing datasets were utilized for all tests. In each test, the entire structure and parameters of the CNN(S) model were copied first, and the last n layers were fine-tuned on the target sets while the first $6-n$ layers were fixed. It should be noted that, the whole CNN has 9 layers in total, the two max pooling layers added after the first two convolutional layers and the flatten layer after the last convolutional layer do not have parameters, therefore, maximally six layers can be fine-tuned. Thus, n refers to the number of tunable layers, and was selected from 1 to 6, where 1 implies that all layers except for the last FC layer are fixed, and 6 implies that the entire network needs to be fine-tuned. A complete fine-tuning and testing procedure was executed 100 times for each selected n, and the averaged max error (MaxE), MAE, RMSE, NEE of 100 runs are summarized in Table 6, where FC stands for the fully-connected layer, 2FC refers to the last two fully-connected layers, Conv represents the convolutional layer, and the number prefixing to Conv stands for the number of convolutional layers involved in fine-tuning from back to front.
As shown in Table 6, the estimation error decreases as the number of fine-tuned layers increases. When only fully-connected layers are fine-tuned, the NEEs are all greater than 1.3%, while if one or more convolutional layers are fine-tuned, the NEE are decreased by at least 30.88% (decreasing from1.36% to 0.94%). This improvement implies that for batteries with different specifications and subject to different charge/discharge policies, they have different high-level specific features projected at the last few convolution layers. The specific high-level features learned from the source dataset can not precisely describe the target dataset, thus they need to be learned using the target dataset.

Further, Figure 10 shows the number of neurons selected in FC1 and FC2 layer of the PCNN(S)-TL model during the network pruning process for different number of fine-tuned layers n. Two observations can be concluded from this bar chart. Firstly, the number of neurons selected in FC2 layer almost does not change with n. Secondly, for the FC1 layer, as the number of fine-tuned layers increases, the number of selected neurons first increases until it reaches the maximum value at $n = 3$, and then gradually converges. Only relatively fewer neurons are selected in FC1 layer when only fine-tune the last one or two layers ($n = 1, 2$), while the maximum

Table 6: Comparison of the model performance on battery capacity estimation with different number of fine-tuned layers.

Assess	Fine-tuned layers (from back to front)	FC n=1	2FC n=2	2FC + Conv n=3	2FC + 2Conv n=4	2FC + 3Conv n=5	2FC + 4Conv n=6
MaxE (Ah)	fine-tune	0.0644	0.0475	0.0377	0.0292	0.0277	0.0275
	fine-tune & pruning	0.0578	0.0478	0.0395	0.0294	0.0273	0.0265
MAE (Ah)	fine-tune	0.0271	0.0177	0.0120	0.0112	0.0112	0.0113
	fine-tune & pruning	0.0262	0.0182	0.0121	0.0109	0.0113	0.0109
RMSE (Ah)	fine-tune	0.0321	0.0218	0.0151	0.0135	0.0132	0.0135
	fine-tune & pruning	0.0307	0.0223	0.0154	0.0132	0.0132	0.0129
NEE (%)	fine-tune	2.01	1.36	0.94	0.84	0.83	0.84
	fine-tune & pruning	1.92	1.39	0.96	0.83	0.83	0.81

number of neurons is selected when the last convolutional layer is fine-tuned together with the fully-connected layers ($n = 3$). Then the number of selected neurons decreases as the number of fine-tuned convolutional layers increases. This again implies that the last few convolutional layers can extract more specific features. When $n = 1, 2$, the number of useful specific features for interpreting the target data is insufficient, as a consequence, only a limited neurons in FC1 are needed which can contribute to the final specific features captured in FC2, and the estimation performance can not be maintained (Table 6). When n increases from 1 to 3, the increase of the number of selected neurons and the decrease of the estimation error reveals that as the number of fine-tuned layers increases, the number of useful extracted features increases and the number of redundant neurons decreases. However, when n is further increased from 3 to 6, the features extracted from the data have already been fully exploited through the fine-tuning of the convolutional layers proceeding to FC1 and FC2, therefore the number of selected neurons in FC1 and FC2 can be reduced. This is observable from Fig. 6 that as n increases from 3 to 6, the number of selected neurons in FC1 and FC2 is slightly decreased. The aforementioned analysis reveals that there is a trade-off in the transfer learning between the number of layers required to be fine-tuned, the model complexity, and associated computational effort.

In summary, according to Table 6 and Figure 10, fine-tuning 4 layers (2 fully-connected layers and 2 convolutional layers) are the best trade-off among estimation performance, model complexity, fine-tuning effort and computational cost.

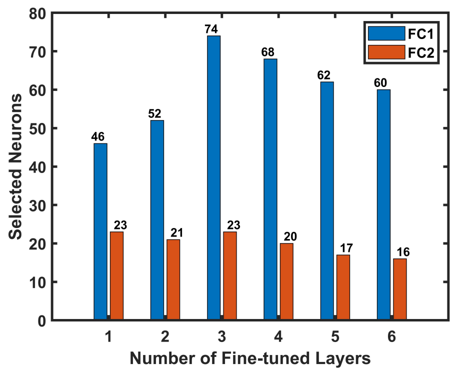


Figure 10: The number of neurons selected in the last two fully-connected layers of the PCNN(S)-TL model versus the number of fine-tuned layers

5.4. Discussions

By incorporating the concepts of transfer learning and network pruning, the final model updated offline using the small target dataset can produce more accurate online capacity estimation results with a more compact structure and lower computation cost. With real-time recorded signals (i.e. current, voltage and charge capacity) of a partial charging curve, the proposed model can quickly estimate the capacity online. These distinctive features open up a great potential for implementing the model in embedded systems for industrial applications. Note that the datasets used in this study only include one testing temperature, while in practice, the battery operating temperature varies. This can be tackled by adding the temperature as an additional input variable to the model. This study has also demonstrated that our proposed method is sufficiently robust to varying operating temperatures and different charging policies, as the batteries used to generate the source dataset and the target dataset were tested at different temperatures and charging/discharging policies. Further, we have demonstrated the effectiveness of our proposed method on battery capacity estimation on the similar type of batteries, though their specifications and testing profiles are different. However, whether the transfer learning can be applicable to different battery types is still an open question.

Finally, in this paper, we first build the CNN model offline using both source data and target data, and then use the developed model for online capacity estimation. A potential limitation for this method is that the parameters may drift as the battery cycling life increases, and the estimation error may increase and eventually exceed tolerable limit. This problem however can be overcome by updating the model parameters regularly using cycling test data which could be collected when the battery systems are under maintenance, or in other occasions where managed charging and discharging tests can be conducted. Given the operation safety is a paramount requirement for battery powered systems, regular maintenance is a necessity, and we strongly recommend including cycling test as one of the key battery maintenance procedures.

6. Conclusions

This paper has proposed a Convolutional Neural Network-based battery capacity estimation method, which combines the transfer learning and network pruning techniques to improve the estimation accuracy on small degradation dataset and improve the computational efficiency and compactness of the convolutional neural network model. The transferred and pruned PCNN(S)-TL model can achieve fast online capacity estimation only using partial charging segment with flexible starting point and a fixed length of 225 consecutive points. In the experiment, a large battery degradation dataset is used as the source dataset to build the convolutional neural network model which is then transferred to a small degradation dataset of target batteries with different specifications tested under different cycling profiles, the convolutional neural network model directly trained on the target dataset and separately pruned/transferred models are compared to verify the performance improvements of the proposed method. The experimental results confirm that:

- The estimation performance of the convolutional neural network model on the small target dataset can be improved through the transfer learning, and the normalized estimation error is decreased by 22.52%.
- Applying pruning technique to the transferred convolutional neural network model, the estimation accuracy is further improved, its normalized estimation error is 24.32% smaller than the convolutional neural network model directly trained on the target dataset.
- By selecting neurons with significant contributions and re-assign weights to the selected neurons using fast recursive algorithm in the fully-connected layers, both the model size and computational cost of the pruned model are significantly reduced, while the estimation

performance are maintained. Substantial test results have confirmed the efficacy of the proposed method, achieving up to 68.34% size reduction and 80.97% computation savings.

In summary, test results have revealed that compared to the original convolutional neural network model, both the model size and computational cost of the proposed model are significantly reduced, while the estimation performance on small degradation dataset is also improved.

7. Acknowledgments

YH. LI would like to thank the China Scholarship Council for sponsoring her research. The work is partially supported by SP Energy Networks on the project 'A holistic approach for power system monitoring to support DSO transition' and EPSRC funded project on 'Creating Resilient Sustainable Microgrids through Hybrid Renewable Energy Systems' under grant EP/R030243/1.

Appendix A. Properties of matrix $\mathbf{R}_k$ in fast recursive algorithm

Before introducing the fast recursive algorithm, the recursive matrix $\mathbf{R}_k$ was defined in Equation (3). When the candidate model terms $\{\varphi_j, j = 1, ..., S\}$ in regression matrix $\mathbf{\Psi}$ are mutually linearly independent, $\mathbf{R}_k$ will has the following distinguished properties:

$$\mathbf{R}_{k+1} = \mathbf{R}_k - \frac{\mathbf{R}_k \varphi_{k+1} \varphi_{k+1}^T \mathbf{R}_k^T}{\varphi_{k+1}^T \mathbf{R}_k \varphi_{k+1}}, \; k = 0, 1, ..., (S-1) \tag{A.1}$$

$$\mathbf{R}_k^T = \mathbf{R}_k, \quad \mathbf{R}_k \mathbf{R}_k = \mathbf{R}_k, \quad k = 0, 1, ..., S \tag{A.2}$$

$$\mathbf{R}_k \mathbf{R}_j = \mathbf{R}_j \mathbf{R}_k = \mathbf{R}_k, \quad k \geq j; \; k, j = 0, 1, ..., S \tag{A.3}$$

$$\mathbf{R}_k \varphi_j = 0, \quad \forall j \in \{1, ..., k\} \tag{A.4}$$

With these properties, the fast recursive algorithm can then be derived. For more details please refer to [42].

References

[1] K. Liu, K. Li, Q. Peng, C. Zhang, A brief review on key technologies in the battery management system of electric vehicles, Frontiers of mechanical engineering 14 (1) (2019) 47–64.

[2] Y. Li, M. Vilathgamuwa, B. Xiong, J. Tang, Y. Su, Y. Wang, et al., Design of minimum cost degradation-conscious lithium-ion battery energy storage system to achieve renewable power dispatchability, Applied Energy 260 (2020) 114282.

[3] W. Jing, C. H. Lai, W. S. Wong, M. D. Wong, A comprehensive study of battery-supercapacitor hybrid energy storage system for standalone pv power system in rural electrification, Applied Energy 224 (2018) 340–356.

[4] D.-I. Stroe, E. Schaltz, Lithium-ion battery state-of-health estimation using the incremental capacity analysis technique, IEEE Transactions on Industry Applications 56 (1) (2019) 678–685.

[5] M. Ouyang, X. Feng, X. Han, L. Lu, Z. Li, X. He, A dynamic capacity degradation model and its applications considering varying load for a large format li-ion battery, Applied Energy 165 (2016) 48–59.

[6] K. K. Sadabadi, P. Ramesh, P. Tulpule, G. Rizzoni, Design and calibration of a semi-empirical model for capturing dominant aging mechanisms of a pba battery, Journal of Energy Storage 24 (2019) 100789.

[7] D. Zhang, S. Dey, H. E. Perez, S. J. Moura, Real-time capacity estimation of lithium-ion batteries utilizing thermal dynamics, IEEE Transactions on Control Systems Technology 28 (3) (2019) 992–1000.

[8] K. Liu, C. Zou, K. Li, T. Wik, Charging pattern optimization for lithium-ion batteries with an electrothermal-aging model, IEEE Transactions on Industrial Informatics 14 (12) (2018) 5463–5474.

[9] Q. Yu, R. Xiong, R. Yang, M. G. Pecht, Online capacity estimation for lithium-ion batteries through joint estimation method, Applied Energy 255 (2019) 113817.

[10] L. Zheng, L. Zhang, J. Zhu, G. Wang, J. Jiang, Co-estimation of state-of-charge, capacity and resistance for lithium-ion batteries based on a high-fidelity electrochemical model, Applied Energy 180 (2016) 424–434.

[11] R. Xiong, L. Li, Z. Li, Q. Yu, H. Mu, An electrochemical model based degradation state identification method of lithium-ion battery for all-climate electric vehicles application, Applied Energy 219 (2018) 264–275.

[12] Y. Zheng, C. Qin, X. Lai, X. Han, Y. Xie, A novel capacity estimation method for lithium-ion batteries using fusion estimation of charging curve sections and discrete arrhenius aging model, Applied Energy 251 (2019) 113327.

[13] C. Zhang, K. Li, S. Mcloone, Z. Yang, Battery modelling methods for electric vehicles-a review, in: 2014 European Control Conference (ECC), IEEE, 2014, pp. 2673–2678.

[14] S. Li, S. Pischinger, C. He, L. Liang, M. Stapelbroek, A comparative study of model-based capacity estimation algorithms in dual estimation frameworks for lithium-ion batteries under an accelerated aging test, Applied energy 212 (2018) 1522–1536.

[15] R. R. Richardson, C. R. Birkl, M. A. Osborne, D. A. Howey, Gaussian process regression for in situ capacity estimation of lithium-ion batteries, IEEE Transactions on Industrial Informatics 15 (1) (2018) 127–138.

[16] Y. Li, H. Sheng, Y. Cheng, D.-I. Stroe, R. Teodorescu, State-of-health estimation of lithium-ion batteries based on semi-supervised transfer component analysis, Applied Energy 277 (2020) 115504.

[17] M. Berecibar, F. Devriendt, M. Dubarry, I. Villarreal, N. Omar, W. Verbeke, J. Van Mierlo, Online state of health estimation on nmc cells based on predictive analytics, Journal of Power Sources 320 (2016) 239–250.

[18] D. Yang, Y. Wang, R. Pan, R. Chen, Z. Chen, State-of-health estimation for the lithium-ion battery based on support vector regression, Applied Energy 227 (2018) 273–283.

[19] K. Liu, Y. Li, X. Hu, M. Lucu, W. D. Widanage, Gaussian process regression with automatic relevance determination kernel for calendar aging prediction of lithium-ion batteries, IEEE Transactions on Industrial Informatics 16 (6) (2019) 3767–3777.

[20] X. Li, C. Yuan, X. Li, Z. Wang, State of health estimation for li-ion battery using incremental capacity analysis and gaussian process regression, Energy 190 (2020) 116467.

[21] X. Feng, C. Weng, X. He, X. Han, L. Lu, D. Ren, M. Ouyang, Online state-of-health estimation for li-ion battery using partial charging segment based on support vector machine, IEEE Transactions on Vehicular Technology 68 (9) (2019) 8583–8592.

[22] P. Guo, Z. Cheng, L. Yang, A data-driven remaining capacity estimation approach for lithium-ion batteries based on charging health feature extraction, Journal of Power Sources 412 (2019) 442–450.

[23] X. Li, Z. Wang, L. Zhang, Co-estimation of capacity and state-of-charge for lithium-ion batteries in electric vehicles, Energy 174 (2019) 33–44.

[24] G.-W. You, S. Park, D. Oh, Diagnosis of electric vehicle batteries using recurrent neural networks, IEEE Transactions on Industrial Electronics 64 (6) (2017) 4885–4893.

[25] Y. Li, C. Zou, M. Berecibar, E. Nanini-Maury, J. C.-W. Chan, P. van den Bossche, J. Van Mierlo, N. Omar, Random forest regression for online capacity estimation of lithium-ion batteries, Applied energy 232 (2018) 197–210.

[26] S. Shen, M. Sadoughi, X. Chen, M. Hong, C. Hu, A deep learning method for online capacity estimation of lithium-ion batteries, Journal of Energy Storage 25 (2019) 100817.

[27] K. Kaur, A. Garg, X. Cui, S. Singh, B. K. Panigrahi, Deep learning networks for capacity estimation for monitoring soh of li-ion batteries for electric vehicles, International Journal of Energy Research (2020).

[28] Y. Li, H. Sheng, Y. Cheng, D.-I. Stroe, R. Teodorescu, State-of-health estimation of lithium-ion batteries based on semi-supervised transfer component analysis, Applied Energy 277 (2020) 115504.

[29] Y. Tan, G. Zhao, Transfer learning with long short-term memory network for state-of-health prediction of lithium-ion batteries, IEEE Transactions on Industrial Electronics 67 (2019) 8723 – 8731.

[30] S. Shen, M. Sadoughi, M. Li, Z. Wang, C. Hu, Deep convolutional neural networks with ensemble learning and transfer learning for capacity estimation of lithium-ion batteries, Applied Energy 260 (2020) 114296.

[31] Y. Li, J. Tao, Cnn and transfer learning based online soh estimation for lithium-ion battery, in: 2020 Chinese Control And Decision Conference (CCDC), IEEE, 2020, pp. 5489–5494.

[32] Y. Li, K. Li, X. Liu, L. Zhang, Fast battery capacity estimation using convolutional neural networks, Transactions of the Institute of Measurement and Control (2020) 0142331220966425.

[33] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on knowledge and data engineering 22 (10) (2009) 1345–1359.

[34] M. M. Ghazi, B. Yanikoglu, E. Aptoula, Plant identification using deep neural networks via optimization of transfer learning parameters, Neurocomputing 235 (2017) 228–235.

[35] W. Zhou, S. Newsam, C. Li, Z. Shao, Learning low dimensional convolutional neural networks for high-resolution remote sensing image retrieval, Remote Sensing 9 (5) (2017) 489.

[36] S. Anwar, K. Hwang, W. Sung, Structured pruning of deep convolutional neural networks, ACM Journal on Emerging Technologies in Computing Systems (JETC) 13 (3) (2017) 1–18.

[37] Z. Chen, J. Lin, S. Liu, Z. Chen, W. Li, J. Zhao, W. Yan, Exploiting weight-level sparsity in channel pruning with low-rank approximation, in: 2019 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE, 2019, pp. 1–5.

[38] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, in: Advances in neural information processing systems, 2015, pp. 1135–1143.

[39] X. Dong, S. Chen, S. Pan, Learning to prune deep neural networks via layer-wise optimal brain surgeon, in: Advances in Neural Information Processing Systems, 2017, pp. 4857–4867.

[40] H. Hu, R. Peng, Y.-W. Tai, C.-K. Tang, Network trimming: A data-driven neuron pruning approach towards efficient deep architectures, arXiv preprint, 2016, arXiv:1607.03250.

[41] A. Bondarenko, A. Borisov, L. Aleksejeva, Neurons vs weights pruning in artificial neural networks, in: Proceedings of the 10th International Scientific and Practical Conference. Volume III, Vol. 22, 2015, p. 28.

[42] K. Li, J.-X. Peng, G. W. Irwin, A fast nonlinear model identification method, IEEE Transactions on Automatic Control 50 (8) (2005) 1211–1216.

[43] Y. Wang, K. Li, S. Gan, C. Cameron, Missing data imputation with ols-based autoencoder for intelligent manufacturing, IEEE Transactions on Industry Applications 55 (6) (2019) 7219–7229.

[44] K. A. Severson, P. M. Attia, N. Jin, N. Perkins, B. Jiang, Z. Yang, M. H. Chen, M. Aykol, P. K. Herring, D. Fraggedakis, et al., Data-driven prediction of battery cycle life before capacity degradation, Nature Energy 4 (5) (2019) 383–391.

[45] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010, pp. 249–256.