Journal of Energy Storage 86 (2024) 111396

Contents lists available at ScienceDirect

Journal of Energy Storage

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

Research Papers

A structural pruning method for lithium-ion batteries remaining useful life prediction model with multi-head attention mechanism

Yang Ge *, Jiaxin Ma, Guodong Sun

School of Mechanical Engineering, Changshu Institute of Technology, Changshu 215500, China

ARTICLE INFO

Keywords:
RUL prediction
Lithium-ion batteries
Multi-head attention
Network pruning
Self-adaptation

ABSTRACT

The utilization of data-driven models in predicting the remaining useful life (RUL) of lithium-ion batteries has gained traction. However, deploying these models on edge devices presents challenges attributed to their limited generalization capacity and substantial computational demands amidst varying operational conditions. In this study, we formulate a RUL prediction model for lithium-ion batteries by leveraging a multi-layer convolutional network alongside a multi-head attention mechanism. Through the incorporation of a domain adapter and a residual structure, we bolster the model's generalization and anti-overfitting capabilities. Additionally, we introduce an innovative global pruning strategy that hinges on scaling coefficients and adaptive sparse pruning. This strategy outperforms existing pruning methods in terms of model parameter reduction, prediction accuracy, and operational efficiency. The amalgamation of our proposed model and pruning strategy proffers a viable avenue for implementing data-driven models within embedded battery management systems.

1. Introduction

Lithium-ion batteries have emerged as pivotal energy supply components in modern industries, permeating nearly every facet of daily life, such as aerospace, transportation, portable electronic devices, and medical applications. However, these batteries have finite operational lifespans, which vary depending on usage scenarios, and do not have a standardized threshold or a consistent performance. In some domains, such as medical treatment and electric vehicles, it is crucial to accurately estimate the RUL of lithium-ion batteries, as any failure may pose serious safety risks. Therefore, the precise real-time prediction of battery longevity has become a pressing concern and attracted increasing attention [1].

With the advancement of production technology, the electrochemical system of lithium-ion batteries has become complex, and the failure modes have shown diversity. The degradation data of lithium-ion batteries have large dispersion, which leads to large errors in the traditional mathematical model prediction methods, such as electrochemical model [2], equivalent circuit model [3], etc. Data-driven prediction methods can automatically learn the degradation rules of batteries under the condition of sufficient labeled training data, thus skipping the exploration of complex electrochemical mechanisms, and gradually become a hot topic of research [4]. Common methods include grey prediction [4], particle filter [5,6], least squares support vector machine [7], ensemble empirical mode decomposition [8], Wiener process model [9], Gaussian process regression [10,11], deep autoencoder network [12], long short-term memory neural network [13,14], t-SNE [15], deep learning neural network [16,17], gated recurrent unit neural network [18–20], Box-Cox transformation [21], Integrated approach [22–25], temporal convolutional network [26], etc. Ly et al. [27] employed the concept of T-shape data and proposed multiple non-crossing quantiles Long Short-Term Memory. Among these methods, deep learning models often have better regression prediction performance than shallow models [28], but they still face the problem of weak generalization ability when predicting battery RUL under varying working conditions [29]. For this purpose, many adaptive methods, transfer learning methods, etc. have been introduced into battery RUL prediction. Wang et al. [30] established an autoregressive integrated moving average-long short-term memory combined model. Qu et al. [31] introduced particle swarm optimization and attention mechanism into deep learning model to improve the prediction accuracy of the model. Deep learning models often require sufficient training samples, otherwise the prediction accuracy is hard to guarantee, and there are few samples of new batteries. For this reason, Zhang et al. [32] put forward a novel deep adaptive continuous time-varying cascade network, which is grounded on extreme learning machines. This

* Corresponding author.
E-mail address: geyang@cslg.edu.cn (Y. Ge).

https://doi.org/10.1016/j.est.2024.111396
Received 8 October 2023; Received in revised form 8 March 2024; Accepted 18 March 2024
Available online 23 March 2024

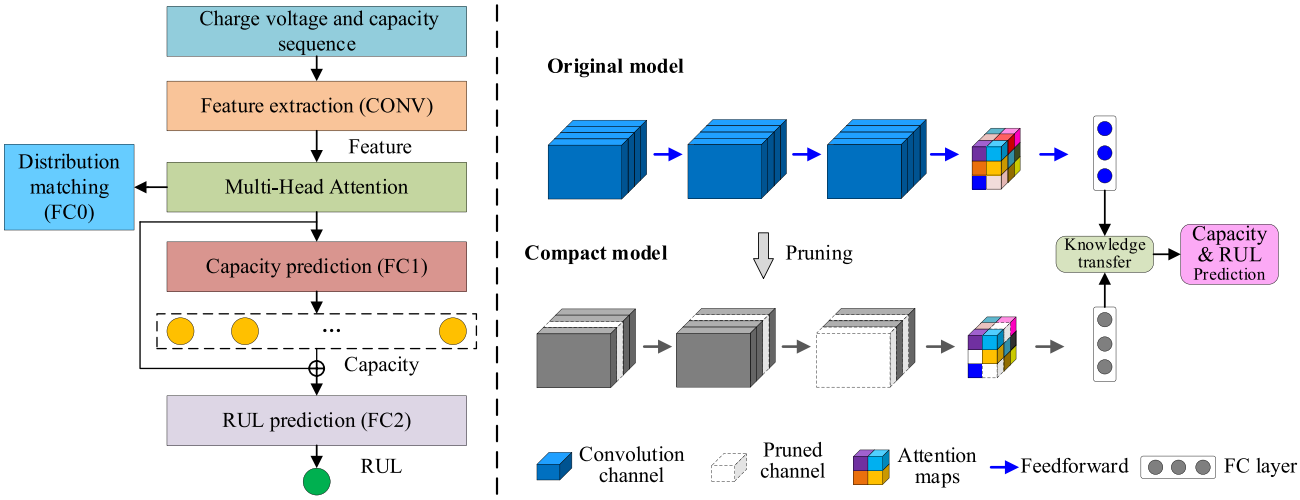

Fig. 1. The proposed predictive framework.

proposal was particularly designed to operate under conditions with limited sample sizes. In addition, bootstrap random vector functional link model [33], capsule network [34], stacked denoising autoencoder [35] and other methods also appeared in recent research, which are used to improve the battery RUL prediction accuracy across working conditions.

In fact, although deep learning models have very excellent prediction effects, they still face many problems in practical engineering applications. One of the urgent problems is the high computational cost and hardware requirements of deep models. We know that in order to reduce costs, the management devices of lithium-ion batteries are often embedded microprocessors with low computing power, and it is obviously impractical to use these devices to run deep learning models. For this reason, the compression and pruning methods of large models have become an effective way to solve this problem. Li et al. [36] proposed a neuron selection method based on contribution to reduce the convolutional network-based battery RUL prediction model, greatly reducing the computational cost. Many lossless compression techniques have been reported, such as multi-objective pruning based on feature mapping selection [37], neural architecture search [38], pruning compensation technique [39], etc. These methods and strategies have been proposed to improve the compression ratio of the pruning model and ensure the prediction accuracy of the pruning model. For multi-layer convolutional networks, conventional pruning methods basically prune each layer according to the same proportion, but the contribution of each convolutional layer is often inconsistent, which can easily lead to over-pruning and under-pruning problems in different convolutional layers [40].

In view of the problems of weak generalization ability and high computational cost of battery RUL prediction models on edge devices, which make it difficult to adapt the prediction models to embedded systems, this paper provides solutions for two problems: (1) how to construct a prediction model with stronger generalization ability; (2) how to reduce the computational cost of the prediction model, so that it can be applied to embedded systems on edge devices. The main contributions of this paper are as follows:

First, we propose a new joint prediction model of battery capacity and RUL based on multi-layer convolutional network and multi-head attention mechanism. The model uses partial charging data of lithium-ion batteries as input, extracts features based on multi-layer convolutional network, and automatically pays attention to the correlation between the extracted features and lithium-ion batteries degradation by using multi-head attention mechanism, which improves the expressive ability of the model. In addition, we also add domain adapters to the prediction model to force the feature extractor to extract common features of different working conditions and degradation stages, and add residual structure to enhance the anti-overfitting ability of the model.

Second, we propose a new global pruning strategy based on scaling coefficients and adaptive sparsity for multi-layer convolutional structure. We introduce trainable adaptive scaling coefficients for each convolutional kernel, which on the one hand represent the importance of each convolutional kernel in the corresponding convolutional layer, and on the other hand construct a global importance evaluation formula based on scaling coefficients. Using this formula, we can evaluate the contribution of each convolutional kernel in the whole model, and use this global importance to prune the model. Compared with other pruning methods, this method can achieve higher model parameter removal rate, better prediction accuracy and lower computational cost.

The rest of this paper is organized as follows: Section 2 constructs a lithium-ion batteries RUL prediction model based on multi-layer convolutional network and multi-head attention mechanism. Section 3 proposes a global adaptive pruning strategy. Section 4 experimentally verifies the superiority of the proposed model and pruning strategy. Section 5 is the summary of the paper.

2. RUL prediction model

Extracting the common features of the data is one of the keys to using deep learning models to extract features and predict RUL for battery degradation data with dispersion caused by usage scenarios, manufacturing differences, etc. For this purpose, this paper constructs a multi-layer convolutional network for feature extraction, and then uses a multi-head attention layer to capture the common information of the features, which improves the expression and generalization ability of the prediction model. In practical applications, battery RUL prediction is usually performed on edge devices. Considering the limited computing power of edge devices, this paper proposes a pruning strategy based on scaling coefficients and sparsity values, which compresses the prediction model, reduces its computational cost, and facilitates practical applications.

Structural pruning plays a pivotal role in the modeling process, particularly in the field of machine learning and neural networks. It is a technique that involves the removal of non-contributing or less-contributing nodes or connections in a model, thereby simplifying the model's architecture without significantly compromising its performance. The importance of structural pruning lies in its ability to reduce the computational complexity of models, making them more efficient and faster to train. This is especially crucial in real-world applications where computational resources and time are often limited. Moreover, structural pruning can help mitigate the issue of overfitting by

Table 1
Parameters of the feature extraction module.

Layer name	Input channels	Output channels	Kernel size	Activation function
Conv 1	2	32	400	Leaky ReLU (0.5)
Conv 2	32	64	400	Leaky ReLU (0.5)
Conv 3	64	128	195	Leaky ReLU (0.5)

eliminating unnecessary complexity in the model, leading to improved generalization on unseen data. In essence, structural pruning is a powerful tool in model optimization, striking a balance between model complexity and performance, and hence, its consideration in the modeling process is of paramount importance.

2.1. General framework

The proposed prediction model framework is shown in Fig. 1. The feature extraction module of the prediction model consists of three convolutional layers, and the specific parameters are shown in Table 1. The multi-head attention layer is located after the feature extractor, which is used to capture the common information of the features. The distribution adapter consists of a fully connected layer, which is used to adjust the distribution differences between different degradation stages and working conditions, and to learn the common knowledge of different stages and conditions through the adapter. The battery capacity and RUL predictors are composed of a fully connected layer each. The enhanced features generated by the multi-head attention layer are used as the input of the capacity predictor, and the enhanced features and the capacity prediction results are connected by residuals and used as the input of the battery life predictor. The details of model training and testing are given in the experimental section.

Multi-head attention is a powerful mechanism in deep learning, which introduces multiple attention heads, transforms the input into different linear spaces, calculates the attention of each head separately, and then fuses the attention results of multiple heads, to capture the information of different feature spaces. It can enhance the model's ability to learn different feature representations and improve the model's expressive ability. Compared with single-head attention, its advantage is that it allows the model to pay attention to different parts of the input sequence, making it more flexible in dealing with various features. In addition, it can also compute in parallel, improving the model's training and inference efficiency to some extent. The basic principle is as follows:

Assuming that the input signal segment is $X \in \mathrm{R}^{s\times t}$, the single-head self-attention mechanism operation can be expressed as:

$$A(X) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where $Q = XW_q$, $K = XW_k$, $V = XW_v$ represent the query, key and value respectively, $W_q, W_k \in \mathrm{R}^{t\times d_h}$, $W_v \in \mathrm{R}^{t\times d_v}$, and d_k is the scaling factor.

The expression of multi-head merging is as follows:

$$MHA(X) = Concat(A_1(X), \cdots, A_h(X))W_0 \tag{2}$$

where $A_i(X)$ represents the attention output of the i-th head, h is the number of heads, which is set to $h = 8$ in this paper, *Concat* means concatenating the outputs of these subspaces, and W_0 is the weight matrix of the output, $W_0 \in \mathrm{R}^{d_v\times d_v}$.

2.2. Loss function

We use the mean square error (MSE) as the loss function of the prediction results, which includes the battery discharge capacity loss (Eq. (3)) and the RUL loss (Eq. (4)).

$$L_C = \frac{1}{N_C}\sum_{i=1}^{N_C}\left(y_C^i - \hat{y}_C^i\right)^2 \tag{3}$$

where N_C represents the number of predicted capacities, y_C^i represents the true value of capacity, and $\hat{y}_C^i$ represents the predicted value of capacity.

$$L_{RUL} = \frac{1}{N_R}\sum_{i=1}^{N_R}\left(y_{RUL}^i - \hat{y}_{RUL}^i\right)^2 \tag{4}$$

where N_R represents the number of predicted RUL, y_{RUL}^i and $\hat{y}_{RUL}^i$ represent the true value and the predicted value of RUL respectively.

Simultaneously, aiming to mitigate distribution discrepancies across distinct degradation stages and operational conditions, we devised a distribution adapter. This adapter is conceived to assimilate the shared insights from various stages and conditions. Comprising a fully connected layer enhanced by a sigmoid activation function, the adapter's optimization is governed by the binary cross-entropy loss, as depicted in Eq. (5).

$$L_{\mathrm{match}} = -log(m_\zeta(f_\theta(x_i)))\tag{5}$$

where $m_\zeta(\cdot)$ represents the adaptation operation, and $f_\theta(\cdot)$ represents feature extraction.

Therefore, the overall loss function of the prediction model can be expressed as:

$$L_{Ori} = L_{RUL} + \beta L_C + \gamma L_{\mathrm{match}} \tag{6}$$

where β and γ are adjustment parameters.

3. Structured pruning strategy

The prediction model presented in this paper encompasses a series of convolutional layers, and the quantity of filters within these convolutional layers significantly influences the overall operational efficiency of the model. Without compromising the integrity of the original network architecture, a reduction in the number of filters within the convolutional layers can profoundly alleviate the parameter load and computational demands of the entire model. To this end, this paper introduces a pruning strategy founded on scaling coefficients and sparsity metrics. Initially employing L1 regularization to induce sparsity in the model, the strategy subsequently prunes filters within the convolutional layers based on parameter characteristics and scaling coefficients associated with these layers.

3.1. Sparse training

Model sparsification yields sparse network parameters, thereby streamlining subsequent model pruning efforts. Conventional sparsification techniques frequently involve introducing a penalty term into the loss function during model training, resulting in a broad sparsification effect across all network parameters. In contrast, our approach integrates a scaling coefficient post each convolutional layer, contingent on the number of output feature channels within said layer. This entails multiplying the output of each filter in every convolutional layer by its corresponding coefficient, of course, these coefficients remain trainable. The sparsification training occurs concomitantly with the model training process, employing the ensuing loss function:

$$L_S = L_{ori} + \lambda\left(\sum_{ij}|\alpha_{ij}| + \sum_P|\theta_p|\right) \tag{7}$$

where L_{ori} is the original training loss of the model, λ is the regularization coefficient, α_{ij} is the scaling coefficient of the j-th filter in the i-th layer, θ_p is the parameter of the convolutional layer. The operation $|\cdot|$ repre-

Table 2
Battery discharge protocol and life.

NO.	Life	Discharge scheme	NO.	Life	Discharge scheme	NO.	Life	Discharge scheme
1	1504	5C-1C-1C-1C	27	1491	4C-1C-3C-1C	53	1938	3C-2C-1C-1C
2	2678	5C-1C-2C-1C	28	1561	4C-1C-4C-1C	54	1308	3C-2C-2C-1C
3	1858	5C-1C-3C-1C	29	1380	4C-1C-5C-1C	55	2041	3C-2C-3C-1C
4	1500	5C-1C-4C-1C	30	2216	4C-2C-1C-1C	56	2290	3C-2C-4C-1C
5	1971	5C-1C-5C-1C	31	1706	4C-2C-2C-1C	57	1885	3C-2C-5C-1C
6	1143	5C-2C-1C-1C	32	2507	4C-2C-3C-1C	58	1348	3C-3C-1C-1C
7	1678	5C-2C-2C-1C	33	1926	4C-2C-4C-1C	59	2365	3C-3C-2C-1C
8	2285	5C-2C-3C-1C	34	2689	4C-2C-5C-1C	60	2047	3C-3C-3C-1C
9	2651	5C-2C-5C-1C	35	1962	4C-3C-1C-1C	61	1679	3C-3C-4C-1C
10	1751	5C-3C-1C-1C	36	1583	4C-3C-2C-1C	62	2057	3C-3C-5C-1C
11	1499	5C-3C-2C-1C	37	2460	4C-3C-3C-1C	63	2143	3C-4C-1C-1C
12	1386	5C-3C-3C-1C	38	1448	4C-3C-4C-1C	64	1905	3C-4C-2C-1C
13	1572	5C-3C-4C-1C	39	1609	4C-4C-1C-1C	65	1975	3C-4C-3C-1C
14	2202	5C-3C-5C-1C	40	1908	4C-4C-2C-1C	66	2168	3C-4C-4C-1C
15	1481	5C-4C-1C-1C	41	1804	4C-4C-3C-1C	67	1742	3C-4C-5C-1C
16	1938	5C-4C-2C-1C	42	1717	4C-4C-4C-1C	68	2012	3C-5C-1C-1C
17	2283	5C-4C-3C-1C	43	2178	4C-4C-5C-1C	69	2308	3C-5C-2C-1C
18	1649	5C-4C-4C-1C	44	2468	4C-5C-1C-1C	70	1702	3C-5C-3C-1C
19	1766	5C-4C-5C-1C	45	2450	4C-5C-3C-1C	71	1697	3C-5C-4C-1C
20	2657	5C-5C-1C-1C	46	1690	4C-5C-4C-1C	72	1848	3C-5C-5C-1C
21	2491	5C-5C-2C-1C	47	2030	4C-5C-5C-1C	73	1811	2C-4C-1C-1C
22	2479	5C-5C-3C-1C	48	1295	3C-1C-1C-1C	74	2030	2C-5C-2C-1C
23	2342	5C-5C-4C-1C	49	1393	3C-1C-2C-1C	75	2285	2C-3C-3C-1C
24	2217	5C-5C-5C-1C	50	1875	3C-1C-3C-1C	76	1783	2C-2C-4C-1C
25	1782	4C-1C-1C-1C	51	1419	3C-1C-4C-1C	77	1400	2C-1C-5C-1C
26	1142	4C-1C-2C-1C	52	1685	3C-1C-5C-1C			

sents the absolute value, signifying the utilization of L1 regularization. The scaling coefficient α_{ij} is sequentially linked to each channel within the convolutional layer. Its proximity to zero signifies the diminished contribution of the respective filter, indirectly reflecting its importance—this can serve as a basis for evaluating filter significance. By penalizing α_{ij}, it compels the model parameters to undergo sparsification. It's evident that the suggested regularization technique doesn't directly alter any pertinent model parameters; rather, it applies a penalty to a parameter external to the model structure, all the while preserving the model's original configuration.

3.2. Sparsifying model parameters

In this section, we delve into the process of sparsifying model parameters. The formula to compute the sparsity value of the j-th filter in the i-th layer is provided as follows:

$$SV_{ij} = \frac{\sum_{l,m} \sigma(p_{ijlm})}{LM}$$
$$\sigma(x) = \begin{cases} 1, & |x| > T_i \\ 0, & \text{else} \end{cases} \tag{8}$$

Here, SV_{ij} signifies the sparsity value of the j-th filter in the i-th layer, $l \in [1, L]$ relates to the filter depth (input), $m \in [1, M]$ where M indicates the kernel size, p_{ijlm} denotes the convolutional kernel weight. The parameter T_i represents the average value of convolutional kernel weights in the i-th layer and serves as a threshold. When a filter possesses a greater number of coefficients smaller than the average T_i of its corresponding layer, the sparsity value SV_{ij} approaches 0. This implies that the filter is relatively redundant compared to others. The calculation of T_i can be carried out through the subsequent formula:

$$T_i = \frac{\sum_{j,l,m} |p_{ijlm}|}{JLM} \tag{9}$$

where J accounts for the total number of filters within the i-th layer. The employment of the average norm T_i of all filter weights as the threshold SV_{ij} for calculating the sparsity value better captures the filter's significance within the holistic model.

3.3. Pruning steps

By combining the sparsity values of each convolutional layer filter with their scaling coefficients, we define the importance score of a filter as follows:

$$SC_{ij} = \alpha_{ij} SV_{ij} \tag{10}$$

where SC_{ij} denotes the importance score of the j-th filter in the i-th layer, α_{ij} is the scaling coefficient, and SV_{ij} is the sparsity value.

Pruning based on the importance scores of filters is performed through the following steps:

Step 1: Commence by setting the pruning ratio k, initializing the predictive model with random weights W_0, and initializing the scaling coefficients $\alpha_{ij} = 1^d$ alongside the pruning mask $\theta_{ij} = 1^d$, where d represents the dimension of the model parameters.

Step 2: Introduce the regularization coefficient λ and train W_0 for 3000 iterations to procure the subsequent weights W_1.

Step 3: Utilize Eq. (10) to compute the importance scores SC_{ij} of convolutional layer filters within W_1. Arrange them in descending order and denote their indices as id.

Step 4: Set the final $k{\cdot}id$ pruning mask values $\theta_{ij} = 0$, calculate $\theta_{ij} \odot W_1$ to obtain the refined sparse weights W_2.

Step 5: If a filter's weight within W_2 equate to zero, take the measure of resetting all its weights and biases to zero while designating them as untrainable.

Step 6: Engender streamlined weight parameters W_3 that correspond to the novel model network architecture.

Step 7: Conclude by fine-tuning the fresh model for 2 epochs, culminating in the attainment of the pruned model with weights W_4.

The pseudo-code for training is provided as Algorithm 1.

Algorithm 1. Pruning of battery RUL predictive model.

Input: Training data $\{(x_i, RUL_i)\}_{i=1}^N$, batch size, pruning ratio k
Initialize W_0 and b_0 of the model, $\alpha_{ij} = 1^d$, $\theta_{ij} = 1^d$
for epoch = 1, 2, …, max_epoch **do**
training the prediction model with Eq. (7)

(*continued on next page*)

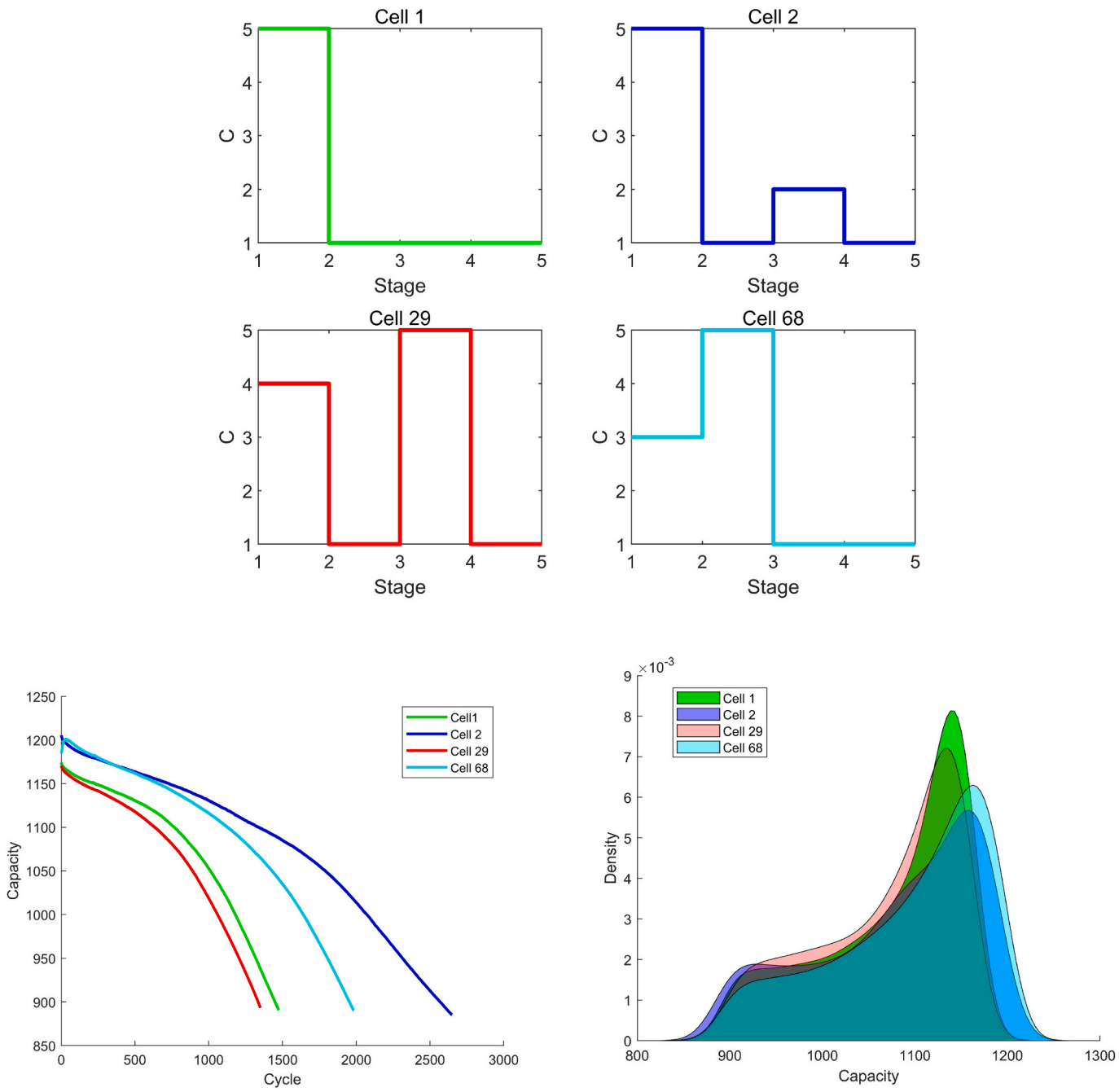

Fig. 2. Discharge scheme and capacity distribution of the batteries.

(*continued*)

4	**end**
5	Export model weights W_1
6	Calculate importance scores SC_{ij} with Eq. (10)
7	Sort SC_{ij} with indices id
8	Set $\theta_{ij} = 0$ of the last k*id
9	Calculate $W_2 = \theta_{ij} \odot W_1$
10	**if** $W_f^i = 0, W_f^i \in W_2$ **do**
11	Set $W_f = 0, b_f = 0$ as untrainable
12	**end**
13	**for** step = 1, 2, …, max_step **do**
14	fine-tuning the model
15	**end**
16	**Output**: the pruned model

4. Experiment

4.1. Experiment description

The experimental dataset encompasses comprehensive lifespan data from 77 lithium-iron phosphate batteries (LFP/graphite A123 APR18650M1A), sourced from the publicly available dataset of Huazhong University of Science and Technology [41]. The batteries possess a nominal capacity of 1.1 Ah and are characterized by a rated voltage of 3.3 V. All batteries underwent cycling charge and discharge procedures in a controlled environment at a constant temperature of 30 °C. While adhering to a uniform rapid charging regimen, each battery, however, followed a distinct discharge protocol, which is outlined as follows:

(1) Charging Protocol: The charging sequence initiated from 0 % to 80 % state of charge (SOC) through a constant current rate of 5C.

Table 3
The detailed structure and parameters of the training model. "400@32" represents a convolution layer with kernel size = 400 and output channel = 32.

Module	Layer(s)	Output size
Input	–	(32,21,000)
Feature extraction	Conv1d (32@400)	(32,32,601)
	Scaling1 ($\alpha_{1j}, j = 1, 2, \cdots, 32$)	(32,32,601)
	Leaky ReLU (0.5)	(32,32,601)
	Conv1d (64@400)	(32,64,202)
	Scaling2 ($\alpha_{2j}, j = 1, 2, \cdots, 64$)	(32,64,202)
	Leaky ReLU (0.5)	(32,64,202)
	Conv1d (128@195)	(32,128,8)
	Scaling3 ($\alpha_{3j}, j = 1, 2, \cdots, 128$)	(32,128,8)
	Leaky ReLU (0.5)	(32,128,8)
Self-attention	Multi-head attention (num_heads = 8)	(32,128,8)
	Leaky ReLU (0.5)	(32,128,8)
Flatten	Flatten	(32,1024)
Capacity prediction	Dense	(32,10)
Concatenate	Concatenate	(32,1034)
RUL prediction	Dense	(32,1)
Distribution matching	Dense (activation: Sigmoid)	(32,1)

Table 4
Comparison of prediction performance between the proposed method and other approaches.

Methods	Capacity (mAh)			RUL (cycles)		
	RMSE	R^2	SMAPE	RMSE	R^2	SMAPE
LSTM [42]	9.15	0.988	0.70	209.35	0.814	78.98
DCNN1 [43]	8.54	0.989	0.72	204.98	0.836	**74.80**
DCNN 2	8.65	0.988	0.69	202.04	0.840	76.44
Our method	**6.18**	**0.995**	**0.48**	**192.17**	**0.858**	77.07

Bold indicates the optimal value.

Subsequently, charging continued from 80 % SOC to 3.6 V at a constant current rate of 1C until achieving a 100 % SOC level, with a predefined current cutoff of C/20.

(2) Discharge Protocol in Four Stages:

Stage 1: Discharge spanned from 100 % to 60 % SOC.

Stage 2: Discharge spanned from 60 % to 40 % SOC.

Stage 3: Discharge spanned from 40 % to 20 % SOC.

Stage 4: Discharge occurred at a constant current of 1C with a voltage termination at 2 V, leading to a depletion from 20 % to 0 % SOC.

For each individual battery cell, the specific discharge scheme is meticulously presented in Table 1. In this tabulated representation, "NO." designates the battery cell number, "Life" quantifies the battery's lifespan in cycles. The notation "$n_1C - n_2C - n_3C - n_4C$" delineates the discharge regimen across the four distinct stages, with "n_i" representing the discharge rate. For instance, Battery #1 underwent a sequential discharge process featuring constant current rates of 5C-1C-1C-1C, in strict accordance with the outlined particulars in Table 2.

During the battery life experiment, voltage, current and capacity data were collected at a fixed time frequency for each charge-discharge cycle, until the maximum capacity of the battery reached 80 % of the nominal capacity (failure threshold). For additional details regarding the experiment, please refer to Reference [41].

4.2. Data processing

We opt for a two-dimensional dataset comprised of charging voltage and capacity to serve as the input for our model. In an effort to streamline training efforts, exclusively the data gathered during the second stage of charging is employed. This pertains specifically to the range from 80 % state of charge (SOC) to the point of first reaching 3.6 V. Owing to the dynamic interplay of battery degradation and operational variables, the data length accrued in each cycle exhibits variability. To ensure data uniformity, we undertake a linear interpolation procedure, expanding each cycle's sample to encompass 100 dimensions.

The sliding window technique is harnessed to extract data from 30 consecutive cycles, thereby constituting a single sample. In a bid to optimize computational efficiency, a cycle is sampled every 3 cycles, translating to the selection of data from a total of 10 cycles. Consequently, each individual sample takes on a configuration of $2 \times 100 \times 10$, encompassing 2 features, 100 sampling points, and 10 cycles.

The model output has two components: discharge capacity and RUL. Since 10 cycles of charging data are inputted, we use the discharge capacity from 10 cycles and the RUL corresponding to the last cycle of the input sequence as output.

We extract 55 sets of battery data out of the total 77 as our training dataset, while allocating the remaining 22 sets for our testing dataset. Specifically, we earmark batteries #1, #2, #4, #9, #10, #12, #13, #15, #18, #19, #20, #23, #24, #36, #39, #44, #57, #65, #66, #71, #73, and #75 for testing purposes. The selection of these particular batteries was made at random, devoid of any specific agenda. It's worth noting that a larger training dataset invariably bolsters the generalization capabilities of our model. However, for the purpose of assessing the efficacy of our method, we are utilizing a 55: 22 ratio. It's pertinent to mention that readers with a keen interest in the matter can adjust this ratio as deemed necessary.

Fig. 2 shows by line or curve four groups' (#1, #2, #29, #68) discharge schemes, capacity degradation curves and capacity degradation distributions. We can see that different discharge schemes lead to different battery capacity degradation trends and corresponding distribution curves. This indicates that battery usage scenarios significantly affect battery life.

The density depicted in Fig. 2 is obtained through Kernel Density Estimation, where a kernel (a Gaussian distribution function) is placed at each observed capacity. An average of all kernels is then computed to estimate the probability density of the entire dataset. The implementation of this process is facilitated by MATLAB's ksdensity function in this study.

4.3. Evaluation metrics

We use the voltage and capacity of the charging process from 55 selected batteries as model input and their discharge capacity and RUL as labels during model training. For prediction we use voltage and capacity from charging process of remaining 22 batteries as input and compare their predicted capacity and RUL with ground truth. We use three metrics to evaluate our prediction results: root mean square error (RMSE), coefficient of determination (R^2) and symmetric mean absolute percentage error (SMAPE). Their formulas are as follows.

(1) Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y^i - \hat{y}^i)^2} \tag{11}$$

Range $[0, +\infty)$. When predicted and true values match exactly RMSE equals 0, indicating a perfect model; larger errors imply larger values.

(2) Coefficient of Determination (R^2)

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}^i - \bar{y})^2}{\sum_{i=1}^{n}(y^i - \bar{y})^2} \tag{12}$$

Higher R2 values closer to 1 indicate better model prediction performance.

(3) Symmetric Mean Absolute Percentage Error (SMAPE)

$$SMAPE = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|\hat{y}^i - y^i|}{(|\hat{y}^i| + |y^i|)/2} \tag{13}$$

Here, $\hat{y}^i$ represents predicted value, y^i represents true value and $\bar{y}$

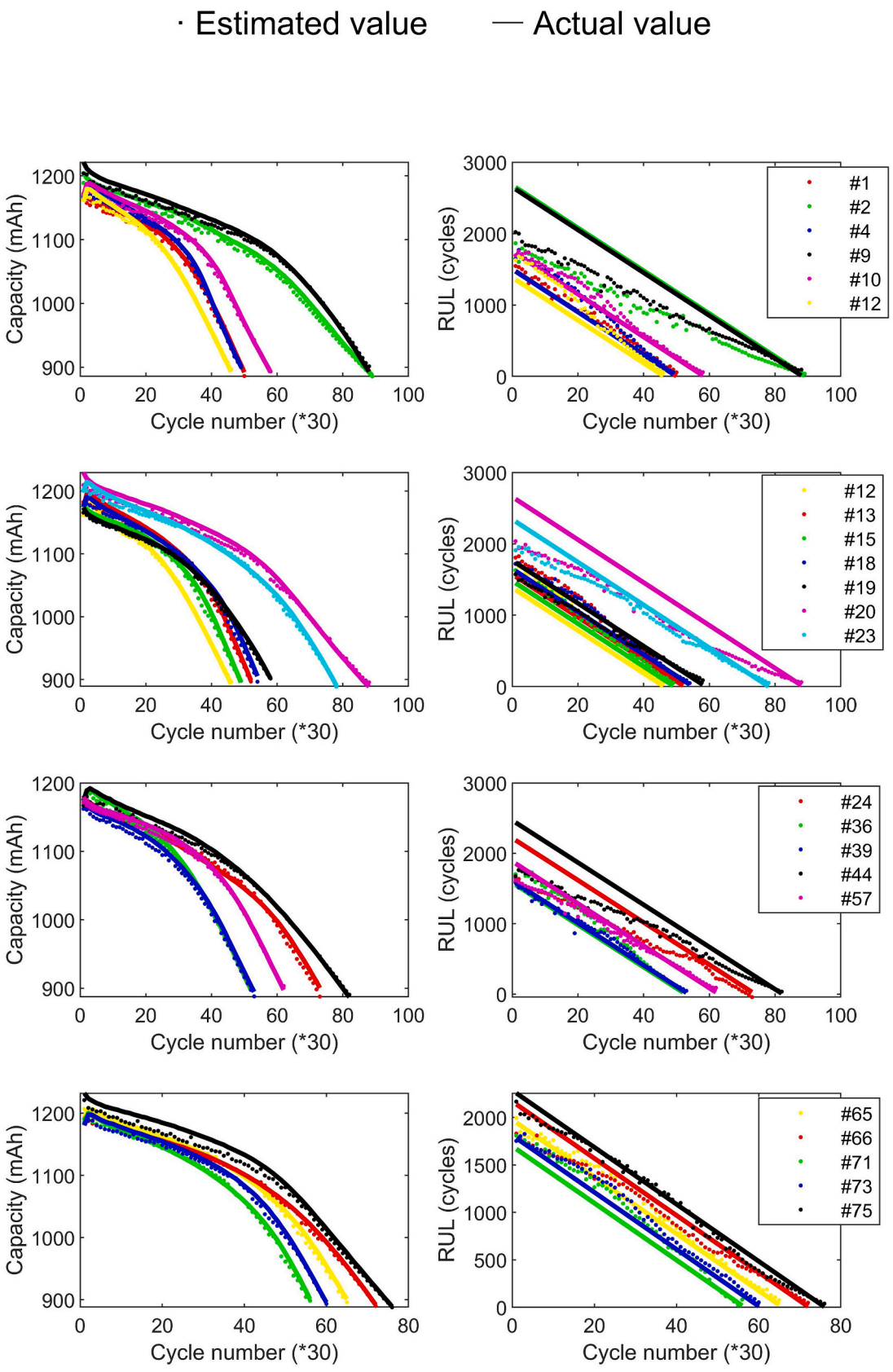

Fig. 3. Comparison between predicted results and ground truth values.

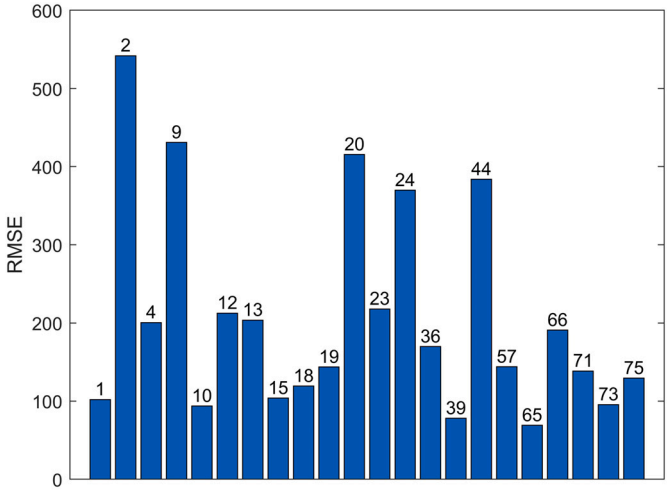

Fig. 4. Predicted results of RMSE for each battery.

represents average value of true samples. A perfect match between predictions and actual results in a SMAPE value of 0, while larger errors lead to higher SMAPE values.

4.4. Prediction results from the original model

After conducting a series of testing iterations, the original model's configuration parameters were established as follows: a learning rate of 0.0001, a training batch size of 32, loss function adjustment parameters set at $\beta = 0.7$ and $\gamma = 0.01$, a multi-head attention layer equipped with 8 heads, and a total of 3000 training epochs. Prior to model training, a max-min normalization technique was applied to normalize all the data. The detailed structure and parameters of the training model are presented in Table 3.

To ascertain the effectiveness of our proposed model, this study has incorporated a selection of common methods for the purpose of comparing prediction outcomes with experimental data. The outcomes are outlined in Table 4. The chosen methods strive to mirror the proposed method's network structure and parameters as closely as possible. Among these, "DCNN1" denotes the model obtained by eliminating the adapter and multi-head attention layer from the proposed method, while keeping other parameters consistent. In parallel, "DCNN2" preserves the adapter while discarding the multi-head attention layer, with the remaining parameters aligning with those of the proposed method.

The values highlighted in bold within the table denote the optimal results. As depicted in Table 4, the proposed model distinctly exhibits superior performance. Graphs illustrating the projected capacity and RUL for the 22 test batteries are presented in Fig. 3. For the purpose of enhancing visual clarity, the values in the graphs are presented at intervals of every 30 cycles to display the outcome of each individual cycle.

The Multi-Head Attention Mechanism is capable of capturing multiple features within the data, while Domain Adapters primarily address the issue of inconsistent data distribution between the source and target domains. As evidenced by the experimental results presented in Table 4, the incorporation of both the Multi-Head Attention Mechanism and Domain Adapters significantly enhances the precision of RUL predictions.

From Figs. 3 and 4, it is evident that the accuracy of capacity prediction surpasses that of RUL prediction. The larger prediction errors in RUL are primarily concentrated within a few battery cells characterized by longer actual lifespans. This discrepancy mainly arises due to the scarcity of long-lifespan training samples, leading to an imbalance in data distribution.

4.5. Model pruning

During the process of sparsity-inducing training, the regularization coefficient was established as $\lambda = 10^{-5}$, while the model parameter learning rate was set to 0.0001. Additionally, the learning rate for the scaling coefficients α_{ij} was designated as 0.01. Following the culmination of the sparsity training phase, pruning was executed in line with the procedure outlined in Section 3.3, guided by the designated pruning ratio. Subsequent to pruning, the refined model underwent two rounds of fine-tuning. With these preparations in place, the model was ready to receive the testing data for prediction. Diverse pruning ratios were implemented, with the corresponding prediction outcomes documented in Table 5.

In Table 5, the inference time signifies the duration required for the trained model to process and predict the testing data, encompassing the completion of capacity and RUL predictions for all 22 battery life cycles. The inference time is notably influenced by computer hardware and its

Table 5
Prediction results under various pruning ratios, note: Δ% refers to the ratio of pruned model parameters to the total model parameters.

Pruning rate %	Capacity (mAh)			RUL (cycles)			Inference time (ms)	Δ%
	RMSE	R^2	SMAPE	RMSE	R^2	SMAPE		
0	6.18	0.995	0.48	**192.17**	**0.858**	77.07	65.53	0
10	5.93	0.995	0.47	198.55	0.852	77.36	45.09	11.01
20	5.93	0.995	0.47	198.56	0.852	77.36	41.91	22.46
30	5.93	0.995	0.47	198.55	0.852	77.36	**37.89**	33.40
40	**5.92**	**0.995**	**0.47**	198.17	0.852	77.33	39.72	44.40
50	6.10	0.994	0.48	198.98	0.851	**76.56**	40.01	55.34
60	10.39	0.981	0.81	218.80	0.798	78.21	38.95	**65.84**

Bold indicates the optimal value.

Table 6
Comparison of prediction results after model pruning (50 % pruned).

Methods	Capacity (mAh)			RUL (cycles)			Inference time (ms)	Δ%
	RMSE	R^2	SMAPE	RMSE	R^2	SMAPE		
Original model	6.18	0.995	0.48	192.17	0.858	77.07	65.53	0
L1 [44]	9.60	0.985	0.75	203.16	0.841	75.30	40.92	48.45
L2 [45]	18.43	0.932	1.39	246.35	0.739	**70.71**	**39.95**	48.45
Our method	**6.10**	**0.994**	**0.48**	**198.98**	**0.851**	76.56	40.01	**55.34**

Bold indicates the optimal value.

occupancy rate. The data presented in Table 4 represents the average time over 5 runs. The hardware and software configuration utilized includes a PyTorch environment, Intel(R) Core(TM) i7–8700 CPU, 48GB DDR4 memory, and an RTX 3090 graphics card. It is discernible that the most favorable outcome is achieved when the pruning ratio stands at 40 %. In practical applications, a higher pruning ratio implies lower computational overhead while satisfying the required prediction accuracy. In our experimentation, we found that when the pruning ratio exceeds 70 %, it leads to the removal of entire convolutional layers, thereby altering the network structure. To showcase the advantages of the proposed pruning approach, we have also compared it against several common filter pruning methods, the specific outcomes of which are documented in Table 6.

The higher parameter pruning rate in the proposed method compared to L1 and L2 is due to the fact that L1 and L2 allocate pruning proportions uniformly across each convolutional layer. In contrast, the proposed method aggregates all filters from the three convolutional layers and performs comprehensive pruning based on their importance. Consequently, the quantity of pruned parameters varies. Evidently, the proposed method prunes a larger number of parameters, indicating the potential for greater reduction in computational overhead. As an increasing number of pruning methods are being introduced, in the context of structural pruning methods, the distinctions between various approaches are relatively subtle. Hence, this study has selected two representative pruning methods for comparative analysis.

In light of the data presented in Table 5, a comprehensive analysis of the model's performance under varying pruning rates is conducted. The pruning rate is defined as the ratio of pruned model parameters to the total parameters. In practical scenarios, a higher pruning rate is indicative of a reduced computational overhead while maintaining the requisite prediction accuracy. The model exhibits optimal prediction performance at a pruning rate of 40 %. This could be attributed to the balance achieved between maintaining prediction accuracy and reducing computational overhead. However, an increase in the pruning rate beyond this point leads to a decline in prediction accuracy, potentially due to the loss of information resulting from excessive pruning.

A comparative analysis with common filter pruning methods, as shown in Table 6, reveals that our proposed method achieves a higher parameter pruning rate than L1 and L2 norms. This is because, unlike L1 and L2 norms which allocate pruning proportions uniformly across each convolutional layer, our method aggregates all filters across the convolutional layers and performs a comprehensive pruning based on their relative importance. This results in a variable quantity of pruned parameters, with our method pruning a larger number of parameters, thereby indicating a potential for a greater reduction in computational overhead. In conclusion, these findings suggest that an appropriate selection of the pruning rate and an effective pruning method can significantly reduce the computational overhead of the model while preserving prediction accuracy. This has profound implications for model optimization and deployment in real-world applications. However, the determination of the optimal pruning rate and method warrants further investigation.

5. Conclusion

This paper introduces an adaptive lithium-ion batteries life prediction model, ingeniously incorporating a domain adapter to automatically glean shared insights from the charging voltage and capacity data across diverse operational conditions and degradation stages. The incorporation of the multi-head self-attention mechanism allows the prediction model to adeptly capture the sensitive information connecting features and lifespan degradation, thereby elevating its generalization capability.

In light of the demand for battery life prediction on edge devices, this study also proposes a sophisticated model pruning strategy grounded in scaling coefficients and sparsity, culminating in the formulation of a global convolutional kernel importance evaluation equation. This innovation yields a higher parameter compression rate in battery life prediction models than existing pruning methods. Comparative experiments substantiate the evident advantages of the proposed approach, furnishing a fresh avenue for real-world engineering applications.

Despite the significant strides our model has made in battery life prediction, there are still limitations and potential directions for future research. While the multi-head self-attention mechanism is capable of capturing sensitive information between features and lifespan degradation, its effectiveness across diverse operational conditions and degradation stages may vary. In terms of future work, enhancing the robustness of the model to handle a wider range of operational conditions and degradation stages is a promising direction. Additionally, exploring other types of data, such as temperature or current, could potentially improve the model's performance and applicability. Lastly, further refining the model pruning strategy to achieve higher compression rates without compromising prediction accuracy is another worthwhile endeavor. These improvements will not only enhance the model's performance but also broaden its applicability in real-world engineering applications.

CRediT authorship contribution statement

Yang Ge: Writing – review & editing, Writing – original draft, Software, Methodology. **Jiaxin Ma:** Resources, Funding acquisition, Data curation. **Guodong Sun:** Visualization, Methodology.

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.

Data availability

Data will be made available on request.

Acknowledgements

Partial support for this study was provided by the Suzhou Science and Technology Foundation of China under Grant SYG202021 and SYG202351, the Jiangsu Provincial Natural Science Research Foundation of China under Grant 21KJA510003.

References

[1] J.C. Kelly, M. Wang, Q. Dai, O. Winjobi, Energy, greenhouse gas, and water life cycle analysis of lithium carbonate and lithium hydroxide monohydrate from brine and ore resources and their use in lithium ion battery cathodes and lithium ion batteries, Resour. Conserv. Recycl. 174 (2021).
[2] C. Lyu, Q.Z. Lai, T.F. Ge, H.H. Yu, L.X. Wang, N. Ma, A lead-acid battery's remaining useful life prediction by using electrochemical model in the particle filtering, framework, Energy 120 (2017) 975–984.
[3] C. Pastor-Fernandez, K. Uddin, G.H. Chouchelamane, W.D. Widanage, J. Marco, A comparison between electrochemical impedance spectroscopy and incremental capacity-differential voltage as Li-ion diagnostic techniques to identify and quantify the effects of degradation modes within battery management systems, J. Power Sources 360 (2017) 301–318.
[4] D.D. Ma, X.G. Qin, Residual life prediction of lithium batteries based on data mining, Comput. Intell. Neurosci. 2022 (2022).
[5] D.J. Gao, Y. Zhou, T.N. Wang, Y.D. Wang, A method for predicting the remaining useful life of lithium-ion batteries based on particle filter using Kendall rank correlation coefficient, Energies 13 (16) (2020).
[6] W.X. Duan, S.X. Song, F. Xiao, Y. Chen, S.L. Peng, C.X. Song, Battery SOH estimation and RUL prediction framework based on variable forgetting factor online sequential extreme learning machine and particle filter, J. Energy Storage (2023) 65.
[7] W. Xiong, G. Xu, Y.M. Li, F. Zhang, P. Ye, B. Li, Early prediction of lithium-ion battery cycle life based on voltage-capacity discharge curves, J. Energy Storage (2023) 62.
[8] W.L. Wu, S.S. Lu, Remaining useful life prediction of lithium-ion batteries based on data preprocessing and improved ELM, IEEE Trans. Instrum. Meas. (2023) 72.

[9] D.X. Shen, L.F. Wu, G.Q. Kang, Y. Guan, Z. Peng, A novel online method for predicting the remaining useful life of lithium-ion batteries considering random variable discharge current, Energy 218 (2021).
[10] J. Xing, H.L. Zhang, J.P. Zhang, Remaining useful life prediction of-lithium batteries based on principal component analysis and improved Gaussian process regression, Int. J. Electrochem. Sci. 18 (4) (2023).
[11] J.B. Li, M. Ye, Y. Wang, Q. Wang, M. Wei, A hybrid framework for predicting the remaining useful life of battery using Gaussian process regression, J. Energy Storage (2023) 66.
[12] L. Ren, L. Zhao, S. Hong, S.Q. Zhao, H. Wang, L. Zhang, Remaining useful life prediction for lithium-ion battery: a deep learning approach, IEEE Access 6 (2018) 50587–50598.
[13] Y.W. Liu, J. Sun, Y.L. Shang, X.D. Zhang, S. Ren, D.T. Wang, A novel remaining useful life prediction method for lithium-ion battery based on long short-term memory network optimized by improved sparrow search algorithm, J. Energy Storage (2023) 61.
[14] G.Z. Lyu, H. Zhang, Q. Miao, Parallel state fusion LSTM-based early-cycle stage lithium-ion battery RUL prediction under Lebesgue sampling framework, Reliab. Eng. Syst. Saf. 236 (2023).
[15] J. Hong, D. Lee, E.R. Jeong, Y. Yi, Towards the swift prediction of the remaining useful life of lithium-ion batteries with end-to-end deep learning, Appl. Energy 278 (2020).
[16] M. Haris, M.N. Hasan, S.Y. Qin, Degradation curve prediction of lithium-ion batteries based on knee point detection algorithm and convolutional neural network, IEEE Trans. Instrum. Meas. (2022) 71.
[17] J.Y. Xia, Q.L. Shi, H.M. Li, M. Zhou, W. Wang, K.L. Wang, et al., Historical data-independent remaining useful life prediction method based on dual-input deep learning neural network, J. Energy Storage (2023) 72.
[18] L.W. Hu, W.B. Wang, G.R. Ding, RUL prediction for lithium-ion batteries based on variational mode decomposition and hybrid network model, Signal Image Video Process. 17 (6) (2023) 3109–3117.
[19] Y.M. Zhu, J.C. Wu, X. Liu, J. Wu, K. Chai, G. Hao, et al., Hybrid scheme through read-first-LSTM encoder-decoder and broad learning system for bearings degradation monitoring and remaining useful life estimation, Adv. Eng. Inform. 56 (2023).
[20] F. Guo, X.W. Wu, L.L. Liu, J.L. Ye, T. Wang, L.J. Fu, et al., Prediction of remaining useful life and state of health of lithium batteries based on time series feature and Savitzky-Golay filter combined with gated recurrent unit neural network, Energy 270 (2023).
[21] X.Y. Zhong, X.J. Song, G.H. Liu, W.X. Zhao, W. Fan, A data-driven method for remaining useful life prediction of rolling bearings under different working conditions, IEEE Trans. Reliab. (2023), https://doi.org/10.1109/TR.2023.3328693.
[22] J.H. Chou, F.K. Wang, S.C. Lo, Predicting future capacity of lithium-ion batteries using transfer learning method, J. Energy Storage (2023) 71.
[23] Q.L. Xie, R.C. Liu, J.H. Huang, J.H. Su, Residual life prediction of lithium-ion batteries based on data preprocessing and a priori knowledge-assisted CNN-LSTM, Energy 281 (2023).
[24] Q. Xu, M. Wu, E. Khoo, Z.H. Chen, X.L. Li, A hybrid ensemble deep learning approach for early prediction of battery remaining useful life, IEEE-CAA J. Autom. Sin. 10 (1) (2023) 177–187.
[25] C. Rincón-Maya, F. Guevara-Carazas, F. Hernández-Barajas, C. Patino-Rodriguez, O. Usuga-Manco, Remaining useful life prediction of lithium-ion battery using ICC-CNN-LSTM methodology, Energies 16 (20) (2023).
[26] L. Li, Y.J. Li, R.Z. Mao, L. Li, W.B. Hua, J.L. Zhang, Remaining useful life prediction for lithium-ion batteries with a hybrid model based on TCN-GRU-DNN and dual attention mechanism, IEEE Trans. Transp. Electrif. 9 (3) (2023) 4726–4740.
[27] S. Ly, J.H. Xie, F.E. Wolter, H.D. Nguyen, Y. Weng, T-shape data and probabilistic remaining useful life prediction for Li-ion batteries using multiple non-crossing quantile long short-term memory, Appl. Energy 349 (2023).
[28] K. Song, D. Hu, Y. Tong, X.G. Yue, Remaining life prediction of lithium-ion batteries based on health management: a review, J. Energy Storage (2023) 57.
[29] H.M. Jiang, H.J. Wang, Y.T. Su, Q.L. Kang, X.H. Meng, L.J. Yan, et al., Multiple health indicators assisting data-driven prediction of the later service life for lithium-ion batteries, J. Power Sources 542 (2022).
[30] Y.Z. Wang, C.Y. Hei, H. Liu, S.D. Zhang, J.G. Wang, Prognostics of remaining useful life for lithium-ion batteries based on hybrid approach of linear pattern extraction and nonlinear relationship mining, IEEE Trans. Power Electron. 38 (1) (2023) 1054–1063.
[31] J.T. Qu, F. Liu, Y.X. Ma, J.M. Fan, A neural-network-based method for RUL prediction and SOH monitoring of lithium-ion battery, IEEE Access 7 (2019) 87178–87191.
[32] M. Zhang, G.Q. Kang, L.F. Wu, Y. Guan, A method for capacity prediction of lithium-ion batteries under small sample conditions, Energy 238 (2022).
[33] S.B. Vilsen, D.I. Stroe, Transfer learning for adapting battery state-of-health estimation from laboratory to field operation, IEEE Access 10 (2022) 26514–26528.
[34] J. Couture, X.K. Lin, Novel image-based rapid RUL prediction for Li-ion batteries using a capsule network and transfer learning, IEEE Trans. Transp. Electrif. 9 (1) (2023) 958–967.
[35] J. Ma, X.Y. Zou, L.L. Sun, Y.J. Cheng, C. Lu, Y.Z. Su, et al., A prediction-based cycle life test optimization method for cross-formula batteries using instance transfer and variable-length-input deep learning model, Neural Comput. Applic. 35 (4) (2023) 2947–2971.
[36] Y.H. Li, K. Li, X. Liu, Y.X. Wang, L. Zhang, Lithium-ion battery capacity estimation-a pruned convolutional neural network approach assisted with transfer learning, Appl. Energy 285 (2021).
[37] P.C. Jiang, Y. Xue, F. Neri, Convolutional neural network pruning based on multi-objective feature map selection for image classification, Appl. Soft Comput. 139 (2023).
[38] H.J. Cheng, Z.D. Wang, L.F. Ma, Z.H. Wei, F.E. Alsaadi, X.H. Liu, Differentiable channel pruning guided via attention mechanism: a novel neural network pruning approach, Complex Intell. Syst. 9 (5) (2023) 5611–5624.
[39] J.N. Liu, R.J. Hao, Q. Liu, W.W. Guo, Prediction of remaining useful life of rolling element bearings based on LSTM and exponential model, Int. J. Mach. Learn. Cybern. 14 (4) (2023) 1567–1578.
[40] K. Kamma, S. Inoue, T. Wada, Pruning ratio optimization with layer-wise pruning method for accelerating convolutional neural networks, IEICE Trans. Inf. Syst. E105D (1) (2022) 161–169.
[41] G.J. Ma, S.P. Xu, B.B. Jiang, C. Cheng, X. Yang, Y. Shen, et al., Real-time personalized health status prediction of lithium-ion batteries using deep transfer learning, Energy Environ. Sci. 15 (10) (2022) 4083–4094.
[42] K. Park, Y. Choi, W.J. Choi, H.Y. Ryu, H. Kim, LSTM-based battery remaining useful life prediction with multi-channel charging profiles, IEEE Access 8 (2020) 20786–20798.
[43] S. Shen, M. Sadoughi, M. Li, Z.D. Wang, C. Hu, Deep convolutional neural networks with ensemble learning and transfer learning for capacity estimation of lithium-ion batteries, Appl. Energy 260 (2020).
[44] A. Kumar, A.M. Shaikh, Y. Li, H. Bilal, B.Q. Yin, Pruning filters with L1-norm and capped L1-norm for CNN compression, Appl. Intell. 51 (2) (2021) 1152–1160.
[45] G. Li, G. Xu, Providing clear pruning threshold: a novel CNN pruning method via L-0 regularisation, IET Image Process. 15 (2) (2021) 405–418.