Journal of Energy Storage 141 (2026) 119508

Contents lists available at ScienceDirect

Journal of Energy Storage

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

Research Papers

CNN-based prediction of multi-variables for lithium-ion batteries optimized with TinyML and deployed on edge devices

Yuqin Weng [a],*, Wenkai Guan [b], Cristinel Ababei [a]

[a] Electrical and Computer Engineering Department, Marquette University, Milwaukee, WI 53233, USA
[b] Division of Science and Mathematics, University of Minnesota, Morris, MN 60450, USA

ARTICLE INFO

Keywords:
Remaining useful life
RUL
Cell temperature
Deep learning
TinyML

ABSTRACT

Estimation or prediction of several battery cell attributes can be very useful in developing operation strategies to prevent battery failures or to schedule battery replacements or updates in electronic devices including electrical vehicles (EVs) that use lithium-ion batteries. Examples of such attributes include state of charge (SoC), remaining useful life (RUL), and temperature. While one can develop individual AI/ML models to predict such attributes, the models may require too much memory and computational capacities that may not be available on resource constrained edge devices that use microcontrollers or small processors. That is why, developing AI/ML models that predict multiple such attributes with a single model may offer a better trade-off between required resources and performance when deployed on edge devices. Therefore, in this paper, we investigate a novel AI/ML model to predict both battery RUL and cell temperature. The proposed novel deep learning model combines a temporal convolutional network (TCN) with a multi-head attention layer that can output two or more variables simultaneously. Simulation results demonstrate that prediction of both RUL and cell temperature can be achieved accurately with a single model and that accuracy is on par with models from existing literature that focused on models for individual predictions. Moreover, we optimize the proposed model using tinyML technologies and deploy it for real time inference on a Raspberry Pi 4 edge device. Experiments show that the model performance is still very good even after such optimizations of the model to make it fit on the edge device.

1. Introduction

Combining artificial intelligence/machine learning (AI/ML) with Internet of Things (IoTs) devices can lead to technologies that are revolutionizing how traditional electronic systems are designed and operated — including battery health management systems in electric vehicles (EVs), hybrid electric vehicles (HEVs), portable power drills, defibrillators, and portable x-ray machines [1]. Regarding battery health management in EVs for instance, there are several vital attributes that can benefit from such technologies, including remaining useful life (RUL, also known as battery cycle life) and cell temperature control used for thermal management [2,3]. This is because accurate RUL and temperature estimation can help develop strategies to prevent failures and thermal runaway [4]. Continuously monitoring the RUL can help develop better battery management strategies and inform maintenance schedules. This in turn can result in longer battery lifespan and reduced maintenance costs [5]. Because significant temperature variations can lead to failures [6,7], temperature monitoring is crucial. Continuous temperature monitoring can help detect significant temperature variations, which in turn can be used to develop operation modes that pro-actively prevent battery pack or cell failure [8,9]. For example, research showed that thermal runaways can be prevented by monitoring and analyzing thermal behavior patterns [10]. That is why, in this paper we focus our attention on prediction and monitoring of RUL and cell temperature as two of the most essential attributes.

2. Related work

The remaining useful life of a battery is the number of battery cycles from its rated capacity to the end of its life (EOL) [11]. The expression for RUL calculation is as follows:

$$RUL = N_{eol} - N \tag{1}$$

where N_{eol} denotes the maximum number of battery cycles, and N is the current number for battery cycles. It is considered that a lithium-ion battery reaches its EOL when the capacity drops to 60%–80% of its

* Corresponding author.
E-mail addresses: yuqin.weng@marquette.edu (Y. Weng), guan0210@morris.umn.edu (W. Guan), cristinel.ababei@marquette.edu (C. Ababei).

https://doi.org/10.1016/j.est.2025.119508
Received 24 March 2025; Received in revised form 5 November 2025; Accepted 16 November 2025
Available online 19 November 2025

rated capacity [12]. The existing literature on RUL prediction can be organized into three different types of approaches or categories: mathematical model based, data-driven, and hybrid approaches [13,14]. The prediction can be accurate for a model based approach when the model simulates the physical world behavior closely. However, the challenge is that detailed battery chemistry and physics knowledge are needed. Data-driven methods, employing AI/ML models, can easily capture the complex relationships and patterns inside data and they have become very popular recently. AI/ML models are flexible and they can be easily adapted to different battery types and different operation conditions with minimal re-training and optimization. However, model driven approaches suffer from a lack of interpretability and they rely on large amounts and high-quality datasets. Hybrid approaches combine the first two methods to benefit from the advantages offered by both, and they require insights into both physical modeling and data analytics.

Examples of model based RUL prediction are the studies in [13,15]. In [15], an algorithm is presented that combines an unscented Kalman filter (AUKF) with a genetic algorithm optimized support vector regression (GA-SVR) method to predict the battery RUL. A mamba-based state space model is proposed in [16] and shown to perform better in accuracy, robustness, and efficiency than previous approaches. In [17], the authors proposed a hybrid convolutional neural network (CNN), which is based on a fusion of three-dimensional CNN and two-dimensional CNN. In [18], the RUL is measured in two stages: before and after battery capacity degradation. A simple mechanism is developed to detect the unhealthy state of the battery, and an LSTM model is used to predict the RUL of the battery. The study in [19] uses an Elastic Net algorithm for battery remaining useful life predictions, combining Ridge Regression (RR) and LASSO regression. In [20,21], several ML algorithms, including random forest regression, decision tree, linear regression, and gradient boosting regression, are used to predict the battery RUL by using the battery life cycle data from the Hawaii National Energy Institute (HNEI). Their results are compared in the papers. Other deep learning and ensemble learning algorithms on data-dependent models for RUL predictions are listed in [22]. Examples of hybrid approaches for RUL prediction can be found in [11,14,23].

Overheating, short battery's remaining useful life, and other chemical reactions inside the batteries are reasons for which manufacturers use accurate thermal management and temperature prediction. Data-driven models for temperature prediction have been widely used due to their efficiency and accuracy [24]. Compared to traditional electrochemical modeling, in which the physical and chemical laws of the model need to be understood, data-driven models are favored in recent literature. Another reason a data-driven approach for temperature prediction and monitoring is useful is that it can eliminate the need to install temperature sensors in battery cells, which may be impractical or very costly [24]. The study in [25] shows that an improved RNN model can estimate the temperature of a Lithium-ion battery. In [26], it was shown that CNN models can help monitor battery pack temperature inside a battery management system better than other ML models. Additional data-driven previous studies that used supervised, unsupervised, and reinforcement learning for temperature prediction can be found in [27].

Existing literature reveals that there are several questions that are not fully answered when it comes to battery management systems: (1) It is not clear if multivariable time-series prediction with AI/ML models would offer better prediction performance than single variable prediction. There is still active research on multivariable prediction [28]. (2) It is not clear how applicable are tinyML technologies to such AI/ML models that are intended to be deployed on edge devices for real time inference that use resource constrained microcontroller units (MCU) or simple general purpose processors such as Raspberry Pi [29]. There are several studies that investigated optimized tinyML models for state of health (SoH) and state of charge (SoC) predictions in the area of battery management systems [30,31]. But, to the best knowledge of the authors, there are no previously reported studies that focused on tinyML models for prediction of RUL and temperature with verification on actual edge devices. Therefore, in this paper, we investigate a novel AI/ML model to predict both RUL and battery cell temperature. The proposed novel deep learning model combines a temporal convolutional network (TCN) with a multi-head attention layer that can output two or more variables simultaneously. We develop the proposed model using three publicly available datasets from battery cycling experiments conducted by Sandia National Lab (SNL) for two different types of lithium batteries. The model is tuned and trained on one dataset and trained and tested on the remaining two datasets with different cycling conditions. Our approach is inspired by insights from [28], where it was reported that deep learning structures like transformers, and attention mechanisms are advantageous for handling long-term dependencies in multivariable time-series data.

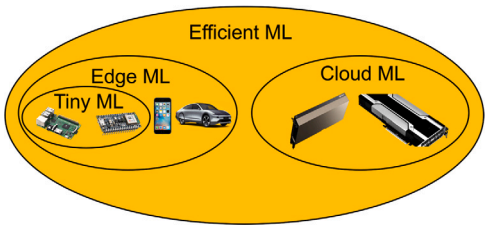

Fig. 1. Description of tinyML as edge ML under the efficient ML umbrella.

The remainder of this paper is structured as follows. Section 3 presents an overview of tinyML technologies. Section 5 describes the datasets used in this work and the testing workflow pipeline. The proposed deep learning model is introduced and explained in Section 4. Results and discussions are presented in Section 6. Finally, Section 7 concludes this work and highlights several directions for future work.

3. Overview of TinyML

3.1. TinyML and applications

TinyML technologies have emerged as a promising approach to achieve energy efficiency at performance levels offered by cloud ML. TinyML represents a new frontier in machine learning research [32]. It further pushes the efficiency boundary, enabling powerful ML models to run on resource constrained low-power devices [32]. As illustrated in Fig. 1, tinyML is one approach to facilitate edge ML and efficient ML. It combines machine learning with embedded or edge devices (e.g., Raspberry Pi, Arduino, and ESP32) for various applications including healthcare and automotive.

TinyML is becoming very popular, and it has been already applied in various application areas as summarized in Fig. 2. One such application areas is that of automotive applications. TinyML has mainly been used in autonomous driving and used for problems such as lane departure warnings, pedestrian detection, and traffic sign recognition. It can also be used in battery health monitoring systems. For example, the study in [33] developed tinyML optimized AI/ML models for individual prediction of RUL and cell temperature for lithium-ion batteries. In this work, we develop similar deep learning models to predict multiple attributes or variables of interest at the same time, in an effort to further reduce the memory capacity and energy consumption required to store and use such models for real time inference on edge devices.

3.2. Deployment on edge

A typical working flow in developing tinyML models and applications involves model development, training, optimization using tinyML, deployment to edge device, and real time inference for the application at hand. A popular choice for model development is Google's TensorFlow (TF) and TensorFlow Lite (TFL) for optimization [35]. Typical

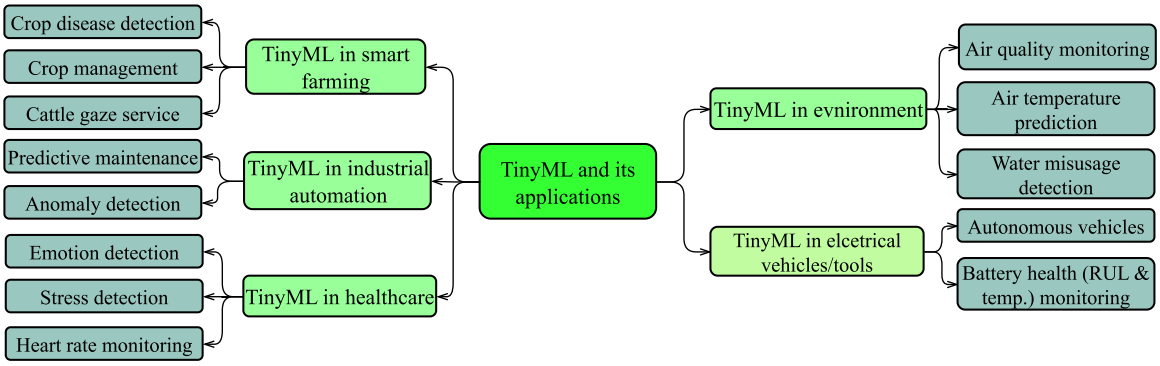

Fig. 2. Overview of tinyML applications in various areas, including smart farming, industrial automation, healthcare, environment, and EVs and electronic devices.

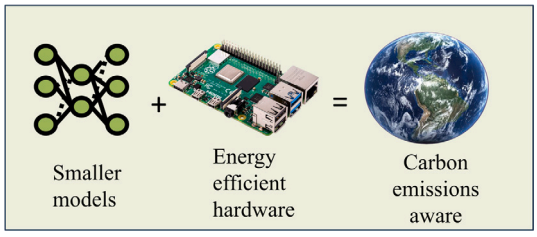

Fig. 3. TinyML technologies enable smaller ML/AI models on more energy-efficient hardware [34].

optimization techniques include model pruning [36,37], which reduces or eliminates unnecessary elements in the model and quantization [37], which simplifies the numerical precision of computations to reduce the model size and improve its speed. Another technique is knowledge distillation [37], which is a technique that creates a smaller learner model that mirrors the functionality of a larger teacher model. These optimization techniques allow for AI/ML models to be reduced significantly, without compromising model accuracy, to be able to easily integrate into regular C/C++ applications deployed onto edge devices.

In addition to efficient algorithms, computationally efficient hardware has also received lots of attention. For example, tensor processing units (TPUs) and field-programmable gate arrays (FPGAs) have been used in tinyML applications. TPUs are specifically tailored to accelerate ML/AI processes. These units significantly improve performance and energy efficiency [38]. Meanwhile, FPGAs can be reprogrammed for specific tasks, offering enhanced performance and reduced energy consumption [38]. By compressing models and reducing the memory space needed for the optimized models, tinyML is a prime example of technologies that aim to achieve carbon awareness (see Fig. 3) by combining smaller models and energy-efficient hardware [34].

4. Proposed deep learning AI/ML model

In this section, we present the proposed AI/ML model. We discuss its architecture and present several specific details on its main components. The simplified diagram that illustrates the structure of the proposed model is presented in Fig. 4. In our previous work [33], we showed that the TCN model offered better performance than other models discussed in [33]. In order to predict the two variables simultaneously, in this paper, we add the multi-head attention mechanism on top of the TCN model to boost prediction performance. This is the main motivation for using the proposed approach.

4.1. Proposed model and loss function

The hyperparameters of the model have been identified via numerous empirical investigations that were driven by the objective of improving model accuracy. That is, all convolution layers in the model — including the first regular convolution and the causal convolution layers inside the TCN block — use 64 kernels and 2 filters. Furthermore, there are four residual blocks inside the TCN block, the number of heads in the attention layer is set to 8, and the learning rate used is 0.001. The loss function used during model development is Huber loss [39]. The Huber loss function is a combination of MAE and RMSE, and thus it combines the properties of both MAE and RMSE [40]. The δ in Huber loss function is a hyperparameter defined by the user; the hyperparameter setting also affects the gradient descent, which significantly influences convergence speed [40]. Consequently, parameter tuning is of great importance for this loss function.

$$Huber = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2, |y_i - \hat{y}_i| \le \delta, \\ \delta|y_i - \hat{y}_i| - \frac{1}{2}\delta^2, |y_i - \hat{y}_i| > \delta, \end{cases} \tag{2}$$

4.2. Time-series modeling

Time-series or sequence modeling is the core of the proposed model, as its inputs are time-series data. Assume an input sequence $\{x_0, x_1, \dots, x_T\}$ is given. The goal is to predict associated outputs $\{y_0, y_1, \dots, y_T\}$. The key observation in time-series modeling is that one must predict the output y_t based only on those inputs that have been already observed, i.e., $\{x_0, x_1, \dots, x_t\}$ [41]. Thus, a time-series modeling network is any function $f : X^{T+1} \to Y^{T+1}$ that produces the mapping:

$$\{\hat{y}_0, \hat{y}_1, \dots, \hat{y}_T\} = f(x_0, x_1, \dots, x_T). \tag{3}$$

This function takes inputs as a sequence of past observations and produces predicted future values. If the function f is fed with inputs of $T+1$ observations, the function will return $T+1$ predicted values. The length of the input, $\{x_0, x_1, \dots, x_t\}$, is called sequence length. The output $\hat{y}_t$ can be a sequence of predictions, $\{\hat{y}_0, \hat{y}_1, \dots, \hat{y}_T\}$, as described by Eq. (3). On the other hand, the output can be a single time-step prediction with a single output, $\hat{y}_T$, or it can be a single time-step prediction with two or more outputs, for example $\hat{y}_T^{(1)}$ and $\hat{y}_T^{(2)}$. The objective of AI/ML model development when dealing with time-series modeling is to learn such function f that minimizes the loss between the actual and predicted outputs as shown in Eq. (4) [41].

$$\min_f L(y_0, y_1, \dots, y_T, f(x_{t+1}, x_{t+2}, \dots, x_T)) \tag{4}$$

where L is the loss function that f tries to minimize. In this work, the sequence length is set to 50, and 4 features are used. Therefore, the input shape for the proposed deep learning model is $(batch\_size, 50, 4)$. Since there are 64 filters for all convolution layers, the output shape for the first convolution layer is $(batch\_size, 50, 64)$.

4.3. Temporal convolutional networks

Convolutional neural networks (CNNs) have been widely used in sequence-to-sequence or time-series predictions. Particularly, 1D CNN

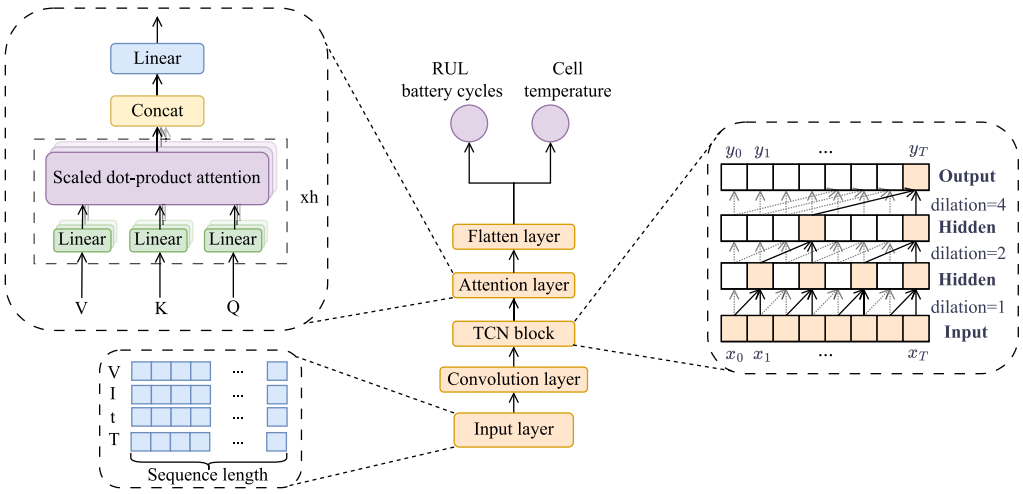

Fig. 4. Simplified diagram to illustrate the architecture of the proposed AI/ML model. The input includes voltage (V), current (I), test time (seconds), which represents the aging of the battery, and environmental temperature (°C). The two most important layers in this model are the temporal convolutional network (TCN) layer and the multi-head attention layer. The output layer has two neurons that provide prediction of RUL in cycles and of cell temperature (°C). Note that "xh" means duplication by "h" times.

Table 1
Wide application domains and examples of the TCN model.

TCN model applications	Examples
Time-series forecasting	PM2.5 prediction and appliances energy time-series forecasting [44]
Healthcare signal processing	Automated epilepsy detection from electroencephalogram (EEG) [45]
Human activity recognition	Action segmentation and detection [46]
Natural Language Processing (NLP)	Text sentiment classification [47]
Anomaly detection	Anomaly detection in cloud–edge collaboration system [48]

have been proven highly efficient in fields like speech recognition, neural language processing, anomaly detection, and air pollution prediction [31]. Compared to 2D CNN, 1D CNN has the advantage of a faster training process, so it needs fewer computational and power resources. On the other hand, a temporal convolutional network (TCN) is a special case of CNN, which further shortens the time of neural networks' training process. The TCN block used in the proposed model is a modified version of the networks originally introduced in [42] as WaveNet for generating raw audio waveforms. The networks are built using a specialized convolution technique, dilated causal convolution. Inside the networks, the convolution filter operates over an expanded receptive field by skipping input data with different steps on different layers. The TCN blocks are proven highly effective for long time-series sequence inputs [43]. Wide application domains and examples of the TCN model can be found in Table 1.

4.3.1. Dilated causal convolution

There are two operation principles for in TCNs: (1) Dimensions of output and the input of a TCN model must be the same, and (2) There is no future data leakage to the past [41], meaning causal convolution only takes the present and past inputs ($\{x_0, x_1, \ldots, x_t\}$). Thus, there is no future information ($\{x_{t+1}, \ldots\}$) involved with the TCN model at time t or earlier. The causal convolution is better than the standard convolution in time-series modeling because if the model uses future information in training, it may lead to the model overfitting during inference because the model is overly optimistic during the training process. These two principles are illustrated in the TCN block from Fig. 4, which displays the form of causal convolution with different dilation rates in each layer. The dilation rate helps expand the receptive field by skipping specific input data. To define a dilated convolution, for a one-dimensional input sequence $x \in X$ and filter $f : \{0, 1, \ldots, k-1\} \to \mathbb{R}$, the dilated convolution operation F of the input sequence s can be defined as [43]:

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i} \tag{5}$$

where the first part, $*_d$ denotes the dilated convolution. Moreover, $(x *_d f)(s)$ represents the dilated convolution for input sequence x at position s, d is the dilation rate, k is the filter size, $x_{s-d \cdot i}$ represents the input element that corresponds to the ith filter after applying the dilation rate. When $d = 1$, the dilated convolution is the same as a regular convolution [41].

4.3.2. Residual block

Deeper and larger TCN models are becoming increasingly crucial in prediction tasks. The residual connections or blocks, first introduced in [49], are helpful for deep neural networks because the residual connections can help the deep neural network avoid the vanishing or exploding gradient problem. The TCN model is usually connected to a residual block, where the residual block can help with deep convolution layers inside the TCN model. As shown in Fig. 5.a, a TCN block has a kernel size of 3, and dilation rates are set to 1. Besides the input layer, each convolution layer, including hidden and output layers, is a residual block. Fig. 5.b provides details of the residual block used in the proposed model. Inside each residual block there are two rounds of a combination of dilated causal convolution, LayerNorm, ReLu, and Dropout. Because each residual block contains two convolutional layers, a TCN block with a dilation rate sequence of length D consists of $2D$ convolutional layers. Thus, the total number of layers in the TCN block, including the input layer is $L_{total} = 2D + 1$. The dilation rate for the TCN block is set to 1 because integer quantization is incompatible with a dilation rate other than 1. Since the dimensions of the output and the input of a TCN model do not change, the output shape for the TCN block remains $(batch\_size, 50, 64)$.

4.4. Multi-head attention layer

A multi-head attention layer is used after the TCN block, as shown in Fig. 4. Attention mechanism has become very popular since its invention [50]. It has been used in larger language models like ChatGPT or

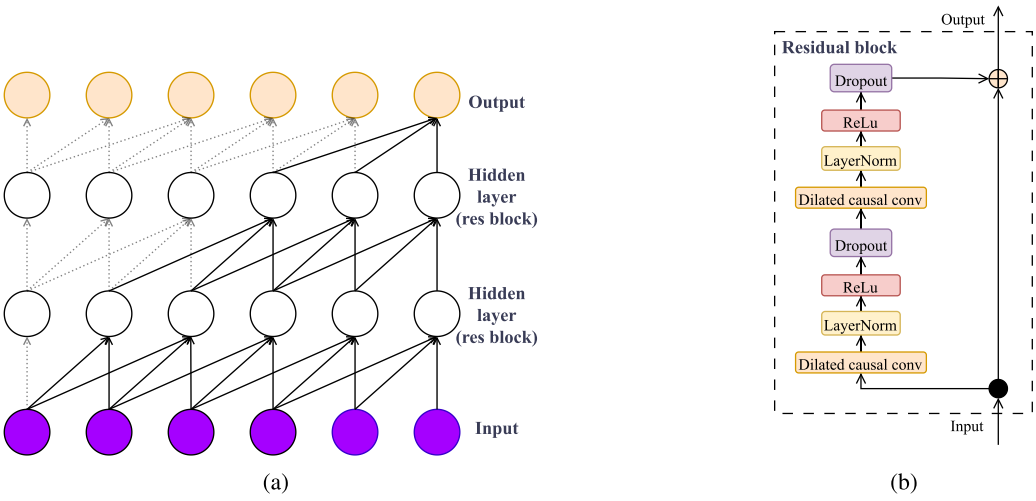

Fig. 5. A TCN block is always associated with a residual block. (a) The kernel size is 3, and dilation rates are set to 1 for the TCN block, (b) Inside each residual block, there are two rounds of dilated causal convolution combinations: LayerNorm, ReLu, and Dropout.

other time-series prediction studies. For example, multi-head attention-based anomaly detection in wireless sensor networks was presented in [51]. Attention-based models for traffic load forecasting were studied in [52]. Inside an attention layer, the input sequence $\{x_0, x_1, \ldots, x_T\}$ is first transformed by a linear projection into three different matrices, namely queries (Q), keys (k), and values (V). Then, a special operation (see Eq. (6) below), is used to compute the output of these three matrices.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6}$$

This unique operation is computed to obtain the weights of attention on the values (V) that consist of a dot-product of the query (Q) with all keys (K) and a division by $\sqrt{d_k}$, where $\sqrt{d_k}$ is the dimension of keys (K) [51]. The division is to prevent large values that could lead to small gradients. In addition, scaled dot-product attention can be computed multiple times in parallel, as shown in the attention layer from Fig. 4. This is called the total number of heads inside the attention layer. Note that the "xh" in Fig. 4 means the scaled dot-product attention is computed "h" times in parallel. Note also that the multi-head attention layer's input and output shapes do not change either. However, the output shape after the flatten layer becomes $(batch\_size, 3200)$. Finally, the output layer has 2 neurons ($\hat{y}_T^{(1)}$ and $\hat{y}_T^{(2)}$), representing RUL and cell temperature.

5. Model development workflow

5.1. Datasets description

The proposed deep learning model is trained and tested using three datasets, publicly available from Sandia National Lab [53]. We present here a brief description of the cycling experiments and collection of these datasets. Sandia National Lab conducted and investigated cycling tests for three lithium-ion commercial cells (different chemistry) with an 18 650 form factor [54]. These include lithium Nickel Manganese Cobalt (NMC), lithium nickel cobalt aluminum oxide (NCA), and lithium iron phosphate (LFP). NMC and NCA batteries are widely used in various applications, including EVs and electrical medical devices [55], because of several advantages, including high energy density, a good RUL, and less sensitivity to cold temperatures. The NMC batteries have the highest energy density among all three batteries [56]. One advantage of higher energy density batteries is that devices can operate for longer times without increasing battery size and weight compared to low energy density batteries. Therefore, the overall size and weight of the higher energy-density batteries are smaller and lighter, which makes EVs or devices more cost-friendly. Although LFP has a lower cost per weight unit and a longer RUL, its energy density is much lower than that of NMC and NCA, which makes it heavier than NMC and NCA batteries. Consequently, the overall cost of LFP batteries is higher than that of NMC and NCA due to less energy storage per weight and volume [57]. This is also why LFP is less popular than NMC and NCA in the American and Western rechargeable battery markets [58]. Consequently, in this work, we only focus on NMC and NCA batteries.

In Sandia's cycling test experiments, different environmental temperature ranges, depth of discharge (DoD), and discharging current (with the same charging rate as 0.5C) are used [59]. The majority DoD of all cycling experiments for the State of Charge (SoC) is 0%–100%, and the rest is 40%–60% and 20%–80%. A battery RUL is defined as an entire charging and discharging process regardless of the difference in DoD. The advantages of using cycle numbers to represent battery remaining useful life include: (1) The cycle index is a more measurable and interpretable way to compare battery replacement and maintenance, which is often used in battery management systems (BMS). (2) The cycle index can be estimated without any direct capacity measurements in real-time, reducing extra capacity calculations and capacity fade estimation, making it easier for real-time applications. (3) The applications for secondary battery usage which do not prioritize battery health as a priority can also be benefited from the cycling life estimation. Furthermore, the capacity of the NMC and NCA battery dataset used in this work (available at [53]), is lower than the nominal capacity due to degradation before the recorded experiment or prior battery usage. However, the cycle index, as the indicator of the RUL, is preserved by observing the charging and discharging process. In addition, experiments are carried out to estimate the battery's entire cycle numbers instead of stopping at EOF, which gives information about RUL before EOF and RUL for secondary usage. The purpose of the AI/ML deep learning model built in this work is to accurately predict both RUL and the cell temperature. The manufacturer-specified operating bounds for NMC and NCA batteries are reported in Table 2.

5.2. Workflow pipeline

The workflow pipeline that describes the main stages of model development, testing, and deployment in hardware is described in Fig. 6. As shown in the first quadrant of this diagram, the dataset

Table 2
NMC and NCA lithium-ion battery manufacturer-specified operating bounds [60].

Battery	NMC	NCA
Nominal capacity (Ah)	3	3.2
Nominal voltage (V)	3.6	3.6
Voltage range (V)	2 to 4.2	2.5 to 4.2
Max discharge current (A)	20	6
Acceptable temperature (°C)	−5 to 50	0 to 45
Nominal mass (g)	47	48.5

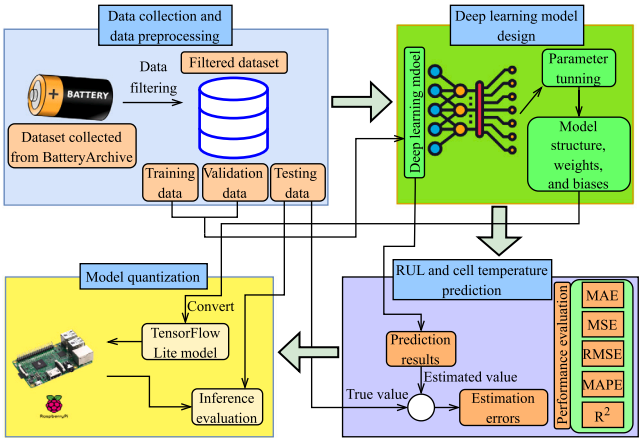

Fig. 6. Overall workflow for AI/ML model development.

is first filtered using a moving average (MA) filtering technique. This technique uses the following expression:

$$MA_k = \frac{p_1 + p_2 + \cdots + p_n}{k} = \frac{1}{k}\sum_{i=1}^{n} p_i \tag{7}$$

The moving average calculates the mean of the data point p_k within a specific time window of length k. We have tested different sizes of the time window for this research; the best size we found is 1500 s. The benefits of MA filtering technique for time-series data analysis include [61]: (1) Noise reduction: reduce noise and fluctuations from the original signal. (2) Signal smoothing: capture the trend of the signal effectively. (3) Simplicity: easy to implement with small computational resources, ideal for real-time applications and for deployment on edge devices. Note that the ambient temperature is also considered as one of the inputs to the model since it significantly affects the battery's internal resistor and degradation rate. To ensure high prediction accuracy, we also incorporate the ambient temperature measurement and apply the MA filtering technique to reduce the fluctuations and noise in the sensor-captured temperature values. Then, the dataset is split into training, validation, and testing portions. The train and validation dataset portions are used for model development, which is done with TensorFlow, as shown in the second quadrant of Fig. 6. Next, as shown in the third quadrant, several error metrics are used to evaluate the model on the test portion of the dataset: mean absolute error (MAE), mean square error (MSE), root mean square error (RMSE), mean absolute percentage error (MAPE), and R^2 score whose expressions are given in the following equations:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{8}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{9}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{10}$$

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

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{12}$$

To keep the development time to a minimum, the proposed AI/ML model is first tuned using the NMC1 battery dataset. Then, the tuned hyperparameters are used to train and test the model also on the other datasets (i.e., NMC2 and NCA).

In the last stage of our experiments, the model is optimized using tinyML technologies; that is, translated into the TensorFlow Lite format using three different types of quantization: 8-bit integer, dynamic range, and 16-bit floating quantization. The details of these three types of quantization will be described later. The final optimized quantized models will be deployed to a Raspberry Pi 4 processor for real time inference testing (as shown in the last quadrant of Fig. 6). Raspberry Pi 4 is used in this work because it is a low-power, single-board computer with a Broadcom BCM2711 processor based on ARM Cortex-A72 architecture, which suits well for tinyML tasks. Another reason Raspberry Pi 4 is used in this research is that while resources-constrained devices like the Arduino are well-suited for simple tasks, they lack the necessary computational resources to support the execution of the multi-variable deep learning model in this work. In contrast, Raspberry Pi 4 provides a more capable and energy-friendly embedded platform for this research. The Raspberry Pi 4 board used in our experiments has 2 GB of LPDDR4 SDRAM, a microSD, and a 2 × micro-HDMI port. It is a popular choice for power-efficient light-weight AI/ML projects due to its ease of use, good performance, and low cost.

6. Results and discussion

6.1. Comparison of the proposed model on different datasets

The proposed model was developed and tuned using the NMC1 battery dataset. There are four features inside the NMC1 battery dataset used in this work. They are current (A), voltage (V), environment temperature (°C), and test time (S). The current is a square wave cycle with a maximum of 1.5 A and a minimum of −3 A, and the voltage ranges from 2 V to 4 V; the test time for one data instance ranges from 10 s to 600 s. These four features are presented in Fig. 7, where the y-axis is the current, voltage, environment temperature, and test time with their corresponding units. The x-axis is the time step index from the first to the last data point recorded in the cycling experiment. The NMC1 battery dataset was tested with a 1C discharging rate, a DoD of 0% to 100%, and an environmental temperature of 15 °C. It has 513 cycles, each varying from 100 to 1000 time steps, and the cell temperature range is from 15.368 °C to 26.291 °C (total range of 10.923). As described in Section 5.2, the MA filtering technique is applied every 1500 s. The NMC dataset has 48,118 data points (time steps) and is divided into training 36,149 data points (75%), validation 8397 data points (17.5%), and testing 3572 data points (7.5%). The training set is used to train the model, and it comprises typically 70%–75% of the whole dataset. The validation set monitors the model's performance during the training process. The testing set, on the other hand, which tests the model on completely unseen datapoints during both the training and validation process, is used to evaluate the model's generalization ability. Note that this split into training, validation and test is common in the literature.

The prediction results obtained with the proposed model before optimization with tinyML technologies (i.e., at TensorFlow level) and using the NMC1 battery dataset are shown in Fig. 8. In these plots, the x-axis represents the time step index, and the y-axis shows RUL in cycles or cell temperature. The "x" markers indicate the true values, and "Δ" markers indicate the predicted values. In addition to the prediction results with both true and predicted values, we plot the true values solely of both RUL and cell temperature (Fig. 8.a and .b) as well for a

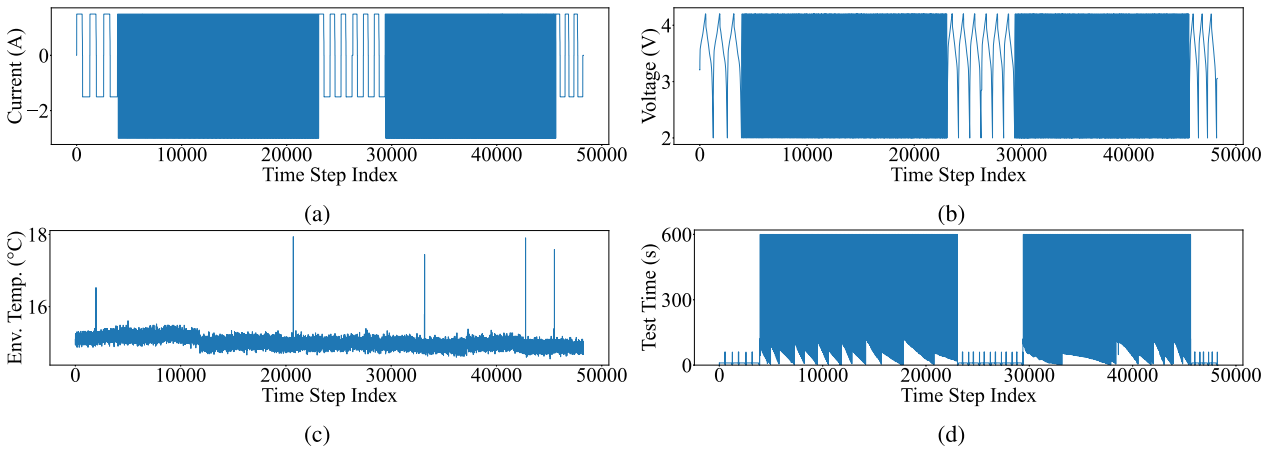

Fig. 7. A total of four features for the NMC1 battery dataset are used in this work; they are (a) current (A), (b) voltage (V), (c) environment temperature (°C), and (d) test time (s).

Table 3
RUL errors for NMC1, NMC2 and NCA battery datasets.

Datasets	MAE	MSE	RMSE	MAPE	R2
NMC1	0.8321	1.0953	1.0466	5.58%	0.9999
NMC2	1.1091	2.2237	1.4912	1.69%	0.9999
NCA	0.8628	1.334	1.155	3.13%	0.9999

Table 4
Cell temperature errors for NMC1, NMC2 and NCA battery datasets.

Datasets	MAE	MSE	RMSE	MAPE	R^2
NMC1	0.3935	0.4485	0.6697	2.2%	0.8539
NMC2	0.1601	0.0432	0.2078	0.63%	0.7509
NCA	0.6295	1.248	1.1171	1.6%	0.8974

clearer and better comparison. We also plot the histograms of absolute errors ($|y_t - \hat{y}_t|$) and absolute percentage errors ($|y_t - \hat{y}_t|/y_t$) because they offer a better picture of how prediction errors are distributed. The error matrices for the NMC1, NMC2, and NCA datasets on TensorFlow level are reported in Tables 3 and 4.

In reporting the performance of the proposed model on the other two datasets, we first note that the absolute numerical ranges of RUL can vary greatly, even for the same type of battery. That is because the discharge rate, DoD, and environmental temperature can all affect the battery's health. To be more specific, the NMC2 battery cell was tested with a 0.5C discharge rate, a DoD of 20% to 80%, and an environment temperature of 25 °C. In this case, the RUL range is from 1 to 1431 and the cell temperature range is from 24.093 °C to 28.065 °C (total range of 3.972). Similarly, the NCA battery cell was tested with a 2C discharge rate, a DoD of 0% to 100%, and an environment temperature of 35 °C. Hence, the RUL range is from 1 to 654 and the cell temperature range is from 33.167 °C to 55.361 °C (total range of 22.194). The reason these three datasets are selected is that they have different discharging rates, DoD, and environmental temperature when they are tested. Despite these differences in datasets, the proposed model appears to be very robust, providing the same level of performance when tested on all three battery datasets. This also proves the model has good scalability for different data distributions.

Because however RUL and cell temperature ranges are different for all three datasets, it is not ideal to compare them directly using MAE, MSE, and RMSE. We still report them here for reference, but, also use MAPE and R^2 scores to compare the investigated datasets. MAE, MSE, and RMSE are still valuable for comparing the degradation in prediction accuracy of different quantization methods, for a given dataset. We observe that RUL, the MAPE varies from 1.69% to 5.58%, and all R^2 values are 0.9999. This indicates the proposed model works well for predicting RUL with only minor errors. The MAPE varies for cell temperature from 0.63% to 2.2% and the R^2 score from 0.7509 to 0.8974. The NMC2 battery dataset has the lowest MAPE but also the lowest R^2 score because MAPE can be favored by a small range of values, like cell temperature in the NMC2 battery dataset. However, the lowest R^2 score means the model cannot explain the overall variance of the data in the NMC2 dataset very well.

The R^2 score plots of both RUL and cell temperature on all three datasets are plotted in Figs. 9 and 10. The dots in these two figures are scatter plots of true and predicted values of battery cycles and cell temperature, while the dashed line represents the ideal model with zero error. The higher R^2 scores indicate better predictive performance of the proposed model. Just as discussed beforehand, for RUL, the predicted values follow well on the true values. On the other hand, the predicted values for cell temperature are slightly off compared to the performance of RUL predictions. The potential reason for this could be that cell temperature is harder to predict than RUL. As seen in Fig. 8.a and .b, the fluctuation of the cell temperature is more significant than that of RUL. In addition, one can see in Fig. 7.c that there are also spikes in the environmental temperature. We suspect that these spikes may be most likely due to erroneous readings during data collection by SNL lab. They are however more difficult to deal with and the MA filtering technique does not address these spikes entirely. As a result, cell temperature prediction is more challenging and this results in the R^2 score being lower than that in the case of RUL prediction. In summary, our proposed model works very well on these three datasets, with particularly excellent prediction accuracy for RUL.

6.2. Comparison of the proposed model to previous work

In this section, we compare our model to recently reported previous works for the predictions of RUL and cell temperature. Results of this comparison are summarized in Table 5. While the best comparison is by MAPE, some previous studies reported only MSE and RMSE. Therefore, the comparison table also provides a normalized MSE or RMSE; the normalization uses MSE or RMSE divided by the range of the true test values. In this way, the comparison to previous work is as fair as possible. In Table 5, row 2 lists the errors for our proposed model, rows 3–9 list RUL and cell temperature errors found in existing literature. When comparing the errors for RUL, the MAPE of the proposed model is lower than [17,18], and the normalized RMSE of our proposed model is also lower than [19]. For cell temperature predictions, the cell temperature estimation in [25] at 5° is close to our proposed model when using normalized MSE. In addition, our proposed model is better than the temperature estimations under two other conditions in [25].

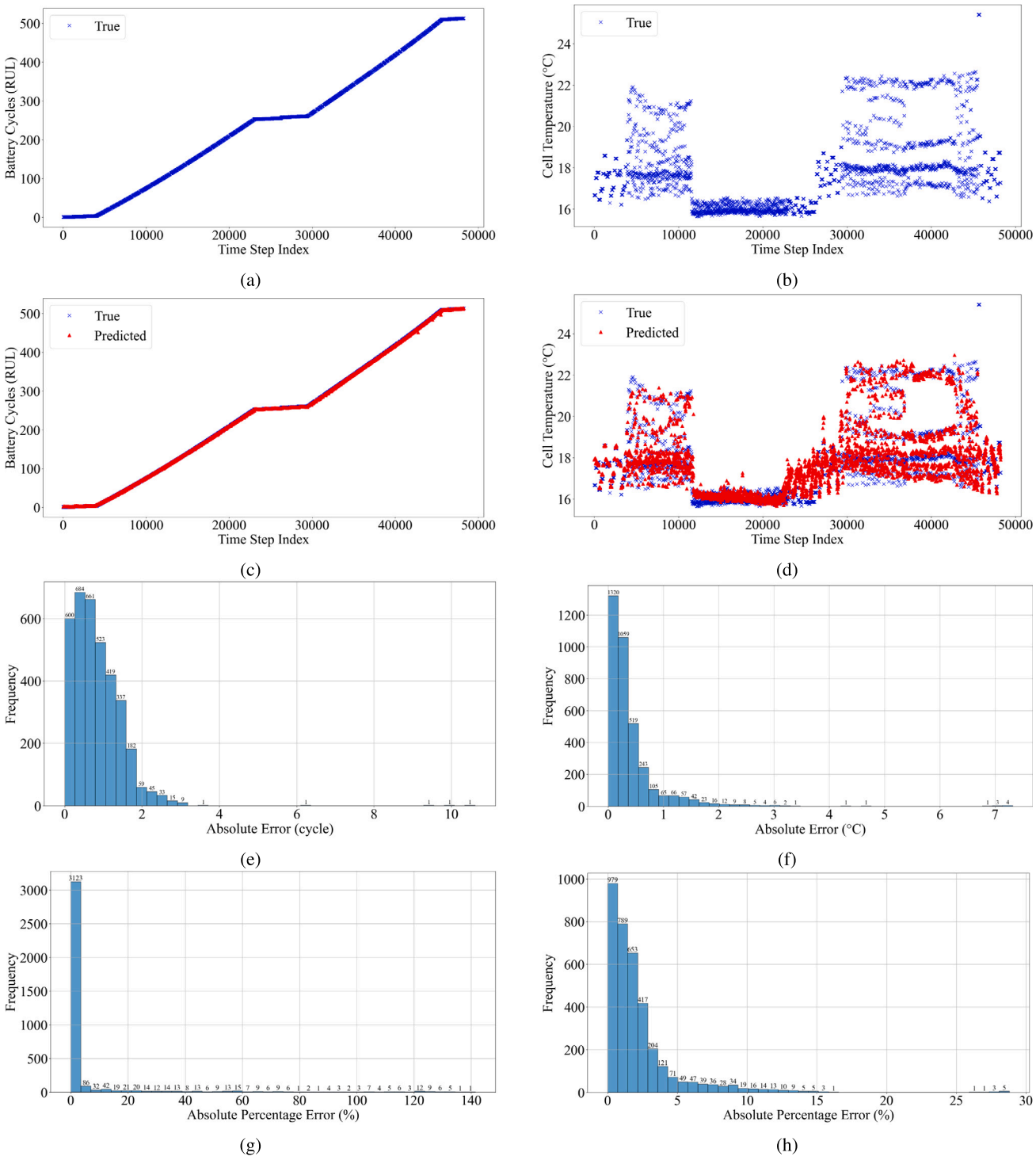

Fig. 8. TensorFlow level model performance evaluation on the test portion of the NMC1 dataset (a) actual values of RUL, (b) actual values of cell temperature, (c) actual and predicted values of RUL, (d) actual and predicted values of cell temperature, (e) absolute error histogram for RUL, (f) absolute error histogram for cell temperature, (g) absolute percentage error histogram for RUL, (h) absolute percentage error histogram for cell temperature.

Finally, we outrun the performance of the model used in [26] with respect to MAPE.

We also present a qualitative comparison of the proposed model to previous studies that focused on multi-task learning for predictions of other attributes, including SoC and state of energy (SoE), capacity and resistance, and other tinyML applications in battery SoC and SoH estimations. This comparison is presented in Table 5, rows 10–13. Our work adds to this body of prior studies, which did not address the problem that we focus on in this paper — that of simultaneously predicting RUL and cell temperature. In [62], the authors investigated forecasting battery capacity and power degradation with multi-task learning. In [63], the authors developed a multi-task learning structure for SoC and SoE. Note that these work did not provide any tinyML implementation. In [30,31], a tinyML solution was presented for SoH and SoC. Here, we only present model performance as reported at TensorFlow level (i.e., models executed on personal computers or servers) and not as reported when models are executed on edge devices. We observe that among all battery attributes that previous studies and our

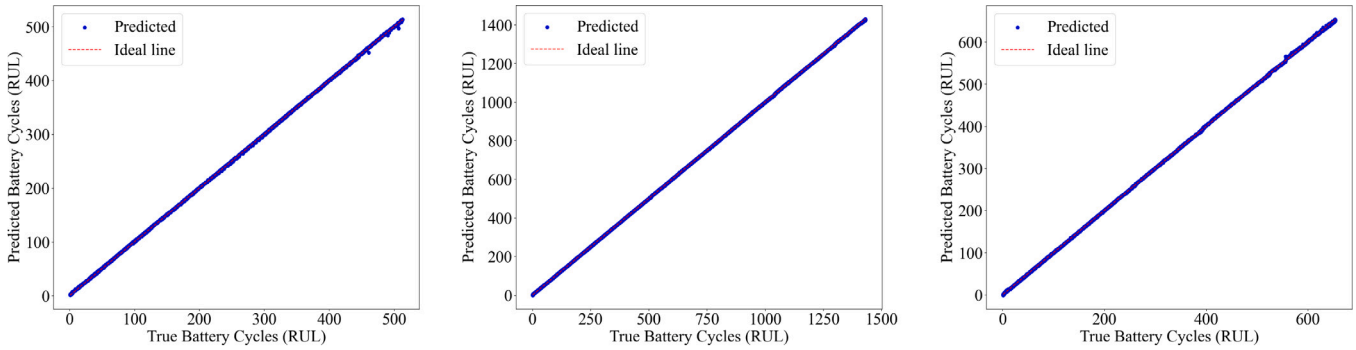

Fig. 9. Comparison of R^2 values for (a) NMC1 (b) NMC2 (c) NCA battery datasets.

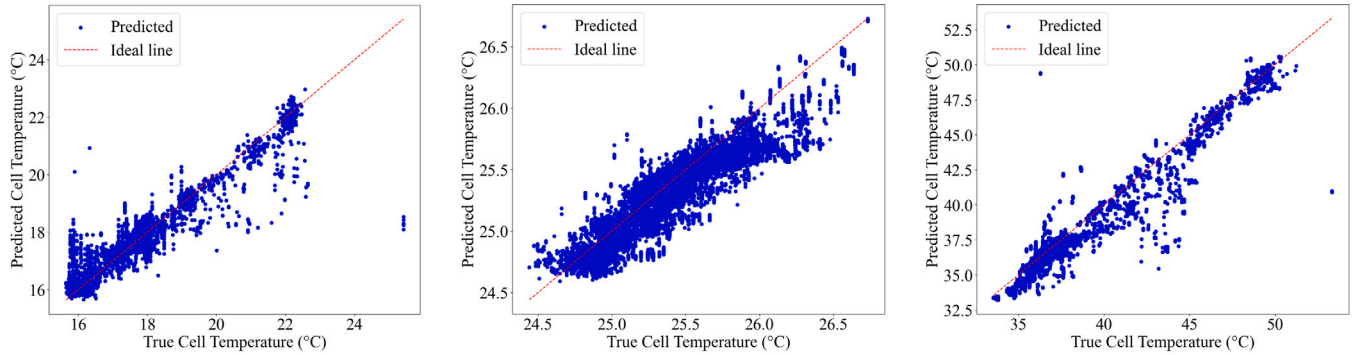

Fig. 10. Comparison of R^2 values for (a) NMC1 (b) NMC2 (c) NCA battery datasets.

Table 5
RUL and cell temperature errors (row 2) compared to existing literature on RUL and cell temperature predictions (rows 3–9), multi-tasking learning (rows 10–11), and other tinyML studies (rows 12–13).

Models	MAPE	Normalized RMSE	Normalized MSE	Multi-variables	TinyML technology
Proposed model (best RUL, cell temp.)	1.69%, 0.63%	0.1%, 5.2%	0.16%, 1.09%	Yes	Yes
Ref. [17] (RUL)	3.6%	NA	NA	No	No
Ref. [18] (RUL)	8.82%	NA	NA	No	No
Ref. [19] (RUL)	NA	4.13%	NA	No	No
Ref. [25] at 5° (cell temp.)	NA	NA	2.8%	No	No
Ref. [25] at 25° (cell temp.)	NA	NA	13.43%	No	No
Ref. [25] at 45° (cell temp.)	NA	NA	6.15%	No	No
Ref. [26] (cell temp.)	around 5%	NA	NA	No	No
Ref. [62] (Multi-tasking, capacity, resistance)	2.37%, 1.24%	NA	NA	Yes	No
Ref. [63] (Multi-tasking, SoC, SoE)	NA	0.7709%, 0.8671%	NA	Yes	No
Ref. [30] (TinyML for SoH)	NA	4.86%	NA	No	Yes
Ref. [31] (TinyML for SoC)	NA	2.6915%	0.072%	No	Yes

work focused on, RUL is predicted with the highest accuracy while temperature is predicted with lower accuracy. As discussed earlier, we note that cell temperature is a more challenging attribute to work with and to predict.

6.3. Lite is all you need

As discussed earlier, there is no existing previous work that focused on optimizing AI/ML deep learning model to predict two or more attributes and then deploy to test on edge devices in the area of battery health monitoring. We are the first to conduct such a study and report here the use of three different quantization methods: 8-bit integer, dynamic range, and 16-bit floating quantization. The model structure, the weights, and biases inside the model represent essentially the AI/ML model; they are stored on disk/memory inside the file that represents the model. The size of the model is therefore related to the total number of weights and associated activations inside the model; hence, the more weights, the larger the model. Converting 32-bit floating point numbers that are used to represent the weights to 8-bit integer numbers can have a significant impact on reducing the model size. The algorithm for 8-bit quantization to generate quantized tensors is called the "min–max" algorithm [30] because the 8-bit integer quantization is a projection of the 32-bit floating number using their min and max values. The equation for 8-bit integer quantization and dequantization is as follows:

$$q = round(\frac{r}{scale} + z)$$
$$r = (q - z) * scale \quad (13)$$

The r, q, and z represent real value, quantized value, and zero point, respectively. The $scale$ is calculated using the min–max difference between the real values and then divided by the min and max difference between the quantized values, which is always 255 for 8-bit integer

Table 6
RUL prediction accuracy for NMC1 dataset — model at TensorFlow level and after optimization with three quantization methods.

	MAE	MSE	RMSE	MAPE	R^2
TensorFlow	0.8321	1.0953	1.0466	5.58%	0.9999
8-bit integer	3.8857	25.4559	5.0454	21.12%	0.999
Degradation	3.0536	24.3606	3.9988	15.54%	0.0001
Dynamic range	0.8271	1.1113	1.0542	5.03%	0.9999
Degradation	0	0.016	0.0076	0	0
16-bit floating	0.8279	1.0873	1.0427	5.62%	0.9999
Degradation	0	0	0	0.04%	0

Table 7
Cell temperature prediction accuracy for NMC1 dataset — model at TensorFlow level and after optimization with three quantization methods.

	MAE	MSE	RMSE	MAPE	R^2
TensorFlow	0.3935	0.4485	0.6697	2.2%	0.8539
8-bit integer	0.6211	0.8383	0.9156	3.43%	0.7269
Degradation	0.2276	0.3898	0.2459	1.23%	0.127
Dynamic range	0.3940	0.4448	0.6669	2.18%	0.8551
Degradation	0	0	0	0	0
16-bit floating	0.3935	0.4479	0.6693	2.2%	0.8541
Degradation	0	0	0	0	0

Table 8
Comparison of models' size and inference time.

	Model size	Reduced	Inference time
TensorFlow	2.27 MB	NA	0.52 ms
8-bit integer	176 KB	13x	5.03 ms
Dynamic range	164 KB	14x	3.03 ms
16-bit floating	246 KB	9x	4.07 ms

Table 9
Comparison of RUL and cell temperature prediction accuracy for NMC1 dataset at TensorFlow Lite level between this and previous work.

	MAE (best at TensorFlow Lite level)
RUL (TCN+attention)	0.8271
RUL (TCN)	1.88
Cell temp. (TCN+attention)	0.3935
Cell temp. (TCN)	0.25

quantization. On the other hand, the zero point value can be symmetric or asymmetric depending on whether z equals zero. If z is zero, this simplifies both processes for quantization and dequantization. In this work, since post-training quantization from TensorFlow is adopted for all three quantization methods, the weights are quantized in symmetric format and signed integer type, and activations are quantized in asymmetric format and signed integer type due to their asymmetric nature [30]. This quantization scheme is called Symmetric (for weights) and Asymmetric (for activations).

The dynamic range quantization also does the 8-bit quantization for the weights; however, it does the fly quantization and dequantization for activations. Only learnable parameters, like weights, are permanently stored in the model, and activations are not stored as weights are. The activations can be quantized dynamically at inference if the deployed device has enough computational power [64]. Thus, dynamic range quantization can preserve more accuracy of the model than 8-bit integer quantization and does not necessarily increase the inference time. As for 16-bit floating quantization, which is called half-precision quantization, it adopts an IEEE 754 floating-point format. The advantage of the 16-bit floating point number over the 32-bit floating point number is that it only requires half the storage size of a 32-bit floating point number. Even so, the quantized 16-bit floating model is usually larger than the two before, but it may have the least degradation among all three quantized models.

The proposed model was successfully quantized using the three different methods, then deployed and tested on the Raspberry Pi 4 edge device, as shown in the experimental set-up from Fig. 11. Degradations of each quantized model compared to the original TensorFlow level model are reported in Tables 6 and 7. The degradation is calculated as the absolute difference between each term in the table for all three quantization methods and the original TensorFlow model, which is a reference for analyzing the performance of the quantization models. If there is no degradation or the error term for the quantized model is slightly better than the original model, the degradation is denoted as 0 in these tables. Some results of the quantized model are slightly better than those of the original model because quantization can act as regularization or smoothing for the data, helping to reduce the errors slightly. In addition, the model size and inference time are also listed and compared in Table 8. We observe that the model size is reduced by up to 14x. The inference time is calculated for one datapoint inference and reported as the average of five values of the same experiments. The average inference time for predicting RUL and cell temperature individually on an embedded device in our previous work [33] was 5 ms. However, the best average inference time in this updated work is 3 ms, which surpasses our previous work. More importantly, the proposed multi-variable prediction model demonstrates faster inference time than the single-variable model, indicating a clear advantage in computational resource efficiency. The comparison of the accuracy of the proposed model (TCN and attention layer) and the previous work model (TCN) is also concluded in Table 9. Since our previous work only reported the MAE at the TensorFlow Lite level, we will compare RUL and cell temperature by MAE. The proposed model's performance on RUL is better, while the cell temperature is slightly worse than our previous work. The reason why the cell temperature is harder to capture can be referred to in the last paragraph of Section 6.1. Note that the proposed model in this work simultaneously predicts two variables, as stated before. The learning complexity increases compared to single-output models. A slight drop in one output's accuracy is within reasonable expectations for multi-task learning, and the proposed model still has two advantages compared to our previous work: (1) higher RUL accuracy, and (2) faster inference time.

Besides accuracy, model size and inference time, the power consumption and runtime memory of the deployment are presented in Table 10. The power consumption is measured with the help of a

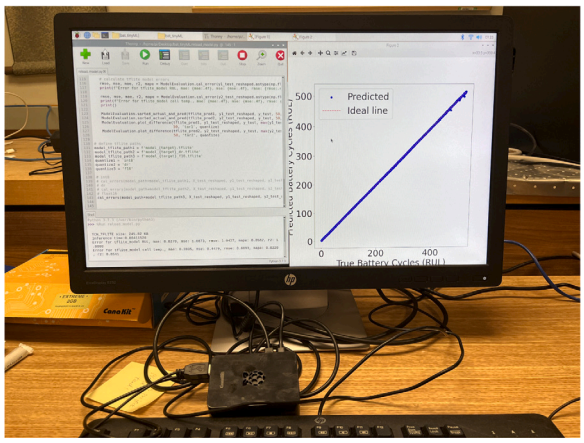

Fig. 11. Experimental set-up for deployment on Raspberry Pi 4.

Table 10
Power consumption and runtime memory usage of the proposed models deployed on the edge device.

Model optim.	Power consumption	Runtime memory
8-bit integer	0.43 W	100 MB
Dynamic range	0.64 W	103 MB
16-bit floating	0.95 W	104 MB

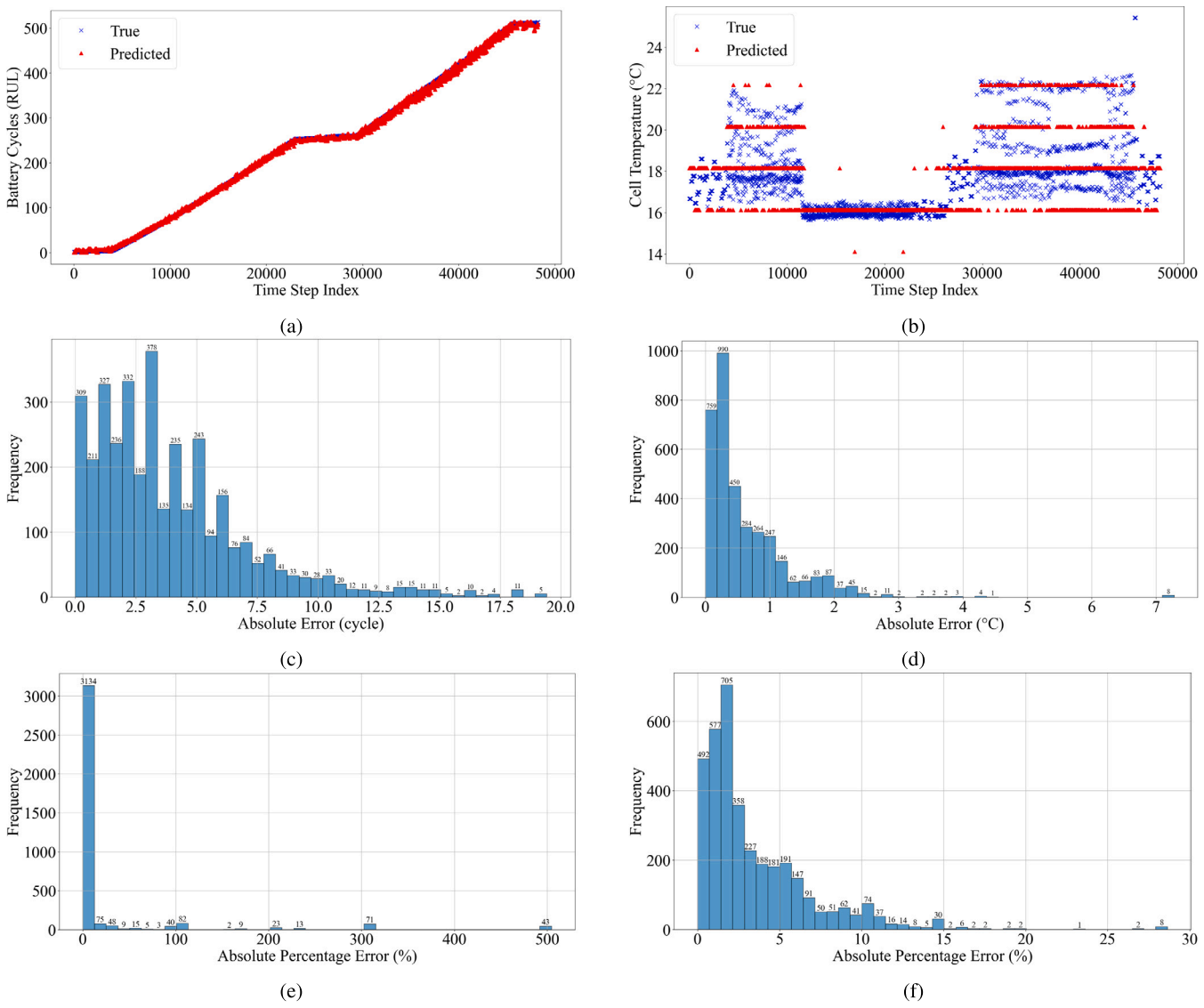

Fig. 12. Performance evaluation of the model optimized with 8-bit integer quantization on the test portion of the NMC1 dataset (a) actual and predicted values of RUL, (b) actual and predicted values of cell temperature, (c) absolute error histogram for RUL, (d) absolute error histogram for cell temperature, (e) absolute percentage error histogram for RUL, (f) absolute percentage error histogram for cell temperature.

USB power tester, while the runtime memory usage is measured using the BpyTOP that is readily available on Linux systems such as that running on the Raspberry Pi. As it can be seen, all three models have small power consumption when used for real-time inferencing on the Raspberry Pi 4 edge device. In terms of the amount of runtime memory usage, we observe that all models use around 100MB, which is only a very small portion of the total memory available on the device of 2 GB; in other words, the proposed models use only 5% of the total memory available. Compared to other previous studies that also used Raspberry Pi for testing [65,66] we note that our models use less memory. For example, the study in [65], which studied models for gesture classification and keyword spotting use up to 11% memory of a Raspberry Pi 4 with a total of 4 GB, while the natural language processing models from [66] use 300–700 MB of memory.

The prediction results of the model optimized with 16-bit floating quantization are not shown because there is nearly no degradation from the original TensorFlow model and to save space. The prediction results of the model optimized with the other two quantization methods are presented in Figs. 12 and 13. We observe that the 8-bit integer quantization leads to larger prediction accuracy degradation, which is undesirable. Therefore, this version of the model is not useful. On the other hand, the model optimized with dynamic range quantization is affected minimally, its degradation is acceptable, and the model size is also very small. In addition, its inference time is the shortest. We conclude that its deployment in real time applications is possible and sufficient (i.e., lite is all you need).

7. Conclusion and future work

In this paper, we proposed a novel AI/ML model to predict simultaneously multiple attributes for lithium-ion batteries. In contrast with approaches that develop separate models for individual attributes or variables, the proposed approach results in models that require less computational capabilities when deployed on resource constrained edge devices. The proposed model focused on the prediction of battery cell temperature and remaining useful life, which are essential attributes used in battery management systems. The model is developed using several public datasets from Sandia National Lab and verified for prediction accuracy using extensive simulations. Furthermore, the model is optimized using tinyML technologies and then deployed on a Raspberry Pi 4 edge device for real time inference, with satisfactory

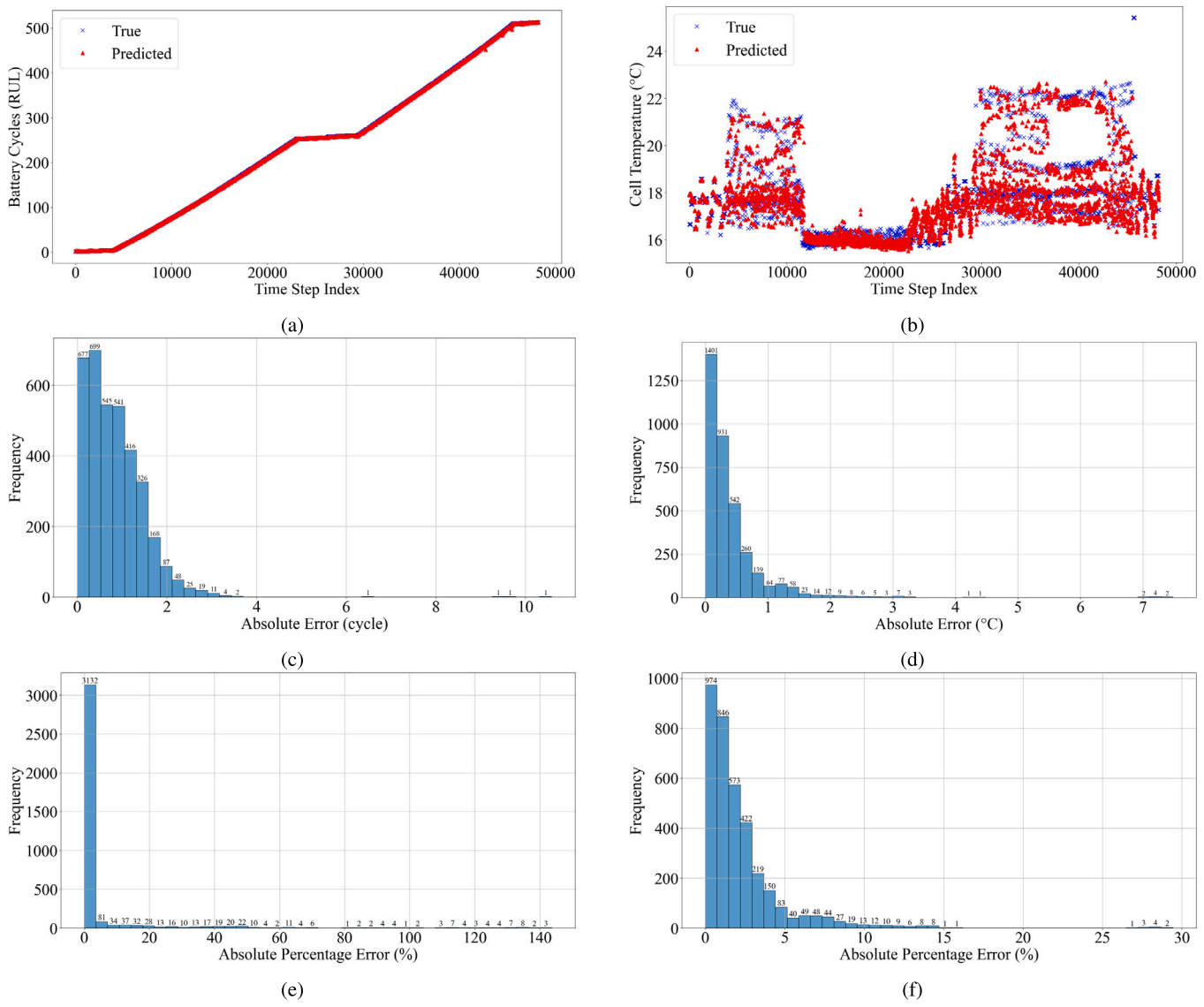

Fig. 13. Performance evaluation of the model optimized with dynamic range quantization on the test portion of the NMC1 dataset (a) actual and predicted values of RUL, (b) actual and predicted values of cell temperature, (c) absolute error histogram for RUL, (d) absolute error histogram for cell temperature, (e) absolute percentage error histogram for RUL, (f) absolute percentage error histogram for cell temperature.

prediction accuracy. Therefore, this work is a step forward towards developing multi-output AI/ML models aimed at deployment on resource constrained edge devices for real time predictions. Feature engineering and analysis for better model generalization ability on different batteries and model size reduction, such as pruning unnecessary layers in the TCN model for less memory consumption, will be the direction of our future work for continuing the tinyML research. The proposed model's generalization capability can be further strengthened through cross-dataset validation, which can also be conducted and explored in future work.

CRediT authorship contribution statement

Yuqin Weng: Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Wenkai Guan:** Software, Methodology, Formal analysis, Conceptualization. **Cristinel Ababei:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Investigation, Conceptualization.

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.

References

[1] N. Arandia, J.I. Garate, J. Mabe, Embedded sensor systems in medical devices: Requisites and challenges ahead, Sensors 22 (9917) (2022).
[2] Y.A. Sultan, A.A. Eladl, M.A. Hassan, S.A. Gamel, Enhancing electric vehicle battery lifespan: Integrating active balancing and machine learning for precise RUL estimation, Sci. Rep. 15 (777) (2025).
[3] A.K.M.A. Habib, M.K. Hasan, G.F. Issa, D. Singh, S. Islam, T.M. Ghazal, Lithium-ion battery management system for electric vehicles: Constraints, challenges, and recommendations, Batteries 9 (2023).

[4] M.O. Tarar, I.H. Naqvi, Z. Khalid, M. Pecht, Accurate prediction of remaining useful life for lithium-ion battery using deep neural networks with memory features, Front. Energy Res. 11 (2023).

[5] Electra, How remaining useful life (RUL) analytics drives sustainability and battery reuse (2024), https://www.electravehicles.com/remaining-useful-life-battery-analytics-sustainable-future/.

[6] H. Fayaz, A. Afzal, A.D.M. Samee, M.E.M. Soudagar, N. Akram, M.A. Mujtaba, R.D. Jilte, M.T. Islam, U. Agbulut, C.A. Saleel, Optimization of thermal and structural design in lithium-ion batteries to obtain energy efficient battery thermal management system (BTMS): A critical review, Arch. Comput. Methods Eng. 29 (2022) 129–194.

[7] Y. Yu, T. Vincent, J. Sansom, D. Greenwood, J. Marco, Distributed internal thermal monitoring of lithium-ion batteries with fibre sensors, J. Energy Storage 50 (2022).

[8] Stellarix, EV battery thermal management system and its importance (2024), https://stellarix.com/insights/articles/ev-battery-thermal-management-system-and-its-importance/.

[9] Y. Ortiz, P. Arevalo, D. Pena, F. Jurado, Recent advances in thermal management strategies for lithium-ion batteries: A comprehensive review, Batteries 10 (83) (2024).

[10] R.D. McKerracher, J. Guzman-Guemez, R.G.A. Wills, S.M. Sharkh, D. Kramer, Advances in prevention of thermal runaway in lithium-ion batteries, Adv. Energy Sustain. Res. 2 (2021).

[11] X. Tang, H. Wan, W. Wang, M. Gu, L. Wang, L. Gan, Lithium-ion battery remaining useful life prediction based on hybrid model, Sustainability 15 (6261) (2023).

[12] X. Meng, W. Wang, X. Cao, G. Li, De Li, A RUL prediction method of lithium-ion battery based on extreme learning machine, J. Phys.: Conf. Ser. (2023).

[13] Y. Cai, W. Li, T. Zahid, C. Zheng, Q. Zhang, K. Xu, Early prediction of remaining useful life for lithium-ion batteries based on CEEMDAN-transformer-DNN hybrid model, Heliyon 9 (2023).

[14] M. Alipour, S.S. Tavallaey, A.M. Andersson, D. Brandell, Improved battery cycle life prediction using a hybrid data-driven model incorporating linear support vector regression and Gaussian, ChemPhysChem 23 (2022).

[15] Z. Xue, Y. Zhang, C. Cheng, G. Ma, Remaining useful life prediction of lithium-ion batteries with adaptive unscented Kalman filter and optimized support vector regression, Neurocomputing 376 (2020).

[16] Y. Liang, S. Zhao, Early prediction of remaining useful life for lithium-ion batteries with the state space model, Energies 17 (6326) (2024).

[17] Y. Yang, A machine-learning prediction method of lithium-ion battery life based on charge process for different applications, Appl. Energy 292 (2021).

[18] D.A. Mittal, H. Bello, B. Zhou, M.S. Jha, S. Suh, P. Lukowicz, Two-stage early prediction framework of remaining useful life for lithium-ion batteries, in: Conference of the IEEE Industrial Electronics Society, 2023.

[19] J. Schaeffer1, G. Galuppini, J. Rhyu, P.A. Asinger, R. Droop, R. Findeisen, R.D. Braatz, Cycle life prediction for lithium-ion batteries: machine learning and more, in: American Control Conference, ACC, 2024, pp. 763–768.

[20] J.N.C. Sekhar, B. Domathoti, E.D.R.S. Gonzalez, Prediction of battery remaining useful life using machine learning algorithms, Sustainability 15 (2023).

[21] C.L. Sravanthi, J.N.C. Sekhar, Predicting remaining useful life of lithium-ion batteries for electric vehicles using machine learning regression, J. Eng. Technol. Ind. Appl. 15 (2023).

[22] C.L. Sravanthi, J.N.C. Sekhar, N.C. Alluraiah, C. Dhanamjayulu, H.K. Pujari, B. Khan, An overview of remaining useful life prediction of battery using deep learning and ensemble learning algorithms on data-dependent models, Int. Trans. Electr. Energy Syst. (2025).

[23] B. Guo, Y. Xu, X. Feng, State-of-health estimation and remaining-useful-life prediction for lithium-ion battery using a hybrid data-driven method, IEEE Trans. Veh. Technol. 69 (2020) 10854–10867.

[24] A.A. Miaari, H.M. Ali, Batteries temperature prediction and thermal management using machine learning: An overview, Energy Rep. 10 (2023).

[25] Y. Li, B. Duan, N. Wang, G. Zhao, P. Gu, C. Zhang, Core temperature estimation method of lithium-ion battery based on an improved recurrent neural network algorithm, in: Chinese Control and Decision Conference, CCDC, 2022, pp. 6320–6324.

[26] F. Kolodziejczyk, B. Mortazavi, T. Rabczuk, X. Zhuang, Machine learning assisted multiscale modeling of composite phase change materials for Li-ion batteries' thermal management, Int. J. Heat Mass Transf. 172 (2021).

[27] A.A. Miaari, H.M. Ali, Batteries temperature prediction and thermal management using machine learning: An overview, J. Energy Rep. 10 (2023) 2277–2305.

[28] J. Zhan, X. Han, M. Ouyang, A.F. Burke, Specialized deep neural networks for battery health prognostics: opportunities and challenges, J. Energy Chem. 87 (2023) 416–438.

[29] R. Sanchez-Iborra, A.F. Skarmeta, TinyML-enabled frugal smart objects: Challenges and opportunities, IEEE Circuits Syst. Mag. 20 (2020) 4–18.

[30] G. Crocioni, D. Pau, J.M. Delorme, G. Gruosso, Li-ion batteries parameter estimation with tiny neural networks embedded on intelligent IoT microcontrollers, IEEE Access (2020).

[31] Y. Mazzi, H.B. Sassi, A. Gaga, F. Errahimi, State of charge estimation of an electric vehicle's battery using tiny neural network embedded on small microcontroller units, Int. J. Energy Res. 46 (2022).

[32] J. Lin, L. Zhu, W.M. Chen, W.C. Wang, S. Han, Tiny machine learning: Progress and futures [feature], IEEE Circuits Syst. Mag. 23 (2023) 8–34.

[33] Y. Weng, W. Guan, C. Ababei, Prediction of remaining useful life and cell temperature for li-ion batteries using TinyML, in: IEEE International Midwest Symposium on Circuits and Systems, MWSCAS, 2021, pp. 562–566.

[34] A. Gupta, The imperative for sustainable AI systems (2024), https://thegradient.pub/sustainable-ai/.

[35] R. David, J. Duke, A. Jain, V.J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev, R. Rhodes, T. Wang, P. Warden, TensorFlow lite micro: Embedded machine learning on TinyML systems, in: Proceedings of Machine Learning and Systems, 2021, pp. 800–811.

[36] S. Mittal, A survey of techniques for improving energy efficiency in embedded computing systems, Int. J. Comput. Aided Eng. Technol. 6 (2014) 440–459.

[37] W. Xu, W. Fang, Y. Ding, M. Zou, N. Xiong, Accelerating federated learning for IoT in big data analytics with pruning, quantization, IEEE Access 9 (2021) 38457–38466.

[38] M. Koot, F. Wijnhoven, Usage impact on data center electricity needs: A system dynamic forecasting model, Appl. Energy 291 (2021).

[39] M. Chen, C. Gao, Z. Ren, Robust covariance and scatter matrix estimation under Huber's contamination model, Ann. Statist. 46 (2018).

[40] S. Ma, D. Li, T. Hu, Y. Xing, Z. Yang, W. Nai, Huber loss function based on variable step beetle antennae search algorithm with Gaussian direction, in: Inter. Conf. on Intelligent Human-Machine Systems and Cybernetics, IHMSC, 2020, pp. 248–251.

[41] S. Bai, J.Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018, ArXiv. https://arxiv.org/abs/1803.01271.

[42] A.V.D. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, N. Kalchbrenner A. Graves, A. Senior, K. Kavukcuoglu, WaveNet: A generative model for raw audio, in: ISCA Workshop on Speech Synthesis Workshop, SSW, 2016.

[43] Y. Liu, J. Li, G. Zhang, B. Hua, N. Xiong, State of charge estimation of lithium-ion batteries based on temporal convolutional network and transfer learning, IEEE Access 9 (2021).

[44] Y. Liu, H. Dong, X. Wang, S. Han, Time series prediction based on temporal convolutional network, in: IEEE/ACIS International Conf. on Computer and Information Science, ICIS, 2019, pp. 300–305.

[45] L. Huang, K. Zhou, S. Chen, Y. Chen, J. Zhang, Automatic detection of epilepsy from EEGs using a temporal convolutional network with a self-attention layer, BioMed. Eng. OnLine 23 (2024).

[46] C. Lea, M.D. Flynn, R. Vidal, A. Reiter, G.D. Hager, Temporal convolutional networks for action segmentation and detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 1003–1012.

[47] D. Cao, Y. Huang, H. Li, X. Zhao, Q. Zhao, Y. Fu, Text sentiment classification based on LSTM-TCN hybrid model and attention mechanism, in: Association for Computing Machinery, 2020, pp. 1003–1012.

[48] J. Chen, W. Chong, S. Yu, Z. Xu, C. Tan, N. Chen, TCN-based lightweight log anomaly detection in cloud-edge collaborative environment, in: International Conference on Advanced Cloud and Big Data, CBD, 2022, pp. 13–18.

[49] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conf. on Computer Vision and Pattern Recognition, 2015.

[50] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Johnes, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Conf. on Neural Information Processing Systems, NIPS, 2017.

[51] M. Matar, T. Xia, K. Huguenard, D. Huston, S. Wshah, Multi-head attention based Bi-LSTM for anomaly detection in multivariate time-series of WSN, in: Inter. Conf. on Artificial Intelligence Circuits and Systems, AISCA, 2023, pp. 1–5.

[52] X. Xu, S. Gao, Z. Jiang, LSTCN: An attention-based deep neural network model combining LSTM and TCN for cellular network traffic prediction, in: Inter. Conf. on Communication and Information Systems, ICCIS, 2021, pp. 34–38.

[53] V. De Angelis, Y. Preger, BatteryArchive.org - insights from a public repository for visualization, SAND2021-6330C analysis, comparison of battery data across institutions, 2021, Sandia National Lab (SNL-NM).

[54] Y. Preger, H.M. Barkholtz, A. Fresquez, D.L. Campbell, B.W. Juba, J. Romàn-Kustas, S.R. Ferreira, B. Chalamala, Degradation of commercial lithium-ion cells as a function of chemistry and cycling conditions, J. Electrochem. Soc. 167 (2020).

[55] GREPOW Rechargeable Battery, NMC vs NCA battery cell: What's the difference? (2024), https://www.grepow.com/blog/nmc-vs-nca-battery-cell-what-is-the-difference.html.

[56] F. Shadram, S. Bhattacharjee, S. Lidelöw, J. Mukkavaara, T. Olofsson, Exploring the trade-off in life cycle energy of building retrofit through optimization, Appl. Energy 269 (2020).

[57] Wikipedia, Lithium iron phosphate battery, https://en.wikipedia.org/wiki/Lithium_iron_phosphate_battery.

[58] IEA, Trends in batteries (2023), https://www.iea.org/reports/global-ev-outlook-2023/trends-in-batteries.

[59] G.D. Reis, C. Strange, M. Yadav, S. Li, Lithium-ion battery data and where to find it, J. Energy AI 5 (2021).

[60] Y. Preger, H.M. Barkholtz, A. Fresquez, D.L. Campbell, B.W. Juba, J. Roman-Kustas, S.R. Ferreira, B. Chalamala, Degradation of commercial lithium-ion cells as a function of chemistry and cycling conditions, J. Electrochem. Soc. (2020) 34–38.
[61] The moving average filter (2018), https://www.advsolned.com/the-moving-average-filter/.
[62] W. Li, H. Zhang, B.v. Vlijmen, P. Dechent, D.U. Sauer, Forecasting battery capacity and power degradation with multi-task learning, Energy Storage Mater. 53 (2022).
[63] X. Bao, Y. Liu, B. Liu, H. Liu, Y. Wang, Multi-state online estimation of lithium-ion batteries based on multi-task learning, Energies 16 (2021).
[64] Google, Post-training dynamic range quantization (2025), https://ai.google.dev/edge/litert/models/post_training_quant.
[65] A. Dey, S. Srivastava, G. Singh, R.G. Pettit, Real-time performance benchmarking of tinyml models in embedded systems (PICO: performance of inference, CPU, and operations), in: International Symposium on Real-Time Distributed Computing Conference, ISORC, 2025.
[66] S. Sarkar, M.F. Babar, M.M. Hassan, M. Hasan, S.K.K. Santu, Processing natural language on embedded devices: How well do transformer models perform? Assoc. Comput. Mach. (2024).