# Improved LSTM-Based Battery SOH Estimation with Differential Evolution Hyperparameter Optimization

Karthickumar Ponnambalam[1], B. Sivaneasan[2], A. Sharma[1], S. S. Lee[1]

[1] *Newcastle University, Singapore*
*k.ponnambalam2@newcastle.ac.uk*
*anurag.sharma@newcastle.ac.uk szesing.lee@newcastle.ac.uk*

[2] *Singapore Institute of Technology, Singapore*
*sivaneasan@singaporetech.edu.sg*

## Abstract

The reliable operation of battery energy storage systems hinges on prompt and accurate assessment of cell health. This work proposes a deep learning framework for lithium-ion battery State of Health (SoH) estimation built on Long Short Term Memory (LSTM) networks whose hyperparameters are tuned via Differential Evolution (DE). Using aging datasets from NASA's Open Data Portal, we search over key design choices (e.g., batch size, hidden units, activation, and loss) to obtain robust configurations without manual trial and error. Building on our prior GA-LSTM study, which reported Root Mean Square Error (RMSE) reductions of 12.4% to 76.79% over a Particle Swarm optimized LSTM (PA-LSTM). This paper replaces the genetic search with DE and evaluates the resulting DE-LSTM under identical data splits and metrics. Empirically, DE-LSTM delivers an additional RMSE reduction of 38% to 91% relative to our GA-LSTM baseline and consistently improves predictive stability across diverse degradation trajectories. These findings indicate that DE driven hyperparameter optimization offers a strong and scalable path to high precision SoH prediction, advancing beyond both the previously published PA-LSTM benchmark and our earlier GA-based optimization.

*Index Terms*: Battery, energy storage, state-of-health, LSTM, genetic algorithm, Differential Evolution, hyperparameter optimization, prediction.

## 1. Introduction

Historically, estimation of the State of Health (SoH) and Remaining Useful Life (RUL) of Li-ion batteries has relied on physics based models. Approaches grounded in battery operating principles such as equivalent circuit and electrochemical models have been used to predict degradation trajectories and infer SoH [1]. However, the coupled, nonlinear physico chemical processes in Li-ion cells make accurate mechanistic modeling difficult and often device specific [2]. As a result, data driven methods that do not require explicit aging dynamics have attracted significant attention. A range of machine learning techniques including support vector machines (SVM), Gaussian process regression (GPR), monotonic echo state networks (MONESNs), dynamic Bayesian networks (DBNs), and recurrent neural networks such as long short term memory (LSTM) have been explored for SoH/RUL prediction.

Motivated by this shift toward learning based solutions, this paper proposes a Differential Evolution (DE) tuned LSTM for battery SoH estimation. In the proposed framework, DE acts as a population based hyperparameter optimizer that searches over architectural and training choices (e.g., hidden units, learning rate, dropout, batch size) to improve predictive accuracy and efficiency without manual trial and error. The resulting DE-LSTM aims to provide a practical, data efficient alternative for battery health monitoring under diverse operating conditions.

## 2. Literature Review

### 2.1. Degradation Prediction

Extensive work has examined how lithium-ion batteries age over time. Broadly, degradation mechanisms are grouped into two categories *calendar aging* and *cycle aging* [3]. Calendar aging reflects time dependent loss that occurs even when a cell is not cycled; it is strongly influenced by temperature and the average state of charge (SoC). In contrast, cycle aging is driven by charge/discharge use and depends on factors such as temperature, SoC, current (or C-rate), and depth of discharge. Prior studies report that calendar aging often follows a non linear progression, whereas cycling induced aging can appear approximately linear with respect to accumulated throughput [4].

Because the underlying electrochemical pathways are complex and vary across chemistries, *data driven* approaches have become a practical alternative for analyzing and forecasting degradation without detailed priors on cell physics. These methods learn relationships between measured signals and capacity loss directly from data and can therefore be applied across battery types [5]. A typical workflow extracts features from voltage, current, and temperature measurements, identifies temporal trends or health indicators, and then fits a regression model to predict capacity or SoH [6]. Algorithms explored in the literature include support vector machines (SVM), relevance vector machines (RVM), dynamic Bayesian networks (DBN), hidden Markov models (HMM) [7], and artificial neural networks (ANN) [8]. More recently, deep neural networks particularly recurrent architectures are favored for their ability to learn informative representations and adapt to non stationary operating conditions, making them well suited for battery degradation prediction.

### 2.2. State of Health

The state of health (SoH) is the most common metric for assessing lithium-ion battery ageing, defined as the ratio of a cell's usable capacity to its initial capacity [9]. In this work, SoH at cycle $p$ is given by:

$$SoH_p = \frac{C_p}{C_0}, \tag{1}$$

where $C_p$ is the measured capacity at cycle $p$ and $C_0$ the baseline capacity. SoH is often expressed as a percentage ($100 \times SoH_p$). While resistance or power based definitions exist, this study adopts the capacity ratio for consistency with the NASA datasets and prior studies.

### 2.3. Evaluation of Predictive Models for Battery SoH

Battery state of health (SoH) forecasting has been studied using diverse machine learning methods, each with advantages and limitations [10]. Below we summarize commonly used approaches:

- **Linear regression:** Simple and effective for near-linear trends, but performs poorly with the nonlinear effects of electrochemical ageing [11].
- **Decision trees:** Capture nonlinear splits and feature interactions, yet often overfit noisy, high dimensional data and show weak generalization [12].
- **Random forests:** Improve robustness over single trees but fail to model the long range temporal dependencies in degradation sequences [13].
- **CNNs:** Strong in extracting spatial features, but less suited to sequential SoH data unless transformed into time frequency representations [14].
- **LSTMs:** Unlike regression or tree-based models that struggle with sequential dependencies, and CNNs that require data transformation, LSTMs are explicitly designed for time series analysis. They capture both short and long term temporal patterns directly from sensor streams, reducing reliance on handcrafted features and consistently providing more accurate SoH/RUL predictions [15].

### 2.4. Optimization of Predictive Models

Designing and training neural networks effectively is key to reliable performance, typically measured by accuracy and F1 score in classification or RMSE/MAE in regression. For battery SoH prediction, the main challenge is ensuring robust generalization under noisy and limited data. This is essential for safe operation, maintenance planning, and extending battery life. Our work addresses this through systematic hyperparameter tuning to minimize error and stabilize performance across diverse conditions.

- **LSTM Hyperparameter Tuning:** LSTM models are highly sensitive to hyperparameters such as learning rate, hidden size, depth, dropout, sequence length, and batch size. Many of these are discrete or conditional, for example, adding layers activates additional parameters. In practice, factors like limited training budgets, gradient clipping, and early stopping make the outer optimization problem non smooth, which reduces the effectiveness of gradient based tuning. Evolutionary optimization offers a strong alternative because it (i) does not rely on outer gradients, (ii) can handle mixed discrete and continuous search spaces, (iii) preserves diversity within the population to balance exploration and exploitation, and (iv) remains robust against noise and local minima. These properties make evolutionary methods well suited for LSTM hyperparameter and architecture search.
- **Population based optimizers considered in this study:** We focus on three well established algorithms: *Particle Swarm Optimization (PSO)*, referred to here as *PA* for consistency with the manuscript [16], along with *Genetic Algorithms (GA)* and *Differential Evolution (DE)*.
  - **PA (Particle Swarm Optimization):** PA models each solution as a "particle" that moves through the search space by combining inertia, a pull toward its own best position, and attraction to the swarm's global best. This cooperation often yields rapid early gains, and PA is straightforward to parallelize. However, it can converge prematurely without diversity-preserving mechanisms (e.g.velocity damping) and may stall on rugged landscapes [17]. PA-LSTM has previously been applied to battery SoH prediction [16].
  - **GA (Genetic Algorithms):** GAs evolve a population of solutions through selection, crossover, and mutation. Fitter candidates are favored for reproduction, while genetic operators introduce diversity to explore new regions [18]. This approach has proven effective for neural network hyperparameter search [19], and in our earlier work GA–LSTM achieved notable improvements over PA–LSTM for SoH prediction [20].
  - **DE (Differential Evolution):** DE generates new solutions by adding a scaled difference between two population vectors to a base vector, followed by crossover and greedy selection [21]. With only a few parameters and strong balance between exploration and exploitation, DE converges reliably and efficiently. It has been widely applied to neural network optimization [22] and has shown promise as a robust alternative to GA for LSTM tuning [23].

Relative to PA and GA, DE's difference vector mutation adapts step sizes to the population's current diversity, naturally balancing exploration and exploitation with minimal tuning [24, 25]. It operates natively in continuous domains and accommodates integer/binary decisions via simple encodings [26]. These traits suit the multi modal, noisy objectives common in LSTM hyperparameter and architecture search, where diversity aware proposals help avoid premature convergence and deliver stable improvements with modest effort [27].

### 2.5. Look-back Training for Incremental Learning

Traditional approaches often train on pooled data from many cells, which can be computationally expensive and may obscure the distinct ageing behavior of individual batteries. Incremental learning provides a more efficient alternative by updating the model only with newly acquired data, reducing computation and allowing real-time adaptation [28]. Within this framework, *look-back* training uses a fixed window of each battery's past cycles to capture its specific degradation trajectory. This approach supports more tailored modeling and enhances the precision of long-term state-of-health estimation [29].

This approach aligns well with time series SoH prediction. By using each cell's historical sequences, the model learns evolving dependencies and drift, which enhances the accuracy of future health estimates [30]. Within our framework, look-back training complements DE based hyperparameter optimization by supplying consistent, cell specific temporal context while the optimizer selects robust LSTM configurations.

### 2.6. Research Gap

Recent advancements in battery health prediction have improved accuracy and performance; however, notable gaps remain. Current models often struggle to balance predictive precision with computational efficiency, and there is an increasing demand for data efficient approaches that can generalize effectively across diverse battery chemistries, designs, and operating conditions [31]. While transfer learning and domain adaptation techniques have shown potential in bridging performance gaps between different battery devices, their integration into practical battery management systems (BMS) is still limited.

One of the critical shortcoming lies in the absence of lightweight, data driven BMS solutions that can be easily tailored to specific battery types while requiring minimal training data. Additionally, few existing methods support real time predictive analytics capable of continuous State of Health (SoH) monitoring and timely decision making in operational environments. This is particularly important for applications where rapid detection of degradation patterns can prevent failures and optimize battery usage.

To address these challenges, novel strategies such as Look-back training [29] which enables models to leverage historical sequences for capturing temporal dependencies and optimization techniques like Differential Evolution [32] offer promising directions. These methods can improve adaptability, enhance generalization, and sustain high performance even in data constrained settings. There is a pressing need for a unified machine learning framework that combines temporal sequence modeling with robust optimization to enable domain adaptation, efficient learning, and real-time applicability. Integrating look-back training with evolutionary optimization not only addresses current limitations in live battery system training but also facilitates continuous adaptation to dynamic battery behavior, ultimately advancing the capabilities of State of Health (SoH) prediction models in real world scenarios.

## 3. METHODOLOGY

Our approach is structured into three sequential phases as shown in Fig.1: (1) Data Process, (2) SOH Prediction, and (3) Differential Evolution Optimization.

### 3.1. Data Process

Commonly used sources of battery degradation data include the NASA PCoE Battery Data Repository [33], the University of Maryland's Center for Advanced Life Cycle Engineering (CALCE), and the Oxford Battery Degradation Dataset. In this study, the NASA dataset was selected. Specifically, three Li-ion cells with serial numbers B0005, B0006, and B0018 were used. These are 18650 format batteries with a nominal capacity of 2 Ah. The cells were subjected to three operating modes: (1) charging, (2) discharging, and (3) electrochemical impedance spectroscopy, each performed under different temperature conditions. Testing continued until the end-of-life (EOL) criterion was reached, defined as a 30% reduction in rated capacity, from 2 Ah down to 1.4 Ah.

### 3.2. Data Analysis and Preprocessing

In this paper, the capacity dataset from NASA, facilitates manual SoH calculation, which is used to train the model.

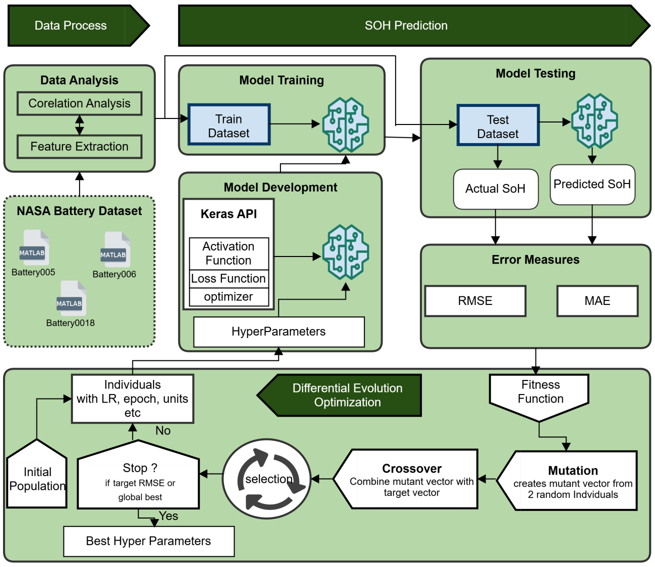


Figure 1. Methodology

Initially, the correlation between capacity and cycle data were examined. As shown in Fig2, battery capacity declines with increasing cycle numbers, with capacities falling below 1.4Ah after approximately 163 cycles, mirroring a consistent trend across all NASA Li-ion batteries.

As SoH is derived from capacity as given by (1), it can also be observed from Fig. 3 that the SOH reduces as the number of cycle increases.

Most existing studies on battery degradation rely on pooled data from multiple cells, which is computationally demanding. In this work, we adopt an incremental (or look-back) learning approach, where model parameters are updated as new data arrive, improving efficiency with less data and computational power [28]. A generic function was implemented to generate look-back datasets from NASA's raw discharge time series data. Look-back training uses a fixed window of past observations (e.g., the last 30 cycles) to predict the next value, such as degradation or state of health, thereby capturing temporal patterns that improve forecasting accuracy. As new data arrive, the window shifts forward and the model refines its predictions, enabling it to adapt to evolving behaviors, for example, learning that frequent high discharge cycles accelerate capacity fading. Continuous updates enable the model to use past data for forecasting upcoming cycles, offer-

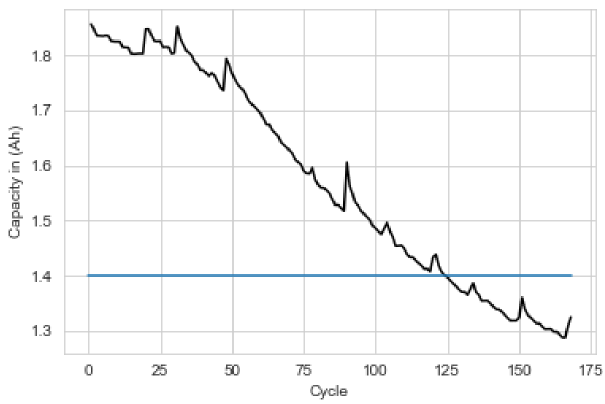


Figure 2. Cycle Vs Capacity

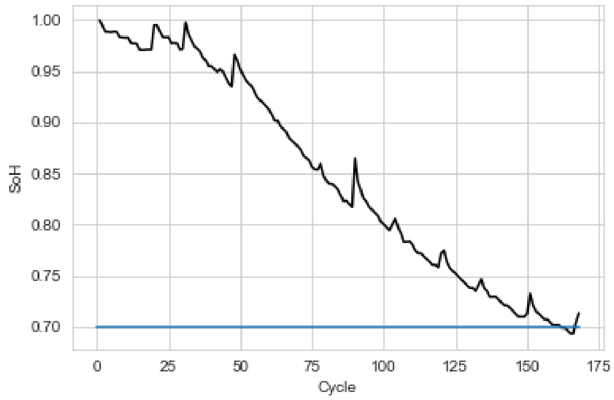


Figure 3. Cycle Vs SoH

ing actionable insights for battery management. Algorithm 1 presents the look-back dataset generation process.

**Algorithm 1** Construction of Look-Back Training Dataset

**Require:** Time series data $batteryData$, look-back window size $windowSize$
**Ensure:** Input sequences $X$, target values $Y$

Initialize $X \leftarrow [], Y \leftarrow []$
**for** $i = 0$ to $len(batteryData) - windowSize$ **do**
    $sequence \leftarrow batteryData[i : (i + windowSize)]$
    Append $sequence$ to $X$
    $label \leftarrow batteryData[i + windowSize]$
    Append $label$ to $Y$
**end for**
**return** $X, Y$

Algorithm 1 **Construction of Look-Back Training Dataset** constructs training samples by pairing short sequences of past observations with the next expected value. Given a time series and a chosen window length, it iteratively collects subsequences of length *windowSize* and stores them in $X$. For each subsequence, the element that immediately follows is stored in $Y$ as its target. In this way, $X$ contains the input sequences representing recent state-of-health (SoH) history, while $Y$ holds the corresponding SoH values for the subsequent cycle. Adjusting the window size controls how much historical information is included, enabling the model to learn degradation patterns and improve predictive accuracy.

### 3.3. SOH Prediction

**Model Development:** In this work, a Long Short-Term Memory (LSTM) network is combined with a fully connected multilayer neural network. During training, input sequences are passed through successive transformations (both linear and nonlinear) in a process referred to as forward propagation. The resulting prediction is then compared against the true label using a loss function, which quantifies the prediction error. To reduce this error, we apply gradient descent optimization, where each trainable weight parameter is iteratively updated. The update rule is defined as:

$$W_t = W_{t-1} - \eta \nabla \mathcal{L}(W_{t-1}) \tag{2}$$

Here, $\eta$ represents the learning rate, $\mathcal{L}(W_{t-1})$ denotes the loss function evaluated at the previous weights, and $\nabla \mathcal{L}(W_{t-1})$ is its gradient. Parameters are adjusted in the opposite direction of the gradient vector, if the gradient component is negative, the weight is increased, while a positive gradient leads to a reduction. This iterative process, known as backpropagation, gradually tunes the weights to minimize the loss and improve prediction accuracy.

For implementation, the Keras API is employed to define LSTM layers, specify input dimensions, and configure training. The selection of hyperparameters such as the number of LSTM layers, learning rate, batch size, and training epochs, strongly influences model performance. However, manual tuning is computationally expensive due to the large search space. To overcome this challenge, we introduce a Differential Evolution (DE) based strategy that automatically explores hyperparameter configurations, thereby improving both training efficiency and predictive performance [34].

**Training:** The proposed model employs an incremental learning strategy, where parameters are updated as new battery degradation data become available while preserving knowledge from earlier cycles. The LSTM network is first trained on an initial portion of the dataset and subsequently refined by incorporating additional data, avoiding the need to retrain on the entire history. This enables real-time adaptation to changing battery conditions and supports more accurate SoH forecasting. By continually updating its internal representation, the LSTM model improves its understanding of degradation dynamics and provides a foundation for predictive maintenance.

**Testing:** Experimental evaluation is conducted on three cells from the NASA battery dataset: *B0005*, *B0006*, and *B0018*. For each cell, look-back sequences are prepared and partitioned into training and testing sets. A generic training function, parameterized by multiple hyperparameters, is used to fit the model with a chosen fraction of the data, where *train_x* represents the input sequences and *train_y* the corresponding targets. The remaining portion (*test_x*) is employed to generate predicted SoH values ($\hat{y}$). Model accuracy is then assessed by comparing these predictions against the ground truth (*test_y*) using the Root Mean Squared Error (RMSE) metric.

### 3.4. Differential Evolution Optimization

The training and testing workflow is driven by a Differential Evolution (DE) algorithm, which is used to automatically tune key hyperparameters of the LSTM model for battery SoH prediction. These hyperparameters include the number of LSTM layers, learning rate, dropout rate, batch size, and other structural or training-related parameters. DE is a population based evolutionary algorithm that maintains a set of candidate hyperparameter configurations, which evolve across successive generations to improve model performance. As illustrated in Fig.4, the process begins with the random initialization of a population of candidate solutions. In each generation, DE performs a *mutation* step, where a new candidate (mutant vector) is created by combining the weighted difference of two randomly selected solutions with a third solution. Next, during the *crossover* step, elements of this mutant vector are recombined with the current candidate (target vector) to form a trial solution. The trial solution is then evaluated by training and validating the LSTM model, and its performance (fitness) is measured using error metrics such as RMSE or MAE.

Finally, in the *selection* step, the trial solution replaces the target if it achieves lower validation error. This procedure is repeated for all individuals in the population, ensuring that each new generation either maintains or improves the quality of solutions. The cycle of mutation, crossover, and selection continues until a stopping criterion is reached, such as a predefined number of generations or achieving the desired validation accuracy.

### 3.4.1. Search Space Definition for DE-LSTM

To ensure clarity and reproducibility, Table 1 lists the key hyperparameters optimized by Differential Evolution (DE) along with their explored ranges.

Table 1. Search Space for DE-LSTM Hyperparameters

| **Parameter** | **Range / Set** |
|---|---|
| $H$ | {8, 16, 32, 64, 128, 256} |
| $W$ | 1–30 |
| $B$ | GPU: 32–512; CPU: 8–128 |
| $w_{p1}, w_{p2}$ | [–0.05, 0.05] |

*Abbreviations and Notes:* $H$ – number of LSTM hidden units controlling model capacity. $W$ – look-back window size (number of previous cycles). $B$ – batch size, which affects training speed and gradient stability. $w_{p1}, w_{p2}$ – optional small weight perturbations for regularization. These ranges were empirically determined to balance accuracy, convergence stability, and computational efficiency during DE optimization.

### 3.4.2. Operational form (DE-LSTM)

At generation $g$, let $\{\mathbf{x}_i^{(g)}\}_{i=1}^{NP}$ denote candidate hyperparameter vectors (e.g., units, learning rate, batch size, epochs), and let $\bar{J}(\cdot)$ be the validation error (RMSE/MAE) to minimize. A common and effective instantiation is *DE/best/1* with binomial crossover:

$$\textbf{Mutation:}\quad \mathbf{v}_i^{(g)} = \mathbf{x}_*^{(g)} + F\big(\mathbf{x}_{r_1}^{(g)} - \mathbf{x}_{r_2}^{(g)}\big), \tag{3a}$$

$$\textbf{Crossover:}\quad u_{i,j}^{(g)} = \begin{cases} v_{i,j}^{(g)}, & \mathrm{rand}_j \le CR \text{ or } j = j_{\mathrm{rand}}, \\ x_{i,j}^{(g)}, & \text{otherwise}, \end{cases} \tag{3b}$$

$$\textbf{Selection:}\quad \mathbf{x}_i^{(g+1)} = \begin{cases} \mathbf{u}_i^{(g)}, & \bar{J}(\mathbf{u}_i^{(g)}) \le \bar{J}(\mathbf{x}_i^{(g)}), \\ \mathbf{x}_i^{(g)}, & \text{otherwise}. \end{cases} \tag{3c}$$

*Notes.* $\mathbf{x}_i^{(g)}$ is the $i$-th candidate hyperparameter vector at generation $g$; $\mathbf{x}_*^{(g)}$ is the current best; $r_1 \neq r_2 \neq i$ are distinct random indices. $F \in (0, 2)$ is the differential weight (step size scale) and $CR \in [0, 1]$ is the crossover probability. $\mathbf{v}_i^{(g)}$ is the mutant; $\mathbf{u}_i^{(g)}$ the trial. $\bar{J}(\cdot)$ is the validation error (e.g., RMSE/MAE) minimized during selection.

**Explanation:** In each generation, Differential Evolution (DE) maintains a population of LSTM hyperparameter candidates $\mathbf{x}_i^{(g)}$ and the current best solution $\mathbf{x}_*^{(g)}$. A *mutant* $\mathbf{v}_i^{(g)}$ is created by perturbing the best solution with a scaled difference of two other candidates, where the factor $F \in (0, 2)$ controls step size, larger $F$ explores broadly, smaller $F$ refines locally. A *trial* $\mathbf{u}_i^{(g)}$ is then formed by combining the mutant and the current candidate gene by gene with crossover probability $CR \in [0, 1]$, determining how much mutant information is introduced. The trial is trained and validated, and DE applies *greedy selection*: if the trial performs no worse than the current candidate (based on error metrics such as RMSE or MAE), it replaces it. Through repeated mutation, crossover, and selection, validation error decreases across generations, producing tuned hyperparameters. The final

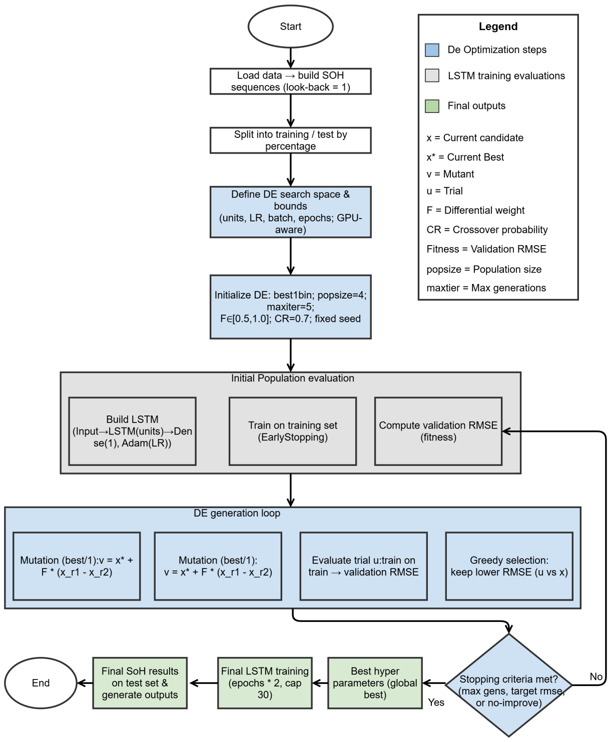


Figure 4. DE Optimization Process for LSTM

LSTM is then retrained with these optimized settings and tested for SoH prediction.

### 3.4.3. Computational Complexity

While Differential Evolution (DE) improves predictive performance, it also introduces additional computational overhead by evaluating multiple LSTM configurations across generations. The overall cost scales approximately as $\mathcal{O}(N_P \times G \times T_{\text{train}})$, where $N_P$ is the population size, $G$ the number of generations, and $T_{\text{train}}$ the average training time for one configuration. In this work, DE was executed with $N_P = 20$ and $G = 30$, which provided stable convergence without excessive computation. The framework was first validated on the free version of Google Colab (NVIDIA T4 GPU, 16 GB RAM) and later extended to a university's Linux server with an 8-core Xeon CPU (16 threads), NVIDIA RTX 2080 Ti GPU (11 GB VRAM), 128 GB RAM, and SSD storage. Under this configuration, the complete DE LSTM optimization, including training and testing for all three NASA cells (B0005, B0006, B0018) and three training fractions (30%, 50%, 70%) required approximately 12 hours of continuous execution.

After identifying the optimal hyperparameters, a final LSTM model was rebuilt using those values to form the optimized DE–LSTM network. This model can then be deployed on standard hardware for further look-back training or real time inference without the need for high end computation, as all weights and hyperparameters are already tuned. Consequently, while the DE search phase is computationally intensive, the resulting model remains lightweight and practical for Battery Management System (BMS) applications.

## 4. RESULTS

This study evaluates the effectiveness of Differential Evolution tuned LSTM (DE–LSTM) models for predicting battery state-of-health (SoH). The evaluation is conducted on the same NASA degradation datasets used in prior benchmarks, including PA–LSTM[16] and our earlier GA–LSTM approach[20]. Table2 summarizes the root mean squared error (RMSE) values obtained across three training fractions (30%, 50%, and 70%) for batteries B0005, B0006, and B0018. Baseline comparisons include simple LSTM, simple RNN, and relevance vector machine (RVM).

Overall, DE–LSTM consistently outperforms the plain baselines and provides remarkable improvements over PA–LSTM across most training scenarios. At the 70% training condition, DE–LSTM achieves RMSEs of 0.00055, 0.00070, and 0.00163 for B0005, B0006, and B0018, respectively. These values correspond to error reductions of 90.9%, 91.2%, and 86.9% compared with PA–LSTM. Relative to our previously reported GA–LSTM results, DE–LSTM further lowers the error by 89.6% (B0005), 86.1% (B0006), and 38.1% (B0018) under the same 70% training condition. Similarly, For the 30% and 50% training fractions, DE–LSTM continues to demonstrate robust performance, showing improvements over PA–LSTM in nearly all cases.

Fig 5 present the outcomes under 70% look-back training condition. In this plot, the black curve denotes the observed SoH during the training phase, the green curve represents the true degradation trajectory for the remaining cycles, and the blue curve corresponds to the DE–LSTM prediction. The close match between predicted and actual values highlights the ability of DE-based hyperparameter tuning to stabilize training and accurately capture long-term degradation behavior. These findings confirm that, particularly in data-rich settings, DE–LSTM provides more reliable and precise SoH predictions compared to GA–LSTM and PA–LSTM.

## 5. CONCLUSION

This paper presented a Differential Evolution optimized LSTM (DE–LSTM) framework combined with a novel look-back training strategy for state-of-health (SoH) prediction in lithium-ion batteries. The proposed approach not only identifies optimal LSTM hyperparameters and enhances prediction accuracy, but also introduces domain interoperability: instead of relying on pooled datasets from multiple batteries, the model learns from each battery's own degradation history, enabling efficient and transferable forecasts of remaining useful life (RUL) across similar battery types. Evaluations on NASA datasets demonstrated significant error reductions compared with baseline models, including simple LSTM, RNN, RVM, GA–LSTM, and PA–LSTM with improvements of up to 91% in data rich settings. Looking ahead, our future work will focus on deploying this method within real-time Battery Management Systems (BMS) and validating its effectiveness through experiments on physical battery cells. By collecting in-house degradation data and extending evaluation to diverse chemistries and operating conditions, we aim to further establish the generalizability of the framework and support its adoption in next generation energy storage applications.

## REFERENCES

[1] Shengbo Eben Li, Baojin Wang, Huei Peng, and Xiaosong Hu. An electrochemistry-based impedance model for lithium-ion batteries. *Journal of Power Sources*, 258:9–18, 7 2014.

[2] Jia Jianfang, Keke Wang, Shi Yuanhao, Wen Jie, and Zeng Jianchao. Multi-Scale Prediction of RUL and SOH for Lithium-Ion Batteries Based on WNN-UPF Combined Model. *Chinese Journal of Electronics*, 30(1), 2021.

[3] Eberhard Meissner and Gerolf Richter. The challenge to the automotive battery industry: the battery has to become an increasingly integrated component within the vehicle electric power system. *Journal of Power Sources*, 144(2):438–460, 6 2005.

[4] Eduardo Redondo-Iglesias, Pascal Venet, and Serge Pelissier. Calendar and cycling ageing combination of batteries in electric vehicles. *Microelectronics Reliability*, 88-90:1212–1215, 9 2018.

[5] Tan Afj, Tomacruz Jgt, Remolona Mfm, Guest Editors, Petar S Varbanov, Yee Van Fan, Jiří J Klemeš, Sandro Nižetić, Lerissah D Lim, Andrei J Felix Tan, Jan T Goran Tomacruz, Michael T Castro, Miguel M Francisco Remolona, and Joey D Ocon. CHEMICAL ENGINEERING TRANSACTIONS Cyclic Degradation Prediction of Lithium-Ion Batteries using Data-Driven Machine Learning.

[6] Wasim Ahmad, Sheraz Ali Khan, M. M.Manjurul Islam, and Jong Myon Kim. A reliable technique for remaining useful life estimation of rolling element bearings using dynamic regression models. *Reliability Engineering & System Safety*, 184:67–76, 4 2019.

[7] Thanh Trung Le, Florent Chatelain, and Christophe Bérenguer. Multi-branch hidden Markov models for remaining useful life estimation of systems under multiple deterioration modes. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 230(5):473–484, 10 2016.

[8] S Saon, T Hiyama Computers Mathematics with Applications, and undefined 2010. Predicting remaining useful life of rotating machinery based artificial neural network. *Elsevier*.

[9] Qianglong Li, Dezhi Li, Kun Zhao, Licheng Wang, and Kai Wang. State of health estimation of lithium-ion battery based on improved ant lion optimization and support vector regression. *Journal of Energy Storage*, 50:104215, 6 2022.

[10] Xiao Sheng Si, Wenbin Wang, Chang Hua Hu, and Dong Hua Zhou. Remaining useful life estimation - A review on the statistical data driven approaches. *European Journal of Operational Research*, 213(1):1–14, 8 2011.

[11] Søren B. Vilsen and Daniel Ioan Stroe. Battery state-of-health modelling by multiple linear regression. *Journal of Cleaner Production*, 290:125700, 3 2021.

[12] Lijun Qian, Liang Xuan, and Jian Chen. Battery soh estimation based on decision tree and improved support vector machine regression algorithm. *Frontiers in Energy Research*, 11:1218580, 6 2023.

[13] Kodjo S.R. Mawonou, Akram Eddahech, Didier Dumur, Dominique Beauvois, and Emmanuel Godoy. State-of-health estimators coupled to a random forest approach for lithium-ion battery aging factor ranking. *Journal of Power Sources*, 484:229154, 2 2021.

[14] Yuxuan Zheng, Jiaxiang Hu, Jianjun Chen, Huiwen Deng, and Weihao Hu. State of health estimation for lithium battery random charging process based on cnn-gru method. *Energy Reports*, 9:1–10, 5 2023.

[15] Hanlei Sun, Jianrui Sun, Kun Zhao, Licheng Wang, and Kai Wang. Data-driven ica-bi-lstm-combined lithium battery soh estimation. *Mathematical Problems in Engineering*, 2022, 2022.

[16] Jiantao Qu, Feng Liu, Yuxiang Ma, and Jiaming Fan. A neural-network-based method for rul prediction and soh monitoring of lithium-ion battery. *IEEE Access*, 7:87178–87191, 2019.

[17] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization: An overview. *Swarm Intelligence*, 1, 10 2007.

[18] Mujahid Tabassum. A genetic algorithm analysis towards optimization solutions. *International Journal of Digital Information and Wireless Communications*, 4:124–142, 2014.

[19] Mohammed Amine Janati Idrissi, Hassan Ramchoun, Youssef Ghanou, and Mohamed Ettaouil. Genetic algorithm for neural network architecture optimization. In *2016 3rd International Conference on Logistics Operations Management (GOL)*, pages 1–4, 2016.

[20] Karthickumar Ponnambalam, B Sivaneasan, A Sharma, SS Lee, and Prasun Chakrabarti. Battery soh estimation using lstm with genetic algorithm-based hyperparameter optimization. In *TENCON 2024-2024 IEEE Region 10 Conference (TENCON)*, pages 1632–1635. IEEE, 2024.

[21] Rainer Storn. Differrential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical report, International Computer Science Institute*, 11, 1995.

[22] Jianyong Sun, Xin Liu, Thomas Bäck, and Zongben Xu. Learning adaptive differential evolution algorithm from optimization experiences by policy gradient. *IEEE Transactions on Evolutionary Computation*, 25(4):666–680, 2021.

Table 2. Results Comparison

| Test Results | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Algorithm | B0005 | | | B0006 | | | B0018 | | |
| | 30% | 50% | 70% | 30% | 50% | 70% | 30% | 50% | 70% |
| Simple LSTM | 0.2386 | 0.0248 | 0.0061 | 0.2117 | 0.0254 | 0.0161 | 0.0654 | 0.0416 | 0.0142 |
| Simple RNN | 0.1708 | 0.0301 | 0.0165 | 0.1349 | 0.0281 | 0.0089 | 0.0331 | 0.0193 | 0.0166 |
| RVM | 0.0731 | 0.0139 | 0.0141 | 0.1148 | 0.0509 | 0.0391 | 0.0215 | 0.0347 | 0.0147 |
| PA-LSTM | 0.0119 | 0.0110 | 0.0060 | 0.0197 | 0.0159 | 0.0079 | 0.0208 | 0.0152 | 0.0124 |
| GA LSTM | 0.0465 | 0.0184 | 0.0053 | 0.0065 | 0.0080 | 0.0050 | 0.0380 | 0.0559 | 0.0026 |
| DE LSTM | **0.0041** | **0.0017** | **0.0005** | **0.0025** | **0.0015** | **0.0007** | **0.0052** | **0.0017** | **0.0016** |
| Improvement from PA LSTM | 65.20% | 84.60% | 90.90% | 87.40% | 90.40% | 91.20% | 74.80% | 89.10% | 86.90% |
| Improvement from GA LSTM | 91.10% | 90.80% | 89.60% | 61.60% | 80.80% | 86.10% | 86.20% | 97.00% | 38.10% |

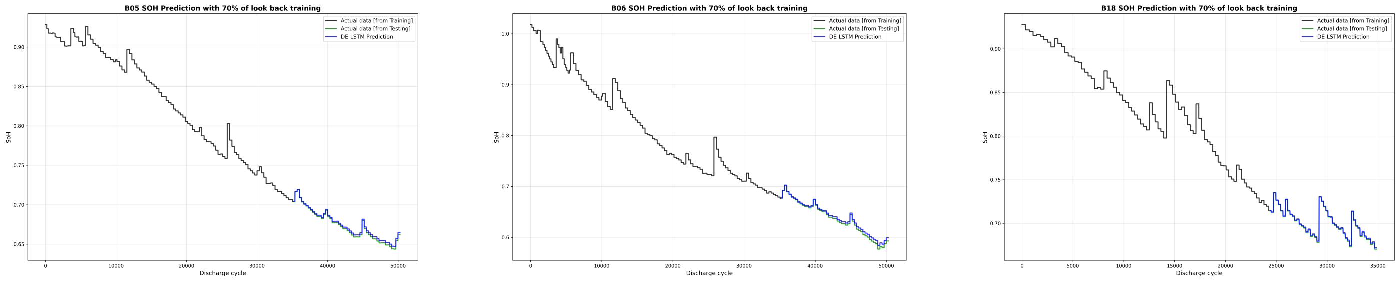


Figure 5. Actual vs Predicted SoH (Training with 70% cycles)


[23] K Vijayaprabakaran and K Sathiyamurthy. Towards activation function search for long short-term model network: A differential evolution based approach. *Journal of King Saud University-Computer and Information Sciences*, 34(6):2637–2650, 2022.

[24] Swagatam Das, Sankha Subhra Mullick, and Ponnuthurai N. Suganthan. Recent advances in differential evolution – an updated survey. *Swarm and Evolutionary Computation*, 27:1–30, 2016.

[25] Swagatam Das and Ponnuthurai N. Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, 2011.

[26] Gary Pampara, Andries P. Engelbrecht, and Nelis Franken. Binary differential evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 1873–1879, 2006.

[27] Lu Peng, Shan Liu, Rui Liu, and Lin Wang. Effective long short-term memory with differential evolution algorithm for electricity price prediction. *Energy*, 162:1301–1314, 2018.

[28] Simone Disabato and Manuel Roveri. Incremental on-device tiny machine learning. pages 7–13, 11 2020.

[29] Atallo Kassaw Takele and Balázs Villányi. Resource aware long short-term memory model (ralstmm) based on-device incremental learning for industrial internet of things. *IEEE Access*, 11:63107–63115, 2023.

[30] Ade Pitra Hermawan, Dong Seong Kim, and Jae Min Lee. Sensor failure recovery using multi look-back lstm algorithm in industrial internet of things. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2020-September:1363–1366, 9 2020.

[31] Nikolaos Gorgolis, Ioannis Hatzilygeroudis, Zoltan Istenes, and Lazlo N.Grad Gyenne. Hyperparameter optimization of lstm network models through genetic algorithm. *10th International Conference on Information, Intelligence, Systems and Applications, IISA 2019*, 7 2019.

[32] D. Whitley, T. Starkweather, and C. Bogart. Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing*, 14:347–361, 8 1990.

[33] Li-ion Battery Aging Datasets — NASA Open Data Portal.

[34] Nikolaos Gorgolis, Ioannis Hatzilygeroudis, Zoltan Istenes, and Lazlo – Grad Gyenne. Hyperparameter optimization of lstm network models through genetic algorithm. In *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–4, 2019.