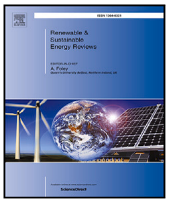
Renewable and Sustainable Energy Reviews 156 (2022) 111843

Contents lists available at ScienceDirect

Renewable and Sustainable Energy Reviews

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



电动汽车锂电池健康状态估计和剩余使用寿命预测的端到端神经网络框

# An end-to-end neural network framework for state-of-health estimation and remaining useful life prediction of electric vehicle lithium batteries

Penghua Li [a,*], Zijian Zhang [a], Radu Grosu [b], Zhongwei Deng [c,*], Jie Hou [a], Yujun Rong [d], Rui Wu [e]

[a] College of Automation, Chongqing University of Posts and Telecommunications, Chongqing, 400065, China
[b] Vienna University of Technology Institute of Computer Engineering, Treitlstrasse 3/3, 1040 Vienna, Austria
[c] State Key Laboratory of Mechanical Transmissions, Department of Automotive Engineering, Chongqing University, Chongqing 400044, China
[d] China Mobile (Hangzhou) Information Technology Co., Ltd., Hangzhou, 310000, China
[e] Chongqing Chang'an Automobile Co., Ltd., Chongqing, 400023, China

## ARTICLE INFO

*Keywords:*
Lithium battery
State-of-health
Remaining useful life
End-to-end framework
Bayesian optimization

## ABSTRACT

This study proposes an end-to-end prognostic framework for state-of-health (SOH) estimation and remaining useful life (RUL) prediction. In such a framework, a hybrid neural network (NN), i.e., the concatenation of one-dimensional convolutional NN and active-state-tracking long–short-term memory NN, is designed to capture the hierarchical features between several variables affecting battery degeneration, as well as the temporal dependencies embedded in those features. The prior distribution over hyperparameters, specified to the popular NNs applied in SOH or RUL tasks, is built through the Kolmogorov–Smirnov test. Such prior distribution is regarded as a surrogate to investigate the degeneration data's impact on modeling such NNs. Based on such a surrogate, a Bayesian optimization algorithm is proposed to build SOH and RUL models, selecting the most promising configuration automatically in the sequential evolution progress of hyperparameters. Compared with the existing NNs, the experiments indicate that our method hits a lower average RMSE 0.0072 and global average RMSE 0.0269 for SOH and RUL tasks. Code and models are available at https://github.com/Lipenghua-CQ/CNN-ASTLSTM.

## 1. Introduction

Lithium-ion batteries, as the systems of choice for customized energy storage, have been widely developed to satisfy the electric transportation industry's ever-growing demand [1,2]. To use lithium batteries safely, reliably, and economically, it is of paramount importance to accurately and robustly monitor lithium batteries' critical internal states, employing capacity or internal resistance to build health indicators such as state-of-health (SOH) and remaining useful life (RUL) [3]. However, direct online measurement of capacity or internal resistance is a perennial challenge in battery health management [4,5].

To cope with the challenge, many model-based and data-driven methods have been developed over the past decade. Although various classifications for the model-based methods emerge according to different perspectives, it is common to divide them into two main categories, i.e., mathematical model and mechanism model. The mathematical model, e.g., incremental capacity analysis [6], differential voltage analysis [7], provides a non-destructive means of characterizing cells and has been widely used for estimating SOH [8]. The mechanism model either applies partial differential equations to simulate mass and charge transfer kinetics closely related to aging, e.g., the electrochemical model [9] famous for the P2D model [10], or analyzes the physical and chemical reactions to describe the battery dynamic response and degradation behavior, e.g., the equivalent circuit model [11]. The authors in Ref. [12] evaluated the P2D model, semi-empirical model, and Gaussian process regression model for calendar aging prediction of lithium-ion batteries. They also highlighted both the challenges and prospects of employing each model to assist the industrial and academic research communities. The data-driven methods [13], especially neural networks (NNs), are gaining increasing interest in recent years since they are adaptive, flexible, and free of any physicochemical mechanisms [3]. Many studies develop big data and artificial intelligence methods to improve the management of the life cycle of lithium-ion batteries [14]. In Ref. [15], the voltage curve properties are fed to a polynomial NN with a group method of data handling for SOH estimation. The later work [16] extracts multiple terminal voltage features from one cell's aging process, then feeds such features to a prior knowledge-based NN to estimate SOH. For obtaining more information embedded in battery degradation data, some studies use multiple

* Corresponding authors.
*E-mail addresses:* lipenghua88@163.com (P. Li), dengzhongw@cqu.edu.cn (Z. Deng).

https://doi.org/10.1016/j.rser.2021.111843
Received 12 July 2021; Received in revised form 5 October 2021; Accepted 29 October 2021
Available online 4 December 2021

variables, instead of a single variable, as the NNs' inputs during SOH tasks. For instance, the researchers [17] regard the state-of-charge (SOC), current and impedance spectroscopy of one cell as the inputs of recurrent NNs (RNNs) when using such RNNs to estimate internal resistance and capacity in parallel. Similarly, the voltage, current, and temperature variations are fed to RNN in another SOH task [18]. To improve the NN-based estimator's generalization ability, the data of multiple cells, beyond multiple variables per cell, is used to train a more generic framework, e.g., the convolutional NN (CNN) prognostic framework [19] and the active-state-tracking long–short-term memory NN (AST-LSTM NN) prognostic framework [20].

As a supplement to SOH, RUL is often predicted along with SOH. Recent studies [20–26] mostly employ LSTM NNs predicting RUL due to their outstanding capability of processing long–short-term dependencies embedded in time series. These methods differ mainly in input type, data curation and NN usage. For input type, some studies utilize the capacity features, i.e., several sub-layers coefficients of empirical mode decomposition (EMD) [21], while others [20,22–24] employ the raw recorded capacities. For data curation, [20,22] use multiple cells' data to train an NN-based model, while others [21,23,24] are the opposite, training a model with one cell's data. Regarding NN usage, two main ways can be traced in the literature in order to achieve accurate results. One is to develop individual forecasting model, such as the standard LSTM NN (S-LSTM NN) with dropout technique [22] and Adam algorithm [23], the bidirectional LSTM NN [24] and the AST-LSTM NN [20]. The other is to combine different models, aiming to take individual models' strength in battery time series processing, to predict RUL synthetically, such as the combination of Elman NN and LSTM NN [21], and the hybridization of CNN and LSTM NN [25,26].

While promising results have been reported by the aforementioned approaches, their flexible application in real health prognostics needs further validation for the following concerns. (1) The accurate SOH estimation [26] and RUL prediction [25] give a shred of evidence that hybridization of individual NN-based models can capture various patterns in the battery data concurrently. However, without further evidence showing that such hybridization is not complicated in practical applications, these methods seem to have only theoretical significance. Besides, further investigation is needed on what kind of LSTM NN can be combined with CNN to obtain a more effective model. (2) An available SOH and RUL models require to train NN several times, mostly depending on how the experimenter chooses to parametrize the models, and how many hyperparameters the experimenter chooses to fix at a reasonable default. The difficulty of manually tuning NNs causes great challenge to reproducing and extending the published results [17–26], making even the original investigation of such methods more of an art than a science. Automatic machine learning algorithms, such as the hyperparameter optimization using Bayesian estimation [27], can reduce manual intervention during the NN modeling process. However, owing to the lack of hyperparameters' prior distribution over battery data, these methods are not especially suitable for SOH and RUL tasks.

Our motivation is to propose an end-to-end prognostic framework to address the aforementioned concerns. This framework helps researchers reduce the manual workload of tuning hyperparameters by an improved Bayesian optimization algorithm, simplifying the process of employing various NNs to build SOH/RUL models. Besides, we also hope that it can help researchers to get a model with higher accuracy that is lightweight enough to be ported to an embedded platform with better performance in the future. To ensure the transparency of our work, we use a public dataset like the NASA dataset, dedicated to the SOH/RUL problem, so that other researchers are able to, if they want, test the model obtained through our framework by themselves. The contributions of this study are summarized as follows correspondingly.

- Compared with the existing CNN-LSTM NN model, a novel hybridization of AST-LSTM NN and one-dimensional CNN, called CNN-ASTLSTM NN, is proposed to capture the degeneration data's features hierarchically and actively learn the time dependencies embedded in such features. Besides, to investigate the practical performance of NN-based methods, several current individual and hybrid models, including the CNN-ASTLSTM NN, are well trained in our framework and comprehensively evaluated from their predicted error, parameters, FLOPs, latency, storage size, and training time.
- An improved Bayesian optimization algorithm is proposed to achieve the NNs' automatic hyperparameter configuration during SOH and RUL tasks, reducing the manual intervention in the NNs' training process and making the results easier to promote and reproduce. The Kolmogorov–Smirnov (KS) test is employed to obtain the prior distribution over the NNs' loss function values. We regard the loss distribution as a surrogate distribution of the hyperparameters specified to the NNs. Based on the prior distribution over hyperparameters, a Bayesian probability surrogate model is built to select automatically the most promising hyperparameters to evaluate the true loss function.

## 2. Methodology

The proposed end-to-end prognostic framework consists of the data-end (Fig. 1(a)), automatic modeling (Fig. 1(b)), and result-end (Fig. 1(c)). At the data-end, we curate the measured battery samples to be applicable for SOH and RUL tasks. In the automatic modeling process, the prior distribution over hyperparameters of NNs is investigated by the KS test. Then, we combine such prior distribution and the posterior hyperparameter distribution generated during the subsequent training progress to tune the NN-based models automatically. The result-end outputs the expected SOH or RUL.

### 2.1. Design of CNN-ASTLSTM NN

Let $\mathbf{U}_n \in \mathbb{R}^{W^{U_n} \times C^{U_n}}$ and $\mathbf{F}_n \in \mathbb{R}^{W^{F_n} \times C^{F_n}}$ be the inputs and feature maps of the $n$th ($n \in \{1, 2, \dots, N\}$) one-dimensional convolution layer, respectively. The superscripts $W$ and $C$ denote the data width and channel, respectively. In particular, $\mathbf{U}_1$ is the curated samples, where the data width and channel represent sample length and sample type (discharge-process capacity and its corresponding voltage, current, battery temperature, sampling time), respectively. The outputs of $C^{F_n}$ filters (kernels) $\mathbf{K}_n \in \mathbb{R}^{W^{K_n} \times C^{U_n} \times C^{F_n}}$, i.e., the feature maps of $n$th convolution layer, can be obtained by:

$$\mathbf{F}_{n,(x,c_{out})} = \sigma\left( \sum_{i,c_{in} \in \mathcal{N}^{\mathbf{K}}{}_n} \mathbf{K}_{n,i,c_{in},c_{out}} \mathbf{U}_{n,s\times x+i,c_{in}} \right) \tag{1}$$

where $(0 \leqslant x \leqslant W^{F_n} - 1, 0 \leqslant c_{out} \leqslant C^{F_n} - 1)$, $\sigma$, $\mathcal{N}^{K_n}$ and $s$ represent the position coordinates, activation function, local neighborhood and stride size, respectively. Note that the range of $\mathcal{N}^{K_n}$ is given as $\left\{ \left(i, c_{in}\right) | 0 \leqslant i \leqslant W^{K_n} - 1,\ 0 \leqslant c_{in} \leqslant C^{U_n} - 1 \right\}$. After *max* pooling, the inputs of $n + 1$th convolution, $\mathbf{U}_{n+1} \in \mathbb{R}^{W^{U_{n+1}} \times C^{F_n}}$, can be given as:

$$\mathbf{U}_{n+1,(x,c_{out})} = \mathbf{Y}_{n,(x,c_{out})} = \max_{0 \leqslant j \leqslant Q-1} \left\{ \mathbf{F}_{n,d\times x+j,c_{out}} \right\} \tag{2}$$

where $\mathbf{Y}_{n,(x,c_{out})}$, $(0 \leqslant x \leqslant W^{U_{n+1}} - 1, 0 \leqslant c_{out} \leqslant C^{F_n} - 1)$, $Q$ and $d$ are the downsampling outputs, position coordinates, pooling kernel and stride size, respectively.

In each convolutional layer, the $C^{F_n}$ filters are learned to express informative combinations by fusing spatial and channel-wise information together within local receptive fields. By stacking a series of convolutional layers interleaved with non-linearities and downsampling, the one-dimensional CNN can capture hierarchical patterns [28] between several variables affecting battery degeneration. Then an AST-LSTM NN [20] (Fig. 1(d)) is used to learn with the long–short-term dependencies embedded in such patterns.

For the $l$th ($l \in 1, 2, \dots, L$) AST-LSTM layer, let $\mathbf{u}_{l,t} \in \mathbb{R}^{|\mathbf{u}_{l,t}|}$ be the inputs at time $t$, $M$ be the number of blocks and $\mathbf{h}_{l,t-1} \in \mathbb{R}^M$ the

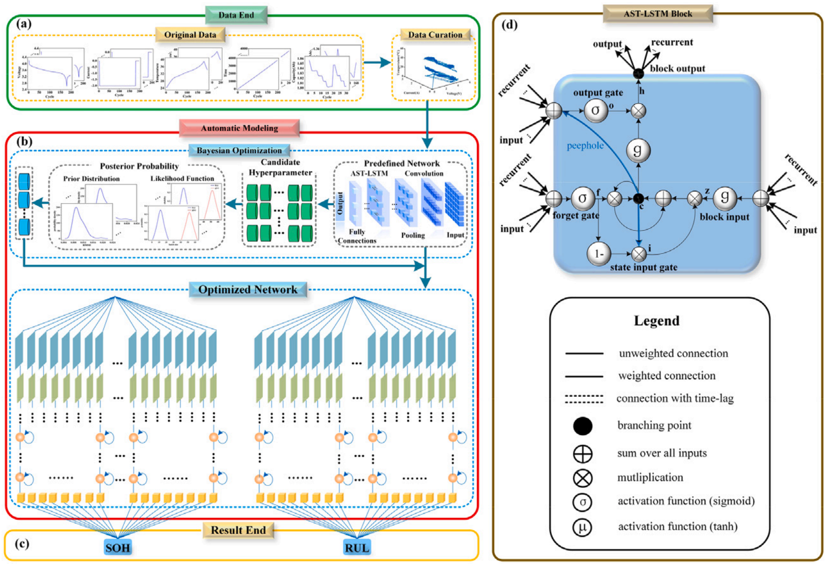


**Fig. 1.** The end-to-end prognostic framework, where (a)–(d) are the data-end, automatic modeling process, result-end, and AST-LSTM block, respectively.

block outputs at time $t-1$. Note that $|\cdot|$ denotes a dimension operator. Obviously, $\mathbf{u}_{1,t} = \mathbf{Y}_{N,(x,c_{out})}$, $\mathbf{u}_{l+1,t} = \mathbf{h}_{l,t}$, where $max(t) = |x| = W^{U_N}$, $|\mathbf{u}_{1,t}| = |c_{out}| = C^{F_N}$. The forget gate outputs $\mathbf{f}_{l,t} \in \mathbb{R}^M$ and candidate gate outputs $\mathbf{z}_{l,t} \in \mathbb{R}^M$ are given as:

$$\mathbf{f}_{l,t} = \sigma\left(\mathbf{k}_{l,fu}\cdot\mathbf{u}_{l,t} + \mathbf{k}_{l,fh}\cdot\mathbf{h}_{l,t-1} + \mathbf{b}_{l,f}\right) \tag{3}$$

$$\mathbf{z}_{l,t} = g\left(\mathbf{k}_{l,zu}\cdot\mathbf{u}_{l,t} + \mathbf{k}_{l,zh}\cdot\mathbf{h}_{l,t-1} + \mathbf{b}_{l,z}\right) \tag{4}$$

where $\mathbf{k}_{l,fu} \in \mathbb{R}^{M\times|\mathbf{u}_{l,t}|}$, $\mathbf{k}_{l,fh} \in \mathbb{R}^{M\times M}$, $\mathbf{b}_{l,f} \in \mathbb{R}^M$, $\mathbf{k}_{l,zu} \in \mathbb{R}^{M\times|\mathbf{u}_{l,t}|}$, $\mathbf{k}_{l,zh} \in \mathbb{R}^{M\times M}$ and $\mathbf{b}_{l,z} \in \mathbb{R}^M$ are the input, recurrent and bias weights in the forget gate layer and candidate gate layer, respectively. The *logistic sigmoid* $\sigma(\cdot)$ and *hyperbolic tangent* $\tanh(\cdot)$ are activation functions of forget gate and candidate gate, respectively. The fused state inputs $\mathbf{i}_{l,t} \in \mathbb{R}^M$ and state outputs $\mathbf{c}_{l,t} \in \mathbb{R}^M$ are written as:

$$\mathbf{i}_{l,t} = \left(1-\mathbf{f}_{l,t}\right)\odot\sigma\left(\mathbf{c}_{l,t-1}\odot\mathbf{p}_{l,i}\right) \tag{5}$$

$$\mathbf{c}_{l,t} = \mathbf{c}_{l,t-1}\odot\mathbf{f}_{l,t} + \mathbf{i}_{l,t}\odot\mathbf{z}_{l,t} \tag{6}$$

where $\mathbf{c}_{l,t-1} \in \mathbb{R}^M$ and $\mathbf{p}_{l,i} \in \mathbb{R}^M$ represent the cell states at time $t-1$ and weights of peephole over old cell states, respectively. The symbol $\odot$ denotes point-wise multiplication. The output gate outputs $\mathbf{o}_{l,t} \in \mathbb{R}^M$ and block outputs $\mathbf{h}_{l,t} \in \mathbb{R}^M$ are given as:

$$\mathbf{o}_{l,t} = \sigma\left(\mathbf{k}_{l,ou}\cdot\mathbf{u}_{l,t} + \mathbf{k}_{l,oh}\cdot\mathbf{h}_{l,t-1} + \mathbf{p}_{l,o}\odot\mathbf{c}_{l,t} + \mathbf{b}_{l,o}\right) \tag{7}$$

$$\mathbf{h}_{l,t} = \mathbf{o}_{l,t}\odot g\left(\mathbf{c}_{l,t}\right) \tag{8}$$

where $\mathbf{k}_{l,ou} \in \mathbb{R}^{M\times|\mathbf{u}_{l,t}|}$, $\mathbf{k}_{l,oh} \in \mathbb{R}^{M\times M}$ and $\mathbf{b}_{l,o} \in \mathbb{R}^M$ are the input, recurrent and bias weights in the output gate layer, respectively. The weights of peephole over new cell states are denoted as $\mathbf{p}_{l,o} \in \mathbb{R}^M$. Finally, the outputs of CNN-ASTLSTM NN, $\mathbf{y}_{L+1,t} \in \mathbb{R}^{|\mathbf{y}_{L+1,t}|}$, are written as:

$$\mathbf{y}_{L+1,t} = \mathbf{k}_{L+1}\mathbf{h}_{L,t} \tag{9}$$

where $\mathbf{k}_{L+1} \in \mathbb{R}^{|\mathbf{y}_{L+1,t}|\times|\mathbf{h}_{L,t}|}$ are the full connection weights.

### 2.2. Hyperparameter optimization of CNN-ASTLSTM NN

The ultimate objective of leveraging CNN-ASTLSTM NN for SOH estimation or RUL prediction is to use such network to find a nonlinear function $\mathbb{F}$ that minimizes estimation or prediction error over i.i.d. battery samples $\mathbf{U}$ from a natural distribution $\mathcal{G}_{\mathbf{U}}$. Following [29], this objective can be reformulated as using a learning algorithm $\mathcal{A}$ to map a finite battery dataset $\mathcal{D}_{\mathbf{U}}$ into such function $\mathbb{F}$ that minimizes expected loss $\mathcal{L}\left(\mathcal{D}_{\mathbf{U}};\mathbb{F}\right)$. The actual learning algorithm is denoted as $\mathcal{A}_{\boldsymbol{\theta}}$ since it produces $\mathbb{F}$ through the optimization of a training criterion w.r.t. a set of parameters $\boldsymbol{\theta}$, called *hyper-parameter*, in high dimensional space $\boldsymbol{\Theta}$. This allows the hyperparameter optimization problem to be written as:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\Theta}{\arg\min}\,\mathbb{E}_{\mathbf{U}\sim\mathcal{G}_{\mathbf{U}}}\left[\mathcal{L}\left(\mathbf{U};\mathcal{A}_{\boldsymbol{\theta}}\left(\mathcal{D}_{\mathbf{U}}\right)\right)\right] \tag{10}$$

where $\mathbb{E}_{\mathbf{U}\sim\mathcal{G}_{\mathbf{U}}}(\cdot)$ is the expectation of loss (also called generalization error). However, the expectation over the unknown natural distribution $\mathcal{G}_{\mathbf{U}}$ is difficult to evaluate directly. To tackle such issue, following [30] and considering $k$-fold cross-validation [31] in CNN-ASTLSTM NN's training process, Eq. (10) is redefined as:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}_c,\boldsymbol{\theta}_a\in\Theta^{(i)}}{\arg\min}\,\frac{1}{k}\sum_{i=1}^{k}\mathcal{L}\left(\mathcal{A}^{(i)}_{\boldsymbol{\theta}_c},\mathcal{A}^{(i)}_{\boldsymbol{\theta}_a},\mathcal{D}^{(i)}_{\mathbf{U},train},\mathcal{D}^{(i)}_{\mathbf{U},valid}\right) \tag{11}$$

where $\mathcal{A}^{(i)}_{\boldsymbol{\theta}_c}$ and $\mathcal{A}^{(i)}_{\boldsymbol{\theta}_a}$ present the learning algorithms specified by CNN hyperparameters $\boldsymbol{\theta}_c$ and AST-LSTM NN hyperparameters $\boldsymbol{\theta}_a$, respectively. The training and valid data over $i$th cross-validation are denoted as $\mathcal{D}^{(i)}_{\mathbf{U},train}$ and $\mathcal{D}^{(i)}_{\mathbf{U},valid}$, respectively. Thorough Eq. (11), the aforementioned problem is reformulated as a single combined hierarchical hyperparameter optimization problem with tree-structured space $\boldsymbol{\Theta} = \Theta^{(1)}\cup\cdots\cup\Theta^{(k)}\cup\left\{\boldsymbol{\theta}_r\right\}$, where $\boldsymbol{\theta}_r$ is the root-level hyperparameters of each subspace $\Theta^{(i)}$.

In principle, Eq. (11) can be solved in various ways, e.g., sequential model-based optimization (SMBO) [32], Gaussian process (GP) optimization [33] and Bayesian optimization [27]. Specifically, we chose the expected improvement (EI) criterion [30] to approximate Eq. (11) with a cheaper surrogate. The EI is an expectation under *CNN-ASTLSTM NN* producing $\mathbb{F}$: $\mathbf{U}\to\mathbf{y}$ that $\mathbb{F}(\mathcal{D}_{\mathbf{U}})$ will exceed negatively a threshold $e^*(\boldsymbol{\theta}_{c,a})$, which is given as:

$$EI_{e^*}\left(\boldsymbol{\theta}_{c,a}\right) := \int_{-\infty}^{\infty}\max\left(e^*-e,0\right)P_{\mathbb{F}}\left(e\middle|\boldsymbol{\theta}_{c,a}\right)de \tag{12}$$

where $e\left(\boldsymbol{\theta}_{c,a}\right) = \mathcal{L}\left(\mathcal{A}_{\boldsymbol{\theta}_c},\mathcal{A}_{\boldsymbol{\theta}_a},\mathcal{D}_{\mathbf{U},train},\mathcal{D}_{\mathbf{U},valid}\right)$ denotes the loss w.r.t. hyperparameter configurations $\boldsymbol{\theta}_c$ and $\boldsymbol{\theta}_a$. Unlike GP optimization [33] modeling $P_{\mathbb{F}}\left(e\middle|\boldsymbol{\theta}_{c,a}\right)$ directly, we follow [27] to model $P_{\mathbb{F}}\left(\boldsymbol{\theta}_{c,a}\middle|e\right)$ and $P_{\mathbb{F}}(e)$. We model $P_{\mathbb{F}}\left(\boldsymbol{\theta}_{c,a}\middle|e\right)$ as one of two density estimates, conditional on whether $e$ is greater or less than the given threshold value $e^*$:

$$P_{\mathbb{F}}\left(\boldsymbol{\theta}_{c,a}\middle|e\right)=\begin{cases}\ell(\boldsymbol{\theta}_{c,a}) & \text{if e} < e^*\\ g(\boldsymbol{\theta}_{c,a}) & \text{if e} \geqslant e^*\end{cases} \tag{13}$$

where $\ell(\cdot)$ denotes a density estimate formed by using observations $\left\{\mathcal{D}^{(i)}_{\mathbf{U},train},\ \mathcal{D}^{(i)}_{\mathbf{U},valid}\right\}$, so that corresponding loss $e(\boldsymbol{\theta}_{c,a})$ is less than $e^*$, and $g(\cdot)$ is a density estimate learned from the remaining observations with corresponding loss greater than or equal to $e^*$. Note that $e^*$ is selected as a $\gamma$-quantile, so that $P_{\mathbb{F}}(e < e^*) = \gamma$. Ref. [27] presents EI is proportional to closed-form expression, i.e., $EI_{e^*}\left(\boldsymbol{\theta}_{c,a}\right) \propto \left(\gamma + \frac{g(\boldsymbol{\theta}_{c,a})}{\ell(\boldsymbol{\theta}_{c,a})}(1-\gamma)\right)^{-1}$, which can be derived as follows:

$$EI_{e^*}\left(\boldsymbol{\theta}_{c,a}\right)=\int_{-\infty}^{e^*}\left(e^*-e\right)\frac{P_{\mathbb{F}}\left(\boldsymbol{\theta}_{c,a}|e\right)P_{\mathbb{F}}(e)}{P_{\mathbb{F}}\left(\boldsymbol{\theta}_{c,a}\right)}de \tag{14}$$

$$\begin{aligned}\int_{-\infty}^{e^*}(e^*-e)P_{\mathbb{F}}\left(\boldsymbol{\theta}_{c,a}|e\right)P_{\mathbb{F}}(e)\,de &= \ell\left(\boldsymbol{\theta}_{c,a}\right)\int_{-\infty}^{e^*}(e^*-e)P_{\mathbb{F}}(e)\,de\\ &= e^*\gamma\ell\left(\boldsymbol{\theta}_{c,a}\right)-\ell\left(\boldsymbol{\theta}_{c,a}\right)\int_{-\infty}^{e^*}eP_{\mathbb{F}}(e)de\end{aligned} \tag{15}$$

$$\begin{aligned}P_{\mathbb{F}}\left(\boldsymbol{\theta}_{c,a}\right) &= \int_{\mathbb{R}}P_{\mathbb{F}}\left(\boldsymbol{\theta}_{c,a}|e\right)P_{\mathbb{F}}(e)\,de\\ &= \gamma\ell\left(\boldsymbol{\theta}_{c,a}\right)+(1-\gamma)g\left(\boldsymbol{\theta}_{c,a}\right)\end{aligned} \tag{16}$$

$$EI_{e^*}\left(\boldsymbol{\theta}_{c,a}\right)=\frac{e^*\gamma\ell\left(\boldsymbol{\theta}_{c,a}\right)-\ell\left(\boldsymbol{\theta}_{c,a}\right)\int_{-\infty}^{e^*}eP_{\mathbb{F}}(e)de}{\gamma\ell\left(\boldsymbol{\theta}_{c,a}\right)+(1-\gamma)g\left(\boldsymbol{\theta}_{c,a}\right)}. \tag{17}$$

An one-dimensional Parzen estimator is created to model the densities of $\ell(\cdot)$ and $g(\cdot)$, which is given as:

$$\widehat{P}_{\ell/g}\left(\boldsymbol{\theta}_{c,a}\right)=\frac{1}{kh}\sum_{i=1}^{k}\mathbf{K}_G\left(\frac{\mathcal{L}\left(\mathcal{A}_{\boldsymbol{\theta}_{c,a}},\mathcal{D}_{\mathrm{U}}\right)-\mathcal{L}\left(\mathcal{A}^{(i)}_{\boldsymbol{\theta}_{c,a}},\mathcal{D}^{(i)}_{\mathrm{U}}\right)}{h}\right) \tag{18}$$

where $\mathbf{K}_G$ is a Gaussian kernel and $h > 0$ is a smoothing parameter called the bandwidth.

For the prior distribution w.r.t. $P_{\mathbb{F}}(e)$, we borrow from [27] and go one step further. We use KS test [34] to obtain the prior surrogate distribution over hyperparameters. The empirical distribution function $F_e$ for $n$ i.i.d. ordered observations $E_j \in \left\{\left(\mathcal{A}^{(j)}_{\boldsymbol{\theta}_c}, \mathcal{A}^{(j)}_{\boldsymbol{\theta}_a}, \mathcal{D}^{(j)}_{\mathbf{U},train}, \quad \mathcal{D}^{(j)}_{\mathbf{U},valid}\right)\right\}$ is defined as:

$$F_e(e)=\frac{1}{n}\sum_{j=1}^{n}I_{[-\infty,e]}\left(E_j\right) \tag{19}$$

where $I_{[-\infty,e]}\left(E_j\right)$ is the indicator function, equal to 1 if $E_j \leqslant e$ and equal to 0 otherwise. The KS statistic for a given cumulative distribution function $F(e)$ is written as:

$$D_n=\sup_e\left|F_e(e)-F(e)\right| \tag{20}$$

where $\sup_e$ is the supremum of the set of distances. The KS test can be constructed by using the critical values of the Kolmogorov distribution. It rejects the null hypothesis at level $\alpha$ if $\sqrt{n}D_n > K_\alpha$, where $K_\alpha$ is found from

$$P\left(K \leqslant K_\alpha\right)=1-\alpha. \tag{21}$$

### 2.3. Application of CNN-ASTLSTM NN

The problem of applying CNN-ASTLSTM NN to train a SOH estimation model can be described as:

$$SOH := \frac{\mathbb{F}\left(\mathbf{U}^{H_i}_{(\mathrm{V,T,I,t}),t},\mathbf{U}^{H_i}_{(\mathrm{V,T,I,t}),t+1},\dots,\mathbf{U}^{H_i}_{(\mathrm{V,T,I,t}),t+s_w-1}|U^{H_i}_{\mathrm{C}}\right)}{C_0}\times 100\% \tag{22}$$

where $\mathbf{U}^{H_i}_{(\mathrm{V,T,I,t}),t}$, $U^{H_i}_{\mathrm{C}}$, $s_w$ and $C_0$ are the history samples of voltage (V), temperature (T), current (I), and sampling time (t) in $i$th cycle, the history capacity at $i$th cycle, the length of sliding window and the initial capacity, respectively. The online estimation of SOH is given as:

$$\widehat{SOH} := \frac{\mathbb{F}\left(\mathbf{U}^{O_j}_{(\mathrm{V,T,I,t}),t},\mathbf{U}^{O_j}_{(\mathrm{V,T,I,t}),t+1},\dots,\mathbf{U}^{O_j}_{(\mathrm{V,T,I,t}),t+s_w-1}\right)}{C_0}\times 100\% \tag{23}$$

where $\mathbf{U}^{O_j}_{(\mathrm{V,T,I,t}),t}$ are the observed samples of voltage, temperature, current, and sampling time in $j$th cycle.

Given the end-of-life (EOL) as $C_0 \times 70\%$, the actual and predicted RUL are defined as:

$$RUL = q_{eol} - q_s \tag{24}$$

$$\widehat{RUL} = \widehat{q}_{eol} - q_s \tag{25}$$

where $q_{eol}$ and $\widehat{q}_{eol}$ are respectively the actual and predicted cycle at EOL, $q_s$ is the cycle when prediction gets started. The $\widehat{q}_{eol}$ can be obtained by applying CNN-ASTLSTM NN to train a RUL prediction model:

$$\widehat{q}^{(train)}_{eol} := \mathbb{F}\left(U^H_{\mathrm{C},q_s},U^H_{\mathrm{C},q_s-1},\dots,U^H_{\mathrm{C},q_s-s_w+1}|U^H_{\mathrm{C},q_s+1},\dots,U^H_{\mathrm{C},q_s+p_w}\right) \tag{26}$$

where $U^H_{\mathrm{C},q_s}$ and $p_w$ denote the history capacity at starting prediction cycle and length of prediction window, respectively. The online prediction of $\widehat{q}_{eol}$ is described as:

$$\widehat{q}^{(online)}_{eol} := \mathbb{F}\left(U^P_{\mathrm{C},q_s+p_w},\dots,U^P_{\mathrm{C},q_s+1},U^O_{\mathrm{C},q_s},\dots,U^O_{\mathrm{C},q_s-s_w+1+p_w}\right) \tag{27}$$

where $U^P_{\mathrm{C},q_s+p_w}$ and $U^O_{\mathrm{C},q_s}$ are the predicted and observed capacities, respectively. The implementation of our method is described as follows:

**Algorithm 1** SOH estimation and RUL prediction

```
Require: $\Theta$, training set $\{\mathbf{U}^H\}$ and testing set $\{\mathbf{U}^O\}$
Ensure: $\widehat{SOH}_j$ and $\widehat{RUL}$
 1: function $Train_{soh}(\mathbf{U}^{H_i})$
 2: $\mathbf{U}^{H_i} \leftarrow normalize\{\mathbf{U}^{H_i}\}$
 3:     for $e \leftarrow 1$ to $Epoch_{soh}$ do
 4:         for $i \leftarrow 1$ to $Cycle$ do
 5:             $Model_{soh}, Loss_{soh} \leftarrow \mathbb{F}\left(\mathbf{U}^{H_i}_{(\mathrm{V,T,I,t}),t},\cdots,\mathbf{U}^{H_i}_{(\mathrm{V,T,I,t}),t+s_w-1}\middle|U^{H_i}_{\mathrm{C}}\right)$
 6:         end for
 7:     end for
 8:     return $Model_{soh}, Loss_{soh}$
 9: end function
10: function $Train_{rul}(\mathbf{U}^H)$
11: $\mathbf{S}^h \leftarrow normalize\{\mathbf{S}^h\}$
12:     for $e \leftarrow 1$ to $Epoch_{rul}$ do
13:         $Model_{rul}, Loss_{rul} \leftarrow \mathbb{F}\left(U^H_{\mathrm{C},q_s},\cdots,U^H_{\mathrm{C},q_s-s_w+1}\middle|U^H_{\mathrm{C},q_s+1},\cdots,U^H_{\mathrm{C},q_s+p_w}\right)$
14:     end for
15:     return $Model_{rul}, Loss_{rul}$
16: end function
17: function $TPE(\mathcal{D}_{\mathbf{U}}, task)$
18: $H \leftarrow \emptyset$
19:     for $t \leftarrow 1$ to $T$ do
20:         Compute $P_{\mathbb{F}}\left(\mathcal{D}_{\mathbf{U}} \mid e\right)$
21:         if $EI_{e^*}\left(\theta_{c,a}\right) > 0$ then
22:             $Loss_{task} \leftarrow Train_{task}(\mathcal{D}_{\mathbf{U}})$ use $\theta$
23:             $H \leftarrow H \cup \left(\theta_{c,a}, Loss_{task}\right)$
24:         end if
25:         $\theta^*_{task} \leftarrow H$
26:     end for
27:     return $\theta^*_{task}$
28: end function
29: Step 1 Off-line training:
30: $\theta^*_{soh} \leftarrow TPE(\mathbf{U}^{H_i}, soh)$
31: $\theta^*_{rul} \leftarrow TPE(\mathbf{U}^H, rul)$
32: $Model_{soh} \leftarrow Train_{soh}(\mathbf{U}^{H_i})$ use $\theta^*_{soh}$
33: $Model_{rul} \leftarrow Train_{rul}(\mathbf{U}^H)$ use $\theta^*_{rul}$
34: Step 2 On-line testing:
35: for $j \leftarrow 1$ to $Cycle$ do
36:     $\widehat{SOH}_j = \frac{Model_{soh}\left(\mathbf{U}^{O_j}_{(\mathrm{V,T,I,t}),t},\cdots,\mathbf{U}^{O_j}_{(\mathrm{V,T,I,t}),t+s_w-1}\right)}{C_0} \times 100\%$
37: end for
38: $\widehat{q}^{(online)}_{eol} := Model_{rul}\left(U^P_{\mathrm{C},q_s+p_w},\cdots,U^O_{\mathrm{C},q_s-s_w+1+p_w}\right)$
39: $\widehat{RUL} = \widehat{q}_{eol} - q_s$
```

**Table 1**
Priori distribution over the hyperparameters of NNs in SOH task.

| Hyperparameters | Methods | | | | |
|---|---|---|---|---|---|
| | RNN | S-LSTM | AST-LSTM | CNN-LSTM | CNN-ASTLSTM |
| $K_n$ | – | – | – | $\log\mathcal{N}(3.30, 0.18)$ | $\log\mathcal{N}(3.69, 0.14)$ |
| $K_s$ | – | – | – | $\log\mathcal{N}(1.61, 0.18)$ | $\log\mathcal{N}(1.95, 0.11)$ |
| s | – | – | – | $\mathcal{N}(3, 0.45)$ | $\mathcal{N}(4, 0.50)$ |
| $P_s$ | – | – | – | $\mathcal{N}(5, 1.20)$ | $\mathcal{N}(3, 0.50)$ |
| $L_h$ | $\mathcal{U}(1, 4)$ | $\log\mathcal{N}(0.69, 0.18)$ | $\log\mathcal{N}(0.69, 0.18)$ | $\log\mathcal{N}(0.69, 0.10)$ | $\log\mathcal{N}(0.69, 0.10)$ |
| $N_b$ | $\log\mathcal{N}(3.69, 0.34)$ | $\log\mathcal{N}(4.79, 0.14)$ | $\log\mathcal{N}(4.87, 0.14)$ | $\log\mathcal{N}(4.50, 0.18)$ | $\log\mathcal{N}(3.40, 0.26)$ |
| lr | $\log\mathcal{N}(-5.65, 0.26)$ | $\log\mathcal{N}(-6.21, 0.26)$ | $\log\mathcal{N}(-6.21, 0.26)$ | $\log\mathcal{N}(-7.01, 0.18)$ | $\log\mathcal{N}(-6.81, 0.18)$ |
| b | $\log\mathcal{N}(2.89, 0.59)$ | $\log\mathcal{N}(2.89, 0.59)$ | $\log\mathcal{N}(3.22, 0.64)$ | $\log\mathcal{N}(3.00, 0.53)$ | $\log\mathcal{N}(2.30, 0.41)$ |
| e | $\log\mathcal{N}(5.14, 0.18)$ | $\log\mathcal{N}(4.87, 0.18)$ | $\log\mathcal{N}(4.94, 0.18)$ | $\log\mathcal{N}(4.94, 0.14)$ | $\log\mathcal{N}(4.70, 0.18)$ |
| dr | $\log\mathcal{N}(-3.22, 0.26)$ | $\log\mathcal{N}(-3.22, 0.26)$ | $\log\mathcal{N}(-3.69, 0.26)$ | $\log\mathcal{N}(-3.00, 0.18)$ | $\mathcal{U}(0.01, 0.10)$ |

**Table 2**
Priori distribution over the hyperparameters of NNs in RUL task.

| Hyperparameters | Methods | |
|---|---|---|
| | AST-LSTM | CNN-ASTLSTM |
| $K_n$ | – | $\log\mathcal{N}(4.25, 0.22)$ |
| $K_s$ | – | $\log\mathcal{N}(1.50, 0.26)$ |
| s | – | $\log\mathcal{N}(1.25, 0.18)$ |
| $P_s$ | – | $\mathcal{N}(1.40, 0.21)$ |
| $L_h$ | $\log\mathcal{N}(0, 0.18)$ | $\mathcal{N}(1.20, 0.22)$ |
| $N_b$ | $\log\mathcal{N}(4.09, 0.18)$ | $\log\mathcal{N}(3.69, 0.29)$ |
| lr | $\log\mathcal{N}(-7.13, 0.26)$ | $\log\mathcal{N}(-7.26, 0.26)$ |
| b | $\log\mathcal{N}(3.22, 0.30)$ | $\log\mathcal{N}(3.09, 0.26)$ |
| e | $\log\mathcal{N}(4.61, 0.26)$ | $\log\mathcal{N}(4.58, 0.26)$ |
| dr | $\mathcal{N}(5.5e{-}2, 9e{-}3)$ | $\log\mathcal{N}(-3.00, 0.26)$ |

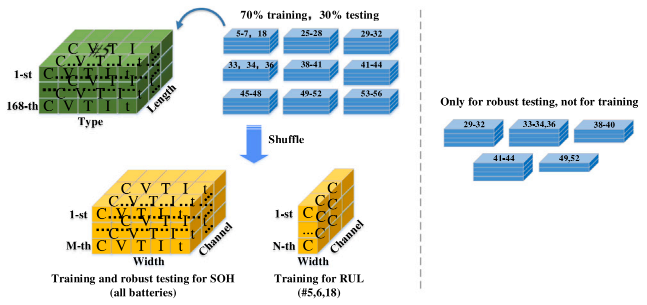


**Fig. 2.** Data curation for SOH and RUL tasks.

## 3. Numerical experiments

For demonstrating the effectiveness of proposed method, the results in [20] are selected as the benchmark in this study. We also harness the proposed optimization algorithm to RNN, S-LSTM NN, AST-LSTM NN [20], and CNN-LSTM NN [35] during the contrast experiments. The hardware and software configurations are the same as [20] except for an additional *hyperopt* [36]. The battery data comes from the NASA data repository [37] and its curation (Fig. 2) is similar to [20].

The discharge-process capacity and its corresponding current, voltage, sampling time, and ambient temperature from 34 cells (#5-7, 18, 25-32, 33-34, 36, 38-56,) are selected as a benchmark dataset. For each cell, we reshape the data into a three-dimensional array. The type dimension includes the data type, i.e., capacity (C), voltage (V), battery temperature (T), current (I), sampling time (t). The sampling dimension stores the specific values of each data type according to the sampling interval. The cycle dimension contains sequentially the No. of discharge-process. We randomly select 70% and 30% as training and test data during the SOH task, respectively. For the RUL task, we choose the first 50 cycles' data from 3 cells (#5, 6,18) as training data and the remaining cycles' data as testing data. In addition, to evaluate various model's robustness, we selected 16 cells (#29-34, 36, 38-44, 49, 52) data for testing capacity prediction. All these data do not participate in the training of these models.

### 3.1. Distribution over CNN-ASTLSTM NN hyperparameters

**SOH Task** For obtaining a priori distribution from the KS test, the hyperparameter configuration space is defined by a generative process drawing valid samples. Specifically, the hyperparameter assignments of CNN-ASTLSTM NN and CNN-LSTM NN are initialized as $\{L_c = 1, K_n = 32, K_s = 4, \mathsf{s} = 4, P_s = 2, L_h = 1, N_b = 32, N_l = 1, \mathsf{lr} = 0.001, \mathsf{b} = 64, \mathsf{e} = 50, \mathsf{dr} = 0.1\}$, where $L_c$, $K_n$, $K_s$, s, $P_s$, $L_h$, $N_b$, $N_l$, lr, b, e and dr denote the layers of CNN, kernels, kernel size, stride size, pool size, hidden layers of AST-LSTM or LSTM NN, blocks, neurons of last layer, learning rate, batch size, epoch and dropout rate, respectively. The initial hyperparameter assignments of RNN, LSTM NN, and ASTLSTM NN are the same as CNN-ASTLSTM NN except that they have not CNN hyperparameters. Note that $L_c$ is not optimized since the feature map length is not sufficient for the subsequent convolution after performing the first convolution and max pooling. According to supervised learning, the $s_w$ and $p_w$ $(N_l)$ also do not perform KS tests because of their manual configuration.

We specify ten random seeds for one of the hyperparameters to perform ten independent trials, keeping the remaining hyperparameters unchanged. We also select the mean square error (MSE) as the loss function. After running ten trials, we obtain ten models with a different configuration of such hyperparameter. An estimator with a standard kernel function is used to estimate the probability densities (Fig. 3) w.r.t. these loss function values.

The KS test is employed to determine which given distribution function is acceptable for the probability densities mentioned above. The candidate cumulative distribution functions are a uniform distribution $\mathcal{U}(a,b)$, a log-uniform distribution $\log\mathcal{U}(a,b)$, a Gaussian distribution $\mathcal{N}(\mu,\sigma)$, and a log-normal distribution $\log\mathcal{N}(\mu,\sigma)$. Here, $(a,b)$, $\mu$ and $\sigma$ are the interval, mean and standard deviation of the distribution. The test level is selected as $\alpha \leqslant 0.1$. If the estimated probability densities from ten trials are not acceptable at level $\alpha$, more trials will perform until such densities are acceptable by one of the given distribution functions. The determined distribution function types of these probability densities are considered the surrogate distribution types over actual hyperparameters. Combining the interval, mean and standard deviation observed during the above trials, Table 1 summarizes the obtained prior distribution over actual hyperparameters.

**RUL Task** It is necessary to reconsider the prior distribution over the NN's hyperparameters during the RUL tasks because of different inputs. We only choose CNN-ASTLSTM NN and AST-LSTM NN to compare since [20] reports AST-LSTM NN hits lower error versus other NNs. The process to build prior hyperparameter distribution for the RUL task, including hyperparameter initialization, probability density estimate, and KS test, is the same for the SOH task. Table 2 lists the prior hyperparameter distribution for the RUL task. Interestingly, even if the two tasks' data curation and goals are different, the distribution

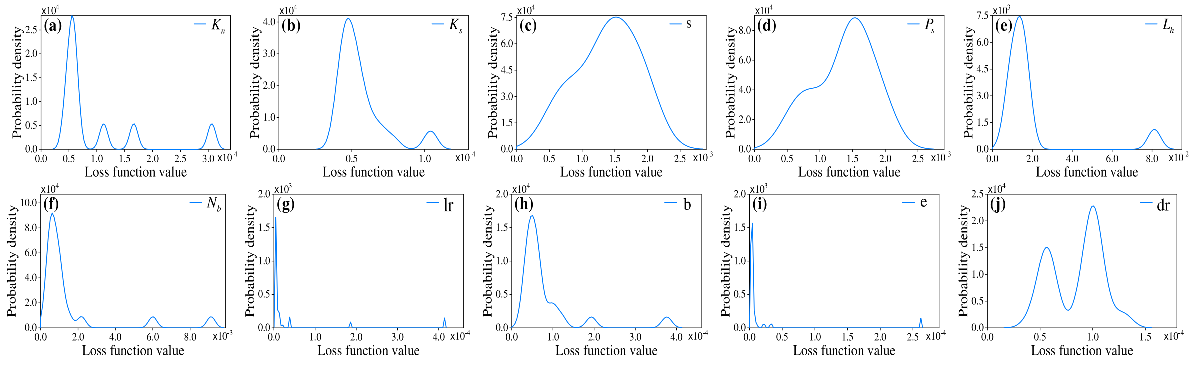


**Fig. 3.** Probability densities of the loss function values over CNN-ASTLSTM NN hyperparameters, where (a)–(j) represent the densities over $K_n$, $K_s$, s, $P_s$, $L_h$, $N_b$, lr, b, e and dr, respectively.

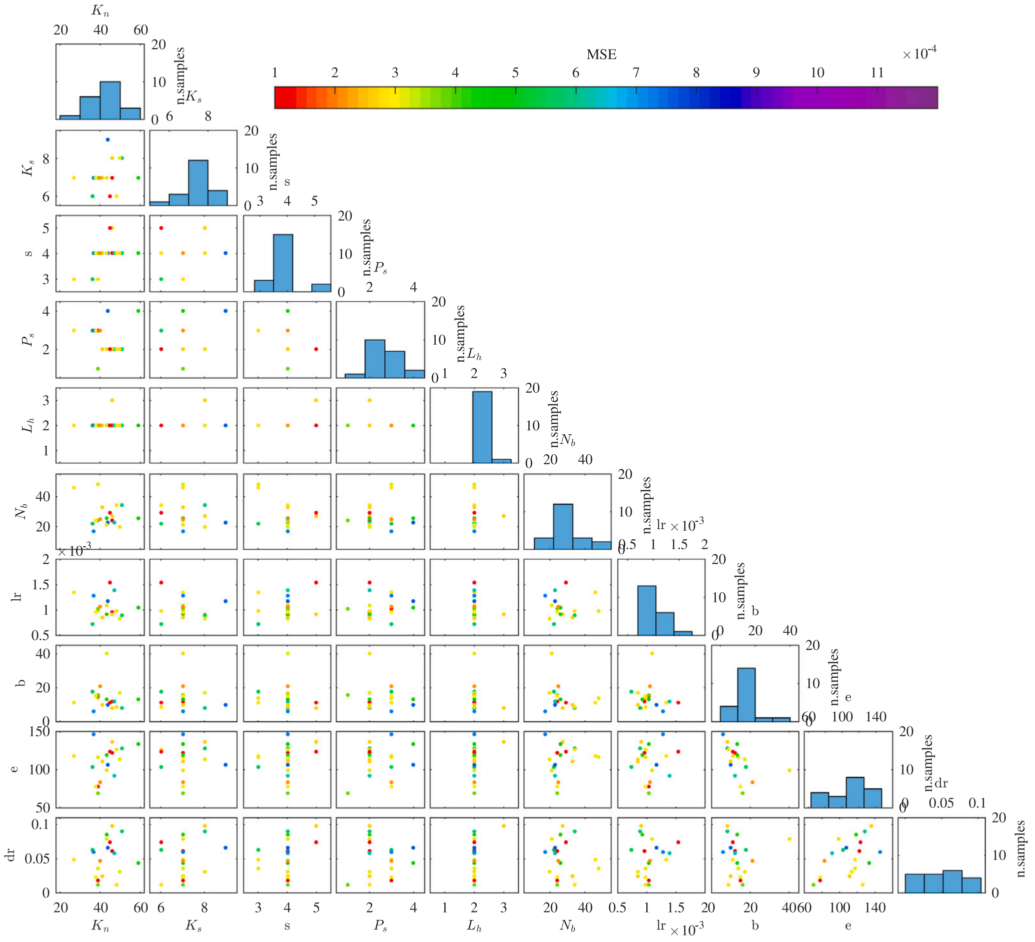


**Fig. 4.** Evolution of hyperparameter search during the optimization progress.

function types of most hyperparameters in the RUL task are the same as those in the SOH task, except for $L_h$, s and dr.

### 3.2. Off-line training for CNN-ASTLSTM NN

The training process of CNN-ASTLSTM NN includes automatic hyperparameter optimization and independent training based on the optimized hyperparameters.

#### 3.2.1. Automatic hyperparameter optimization

**SOH Task** The optimization algorithm works by identifying hyperparameter assignments that could have been drawn from the prior distribution given in Table 1, and that appear promising based on the loss function's value at other points. The complete optimization progress performs 20 runs for all the hyperparameters. After each run of hyperparameters on the MSE loss function, the optimization algorithm makes an educated guess which set of hyperparameters is the most likely to improve the score and should be tried in the next run. This process is done by getting regressor predictions on many points (hyperparameter sets) and choosing the point that is the best guess based on the EI criterion.

Fig. 4 describes the optimization progress comprehensively in the form of scatter plots and histograms. The histogram presents the

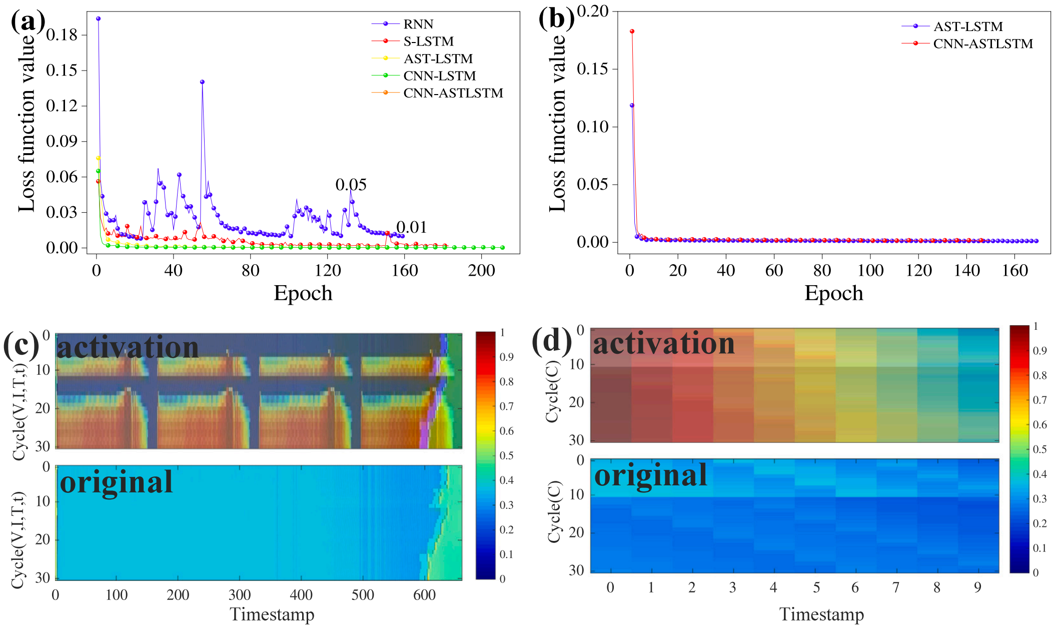


**Fig. 5.** Independent retraining processes, where (a)–(b) are the loss curves of NNs in SOH and RUL tasks, respectively; (c)–(d) are the activation maps of features extracted from CNN-ASTLSTM NN in SOH and RUL tasks, respectively.

**Table 3**
Optimized hyperparameters for the NNs applied in SOH task.

| Hyperparameters | Methods | | | | |
|---|---|---|---|---|---|
| | RNN | LSTM | AST-LSTM | CNN-LSTM | CNN-ASTLSTM |
| $L_c$ | – | – | – | 1 | 1 |
| $K_n$ | – | – | – | 34 | 46 |
| $K_s$ | – | – | – | 5 | 7 |
| s | – | – | – | 3 | 4 |
| $P_s$ | – | – | – | 7 | 2 |
| $L_h$ | 2 | 1 | 2 | 2 | 2 |
| $N_b$ | (32, 33) | 51 | (112, 122) | (76, 104) | (24, 28) |
| lr | 4.8e−3 | 1.4e−3 | 2.8e−3 | 6e−4 | 9e−4 |
| b | 34 | 1 | 14 | 26 | 12 |
| e | 160 | 183 | 153 | 212 | 122 |
| dr | 5.38e−2 | 1.177e−1 | 2.21e−2 | 5.53e−2 | 6.09e−2 |

explored values of each hyperparameter. We can find that each hyperparameter converges to certain parts of space that generate more heavy explorations. The scatter plot presents the sampled values, i.e., each point's position determined by a pair of hyperparameters. Besides, each point's color represents the loss function value (MSE) when such a pair of hyperparameters specify CNN-ASTLSTM NN. Table 3 gives the results of the automatic hyperparameter optimization. When all the NNs hit their best performance, CNN-ASTLSTM NN converges faster than others (e=122) .

**RUL Task** The RUL task's hyperparameters optimization process is the same as that of the SOH task, except for the configuration space drawn from Table 2. The optimization algorithm searches out the best hyperparameters for CNN-ASTLSTM NN and AST-LSTM NN (Table 4). Note the $N_l$, i.e., $p_w$ is manually set as 1, 2, 3, 4, and 5 because of supervised learning.

### 3.2.2. Independent training

Based on the optimized hyperparameters given in Tables 3 and 4, the NNs applied in SOH and RUL tasks are re-trained independently. Fig. 5(a)–(b) demonstrate all the NNs are well-trained. Note the RNN's loss curve shows a more significant fluctuation versus other NNs, but this is acceptable for its relatively low range (only 0.05-0.01=0.04). We also present the activation map of well-trained models in Fig. 5(c)–(d), taking 30 cycles' battery data as an example. We obtain the two original data matrices by reshaping an array with size {30(cycles) × 4(V, I, T, t) × 660(samples)} and an array with size {30(cycles) × 1(C) × 10(samples)}, respectively. For evaluating the model features' effects on prediction, the interest areas with different levels are superimposed on the original data. As increased battery data fed to CNN (from 1 cycle to 30 cycles) lead to expanded interest areas and augmented activation levels (from blue to red), the features captured from CNN provide a more significant impact on prediction.

### 3.3. On-line testing for CNN-ASTLSTM NN

For regular online SOH tests, the remaining 30% of history data per battery is fed to the well-trained SOH model to estimate capacity. The absolute error (AE), root mean square error (RMSE), and average RMSE (ARMSE) are regarded as evaluation indicators:

$$AE_i = |y_i - \hat{y}_i| \tag{28}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{29}$$

$$ARMSE = \frac{1}{M}\sum_{j=1}^{M} RMSE_j \tag{30}$$

where $y_i$, $\hat{y}_i$, $N$, $M$, and $RMSE_j$ represent the actual capacity, estimated capacity, capacity sequence length, battery number, and $j$th battery's RMSE, respectively.

From Fig. 6(a)–(l), we can find that the AE curves of the automatically optimized CNN-ASTLSTM NN are located lower than those of other automatically optimized NNs. Fig. 6(m), a box distribution drawn from Table 5, shows the maximum, up-quartile, median, low-quartile, and minimum of CNN-ASTLSTM NN reach a lower position than those of other NNs. For the robust online SOH test, similar to [20], we use the data of #29-34, 36, 38-44, 49, 51 batteries because of their severe performance degradation (dramatic low capacities and voltages). However, our testing condition is stricter than [20], i.e., we do not use those batteries' data as auxiliary data for model training. Fig. 6(s) plots the RMSEs box distribution by the results presented from Fig. 6(n) to Fig. 6(r). It shows our method hits better performance according to its lower up-quartile, median, low-quartile.

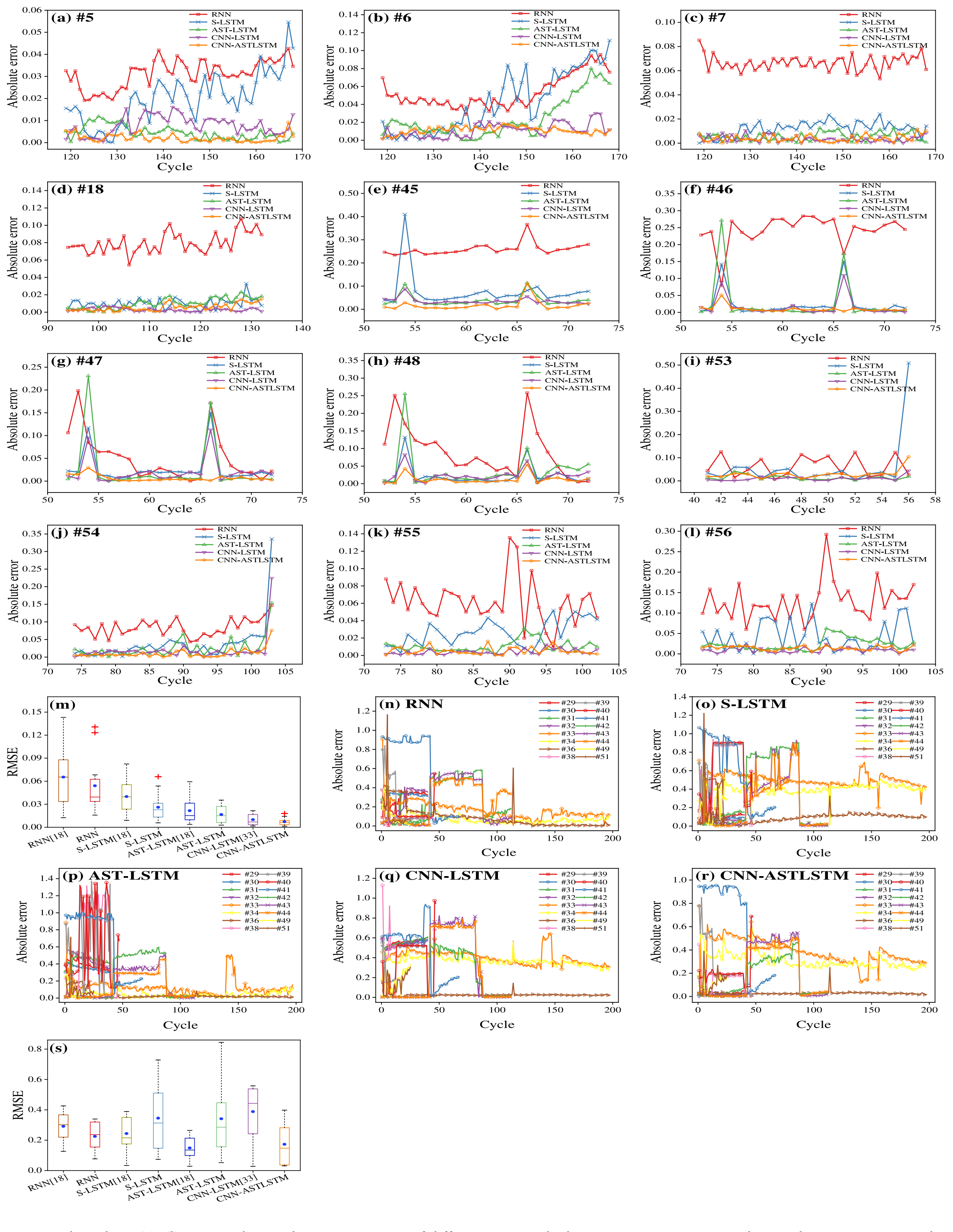


**Fig. 6.** Online SOH test results, where (a)–(l) present the regular test AE curves of different NNs applied on #5, 6, 7, 18, 45-48 and 53-56 batteries, respectively; (n)–(r) sequentially display the robust test AE curves for RNN, LSTM NN, AST-LSTM NN, CNN-LSTM NN and CNN-ASTLSTM NN, when such NNs are applied on #29-34, 36, 38-44, 49, 51 batteries; (m) and (s) are the box distributions of RMSEs for different methods applied for regular and robust tests, respectively.

For the multi-step RUL test, the #5 battery is selected as benchmark because of its typicality [20,22,38]. The multi-step prediction is implemented as updating data simultaneously per prediction [20,22], which is more challenging than counting the number of predictions. The AE indicator is rewritten as $AE = \left|RUL - \widehat{RUL}\right|$. The global ARMSE (GARMSE) and global average AE (GAAE) are introduced to evaluate the global performance of different methods.

$$GARMSE = \frac{1}{m}\sum_{j=1}^{m} ARMSE_j \tag{31}$$

$$GAAE = \frac{1}{m}\sum_{j=1}^{m} AAE_j \tag{32}$$

where $m$ is the types of $p_w$. We denote $ARMSE_j$ and $AAE_j$ to be the ARMSE and AAE when NNs are specified to $j$th type $p_w$. The predicted capacities get closer to the actual capacities when increasing $q_s$, while increasing $p_w$ results in the opposite (Fig. 7(a)–(j)). Fig. 7(k)–(l), drawn from RMSEs and AEs in Table 6, show that CNN-ASTLSTM NN hits better performance due to its lower box distribution.

**Table 4**
Optimized hyperparameters for the NNs applied in RUL task.

| Methods | Hyperparameters | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $L_c$ | $K_n$ | $K_s$ | s | $P_s$ | $L_h$ | $N_b$ | lr | b | e | dr |
| AST-LSTM($p_w$ = 1) | – | – | – | – | – | 1 | 84 | 2.6e−3 | 17 | 170 | 1.1e−2 |
| CNN-ASTLSTM($p_w$ = 1) | 1 | 52 | 6 | 3 | 2 | 1 | 23 | 1.2e−3 | 13 | 150 | 3.5e−2 |
| AST-LSTM($p_w$ = 2) | – | – | – | – | – | 1 | 95 | e−3 | 18 | 81 | 5.6e−2 |
| CNN-ASTLSTM($p_w$ = 2) | 1 | 87 | 3 | 3 | 1 | 1 | 42 | 1.1e−3 | 10 | 95 | 4.8e−2 |
| AST-LSTM($p_w$ = 3) | – | – | – | – | | 1 | 38 | 1.5e−3 | 9 | 76 | 6.6e−2 |
| CNN-ASTLSTM($p_w$ = 3) | 1 | 44 | 5 | 3 | 1 | 1 | 89 | 1.7e−3 | 17 | 231 | 0.11 |
| AST-LSTM($p_w$ = 4) | – | – | – | – | | 1 | 44 | 8.6e−4 | 18 | 116 | 6.3e−2 |
| CNN-ASTLSTM($p_w$ = 4) | 1 | 106 | 3 | 3 | 1 | 1 | 52 | e−3 | 15 | 189 | 3.7e−2 |
| AST-LSTM($p_w$ = 5) | – | – | – | – | | 1 | 59 | 7.9e−4 | 17 | 106 | 6.1e−2 |
| CNN-ASTLSTM($p_w$ = 5) | 1 | 75 | 7 | 5 | 1 | 1 | 72 | 9.7e−4 | 15 | 197 | 0.13 |

**Table 5**
Online RMSEs of different methods for regular SOH test.

| Method | Battery ID | | | | | | | | | | | | ARMSE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | #5 | #6 | #7 | #18 | #45 | #46 | #47 | #48 | #53 | #54 | #55 | #56 | |
| RNN [20] | 0.0162 | 0.0503 | 0.0257 | 0.0125 | 0.0770 | 0.0774 | 0.0634 | 0.0984 | 0.1115 | 0.1432 | 0.0673 | 0.0413 | 0.0654 |
| RNN@AutoML | 0.0159 | 0.0281 | 0.0334 | 0.0402 | 0.1307 | 0.1232 | 0.0363 | 0.0566 | 0.0385 | 0.0439 | 0.0338 | 0.0683 | 0.0541 |
| S-LSTM [20] | 0.0091 | 0.0385 | 0.0196 | 0.0122 | 0.0577 | 0.0276 | 0.0408 | 0.0626 | 0.0825 | 0.0534 | 0.0470 | 0.0289 | 0.0400 |
| S-LSTM@AutoML | 0.0115 | 0.0270 | 0.0067 | 0.0058 | 0.0538 | 0.0235 | 0.0222 | 0.0192 | 0.0659 | 0.0346 | 0.0150 | 0.0280 | 0.0261 |
| AST-LSTM [20] | 0.0038 | 0.0078 | 0.0085 | 0.0122 | 0.0260 | 0.0111 | 0.0368 | 0.0427 | 0.0594 | 0.0179 | 0.0126 | 0.0201 | 0.0216 |
| AST-LSTM@AutoML | 0.0028 | 0.0163 | 0.0032 | 0.0060 | 0.0230 | 0.0353 | 0.0316 | 0.0327 | 0.0088 | 0.0193 | 0.0063 | 0.0139 | 0.0166 |
| CNN-LSTM [35]@AutoML | 0.0044 | 0.0068 | 0.0022 | 0.0017 | 0.0189 | 0.0160 | 0.0174 | 0.0150 | 0.0072 | 0.0216 | 0.0025 | 0.0055 | 0.0099 |
| CNN-ASTLSTM@AutoML | 0.0014 | 0.0059 | 0.0024 | 0.0035 | 0.0143 | 0.0070 | 0.0047 | 0.0088 | 0.0180 | 0.0089 | 0.0035 | 0.0075 | 0.0072 |

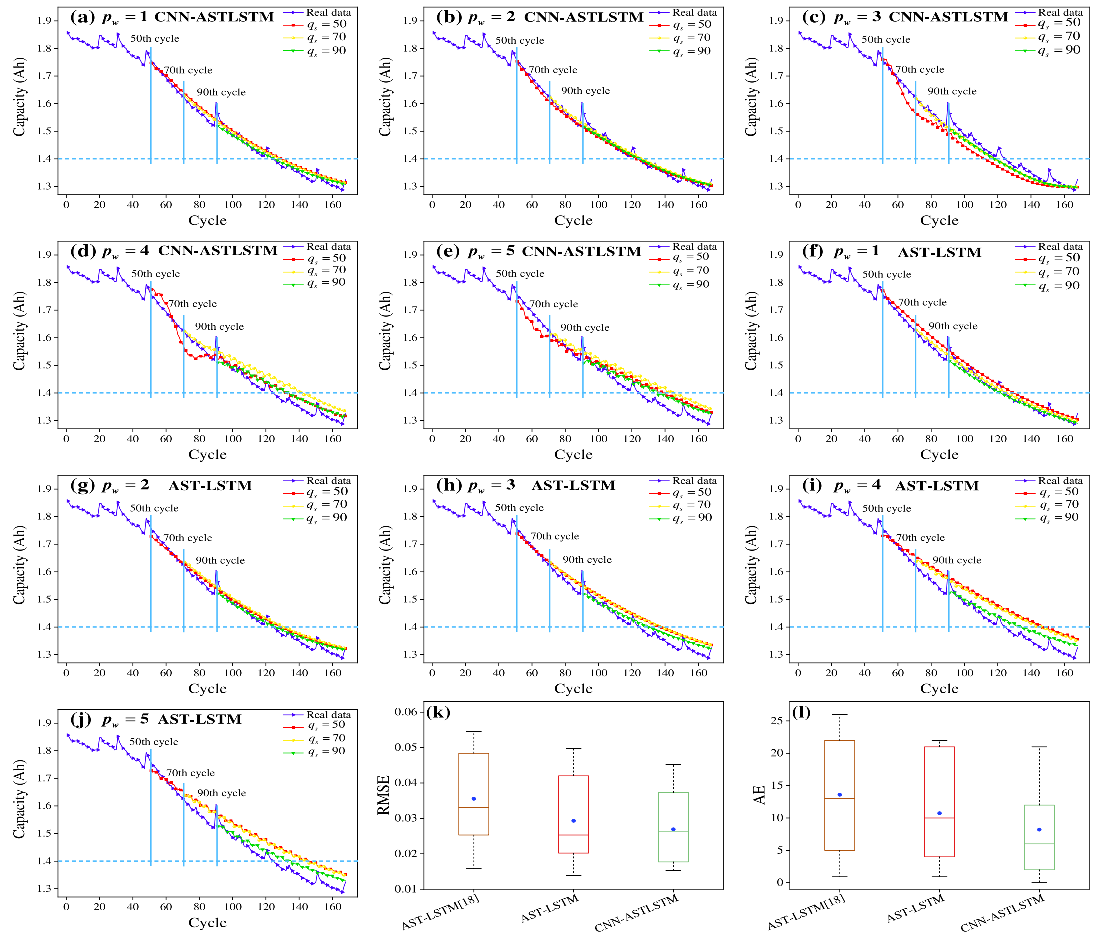


**Fig. 7.** Online multi-step RUL test results, where (a)–(j) present the predicted capacities of CNN-ASTLSTM NN and AST-LSTM NN when $p_w = \{1,2,3,4,5\}$ and $q_s = \{50,70,90\}$, respectively; (k) and (l) are the box distribution of RMSEs and AEs, respectively.

**Table 6**
Prediction errors of different methods for #5 battery.

| Methods | $p_w$ | SP | $\widehat{RUL}$ | RMSE | ARMSE | GARMSE | AE | AAE | GAAE |
|---|---|---|---|---|---|---|---|---|---|
| CNN-ASTLSTM@AutoML | 1 | 50 | 79 | 0.0183 | 0.0177 | 0.0269 | 5 | 3 | 9.4 |
| | | 70 | 58 | 0.0172 | | | 4 | | |
| | | 90 | 34 | 0.0176 | | | 0 | | |
| | 2 | 50 | 72 | 0.0186 | 0.0172 | | 2 | 1.3 | |
| | | 70 | 56 | 0.0153 | | | 2 | | |
| | | 90 | 34 | 0.0177 | | | 0 | | |
| | 3 | 50 | 62 | 0.0452 | 0.0333 | | 12 | 8 | |
| | | 70 | 48 | 0.0262 | | | 6 | | |
| | | 90 | 28 | 0.0284 | | | 6 | | |
| | 4 | 50 | 84 | 0.0320 | 0.0313 | | 10 | 12.7 | |
| | | 70 | 72 | 0.0373 | | | 18 | | |
| | | 90 | 44 | 0.0246 | | | 10 | | |
| | 5 | 50 | 90 | 0.0389 | 0.0350 | | 16 | 16 | |
| | | 70 | 75 | 0.0395 | | | 21 | | |
| | | 90 | 45 | 0.0267 | | | 11 | | |
| AST-LSTM@AutoML | 1 | 50 | 79 | 0.0213 | 0.0180 | 0.0293 | 5 | 3 | 10.7 |
| | | 70 | 55 | 0.0139 | | | 1 | | |
| | | 90 | 31 | 0.0189 | | | 3 | | |
| | 2 | 50 | 78 | 0.0202 | 0.0211 | | 4 | 4 | |
| | | 70 | 60 | 0.0234 | | | 6 | | |
| | | 90 | 36 | 0.0196 | | | 2 | | |
| | 3 | 50 | 87 | 0.0316 | 0.0295 | | 13 | 11 | |
| | | 70 | 69 | 0.0331 | | | 15 | | |
| | | 90 | 39 | 0.0238 | | | 5 | | |
| | 4 | 50 | 96 | 0.0497 | 0.0414 | | 22 | 18 | |
| | | 70 | 76 | 0.0466 | | | 22 | | |
| | | 90 | 44 | 0.0278 | | | 10 | | |
| | 5 | 50 | 95 | 0.0425 | 0.0367 | | 21 | 17.7 | |
| | | 70 | 75 | 0.0420 | | | 21 | | |
| | | 90 | 45 | 0.0253 | | | 11 | | |
| AST-LSTM [20] | 1 | 50 | 70 | 0.0253 | 0.0199 | 0.0592 | 4 | 2 | 13.6 |
| | | 70 | 55 | 0.0159 | | | 1 | | |
| | | 90 | 33 | 0.0186 | | | 1 | | |
| | 2 | 50 | 87 | 0.0331 | 0.0301 | | 13 | 10.3 | |
| | | 70 | 67 | 0.0349 | | | 13 | | |
| | | 90 | 39 | 0.0223 | | | 5 | | |
| | 3 | 50 | 93 | 0.0441 | 0.0403 | | 19 | 15 | |
| | | 70 | 72 | 0.0484 | | | 18 | | |
| | | 90 | 42 | 0.0284 | | | 8 | | |
| | 4 | 50 | 96 | 0.0471 | 0.0416 | | 22 | 18 | |
| | | 70 | 76 | 0.0501 | | | 22 | | |
| | | 90 | 44 | 0.0277 | | | 10 | | |
| | 5 | 50 | 100 | 0.0523 | 0.0456 | | 26 | 22.7 | |
| | | 70 | 80 | 0.0545 | | | 26 | | |
| | | 90 | 50 | 0.0301 | | | 16 | | |

## 3.4. Comprehensive performance analysis for CNN-ASTLSTM NN

As automotive electronics technology is constantly evolving, the next-generation electrical-electronic architecture will transfer the BMS's data processing and algorithm calculating functions to a domain processor. This processor's computing power is more potent than BMS, providing opportunities for neural networks' practical application. Therefore, it is necessary to analyze the aforementioned methods' all-around performance. Our motivation is to find a better trade-off between various indicators, given by application scenarios in the future, to estimate the battery's health state. Besides, we also try to use such analysis to show that automatic hyperparameter optimization can decrease manual intervention, reduce training difficulty, and improve the prediction model's other performance.

The evaluation indicators include FLOPs, parameters, training time, inference time (latency), and storage size. The definition of FLOPs follows [39], i.e. the number of multiply-adds, an indirect metric to measure the computation complexity. The parameters, an indirect indicator affecting the model's storage size, are counted directly by the *model.summary* function built-in Keras. The training time and inference latency are counted by the *time.time* function in the *TIME LIBRARY* of Python. Specifically, we count a latency when the SOH model outputs an estimated capacity, then obtain the mean and standard deviation of these latencies when the networks run through all the testing samples (50 cycles' capacities, #5 battery). The storage size is calculated by the *os.path.getsize* function in *OS LIBRARY* of python.

Table 7 shows that each index of the automatically optimized models is lower than that of the corresponding manually tuned models. All indicators except two (Training time and Inference latency) of our method are better than those of CNN-LSTM NN when using the proposed automatic optimization. It is worth mentioning that the model's accuracy, besides the training costs (Training time, FLOPs, and Parameters) and hardware costs (Inference time and Storage size), should also be considered during the actual application. Therefore, combining Tables 5 and 7, we use a pairwise combination of indicators to draw performance distribution of all the methods. From Fig. 8, we find that our method is located at the bottom left side in each performance distribution, reaching the best trade-off.

## 4. Conclusion

This study presents an end-to-end prognostic framework to automatically extract hierarchical features of battery degeneration data and optimize the NN's hyperparameters. We have described our efforts to harness AST-LSTM NN's idea and the advances in network design to deliver the CNN-ASTLSTM NN. We have also shown how to apply the KS test to get the prior distribution over hyperparameters. This prior distribution is regarded as a surrogate for evaluating the battery data impact on the modeling process of the NNs during SOH or RUL tasks. For automatically selecting the most promising hyperparameter configuration, we also introduced such prior distribution to the Bayesian optimization algorithm to build SOH and RUL models. Compared with the existing NNs, the experiments indicate our method hits lower

**Table 7**
Comprehensive performance of different methods in SOH estimation task.

| Methods | Performance index | | | | |
|---|---|---|---|---|---|
| | FLOPs (Million) | Parameters (Million) | Training time (h) | Inference time (s) | Storage size (KB) |
| RNN@AutoML | 0.050 | 0.003 | 0.219 | 0.032 ± 0.014 | 73 |
| RNN [20] | 0.244 | 0.016 | 0.359 | 0.016 ± 0.012 | 225 |
| S-LSTM@AutoML | 0.197 | 0.011 | 12.713 | 0.037 ± 0.014 | 160 |
| S-LSTM [20] | 1.170 | 0.065 | 0.746 | 0.033 ± 0.023 | 796 |
| AST-LSTM@AutoML | 2.753 | 0.168 | 3.644 | 0.084 ± 0.026 | 2004 |
| AST-LSTM [20] | 1.092 | 0.065 | 0.992 | 0.034 ± 0.025 | 804 |
| CNN-LSTM@AutoML | 1.950 | 0.110 | 0.175 | 0.009 ± 0.028 | 1326 |
| CNN-ASTLSTM@AutoML | 0.237 | 0.014 | 0.200 | 0.014 ± 0.029 | 211 |

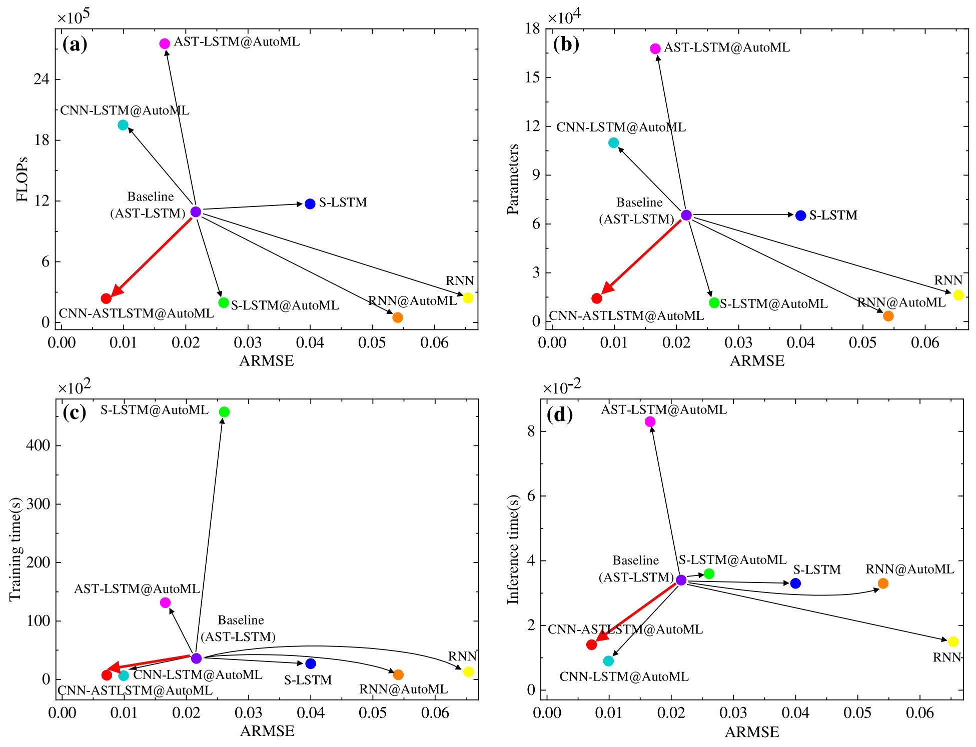


**Fig. 8.** All the methods' performance distribution, where (a)–(b) are the FLOPs-ARMSE, Parameters-ARMSE, Training time-ARMSE, and Inference latency-ARMSE distributions.

ARMSE 0.0072 and GARMSE 0.0269 for SOH and RUL tasks, respectively. We have also employed the indicators that will be considered during the actual future applications, e.g., FLOPs, Parameters, Training time, and Inference latency, to conduct a comprehensive analysis of the existing NN-based methods. Related results show that our method achieves the best trade-off performance.

## CRediT authorship contribution statement

**Penghua Li:** Conceptualization, Methodology, Software, Writing – original draft, Writing – review& editing. **Zijian Zhang:** Software, Validation, Visualization. **Radu Grosu:** Methodology, Validation, Formal analysis. **Zhongwei Deng:** Visualization, Writing – review & editing. **Jie Hou:** Investigation, Data curation. **Yujun Rong:** Funding acquisition, Project administration. **Rui Wu:** Resources, Supervision.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work is supported by Chongqing Outstanding Youth Fund Project (cstc2021jcyj-jqX0001); Science and Technology Research Project of Chongqing Education Commission (KJZD-K202100603); National Natural Science Foundation of China (61903057); Chongqing Natural Science Foundation (cstc2019jcyj-msxmX0129); the Project of Innovation Research Group of Universities in Chongqing (CXQT20016).

## References

[1] Hua Y, Zhou S, Huang Y, Liu X, Ling H, Zhou X, et al. Sustainable value chain of retired lithium-ion batteries for electric vehicles. J Power Sources 2020;478:228753.
[2] Yang S, He R, Zhang Z, Cao Y, Gao X, Liu X. Chain: Cyber hierarchy and interactional network enabling digital solution for battery full-lifespan management. Matter 2020.
[3] Hu X, Feng F, Liu K, Zhang L, Xie J, Liu B. State estimation for advanced battery management: Key challenges and future trends. Renew Sust Energ Rev 2019;114:109334.
[4] Hu C, Ye H, Jain G, Schmidt C. Remaining useful life assessment of lithium-ion batteries in implantable medical devices. J Power Sources 2018;375:118–30.
[5] Harris SJ, Harris DJ, Li C. Failure statistics for commercial lithium ion batteries: A study of 24 pouch cells. J Power Sources 2017;342:589–97.

[6] Weng C, Feng X, Sun J, Peng H. State-of-health monitoring of lithium-ion battery modules and packs via incremental capacity peak tracking. Appl Energy 2016;180:360–8.

[7] Liu G, Ouyang M, Lu L, Li J, Han X. Online estimation of lithium-ion battery remaining discharge capacity through differential voltage analysis. J Power Sources 2015;274:971–89.

[8] Ge M-F, Liu Y, Jiang X, Liu J. A review on state of health estimations and remaining useful life prognostics of lithium-ion batteries. Measurement 2021;109057.

[9] Smith KA, Rahn CD, Wang C-Y. Model-based electrochemical estimation and constraint management for pulse operation of lithium ion batteries. IEEE Trans Control Syst Technol 2009;18(3):654–63.

[10] Kong X, Plett GL, Trimboli MS, Zhang Z, Qiao D, Zhao T, et al. Pseudo-two-dimensional model and impedance diagnosis of micro internal short circuit in lithium-ion cells. J Energy Storage 2020;27:101085.

[11] Liu X, Hui SR. Equivalent circuit modeling of a multilayer planar winding array structure for use in a universal contactless battery charging platform. IEEE Trans Power Electron 2007;22(1):21–9.

[12] Liu K, Ashwin T, Hu X, Lucu M, Widanage WD. An evaluation study of different modelling techniques for calendar ageing prediction of lithium-ion batteries. Renew Sust Energ Rev 2020;131:110017.

[13] Li Y, Liu K, Foley AM, Zülke A, Berecibar M, Nanini-Maury E, Van Mierlo J, et al. Data-driven health estimation and lifetime prediction of lithium-ion batteries: A review. Renew Sust Energ Rev 2019;113:109254.

[14] Lai X, Huang Y, Deng C, Gu H, Han X, Zheng Y, et al. Sorting, regrouping, and echelon utilization of the large-scale retired lithium batteries: A critical review. Renew Sust Energ Rev 2021;146:111162.

[15] Wu J, Wang Y, Zhang X, Chen Z. A novel state of health estimation method of Li-ion battery using group method of data handling. J Power Sources 2016;327:457–64.

[16] Dai H, Zhao G, Lin M, Wu J, Zheng G. A novel estimation method for the state of health of lithium-ion battery using prior knowledge-based neural network and Markov chain. IEEE T Ind Electron 2018;66(10):7706–16.

[17] Eddahech A, Briat O, Bertrand N, Deletage J-Y, Vinassa J-M. Behavior and state-of-health monitoring of Li-ion batteries using impedance spectroscopy and recurrent neural networks. Int J Electr Power 2012;42(1):487–94.

[18] Chaoui H, Ibe-Ekeocha CC. State of charge and state of health estimation for lithium batteries using recurrent neural networks. IEEE Trans Veh Technol 2017;66(10):8773–83.

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

[20] Li P, Zhang Z, Xiong Q, Ding B, Hou J, Luo D, Rong Y, et al. State-of-health estimation and remaining useful life prediction for the lithium-ion battery based on a variant long short term memory neural network. J Power Sources 2020;459:228069.

[21] Li X, Zhang L, Wang Z, Dong P. Remaining useful life prediction for lithium-ion batteries based on a hybrid model combining the long short-term memory and Elman neural networks. J Energy Storage 2019;21:510–8.

[22] Zhang Y, Xiong R, He H, Pecht MG. Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries. IEEE Trans Veh Technol 2018;67(7):5695–705.

[23] Zhou Y, Huang Y, Pang J, Wang K. Remaining useful life prediction for supercapacitor based on long short-term memory neural network. J Power Sources 2019;440:227149.

[24] Yu Y, Hu C, Si X, Zheng J, Zhang J. Averaged Bi-LSTM networks for RUL prognostics with non-life-cycle labeled dataset. Neurocomputing 2020.

[25] Ma G, Zhang Y, Cheng C, Zhou B, Hu P, Yuan Y. Remaining useful life prediction of lithium-ion batteries based on false nearest neighbors and a hybrid neural network. Appl Energy 2019;253:113626.

[26] Fan Y, Xiao F, Li C, Yang G, Tang X. A novel deep learning framework for state of health estimation of lithium-ion battery. J Energy Storage 2020;32:101741.

[27] Bergstra JS, Bardenet R, Bengio Y, Kégl B. Algorithms for hyper-parameter optimization. In: Advances in neural information processing systems. 2011, p. 2546–54.

[28] Hu J, Shen L, Sun G. Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018. p. 7132–41.

[29] Bergstra J, Bengio Y. Random search for hyper-parameter optimization. J Mach Learn Res 2012;13(1):281–305.

[30] Thornton C, Hutter F, Hoos HH, Leyton-Brown K. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining. 2013. p. 847–55.

[31] Kohavi R, et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of IJCAI. 1995. p. 1137–45.

[32] Hutter F, Hoos HH, Leyton-Brown K. Sequential model-based optimization for general algorithm configuration. In: International conference on learning and intelligent optimization. 2011. p. 507–23.

[33] Desautels T, Krause A, Burdick JW. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. J Mach Learn Res 2014;15:3873–923.

[34] Simard R, L'Ecuyer P, et al. Computing the two-sided Kolmogorov-Smirnov distribution. J Stat Softw 2011;39(11):1–18.

[35] Kim T-Y, Cho S-B. Predicting residential energy consumption using CNN-LSTM neural networks. Energy 2019;182:72–81.

[36] Bergstra J, Bardenet R, Kégl B, Bengio Y. Implementations of algorithms for hyper-parameter optimization. In: NIPS workshop on bayesian optimization. 2011. p. 29.

[37] Agogino A, Goebel K. Mill data set. BEST lab, UC berkeley. NASA ames prognostics data repository. 2007, http://ti.are.nasa.gov/prolect/prognostie-data-repository.

[38] Yu J. State of health prediction of lithium-ion batteries: Multiscale logic regression and Gaussian process regression ensemble. Reliab Eng Syst Saf 2018;174:82–95.

[39] Ma N, Zhang X, Zheng H-T, Sun J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the european conference on computer vision. 2018. p. 116–31.