Energy 308 (2024) 132840

Contents lists available at ScienceDirect

Energy

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

A robust adapted Flexible Parallel Neural Network architecture for early prediction of lithium battery lifespan

Lidang Jiang [a], Zhuoxiang Li [a], Changyan Hu [a], Junxiong Chen [b], Qingsong Huang [a], Ge He [a,*]

[a] School of Chemical Engineering, Sichuan University, Chengdu, 610065, Sichuan, PR China
[b] School of Automation Engineering, University of Electronic Science and Technology of China, Chengdu, 610023, Sichuan, PR China

ARTICLE INFO

Keywords:
Neural networks
Lithium batteries
Interpretable machine learning
Deep learning

ABSTRACT

Early prediction of End of Life (EPEOL) is crucial for improving lithium battery efficiency and lifespan. Traditional fixed-architecture neural networks often suffer from underfitting or overfitting due to diverse data distributions. To address this, we propose the Flexible Parallel Neural Network (FPNN). This model integrates modules like InceptionBlock, 3D CNN, 2D CNN, and dual-stream networks. By effectively extracting electrochemical features from video-format data through the 3D CNN and achieving multi-scale feature abstraction with InceptionBlock, FPNN ensures effective module coordination. The model can adaptively adjust the number of InceptionBlocks to handle tasks of varying complexity. Experimental results on the MIT dataset show that FPNN achieves MAPE values of 1.26%, 0.41%, 0.37%, 0.33%, 0.32%, 0.32%, 0.31%, 0.31%, 0.22%, and 0.34% using the first 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 cycle data, respectively. The interpretability of FPNN is reflected in its structural design and flexible unit choices, providing a basis for model interpretation. Comprising multiple modules, FPNN enables smooth feature extraction from electrochemical data through collaborative interaction. The diverse branching structure of the flexible units allows the model to capture features at different scales, learning richer information. Our approach offers a precise, adaptable, and easy-to-understand solution for EPEOL, opening new possibilities in battery health monitoring.

1. Introduction

Since Sony introduced the first commercial lithium-ion batteries (LIBs) in 1991 [1], these batteries have been widely adopted in various fields such as portable electronic devices and electric vehicles, owing to their long service life and low self-discharge characteristics [2–4]. The number of charging and discharging cycles experienced by a battery until its capacity falls to 70%–80% of the initial capacity is considered the battery's End of Life (EOL) [5]. Given the long lifespan characteristic of lithium batteries, using traditional experimental methods to determine their lifespan is not only time-consuming but also costly. Hence, accurately predicting the EOL using early operational data of the battery, or Early Prediction of End of Life (EPEOL), becomes particularly crucial. EPEOL technology can provide vital information and guidance for product design improvements, safety enhancements, maintenance strategy optimization, and cost–benefit analysis, making it of great significance to battery manufacturers, engineers, and end users alike.

The modeling approaches for EPEOL can broadly be categorized into two main types: physics-based modeling and data-driven modeling. Physics-based modeling includes semi-empirical models [6], empirical models [7,8], equivalent circuit models [9,10], and electrochemical models [11–13]. This method requires researchers to have a deep understanding of the physical and chemical processes within batteries and use mathematical models to predict the battery's lifespan. Physics-based modeling excels in explaining battery behavior and considering the interactions among various influencing factors, thereby offering advantages in model accuracy and interpretability. However, this approach often relies on extensive experimental data for parameter fitting and model calibration, making the modeling process time-consuming and the model structure complex. Moreover, the theoretical assumptions of physics-based models may not always be applicable in complex real-world scenarios [14].

Compared to physical knowledge modeling, data-driven modeling [15], particularly machine learning (ML) [16] techniques, exhibit remarkable advantages in terms of flexibility, adaptability, and scalability. Various machine learning methods have been applied to the domain of battery life prediction, such as using Support Vector Machines (SVM) to identify batteries with lifespans that do not meet expectations [17], and employing decision trees [18], SVM [19], and k-nearest neighbors (KNN) [20] for classifying the lifespan of LFP/graphite batteries [21].

* Corresponding author.
E-mail addresses: qshuang@scu.edu.cn (Q. Huang), hege@scu.edu.cn (G. He).

https://doi.org/10.1016/j.energy.2024.132840
Received 4 April 2024; Received in revised form 27 July 2024; Accepted 14 August 2024
Available online 22 August 2024

Nomenclature			
EOL	End of life	MAPE	Mean absolute percentage error
LIBs	Lithium-ion batteries	RNN	Recurrent neural network
EPEOL	Early prediction of end of life	CNN	Convolutional neural network
ML	Machine learning	FPNN	Flexible parallel neural network
SVM	Support vector machine	BMS	Battery management system
KNN	k-nearest neighbors	NOI	Number of inceptionblock
RUL	Remaining useful life	MAE	Mean absolute error
EIS	Electrochemical impedance spectroscopy	RMSE	Root mean squared error
GPR	Gaussian process regression	CC	Constant current
		CV	Constant voltage

Severson et al. [14] trained a simple linear model on the MIT battery dataset, achieving a 9.1% accuracy in predicting remaining useful life (RUL). Zhang et al. [22] constructed an Electrochemical Impedance Spectroscopy (EIS) dataset, and attained high precision in RUL prediction through Gaussian Process Regression (GPR). Yang et al. [23] utilized a Gradient Boosting Regression Trees model, employing features such as voltage, capacity, and temperature, to achieve a 7% Mean Absolute Percentage Error (MAPE) in EPEOL tasks. Fei et al. [24] developed a comprehensive machine learning framework that includes feature extraction, feature selection, and a prediction module, wherein the prediction module integrates various machine learning models such as GPR, SVM, and Elastic Net. On data from the first 100 cycles in the MIT dataset, SVM achieved the lowest MAPE of 8.0%. However, these traditional machine learning algorithms often require manual selection and design of features, a process that adds complexity to the machine learning workflow.

With the continuous advancement in computing power, neural network technologies, emblematic of deep learning [25], have shown tremendous potential in predicting the lifespan of batteries [26]. The capacity of lithium-ion batteries gradually declines with use, thereby affecting their lifespan. Recurrent Neural Networks (RNN) [27] and their variants [28–31] have been widely adopted due to their excellent performance in processing time series data, especially in capturing long-term dependencies within the data. However, these methods rely on the continuity of battery data time series and cannot be parallelized, leading to underutilization of existing computational resources during model training and inference phases. This not only results in significant computational resource wastage but also reduces the timeliness of information retrieval. In the realm of sequence data processing, the emergence of the Transformer model [32], capable of parallel computations, has gradually led to the obsolescence of RNNs and their derivatives. Yet, the Transformer model's memory usage is directly proportional to the length of battery sequence data, which can occupy substantial memory in practical applications, limiting its widespread use [33]. Meanwhile, Convolutional Neural Networks (CNN) not only require less memory and have more relaxed hardware requirements but also enable parallel computation. By preprocessing lithium battery feature data into a format akin to images, two-dimensional CNNs can efficiently extract important spatial features, thereby achieving higher prediction accuracy [34–37]. On the other hand, converting battery data into a format similar to videos and processing it with three-dimensional CNNs can delve into the complex interactions among parameters such as voltage, current, and temperature, enabling comprehensive analysis of electrochemical characteristics. Building on this concept, Yang [38] proposed a hybrid CNN model that significantly improved prediction accuracy, reducing the prediction error to 3.08% using data from only 20 charging cycles.

However, different EPEOL tasks have unique data distributions, requiring models of varying complexity to avoid underfitting or overfitting. While fixed-architecture models can fit electrochemical data, simplistic models may miss complex relationships, and complex models may overfit noise. Regularization techniques help, but finding optimal hyperparameters is complex and time-consuming. In this paper, we propose the Flexible Parallel Neural Network (FPNN), a neural network model capable of rapidly and adaptively adjusting to suit the complexity of different EPEOL tasks. The core of FPNN integrates the Inception-ResNet-A [39], three-dimensional convolutional neural network (3D CNN) [40], two-dimensional convolutional neural network (2D CNN), and dual-stream networks [41]. After preprocessing, each sample is presented in a format similar to videos, where the 3D CNN merges time (depth, i.e., charging capacity index) and spatial (channel, i.e., voltage/current/temperature) features to extract primary features. These primary features are then abstracted at a higher level through InceptionBlock (i.e., the Inception-ResNet-A module), forming multi-scale features. By stacking different InceptionBlocks with residual connections, forming InceptionBlocks, the model further enhances its capability to extract electrochemical features. The model automatically learns the number of InceptionBlocks, allowing FPNN to flexibly adjust its complexity to suit different EPEOL tasks. Significantly, this study also pioneers an efficient one-stop training and inference algorithm that, when coupled with FPNN, achieves flawless predictive performance.

This paper's main contributions are summarized as follows:

(1) Adaptability and Accuracy of FPNN: The FPNN model we propose demonstrates rapid and flexible adaptability to a variety of EPEOL tasks and achieves outstanding prediction accuracy. Whether employing a fixed or variable architecture, FPNN consistently exhibits superior performance.

(2) Interpretability of the Model: The advantages of FPNN primarily lie in its well-designed overall architecture, the structure, and parameter selection of its flexible units. Comprising multiple modules that work in synergy, FPNN can smoothly and progressively extract the required electrochemical features from electrochemical data. The flexible units, implemented via 1×1 convolution kernels, can rapidly integrate information across different channels, effectively consolidating diverse data. Its varied branching structure also enables the model to capture features at different scales, thus learning more enriched information. This unique structural design not only enhances the model's predictive accuracy but also increases the transparency and interpretability of its decision-making process.

(3) One-Stop Training and Inference Algorithm: This paper also presents a one-stop training and inference algorithm suitable for EPEOL tasks and other machine learning tasks.

The structure of the article is as follows: Section 2 provides a detailed introduction to the MIT dataset, including its data sources, types, and its importance and application methods in EPEOL tasks; Section 3 describes the application process of FPNN in actual EPEOL tasks, with special emphasis on data preprocessing steps and the detailed design of the FPNN architecture. It introduces the process for generating model input data, the various components of FPNN and their synergistic working mechanism, as well as the one-stop training and inference algorithm; Section 4 displays the experimental results and provides an in-depth analysis, including performance evaluation of the

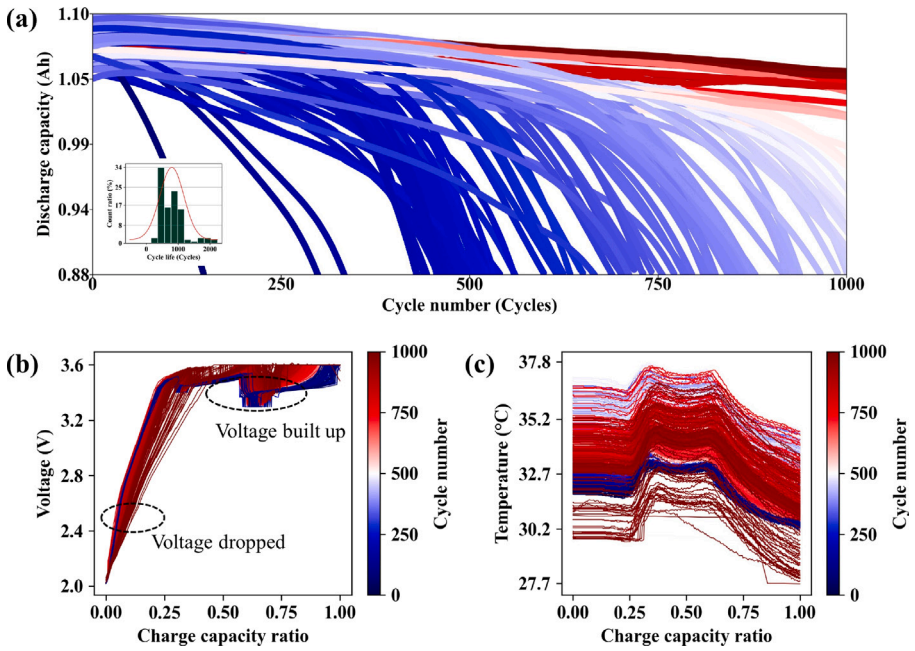

Fig. 1. Provides a comprehensive analysis of lithium-ion battery performance: (a) Using the MIT dataset, it shows capacity fade trends with cycle numbers and an inset graph of battery life distribution. (b) Displays voltage changes of the "b1c23" battery across charging cycles, with black circles highlighting voltage fluctuations. (c) Describes the "b1c23" battery's temperature variation during charging, reflecting thermal management at different stages.

FPNN model under different EPEOL tasks, comparative analysis with other existing models, ablation experiments. And Section 5 analysis of model interpretability; The paper concludes with a summary of its contents.

2. Input data generation

The dataset used in this study, created by Severson et al. [14] in 2019, addresses the issue of rapid charging of batteries and is known as the MIT dataset. This dataset encompasses 124 lithium iron phosphate/graphite batteries from the A123 Systems, tested in a constant temperature environment of 30 °C, though the actual temperature may fluctuate around 30 °C due to the internal electrochemical reactions of the batteries. As shown in Fig. 1(a), the dataset records the nominal capacity of the batteries as 1.1 Ah. A battery is considered to have failed when its discharge capacity falls to 80% of its nominal capacity, and the corresponding number of charge–discharge cycles is regarded as the battery's lifespan. The dataset primarily employs the "C1(Q1)-C2" charging strategy, which involves initial constant current charging at C1 until the charge reaches Q1 (State of Charge, SOC, %), followed by constant current charging at C2 until the SOC reaches 80%, and finally, charging under a 1C CC-CV (Constant Current-Constant Voltage) mode until the cut-off voltage of 3.6 V, with the voltage lower limit set at 2.0 V. By adjusting the values of C1, C2, and Q1, 72 different charging modes are simulated to mimic the varied charging environments in the real world.

In real-world scenarios, the discharge behavior of batteries varies, whereas the charging process is relatively consistent. Based on this observation, our study selects only charging data as the model input to better align with practical application requirements. Within the MIT dataset, there is a mapping relationship between the variation in battery life and changes in charging current; as the number of charge–discharge cycles increases, the battery's voltage gradually decreases at the beginning of charging and gradually increases towards the end of charging, as illustrated in Fig. 1(b). This variation reflects the decline in battery capacity, which significantly impacts battery life. Furthermore, Fig. 1(c) shows the change in temperature with the number of cycles, revealing the evolution of the battery's internal electrochemical characteristics. Therefore, voltage, current, and temperature data for different batteries under various charge–discharge cycles can all serve as important samples containing specific electrochemical information.

3. Methodology

Following a detailed introduction to the dataset, this section will elaborate on the overall workflow for EPEOL tasks. As illustrated in Fig. 2, the workflow begins with the Battery Management System (BMS), responsible for collecting real-time operational data from the batteries. Subsequently, these raw data undergo a series of preprocessing steps to be transformed into a video-like data format, thereby enhancing data processability. The processed data are then fed into the FPNN model for training and prediction tasks. The hyperparameters of FPNN are adjusted using the Bayesian optimization algorithm, where the primary variation in hyperparameter settings across different EPEOL tasks lies in the number of InceptionBlocks (NOI), with other parameters remaining consistent. Ultimately, the model's prediction outcomes are displayed through a meticulously designed data visualization tool. This tool not only presents the predictive achievements but also compares the predicted results with actual data, providing users with an intuitive method of performance evaluation.

3.1. Data preprocessing

Before being fed into the model, the data must undergo a series of preprocessing steps. This study selected voltage, current, and temperature data from the charging phase of the dataset, organizing these data into three separate matrix blocks. Taking voltage data as an example, the horizontal axis represents the cycle number, and the vertical axis represents the charging capacity. As shown in Fig. 3(a), each data sample includes the battery's data from the first cycle and its three most recent cycles. This sample selection strategy, based on a balance between the number of samples and feature similarity, differs from methods used in other studies [38,42]. Once organized into a video-like format, each battery generated n samples, leading to a total

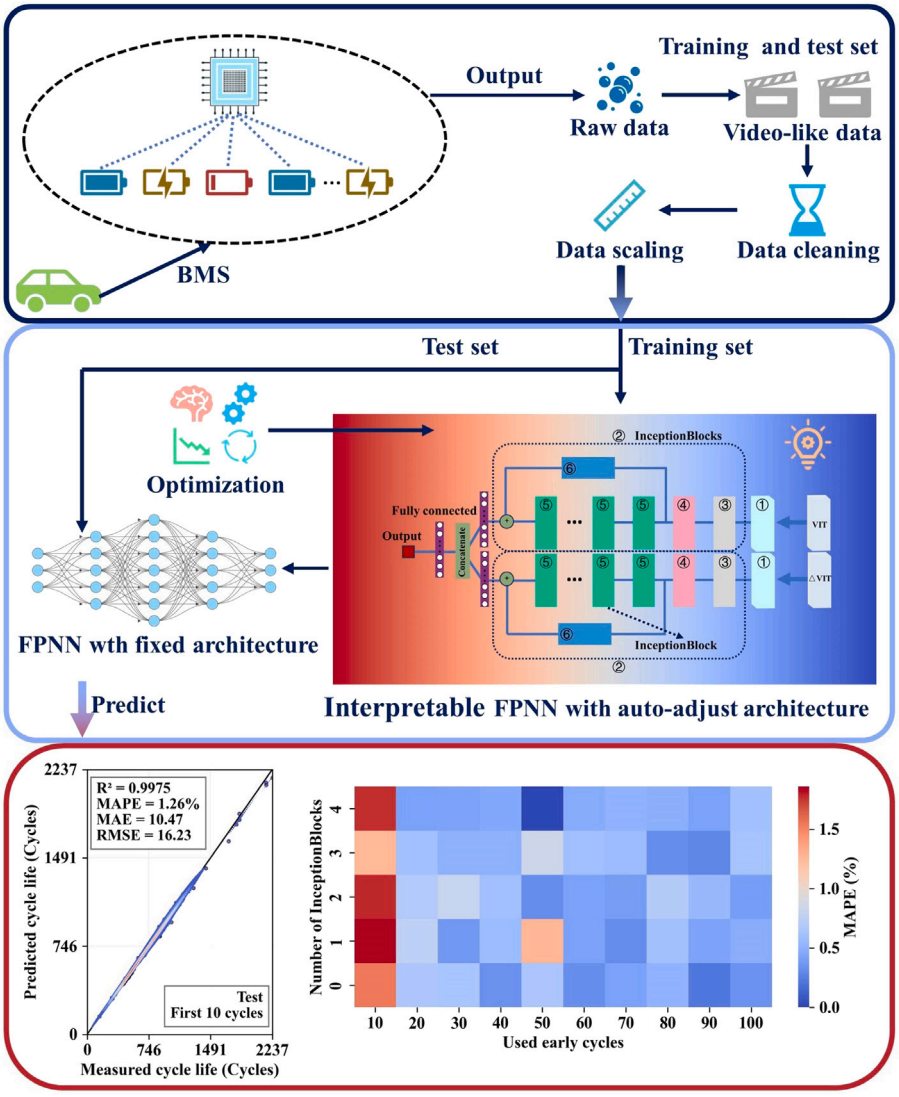

Fig. 2. Illustrates the EPEOL technique roadmap based on FPNN.

of 124n samples due to the 124 batteries. To reduce randomness in the experimental results, this study shuffled data at the sample level rather than the battery level, meaning samples from the same battery do not appear consecutively in the dataset. Following the proportions used in other research [38,42], the first 94n samples were divided into a training set, with the remaining samples forming the test set. Subsequent steps included removing outliers and applying the Savitzky-Golay filter for data smoothing. The features of the training and test sets were normalized to a range of −1 to 1, while the labels were kept as is, directly fed into one branch of FPNN. Additionally, considering the importance of differential data, the battery's voltage, current, and temperature data were differenced against their first cycle's data to extract differential features. These differential features were processed in the same manner as the original features and ultimately fed into another branch of FPNN.

3.2. FPNN (Flexible Parallel Neural Network)

3.2.1. FPNN architecture

In the FPNN architecture, one branch network is tasked with processing standard features, while another handles differential features, ensuring the model comprehensively captures electrochemical information. Within each branch network, electrochemical features x are first processed through a 3D CNN to integrate Voltage, Current, and Temperature (VIT) information, passing primary features to the next layer. The 3DCNN operations include performing 3D convolutions (1), normalization, and applying Leaky ReLU for nonlinearity, to extract nonlinear features of electrochemical data. Specifically, the 3D convolution operation (1) can be represented as:

$$z(x) = C(i,j,k) = \sum_m \sum_n \sum_l x \cdot K(m,n,l) \tag{1}$$

$$x = I(i+m, j+n, k+l) \tag{2}$$

where $C(i,j,k)$ is the output of the 3D convolution operation, corresponding to the value of the element at position (i,j,k) in the output volume. $\sum_m \sum_n \sum_l$ denotes the triple summation over all elements of the convolution kernel. $I(i+m, j+n, k+l)$ is the electrochemical feature input to the current branch network, specified by the position $(i+m, j+n, k+l)$, where i, j and k are the coordinates of the current element being processed, and m, n and l are the row, column, and depth offsets of the convolution kernel, respectively. $K(m,n,l)$ is the weight at position (m,n,l) within the convolution kernel. This processing enables FPNN to efficiently extract and integrate key features from battery data,

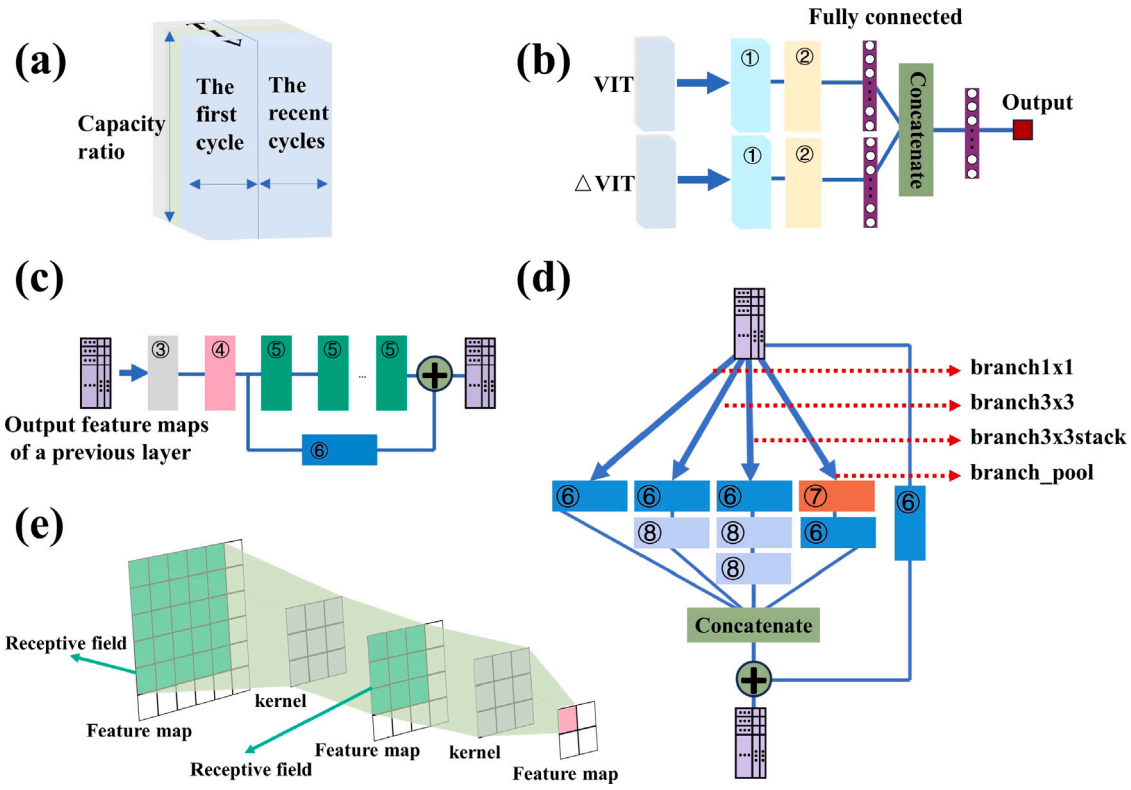

Fig. 3. Detailed architecture and components of the FPNN.

providing a solid foundation for subsequent predictions. The expression for the activation function is as follows,

$$f(z) = \begin{cases} z(x) & \text{if } z(x) > 0, \\ \alpha z(x) & \text{if } z(x) \le 0. \end{cases} \tag{3}$$

Within this context, is the input to the activation function, $f(z)$ is the output from the activation function, and α is a small constant, typically ranging between 0 and 1 . Subsequently, the electrochemical features merged via the 3DCNN are passed to the InceptionBlocks flexible modules for high-level feature extraction, as shown below:

$$O_{\text{blocks}}(x) = g(f(z)) \tag{4}$$

where $f(z)$ denotes the output from the 3D CNN layer, and $O_{\text{block}}(x)$ represents the output from the flexible InceptionBlocks modules, that is, advanced multiscale features. These features are then fed into the fully connected layers (5) within each branch network for nonlinear mapping, preparing for the feature fusion across different branch networks.

$$F_{\text{subnet}}(x) = f\left(W O_{\text{block}}(x) + b\right) \tag{5}$$

Here, $F_{\text{subnet}}(x)$ is the output vector from the fully connected layer within a single branch, where f is the Leaky ReLU (3) activation function, W is the weight matrix of the fully connected layer, and b is the bias vector.

$$O_{FPNN}(x) = W \cdot \left[F_{\text{subnet }1}(x_1), F_{\text{subnet }2}(x_2)\right] + b \tag{6}$$

In this equation, $O_{FPNN}(x)$ is the output of FPNN, and $\left[F_{\text{subnet1}}(x_1), F_{\text{subnet2}}(x_2)\right]$ is the result of concatenating the output vectors from the two branch networks. The forward propagation algorithm for FPNN is detailed in Algorithm 1.

Algorithm 1 Forward Propagation Algorithm of FPNN

Input: Feature x_1, Differential feature x_2
Output: Output of FPNN
Step 1: 3D CNN integrates electrochemical features by Eq. (1).
Step 2: InceptionBlocks extracts advanced electrochemical features by Eq. (4).
Step 3: Fully connected layers in subnetworks further extract features by Eq. (5).
Step 4: Features from different subnetworks are integrated by Eq. (6).

Return: Output of FPNN

As shown in Fig. 3, detailed architecture and components of the FPNN: ① a 3D convolutional layer using 3 × 3 convolutional kernels and 64 channels; ② an InceptionBlocks module; ③ a 2D convolutional layer with a kernel size of 7 × 7 and 64 channels; ④ a max-pooling layer with a pooling kernel size of 3 × 3; ⑤ an InceptionBlock flexible unit; ⑥ a 2D convolutional layer with a kernel size of 1 × 1 and 16 or 24 channels (used as the target channel number for residual connections in other cases); ⑦ an average pooling layer with a pooling kernel size of 3 × 3; and a ⑧ 2D convolutional layer with a kernel size of 3 × 3 and 16 or 24 channels. The figure also shows: (a) Preprocessed FPNN video-like data; (b) FPNN's overall architecture; (c) Detailed structure of the flexible module InceptionBlocks; (d) Specific details of the InceptionBlock flexible unit; (e) Illustration of the receptive field.

3.3. Flexible modules in FPNN: InceptionBlocks

As shown in Fig. 3(c), the electrochemical features $f(z)$ processed by the 3D CNN are passed to the flexible module, InceptionBlocks, for advanced feature extraction. The InceptionBlocks module consists of an

initialization layer followed by multiple InceptionBlock flexible units. The initialization layer is composed of a 7 × 7 convolutional kernel, a 2D CNN with 64 channels, and a max pooling layer with a 3 × 3 pooling kernel (7), aiming to rapidly reduce the dimensionality of feature data. This not only alleviates the model's hardware requirements but also prepares for subsequent feature extraction by the InceptionBlocks.

The operation of the max pooling layer can be represented as:

$$P(i,j) = \max_{0\le m<K, 0\le n<L} I(i\times S+m, j\times S+n) \tag{7}$$

where $P(i,j)$ is the element of the pooling output, $I(i\times S+m, j\times S+n)$ is the element of the input feature $f(z), K, L$ are the dimensions of the pooling window, with K as the height and L as the width. S is the stride, controlling the step size of the sliding window. (i,j) indexes the output feature map, and (m,n) are the relative indexes within the pooling window.

The 2D CNN operation includes 2D convolution (8), normalization, and non-linearization using Leaky ReLU (3) to extract the non-linear electrochemical information:

$$C(i,j) = \sum_m \sum_n I(i+m, j+n)\cdot K(m,n) \tag{8}$$

where $I(i+m, j+n)$ is the element in the input matrix, $K(m,n)$ is the element in the convolution kernel, m and n are the row and column indexes of the convolution kernel, and $C(i,j)$ is the element in the convolution output. The output of the max pooling layer then serves as the input to the first InceptionBlock, with each subsequent InceptionBlock's output passed to the next layer. Inception-ResNetA has been adapted and simplified to form the InceptionBlock: by reducing the number of channels in the feature map to decrease memory usage, thus easing the hardware requirements for model deployment. To ensure the features extracted by the initialization layer are effectively transferred across different InceptionBlocks, residual connections are introduced in this study. The number of InceptionBlocks (NOI) within InceptionBlocks is not fixed, as different EPEOL tasks may require network architectures of varying complexity.

The detailed structure of each flexible unit, the InceptionBlock, is shown in Fig. 3(d). The InceptionBlock takes the output from its preceding layer as input and splits into four branches, each composed of 2D CNNs with varying configurations. The first branch includes a 16-channel 2D convolution layer with a 1 × 1 convolution kernel; the second branch consists of a 16-channel 2D convolution layer with a 1 × 1 convolution kernel followed by a 24 -channel 2D convolution layer with a 3 × 3 convolution kernel; the third branch contains a 16 -channel 2D convolution layer with a 1 × 1 convolution kernel and two 24 -channel 2D convolution layers with 3 × 3 convolution kernels; the fourth branch is composed of an average pooling layer with a 3 × 3 pooling kernel and a sequence of a 24-channel 2D convolution layer with a 3 × 3 convolution kernel. The operation of the average pooling layer can be represented as:

$$P(i,j) = \frac{1}{K\times L}\sum_{m=0}^{K-1}\sum_{n=0}^{L-1} I(i\times S+m, j\times S+n) \tag{9}$$

where $P(i,j)$ is the element in the pooling output, $I(i\times S+m, j\times S+n)$ is the element in the input feature map, $K\times L$ is the size of the pooling window, K is the height of the window, and L is the width. S is the stride, controlling the sliding step of the window. (i,j) indexes the output feature map, and (m,n) are the relative indexes within the pooling window. The outputs of these four branches are concatenated together, forming the output features of the InceptionBlock. To ensure effective transfer of extracted feature information between layers within the InceptionBlock, residual connections (10) are used, which include a linear transformation to adjust the dimensions of the input and output for compatibility.

$$R(I) = F(I) + WI \tag{10}$$

where $F(I)$ is the output of a series of operations (such as convolution, activation, etc.) within the residual block, $R(I)$ is the output of the residual block. I is the input to the residual block. Since the dimensions of $F(I)$ and I do not match, a linear transformation W is added to adjust the dimensions of to match $F(I)$.

In Fig. 3(d), we observe that different branches employ varying numbers of convolutional layers. This design enables the model to learn features at multiple scales. As shown in Fig. 3(e), an example calculation of a 2D CNN layer with two layers of 3 × 3 convolution kernels is displayed. On the final output's 2 × 2 feature map, the pink pixel values are computed based on a 3 × 3 region in the previous layer's feature map, which is referred to as the receptive field in the current feature map. However, this 4 × 4 feature map is also the result of convolution operations from the previous layer. Here, the pixel values in the green 3 × 3 region are computed from a 5 × 5 area in the layer above. This means that receptive fields of different sizes can capture feature information at various scales. Therefore, by designing receptive fields of diverse ranges within the flexible units, FPNN is capable of learning electrochemical features across different scales.

3.4. Training and inference algorithm

This study uses Bayesian optimization [43] to determine the hyperparameters of FPNN. Bayesian optimization employs Gaussian process regression as a surrogate model to provide predicted values and their confidence intervals for prediction points, effectively balancing exploration and exploitation. Among these hyperparameters, NOI is a key parameter that endows FPNN with the ability to adapt to different EPEOL tasks.

Due to the significant differences in data distribution across different data scenarios (10–100 cycles), the optimal hyperparameters for different scenarios may vary significantly. We mainly investigate the impact of the NOI hyperparameter on model performance. Therefore, other hyperparameters should remain constant across different data scenarios (10–100 cycles) to analyze the role of NOI. We chose to perform hyperparameter search in the 60-cycle data scenario and use this set of hyperparameters as the common hyperparameters for other data scenarios to study the role of NOI.

However, since less data in early prediction tasks means earlier prediction of lifespan, allowing for earlier understanding of battery status and saving more resources, we also conducted hyperparameter search in a relatively early data scenario (30 cycles). We did not choose the 10-cycle data scenario for the search because the amount and distribution of data in 10 cycles differ too much from other scenarios. The final choice of 30 and 60 cycles was the result of weighing various factors.

Since experiments conducted in the 30-cycle scenario are more practically significant, this study mainly explores hyperparameter search in the first 30 cycles and analyzes the role of NOI. The experimental data for the hyperparameter search in the first 60 cycles is provided in the supplementary information. Unless otherwise specified, the experiments and analyses in the following text are based on hyperparameters determined in the first 30 cycles.

To comprehensively evaluate the model's predictive performance, this study employs the MAPE (11), Mean Absolute Error (MAE)(12), and Root Mean Square Error (RMSE)(13) as evaluation metrics, with their corresponding mathematical expressions as follows:

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i-\hat{y}_i}{y_i}\right| \tag{11}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i-\hat{y}_i\right| \tag{12}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2} \tag{13}$$

Table 1
Hyperparameters for the model.

Parameter	Value
Batch size	32
Learning rate	0.006364460432422959
Beta1	0.6321610255878152
T_max	50
Weight decay	0.0037294309652403904
Patience	60
Step size	67
Gamma	0.7006417870520055
Warmup epochs	89
Leaky slope	0.05926767475346575
Gradient clip	78.52765595281589
Alpha RMSprop	0.9136110938572766
Initial channels	256
FC1 output dimension	8055
Scheduler type	StepLR
Optimizer type	RAdam
Convolution mode	fan_in
Linear mode	fan_in

where: n is the total number of samples, y_i is the actual value of the ith sample, and $\hat{y}_i$ is the predicted value of the ith sample.

This study introduces a novel one-stop training and inference algorithm, as outlined in Algorithm 2. As previously described, data are first split into training and test sets in Step 1, followed by a series of data preprocessing operations in Step 2. Subsequently, in Step 3, a portion of the training set is allocated as a validation set for parameter tuning during the Bayesian optimization process. The Bayesian optimization phase begins with the initialization of a set of hyperparameters in Step 4, under which the FPNN model is trained. Each training iteration covers multiple epochs, entailing the process of forward and backward propagation (Steps 5 and 6).

Through a complete cycle of forward and backward propagation, the model traverses the entire dataset, facilitating knowledge acquisition and weight optimization. By repeating this process, the model can be quickly optimized. Here, the number of training epochs is set to a high value to ensure sufficient optimization iterations. To maintain training efficiency, an early stopping strategy is implemented: if no reduction in validation loss is observed after several iterations, the current hyperparameter trial is terminated, and step 4 is restarted to explore the next set of hyperparameters. This inner loop continues until the validation loss drops to a predetermined lower threshold. Once hyperparameter tuning is complete, step 7 returns the optimal set of hyperparameters for subsequent model training and inference. Table 1 shows some of the final important hyperparameters:

Before formally proceeding with model training and inference, to fully utilize the data resources, the training and validation sets are merged into a new training set (Step 8). Then, employing the previously identified optimal hyperparameter combination, the model undergoes multiple epochs of iterative training. Given FPNN's architectural adjustment capability, the training epochs are set to 1000 to prevent undertraining due to insufficient iterations. Next, the model is saved and loaded (Step 12), and its performance is evaluated using the test set (Step 13), marking the conclusion of the model inference phase. These inference results will be presented in subsequent sections.

Overall, compared to traditional training and inference algorithms, the main improvements and integrations are as follows:

1. **Full utilization of data**: We divided the dataset into training, validation, and test sets. To ensure the model's generalization ability, hyperparameters were first identified through validation set performance. After determining the hyperparameters, the training and validation sets were combined into a new training set to fully utilize the data. Since the model can control complexity based on the NOI value to avoid underfitting or overfitting, we no longer rely on the validation set to monitor performance. Instead, the number of iterations is set to a large value to ensure sufficient iterations.
2. **Faster convergence**: We used Bayesian optimization to find hyperparameters. Bayesian optimization uses Gaussian process regression as a surrogate model, enabling faster convergence. Additionally, an early stopping strategy was adopted to further accelerate convergence.

Algorithm 2 Training and Evaluation of FPNN

Input: Dataset
Output: Trained Model, Test Loss
Step 1: Split the dataset into training and test sets.
Step 2: Perform data preprocessing, including cleaning, scaling, and sparsifying the data.
Step 3: Further split the training set into a new training set and a validation set.
Repeat
Step 4: Initialize or adjust hyperparameters using Bayesian optimization algorithm.
Do
Repeat
Outer Loop for multiple epochs:
Do
Forward Propagation
Backward Propagation:
Step 5: Calculate gradients based on the training loss.
Step 6: Update network parameters.
End
Until early stopping counter is reached without a decrease in minimum validation loss.
End
Until the validation loss is below the predefined threshold.
Step 7: Return the optimized hyperparameters.
Step 8: Concatenate the training and validation sets to form a new training set.
Step 9: Set the hyperparameters found to be optimal in the previous steps.
Do
Repeat
Outer Loop for multiple epochs:
Do
Forward Propagation
Backward Propagation:
Step 10: Calculate gradients based on the training loss.
Step 11: Update network parameters.
End
Until reach a predefined number of epochs.
End
Step 12: Save the model and load the model.
Step 13: Forward Propagate on the test set to compute the test loss.
Return Trained Model, Test Loss

4. Results and discussion

4.1. Evaluation of FPNN model predictive performance

This section evaluates the EPEOL prediction performance of FPNN under different cycle data, particularly analyzing the impact of NOI (number of InceptionBlocks). The study results show that different NOI settings significantly affect the prediction accuracy of FPNN in EPEOL tasks. As shown in Table 2, using 10-cycle data as an example, when

Table 2
Impact of NOI on FPNN's EPEOL performance.

Used early cycles	Blocks	MAPE (%)	MAE (Cycles)	RMSE (Cycles)
	0	1.55	14.49	29.32
	1	1.86	17.35	36.99
10	2	1.79	16.86	32.27
	3	1.26	10.47	16.23
	4	1.77	15.23	25.00
	0	0.66	7.08	17.50
	1	0.76	6.49	11.72
20	2	0.69	6.24	12.79
	3	0.61	5.57	12.31
	4	0.41	3.71	8.02
	0	0.63	4.86	9.95
	1	0.37	3.38	8.34
30	2	0.79	7.38	36.98
	3	0.50	4.38	8.46
	4	0.42	3.99	16.14
	0	0.33	2.83	5.56
	1	0.57	5.19	17.21
40	2	0.59	5.28	10.81
	3	0.49	4.45	11.98
	4	0.46	3.56	5.61
	0	0.65	5.90	9.89
	1	1.26	11.04	16.35
50	2	0.32	3.22	6.74
	3	0.84	7.64	11.05
	4	NaN	NaN	NaN
	0	0.32	2.41	5.06
	1	0.41	3.53	7.84
60	2	0.41	3.2	5.02
	3	0.59	5.15	10.56
	4	0.46	3.98	8.87
	0	0.41	3.54	6.68
	1	0.31	2.8	7.12
70	2	0.38	3.43	7.96
	3	0.56	5.64	14.14
	4	0.51	5.12	15.27
	0	0.51	4.27	6.41
	1	0.6	4.75	6.01
80	2	0.69	6.45	11.21
	3	0.31	2.99	10.49
	4	0.49	4.13	7.32
	0	0.22	1.77	2.44
	1	0.42	3.38	6.68
90	2	0.55	4.94	8.16
	3	0.29	3.11	9.92
	4	0.41	2.97	4.5
	0	0.34	2.37	3.86
	1	0.48	4.61	10.18
100	2	0.39	3.47	5.78
	3	0.62	4.9	6.9
	4	0.6	4.98	6.74

Note: "NaN" indicates that during the training process, the gradient descent optimization algorithm iterated abnormally, resulting in a calculated Loss value of NaN, making further gradient computation impossible.

Table 3
Prediction errors of different model structures at optimal NOI.

Used early cycles	Detach	MAPE (%)	MAE (Cycles)	RMSE (Cycles)
	None	1.26	10.47	16.23
	Residual	1.55	14.57	26.66
10	3D conv	2.35	22.88	45.68
	A branch	96.07	762.5	832.58
	Initial layers	–	–	–
	None	0.41	3.71	8.01
	Residual	0.89	8.12	32.55
20	3D conv	0.96	8.76	18.19
	A branch	90.41	753.73	853.68
	Initial layers	–	–	–
	None	0.37	3.38	8.33
	Residual	0.39	3.28	8.53
30	3D conv	0.52	4.61	8.78
	A branch	84.17	726.55	842.75
	Initial layers	–	–	–
	None	0.33	2.82	5.56
	Residual	0.45	3.84	7.23
40	3D conv	0.44	3.65	6.64
	A branch	79.17	694.75	825.33
	Initial layers	–	–	–
	None	0.32	3.22	6.74
	Residual	0.63	5.89	10.56
50	3D conv	0.68	5.81	9.94
	A branch	75.44	682.8	829.02
	Initial layers	–	–	–
	None	0.32	2.41	5.06
	Residual	0.38	3.05	5.63
60	3D conv	0.49	3.69	5.67
	A branch	72.1	651.58	805.79
	Initial layers	–	–	–
	None	0.31	2.8	7.12
	Residual	0.42	3.92	13.27
70	3D conv	0.49	4	6.96
	A branch	71.08	659.02	818.57
	Initial layers	–	–	–
	None	0.31	2.99	10.49
	Residual	0.54	4.43	5.65
80	3D conv	0.51	4.58	7.5
	A branch	68.9	630.14	785.73
	Initial layers	–	–	–
	None	0.22	1.77	2.44
	Residual	0.3	2.5	3.61
90	3D conv	0.38	3	3.88
	A branch	64.21	600.75	773.62
	Initial layers	–	–	–
	None	0.34	2.37	3.86
	Residual	0.39	3.36	6.61
100	3D conv	0.37	2.57	4.88
	A branch	64.26	599.38	770.49
	Initial layers	–	–	–

Note: "–" indicates that due to excessive memory consumption by the model after removing the initialization layer in the InceptionBlocks, the ablation experiment could not proceed, and data was not collected; "A branch" refers to the differential feature branch removed in the dual-stream network.

NOI is set to 0, despite not using InceptionBlock, FPNN still achieved a low MAPE of 1.55% through effective collaboration of other modules, demonstrating high accuracy. However, when NOI increased to 1 or 2, MAPE rose to 1.86% and 1.79%, respectively, indicating a decline in model performance, possibly due to inappropriate model structure design. When NOI was set to 3, the model performance was optimal, with MAPE dropping to 1.26%, possibly due to the good synergy of multiple InceptionBlocks making the model architecture best suited for the current task. However, when NOI further increased to 4, MAPE rose again to 1.77%, possibly due to a deeper network weakening the nonlinear information transmission capability.

Using the first 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 cycle data, the optimal NOI values are 3, 4, 1, 0, 2, 0, 1, 3, 0, and 0, respectively, with MAPE values of 1.26%, 0.41%, 0.37%, 0.33%, 0.32%, 0.32%, 0.31%, 0.31%, 0.22%, and 0.34%, respectively. Overall, the model's prediction performance improves with the increase in early data volume. However, when using the first 100 cycle data, the model's optimal performance increases to 0.34%. This might be due to the prediction error being already very small, leading to statistical fluctuations from 0.22% to 0.34%. Additionally, the data volume and distribution of 90 cycles are more similar to 30 cycles, hence showing lower error.

Specifically, when the early data volume is small (10, 20 cycles), a larger NOI (3, 4) is necessary. As the data volume increases (30, 40, 50, 60, 70, 90, 100 cycles), the optimal NOI reduces to 1, 0, 2, 0, 1, 0, and 0, respectively. This indicates that with larger data volumes, a simpler FPNN structure is sufficient to achieve high-accuracy predictions. This may be because, with larger data volumes, a larger NOI makes the optimization space more complex, making it easier for the model to jump to points away from the local optimum, thus making gradient descent-based optimization more challenging. Of course, the case of 80 cycles is an exception, with an optimal NOI of 3, which aligns with the basic statistical rule that most data conforms to general patterns, with some anomalies.

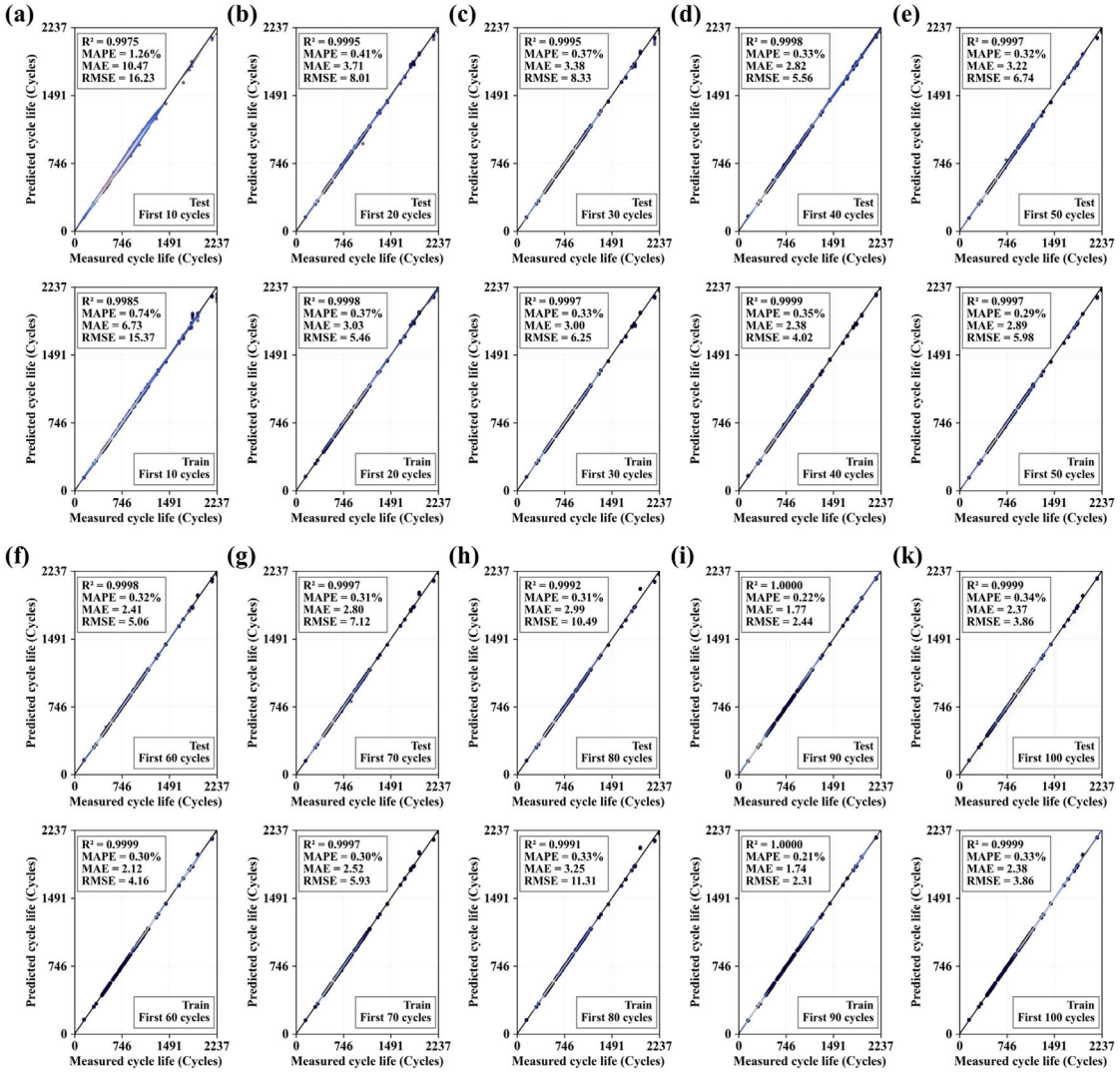

Fig. 4. Presents the performance of the FPNN model in EPEOL tasks with different early data volumes under optimal NOI settings.

This finding suggests that as the data volume increases, some complex components of FPNN may become unnecessary and may even negatively impact prediction performance. Conversely, when the data volume is small, a larger NOI is necessary to achieve higher accuracy with a more complex architecture, better fitting the current data distribution. However, in the supplementary information, when using hyperparameters determined by the first 60 cycles of data as the common hyperparameters, the choice of NOI did not show this pattern. This might be because the learning rate determined by the first 60 cycles of data is much smaller compared to that determined by the first 30 cycles of data, making the model's optimization steps smaller and less likely to jump to unfavorable points, leading to better convergence. Nevertheless, even when using hyperparameters determined by the first 60 cycles of data, NOI still shows slight variations, indicating that the model can adjust NOI to control the structure and complexity to better fit the current data distribution.

Notably, when using 50 cycle data with an NOI of 4, the model's error was NaN. NaN stands for "Not a Number". Since the numerical display range of computers is limited, when values exceed this range, the computer will display NaN. NaN appears when values are too large or too small. During the optimization process of the gradient descent algorithm, the model jumped to a state that resulted in NaN values, which is a low-probability event. This situation might be due to the model's complexity, with more layers and numerical calculations, increasing the probability of numerical overflow.

Table 4
EPEOL performance of other published methods.

Methods	MAPE (%)	RMSE (Cycles)	Used early cycles
SVM [24]	8.00	115.00	100
Linear model [14]	7.50	100.00	100
AlexNet [44]	–	91.51	100
GBRT [23]	7.00	82.80	250
HCNN [38]	3.08	42.00	20
	1.12	13.00	60
	1.26	**16.23**	**10**
Proposed method	**0.32**	**6.74**	**50**
	0.22	**2.44**	**90**

Note: "–" indicates that the data was not available.

Fig. 4 shows the predictive performance of FPNN under optimal NOI settings. The first row in the figure displays the performance of FPNN on the test set when trained with early data from 10, 20, 30, 40, and 50 cycles. Specifically, Fig. 4(a) shows the performance of FPNN trained with 10 cycle data on the EPEOL task, with an overall MAPE of 1.26%. Although some samples exhibit larger prediction errors, most samples have smaller errors, demonstrating high overall prediction accuracy. This verifies the effectiveness of the FPNN model, especially when data is limited. Fig. 4(b)–(e) sequentially show the performance of FPNN trained with 20, 30, 40, and 50 cycle data on the EPEOL task. As the early cycle data increases, the model's prediction performance

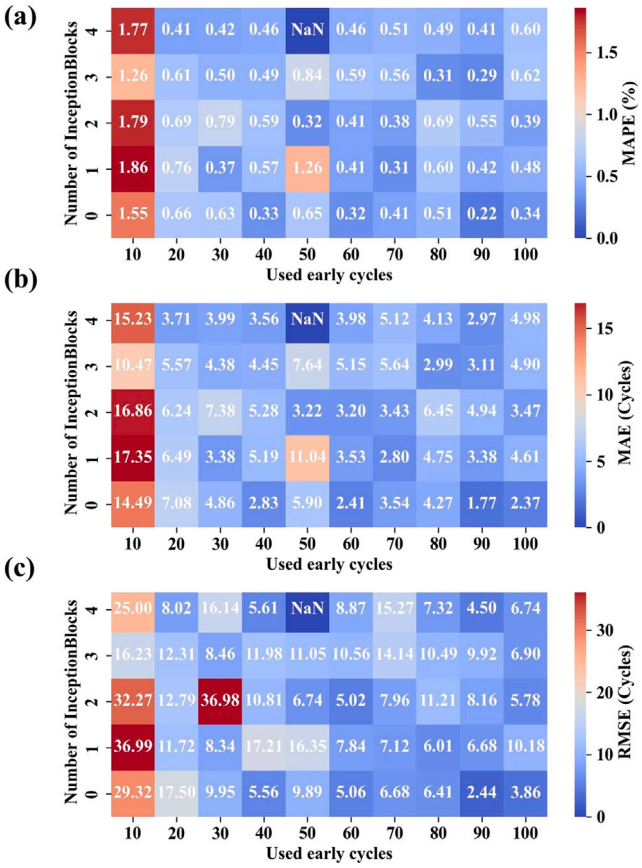

Fig. 5. This figure details the impact of different early cycle data volumes and NOI settings on FPNN's prediction performance: (a) Heatmap of MAPE; (b) Heatmap of MAE; (c) Heatmap of RMSE.

significantly improves, with MAPE below 0.5% in all cases, achieving very high accuracy. This highlights the FPNN's ability to dynamically fit the current data distribution highly.

Fig. 4 second row shows the performance of FPNN on the training set when trained with 20, 30, 40, and 50 cycle data. Except for the case with 40 cycle data, the performance on the training set is generally better than on the test set, indicating that no data leakage occurred during the training process of this study. When trained with the first 40 cycles, the model's performance on the training set (MAPE of 0.35%) is similar to its performance on the test set (MAPE of 0.33%), with slightly better performance on the test set. This may be due to the inherent randomness of machine learning methods. Fig. 4(f)–(k) show similar patterns, which will not be further elaborated here.

"Data leakage" refers to a situation in machine learning where the model unintentionally uses information outside the training dataset during its creation. This leads to overly optimistic performance evaluations because the model has access to data that it typically would not have in real-world scenarios. Simply put, it means the model "saw' information during training that it should not have, resulting in excellent performance during training but poor performance on new, unseen data. This happens because the model relies on invalid information that is not available during actual predictions. Overall, the better performance on the training set compared to the test set indicates that no "data leakage" occurred during the training process in this study, making the results more convincing.

Fig. 5 comprehensively shows the performance of FPNN in the EPEOL task. Specifically, Fig. 5 presents the prediction performance of FPNN when handling early data from the first 10 to 100 cycles, including (a) a heatmap of MAPE changes, (b) a heatmap of MAE changes, and (c) a heatmap of RMSE changes. The results show that the model's MAE and RMSE exhibit the same trend as MAPE. Since MAPE is a relative error metric that can effectively reduce the risk of excessive error value changes due to differences in cycle life, MAPE will be prioritized in subsequent discussions. The conclusions obtained here are consistent with previous analyses and will not be repeated.

Fig. 6(a)–(b) shows the variation in FPNN prediction performance (box plots of MAPE and MAE error distributions) under different early cycle data volumes and NOI settings. The results indicate that the overall volatility of the model's prediction performance is low, with the maximum MAPE not exceeding 5%. As the early data volume increases, the range of the model's prediction error distribution gradually narrows until it fluctuates within a small range, indicating increasingly stable model performance. However, when using early data from 50 cycles, the error volatility increases, which may be due to randomness and chance in model optimization. As shown in Fig. 6(c), with the increase in early data volume, the model's MAE and RMSE gradually decrease until they fluctuate within a small range. Fig. 6(d)–(e) presents the prediction results for the short-lifespan battery "b2c1" and the long-lifespan battery "b1c2" samples. Even in these extreme cases, the model's prediction performance remains excellent, demonstrating the high robustness of the method proposed in this study.

4.2. Analysis of the effects of the parallel network, initial layers, ResNet, and 3D CNN

To gain a deeper understanding of the roles of various FPNN modules and their contributions to prediction accuracy, ablation experiments were conducted, and the results are presented in Table 3. "Ablation experiments" are a method used in machine learning and artificial intelligence to understand the importance and impact of different components of a model. In ablation experiments, specific parts of the model are systematically removed or modified to observe how these changes

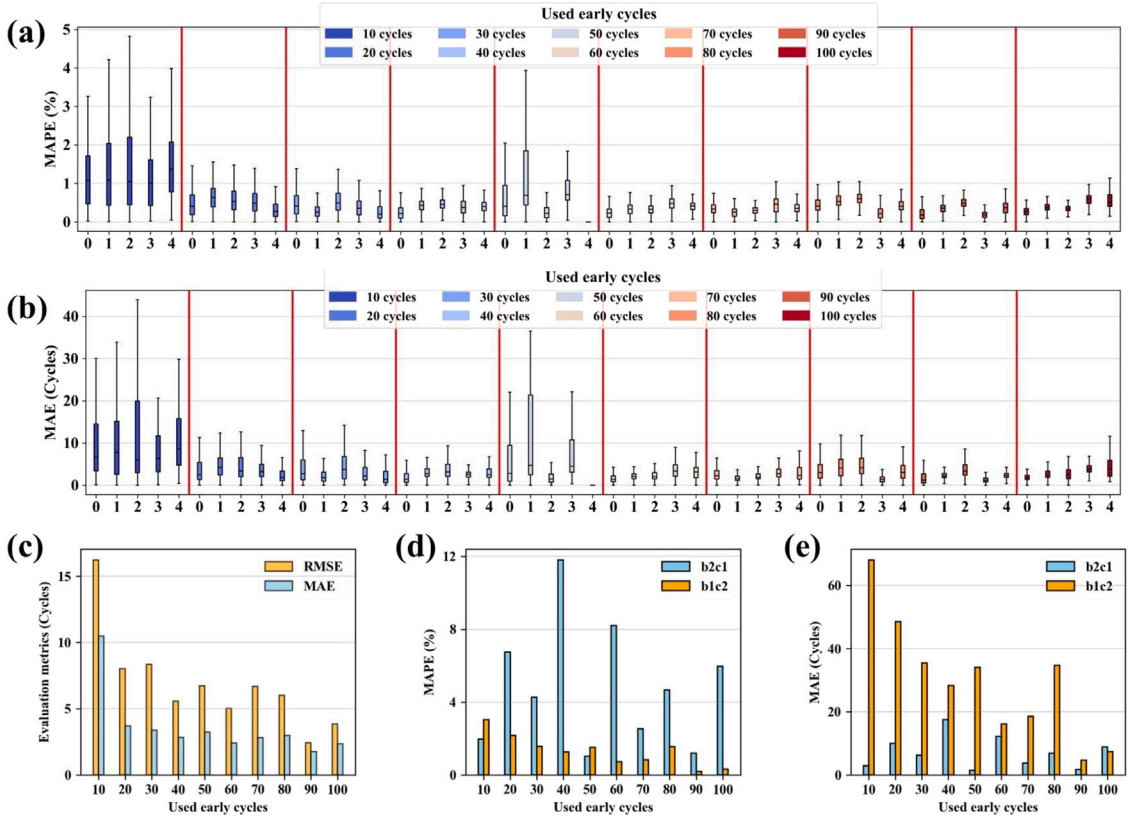

Fig. 6. Comprehensively shows the overall performance of FPNN in the EPEOL task. Box plots of errors: (a) MAPE; (b) MAE; (c) Bar chart of errors for all battery samples; Bar charts of errors for a single sample: (d) MAPE; (e) MAE.

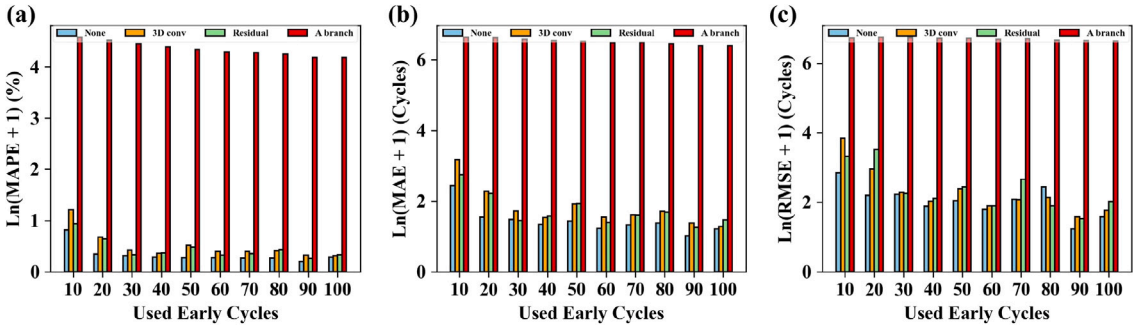

Fig. 7. Results of the FPNN ablation experiments: (a) MAPE; (b) MAE; (c) RMSE.

affect the model's performance. This helps identify which components are essential and how they contribute to the overall effectiveness of the model.

The impact of removing specific components from FPNN was analyzed under the optimal NOI settings. Fig. 7 visually presents the results of the ablation experiments. In the figure, NaN values are replaced with 0, and due to significant differences in data extremes, a simple mathematical transformation was applied, $error_{log} = \ln(1+error)$, where error represents the error evaluation metric and $error_{log}$ is the value on the vertical axis in the figure.

Using early data from only 10 cycles, the results highlight the critical role of the differential feature branch in improving model accuracy. Removing this branch led to a sharp increase in MAPE to 96.07%, with MAE and RMSE rising to 762.5 cycles and 832.58 cycles, respectively. Additionally, removing the residual connections caused a slight increase in MAPE to 1.55%, with corresponding increases in MAE and RMSE, underscoring the importance of residual connections in reducing prediction error. Removing the 3D convolution layer increased MAPE to 2.35%, further confirming the critical role of the 3D convolution layer in feature extraction. Finally, when the initialization layer was removed, the model's memory usage became too high, making training infeasible, which reveals the importance of the initialization layer for model stability.

When trained with early data from 20–100 cycles, the patterns observed when removing each module were similar to those seen with 10 cycles of early data, indicating that each component of the model is crucial for overall performance.

4.3. Comparison with established methods

As shown in Table 4, the proposed FPNN method offers significant performance advantages over other published EPEOL prediction techniques. The FPNN model trained with the first 10, 50, and 90 cycle data achieved MAPE values of 1.26%, 0.32%, and 0.22%, respectively, outperforming other methods such as SVM, linear models, AlexNet, GBRT, and HCNN in terms of MAPE. Additionally, the FPNN model

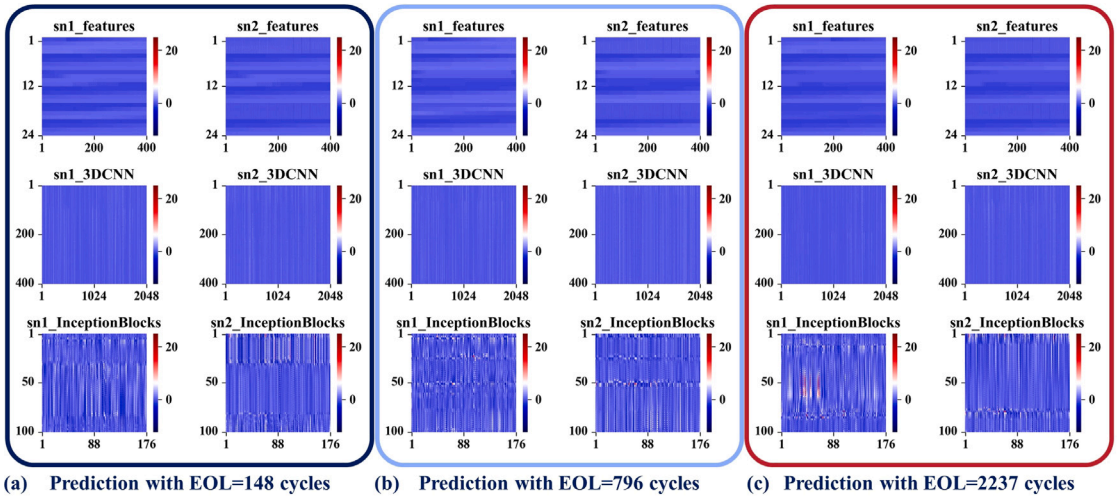

Fig. 8. Reveals the specific roles and mutual influences of various modules in the FPNN model when predicting samples with different EOLs, uncovering the internal mechanism of the model in handling complex EPEOL tasks.

requires significantly less early cycle data, highlighting its excellent capability in improving data utilization efficiency.

In other publications, some use traditional machine learning methods such as SVM, linear models, and GBRT. Generally, these have limited fitting capabilities compared to neural networks and require complex feature engineering. Some publications use early neural network architectures like AlexNet, which are now outdated. While HCNN does demonstrate strong fitting capabilities, our designed FPNN combines the advantages of core modules such as InceptionBlock, along with unique data processing methods and algorithms, exhibiting even stronger data fitting abilities.

Overall, when FPNN performs EPEOL tasks, the optimal NOI shows slight variations when other hyperparameters remain unchanged. This indicates that the model can adjust its structure and complexity by tuning the NOI to better fit the current data distribution.

5. Model interpretability analysis

5.1. Analysis of the synergistic action of FPNN's overall architecture

To delve into the model's interpretability, this section analyzes in detail the internal mechanism of the FPNN model in handling EPEOL tasks. As shown in Fig. 8, the figure selects and presents the prediction cases of individual samples with different EOLs from the early data test set of the first 10 cycles, where "sn" represents the subnet. In Fig. 8(a), a sample with an EOL of 148 inputs electrochemical features composed of voltage, current, temperature, and charging capacity data into one branch of the model. Since this data array is four-dimensional and not conveniently visualized directly, specific dimensions have been flattened for two-dimensional visualization. The first row, first column of the figure shows the flattened electrochemical features, while the first row, second column displays the differential features inputted into another branch network. After the data enters the model, it is first subjected to initial feature fusion by the 3D CNN, with the fused features displayed in the second row, where features appear more uniform. These fused features are then fed into the flexible InceptionBlocks module for extracting advanced electrochemical features. In Fig. 8(b)–(c), the electrochemical feature extraction cases of samples with different EOLs are shown. It is evident that through the synergistic action of multiple modules, the model can differentiate samples of different EOLs and learn their respective advanced electrochemical features. This demonstrates that the FPNN model can precisely capture the complex nonlinear relationship between the electrochemical features of batteries with different lifespans and their EOL.

5.2. Interpretation of learning outcomes of the flexible unit InceptionBlock

The design of the flexible unit InceptionBlock includes four branches with different functions, each composed of a series of meticulously designed convolutional layers. Taking branch 1 × 1 as an example, the 1 × 1 convolution layer mainly handles channel number conversion and feature information integration. This not only reduces the model parameters but also enhances computational efficiency. Within the InceptionBlock, most branches begin with a 1 × 1 convolution kernel, aimed at quickly adjusting the channel number of the feature map, preparing for deeper feature extraction. This rapid channel transition facilitates the fusion of information across different channels, enhancing the model's adaptability to input data and generalization ability. Through optimizing weight parameters, the neural network learns features from data and adapts to the target function. Weight visualization offers insights into the distribution of weights learned by convolutional layers in each branch, aiding in understanding the model's decision-making process.

Fig. 9 shows the heatmaps of convolutional layer weights in the first InceptionBlock within InceptionBlocks of the FPNN model trained with early data from 10 cycles, revealing the roles of different convolutional layers in feature extraction: (a) The convolutional layer with 1 × 1 kernels in branch1 × 1; (b) The convolutional layer with 1 × 1 kernels in branch3 × 3; (c) The convolutional layer with 1 × 1 kernels in branch3 × 3stack; (d) The convolutional layer with 1 × 1 kernels in branch_pool; (e) The convolutional layer with 1 × 1 kernels in residual_conv; (f) The convolutional layer with 3 × 3 kernels in branch3 × 3; (g) The first convolutional layer with 3 × 3 kernels in branch3 × 3stack; (h) The second convolutional layer with 3 × 3 kernels in branch3 × 3stack.

As demonstrated in Fig. 9(a)–(e), the sizes of the convolution kernel weights in the 1 × 1 convolution layer vary, revealing how FPNN differentially processes information across channels. Especially in branch_pool, the 1 × 1 convolution layer following the pooling layer highlights its ability to effectively transform features while reducing dimensions.

Fig. 9(f) showcases the weight distribution of the 3 × 3 convolution kernels in the branch3 × 3, compared to the 1 × 1 convolution kernels, revealing its capability for specialized feature extraction. FPNN, through this approach, reduces dependency on a single feature set, thereby enhancing the model's generalizability. In Fig. 9(g)–(h), the branch3 × 3stack branch achieves a receptive field equivalent to larger convolution kernels by stacking multiple 3 × 3 convolution layers, while maintaining a lower number of parameters.

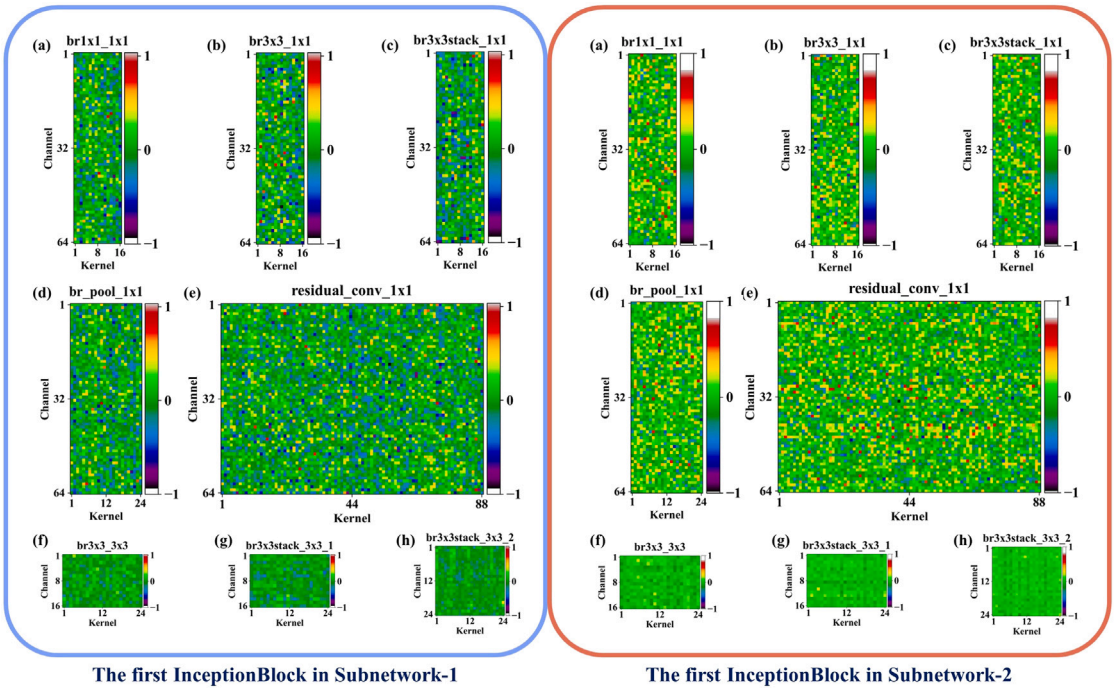

Fig. 9. Shows the heatmaps of convolutional layer weights in the first InceptionBlock within InceptionBlocks of the FPNN model trained with early data from 10 cycles, revealing the roles of different convolutional layers in feature extraction.

By integrating complex nonlinear features extracted from different branches and linear features transmitted through residual connections, FPNN accomplishes effective feature fusion. The multi-layered combination of InceptionBlocks further excavates advanced electrochemical features across a broader scale, significantly enhancing FPNN's predictive accuracy. This section, through in-depth analysis of the learning outcomes within each layer of the flexible unit InceptionBlock, demonstrates FPNN model's strong interpretability.

6. Conclusion

The FPNN model proposed in this study integrates multiple modules, including InceptionBlock, 3D CNN, 2D CNN, and dual-stream networks, significantly enhancing the ability to extract electrochemical features from video-like data. The FPNN model demonstrates high adaptability to different EPEOL tasks by adaptively adjusting the number of InceptionBlocks. Experimental results on the MIT dataset show that the FPNN model achieves excellent prediction accuracy in EPEOL tasks, with MAPE values of 1.26%, 0.41%, 0.37%, 0.33%, 0.32%, 0.32%, 0.31%, 0.31%, 0.22%, and 0.34% when using the first 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 cycle data, respectively.

The study shows that when FPNN performs EPEOL tasks, the optimal NOI shows slight variations when other hyperparameters remain unchanged, indicating that the model can adjust its structure and complexity by tuning the NOI to better fit the current data distribution. Further ablation experiments validated the importance and indispensability of each component in the FPNN architecture. Additionally, visual analysis of the features learned by the overall FPNN architecture and the inner layer weights of the InceptionBlock enhanced the model's interpretability and confirmed its effectiveness in handling complex feature extraction tasks.

Given the similarity between EPEOL tasks and other machine learning tasks in the battery field, these characteristics of FPNN suggest its great potential for broader applications in the battery industry.

CRediT authorship contribution statement

Lidang Jiang: Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Data curation, Conceptualization. **Zhuoxiang Li:** Formal analysis. **Changyan Hu:** Formal analysis. **Junxiong Chen:** Formal analysis. **Qingsong Huang:** Supervision, Funding acquisition. **Ge He:** Writing – review & editing, Supervision, Resources.

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.

Acknowledgments

The authors are grateful for the support of the National Key Research and Development Program of China (No. 2021YFB40005) and the special funding for basic scientific research business expenses in the central universities of China (No. 2022SCU12096).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.energy.2024.132840.

References

[1] Yoshino Akira. The birth of the lithium-ion battery. Angew Chem Int Ed 2012;51(24):5798–800.

[2] Gan Chuanhai, Zhang Chengkun, Wen Weidong, Liu Yingkuan, Chen Juan, Xie Qingshui, Luo Xuetao. Enhancing delithiation reversibility of Li15Si4 alloy of silicon nanoparticles-carbon/graphite anode materials for stable-cycling lithium ion batteries by restricting the silicon particle size. ACS Appl Mater Interfaces 2019;11(39):35809–19.

[3] Sehrawat Poonam, Shabir Abgeena, Julien CM, Islam SS, et al. Recent trends in silicon/graphene nanocomposite anodes for lithium-ion batteries. J Power Sources 2021;501:229709.

[4] Teki Ranganath, Datta Moni K, Krishnan Rahul, Parker Thomas C, Lu Toh-Ming, Kumta Prashant N, Koratkar Nikhil. Nanostructured silicon anodes for lithium ion rechargeable batteries. Small 2009;5(20):2236–42.

[5] Zubi Ghassan, Dufo-López Rodolfo, Carvalho Monica, Pasaoglu Guzay. The lithium-ion battery: State of the art and future perspectives. Renew Sustain Energy Rev 2018;89:292–308.

[6] Han Xuebing, Ouyang Minggao, Lu Languang, Li Jianqiu. A comparative study of commercial lithium ion battery cycle life in electric vehicle: Capacity loss estimation. J Power Sources 2014;268:658–69.

[7] Lin Chun-Pang, Cabrera Javier, Yang Fangfang, Ling Man-Ho, Tsui Kwok-Leung, Bae Suk-Joo. Battery state of health modeling and remaining useful life prediction through time series model. Appl Energy 2020;275:115338.

[8] Ma Qiuhui, Zheng Ying, Yang Weidong, Zhang Yong, Zhang Hong. Remaining useful life prediction of lithium battery based on capacity regeneration point detection. Energy 2021;234:121233.

[9] Fleischer Christian, Waag Wladislaw, Heyn Hans-Martin, Sauer Dirk Uwe. On-line adaptive battery impedance parameter and state estimation considering physical principles in reduced order equivalent circuit battery models: Part 1. Requirements, critical review of methods and modeling. J Power Sources 2014;260:276–91.

[10] Ouyang Tiancheng, Xu Peihang, Chen Jingxian, Lu Jie, Chen Nan. An online prediction of capacity and remaining useful life of lithium-ion batteries based on simultaneous input and state estimation algorithm. IEEE Trans Power Electron 2020;36(7):8102–13.

[11] Allam Anirudh, Onori Simona. Online capacity estimation for lithium-ion battery cells via an electrochemical model-based adaptive interconnected observer. IEEE Trans Control Syst Technol 2020;29(4):1636–51.

[12] Li Weihan, Fan Yue, Ringbeck Florian, Jöst Dominik, Han Xuebing, Ouyang Minggao, Sauer Dirk Uwe. Electrochemical model-based state estimation for lithium-ion batteries with adaptive unscented Kalman filter. J Power Sources 2020;476:228534.

[13] Liu Chang, Wang Yujie, Chen Zonghai. Degradation model and cycle life prediction for lithium-ion battery used in hybrid energy storage system. Energy 2019;166:796–806.

[14] Severson Kristen A, Attia Peter M, Jin Norman, Perkins Nicholas, Jiang Benben, Yang Zi, Chen Michael H, Aykol Muratahan, Herring Patrick K, Fraggedakis Dimitrios, et al. Data-driven prediction of battery cycle life before capacity degradation. Nat Energy 2019;4(5):383–91.

[15] Wang Dong, Yang Fangfang, Tsui Kwok-Leung, Zhou Qiang, Bae Suk Joo. Remaining useful life prediction of lithium-ion batteries based on spherical cubature particle filter. IEEE Trans Instrum Meas 2016;65(6):1282–91.

[16] Jordan Michael I, Mitchell Tom M. Machine learning: Trends, perspectives, and prospects. Science 2015;349(6245):255–60.

[17] Saxena Saurabh, Kang Myeongsu, Xing Yinjiao, Pecht Michael. Anomaly detection during lithium-ion battery qualification testing. In: 2018 IEEE international conference on prognostics and health management. ICPHM, IEEE; 2018, p. 1–6.

[18] Myles Anthony J, Feudale Robert N, Liu Yang, Woody Nathaniel A, Brown Steven D. An introduction to decision tree modeling. J Chemometr: J Chemometr Soc 2004;18(6):275–85.

[19] Noble William S. What is a support vector machine? Nature Biotechnol 2006;24(12):1565–7.

[20] Guo Gongde, Wang Hui, Bell David, Bi Yaxin, Greer Kieran. KNN model-based approach in classification. In: On the move to meaningful internet systems 2003: CoopIS, DOA, and ODBASE: OTM confederated international conferences, CoopIS, DOA, and ODBASE 2003, catania, sicily, Italy, November 3-7, 2003. proceedings. Springer; 2003, p. 986–96.

[21] Zhu Shan, Zhao Naiqin, Sha Junwei. Predicting battery life with early cyclic data by machine learning. Energy Storage 2019;1(6):e98.

[22] Zhang Yunwei, Tang Qiaochu, Zhang Yao, Wang Jiabin, Stimming Ulrich, Lee Alpha A. Identifying degradation patterns of lithium ion batteries from impedance spectroscopy using machine learning. Nature Commun 2020;11(1):1706.

[23] Yang Fangfang, Wang Dong, Xu Fan, Huang Zhelin, Tsui Kwok-Leung. Lifespan prediction of lithium-ion batteries based on various extracted features and gradient boosting regression tree model. J Power Sources 2020;476:228654.

[24] Fei Zicheng, Yang Fangfang, Tsui Kwok-Leung, Li Lishuai, Zhang Zijun. Early prediction of battery lifetime via a machine learning based framework. Energy 2021;225:120205.

[25] LeCun Yann, Bengio Yoshua, Hinton Geoffrey. Deep learning. Nature 2015;521(7553):436–44.

[26] Zhang Yu, Peng Zhen, Guan Yong, Wu Lifeng. Prognostics of battery cycle life in the early-cycle stage based on hybrid model. Energy 2021;221:119901.

[27] Zhao Fen, Li Yinguo, Wang Xinheng, Bai Ling, Liu Tailin. Lithium-ion batteries state of charge prediction of electric vehicles using RNNs-CNNs neural networks. Ieee Access 2020;8:98168–80.

[28] Hong Jichao, Wang Zhenpo, Chen Wen, Wang Le-Yi, Qu Changhui. Online joint-prediction of multi-forward-step battery SOC using LSTM neural networks and multiple linear regression for real-world electric vehicles. J Energy Storage 2020;30:101459.

[29] Ren Lei, Dong Jiabao, Wang Xiaokang, Meng Zihao, Zhao Li, Deen M Jamal. A data-driven auto-CNN-LSTM prediction model for lithium-ion battery remaining useful life. IEEE Trans Ind Inf 2020;17(5):3478–87.

[30] Wei Meng, Ye Min, Li Jia Bo, Wang Qiao, Xu Xinxin. State of charge estimation of lithium-ion batteries using LSTM and NARX neural networks. Ieee Access 2020;8:189236–45.

[31] Yi Yahui, Xia Chengyu, Feng Chao, Zhang Wenjing, Fu Chenlong, Qian Liqin, Chen Siqi. Digital twin-long short-term memory (LSTM) neural network based real-time temperature prediction and degradation model analysis for lithium-ion battery. J Energy Storage 2023;64:107203.

[32] Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N, Kaiser Łukasz, Polosukhin Illia. Attention is all you need. Adv Neural Inf Process Syst 2017;30.

[33] Wang Zili, Liu Yonglu, Wang Fen, Wang Hui, Su Mei. Capacity and remaining useful life prediction for lithium-ion batteries based on sequence decomposition and a deep-learning network. J Energy Storage 2023;72:108085.

[34] Bian Chong, Yang Shunkun, Liu Jie, Zio Enrico. Robust state-of-charge estimation of li-ion batteries based on multichannel convolutional and bidirectional recurrent neural networks. Appl Soft Comput 2022;116:108401.

[35] Chen Dinghong, Zhang Weige, Zhang Caiping, Sun Bingxiang, Cong XinWei, Wei Shaoyuan, Jiang Jiuchun. A novel deep learning-based life prediction method for lithium-ion batteries with strong generalization capability under multiple cycle profiles. Appl Energy 2022;327:120114.

[36] Gong Qingrui, Wang Ping, Cheng Ze, et al. A method for estimating state of charge of lithium-ion batteries based on deep learning. J Electrochem Soc 2021;168(11):110532.

[37] Li Yihuan, Li Kang, Liu Xuan, Wang Yanxia, Zhang Li. Lithium-ion battery capacity estimation—A pruned convolutional neural network approach assisted with transfer learning. Appl Energy 2021;285:116410.

[38] Yang Yixin. A machine-learning prediction method of lithium-ion battery life based on charge process for different applications. Appl Energy 2021;292:116897.

[39] Szegedy Christian, Ioffe Sergey, Vanhoucke Vincent, Alemi Alexander. Inception-v4, inception-resnet and the impact of residual connections on learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 31, 2017.

[40] Ji Shuiwang, Xu Wei, Yang Ming, Yu Kai. 3D convolutional neural networks for human action recognition. IEEE Trans Pattern Anal Mach Intell 2012;35(1):221–31.

[41] Simonyan Karen, Zisserman Andrew. Two-stream convolutional networks for action recognition in videos. Adv Neural Inf Process Syst 2014;27.

[42] Zhang Qisong, Yang Lin, Guo Wenchao, Qiang Jiaxi, Peng Cheng, Li Qinyi, Deng Zhongwei. A deep learning method for lithium-ion battery remaining useful life prediction based on sparse segment data via cloud computing system. Energy 2022;241:122716.

[43] Snoek Jasper, Larochelle Hugo, Adams Ryan P. Practical bayesian optimization of machine learning algorithms. Adv Neural Inf Process Syst 2012;25.

[44] He Ning, Wang Qiqi, Lu Zhenfeng, Chai Yike, Yang Fangfang. Early prediction of battery lifetime based on graphical features and convolutional neural networks. Appl Energy 2024;353:122048.