1 Introduction

The rapid advancements in technology and the growing reliance on portable and efficient energy storage systems have made lithium-ion batteries an indispensable component of modern life. From powering electric vehicles to enabling critical medical devices, lithium-ion batteries are at the forefront of energy innovation. However, their performance and lifespan remain significant challenges, particularly in applications requiring fast charging and long-term reliability. Understanding and predicting the lifespan of lithium-ion batteries is crucial for ensuring safety, optimizing performance, and reducing costs. Numerous researchers are currently exploring charging methods [1, 2, 3, 4, 5, 6, 7] and battery longevity [8, 9, 10, 11, 12, 13, 14, 15, 16] through various approaches with a focus on predicting battery lifespan. These approaches often involve models and extensive data analysis. However, there has been limited exploration into using neural networks to predict battery lifespan in relation to fast charging methods.

In this project, we aim to utilize data-driven methods to predict the lifespan of lithium-ion batteries. By applying deep learning approaches, we seek to develop predictive models that are not only accurate but also less complex than traditional methods. Specifically, we will compare the performance of three types of neural networks: Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and a custom-built neural network. These models will be evaluated based on their ability to handle complex temporal data, their computational efficiency, and their overall predictive accuracy.

Through this project, we aim to bridge the gap between traditional battery lifespan estimation techniques and modern deep learning approaches, ultimately paving the way for innovative solutions in energy storage and management.

2 Battery Lifespan Prediction

Accurately predicting a battery's lifespan and state of health is a complex challenge. The study in [17] proposes a method for estimating the State of Charge (SOC) and State of Health (SOH) of lithium-ion batteries using a dual extended Kalman filter (EKF) technique, which is based on a polynomial battery model. The authors develop a mathematical model that integrates battery dynamics and uses EKF to predict SOC and SOH in real-time. By using the polynomial battery model, the system can effectively handle non-linearities in battery behavior, providing more accurate predictions of the battery's remaining capacity and overall health. Another group of researchers [18] proposed a method which employs a combination of Kalman Filters (KF) and Neural Networks (NNs) to achieve accurate predictions. While KF is used for real-time tracking of the battery's SOC, NN trained with historical data, enhance the accuracy of SOH estimation. By incorporating temperature effects and combining different estimation techniques, the method improves overall prediction reliability.

A common theme in the studies mentioned is the reliance on existing datasets for a specific type of battery to predict lifespan. In this research, entirely new fast-charging profiles have been created, making the prediction of the SOH more challenging when using traditional methods. Utilizing deep learning to train on existing data and then applying it to predict the lifespan of a different battery type, or using transfer learning to retrain with a smaller dataset, could offer a new approach to battery lifespan prediction.

2.1 Data Preprocessing

2.1.1 Data analysist

Data preprocessing is an extremely important task before using data to train various Deep Learning models. In article [19], the author uses a dataset provided by Toyota-MIT. The dataset contains 124 fast charging profiles of lithium-ion phosphate (LFP)/graphite cells, manufactured by A123 Systems (APR18650M1A). At the time of this project, Toyota-MIT had released an additional 45 battery profiles, bringing the total number of profiles in the dataset to 167.

Figure 2.1 displays the dataset sorted by battery lifespan, measured in cycles. The distribution of battery cycle life is not uniform. Only a small number of batteries have a cycle life exceeding 1,200 cycles, with the majority concentrated in the 250 to 750 cycle range. The 750 to 900 cycle range and 1000 to 1250 cycles are also noteworthy due to the very low number of

batteries within it. This imbalance poses a significant challenge for training the model, as the abundance of batteries with fewer cycles tends to dominate the dataset, making accurate predictions within the 750 to 900 cycle range and 1000 to 1250 extremely difficult. An uneven distribution of battery cycle life in the dataset may pose significant challenges for the deep learning model's performance. Models trained on such imbalanced data might struggle to generalize well, particularly for batteries at the extremes of the cycle life spectrum.

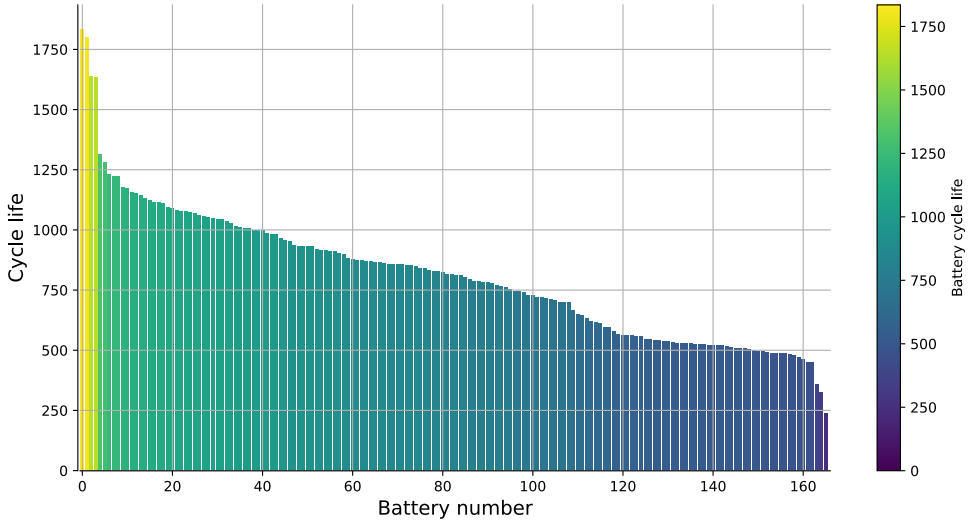


Figure 2.1: Dataset arranged according to battery lifespan.

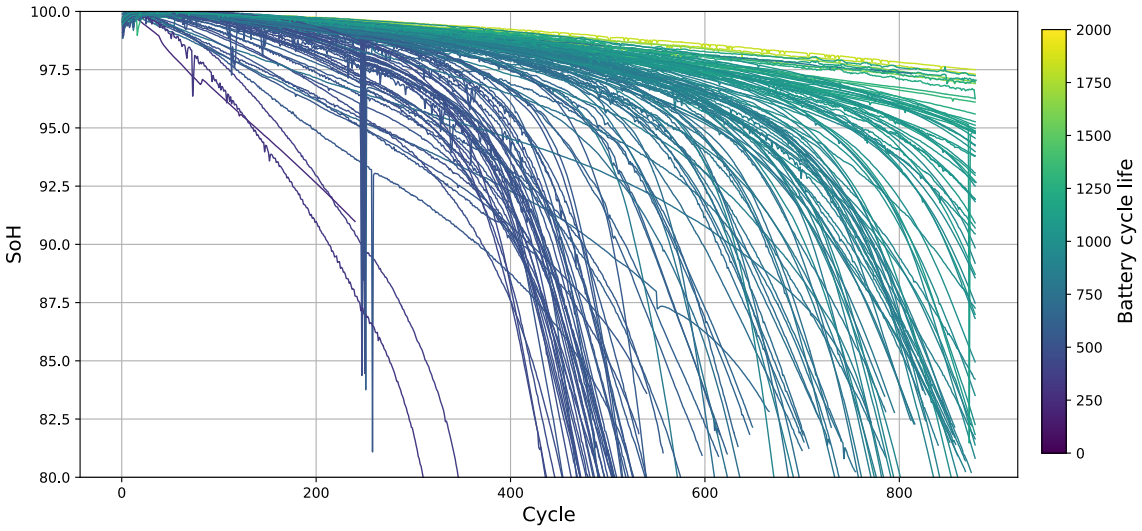


Figure 2.2: SOH over the number of cycles for the entire set of batteries.

The figure 2.2 illustrates the SOH versus the number of cycles for a representative set of batteries from the dataset. Each curve represents the degradation pattern of an individual battery over its lifecycle. The color gradient from dark blue to bright yellow indicates the battery cycle life, with darker shades representing shorter cycle lives and lighter shades corresponding to longer ones. As depicted in the figure, most of the curves follow a generally smooth downward trend, reflecting the typical degradation process of batteries. Nonetheless, several curves exhibit irregularities, such as sharp drops or fluctuations, deviating from the expected smooth decline. These anomalies may indicate sudden capacity loss or measurement errors and are crucial to consider when using this dataset for training deep learning models. Such irregularities could introduce noise into the model, potentially affecting its ability to learn and predict battery behavior accurately. In this project, no attempt will be made to smooth the curve, as the goal is to observe how the deep learning model can handle such errors. Furthermore, the dataset provided by Toyota-MIT contains many failed batteries that have already aged (i.e., they do not reach 80% capacity after a full charge). Therefore, the final dataset that will be used for further processing in this project consists of 164 batteries.

2.1.2 Feature Selection Procedure

In [19], the authors examine a collection of features derived from the Toyota-MIT dataset, which are utilized as input variables in a linear regression model to estimate the Remaining Useful Life (RUL) of batteries at cycle 100. Their method entails extracting a single value for each feature based on the data recorded from cycles 10 to 100 for all batteries. In [20], the author suggested using the 10th cycle as the reference, but instead of using 100 cycles, they used 200 cycles for better performance. Furthermore, their study also indicates that the best performance voltage range for prediction is between 2.0 V and 3.0 V. In this work, this range will be adopted for further analysis.

The figure 2.3 illustrates the discharge capacity curves of a battery across various cycles of its lifespan in the range of voltage from 2.0V to 3.0V. Each curve corresponds to a different cycle, showcasing how the battery's ability to hold charge diminishes over time.

The reduction of each curve on the vertical axis during the cycle shows how the discharge capacity changes as a function of voltage throughout the cycle. It illustrates the overall progression of the battery's aging process. To evaluate the degradation in battery performance, "the discharge capacity curve from cycle 10 is chosen as the baseline or reference"[20]. The choice of cycle 10 as the reference cycle is because the battery needs to be charged for several cy-

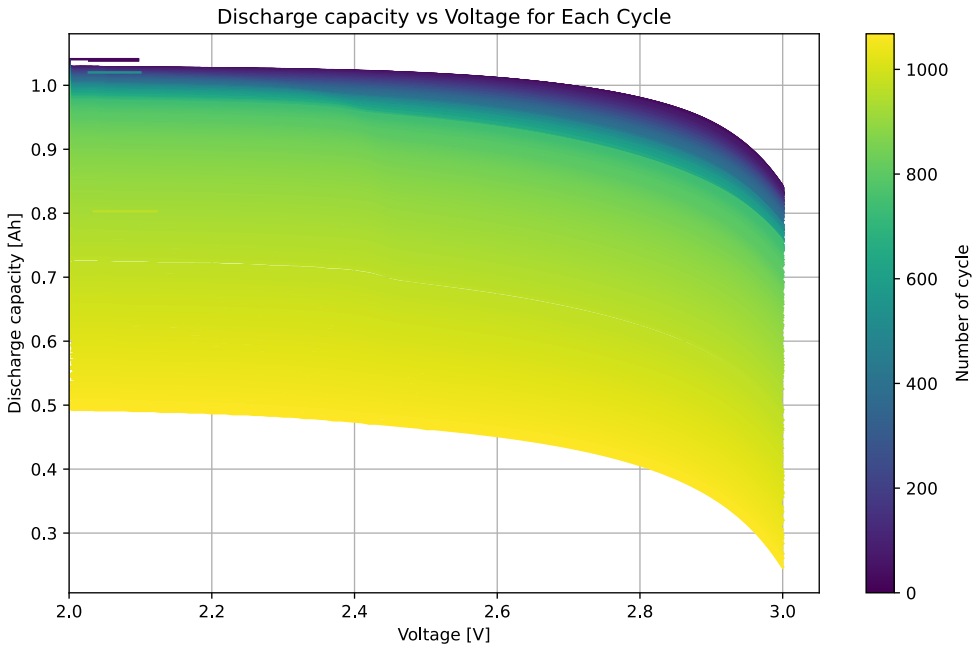


Figure 2.3: The discharging capacity across all cycles for a sample battery is shown as a function of voltage, ranging from 2.0V to 3.0V.

cles [21] to reach its full capacity. Subsequent curves, representing cycles after the 10th, are compared against this reference. Specifically, the difference in the area under the curve between the 10th cycle and any given later cycle provides a measure of the decline in the battery's performance. This difference indicates how much the battery's energy output has decreased relative to its performance at the 10th cycle, giving insight into the battery's aging process and remaining useful life.

The figure 2.4 illustrates the variation in discharge capacity ΔQ_{200} as a function of voltage. The curve shifting upward and to the right indicates a reduction in the battery's capacity retention ability with each cycle.

Figure 2.5 depicts the variation in discharge capacity curves between the reference cycle 10 and cycle 200 for all batteries. It is more clear that the battery with the faster charger profile shows a significant decrease in capacity over its cycle life compared to the others.

Feature 1 (Ftr1) is derived by calculating the common logarithm of the variance found in the difference curves. In a similar manner, Feature 2 (Ftr2) and Feature 3 (Ftr3) are obtained by applying the common logarithm to the maximum and minimum values of the difference curves, respectively. These features are computed for each battery and for every cycle, with the ex-

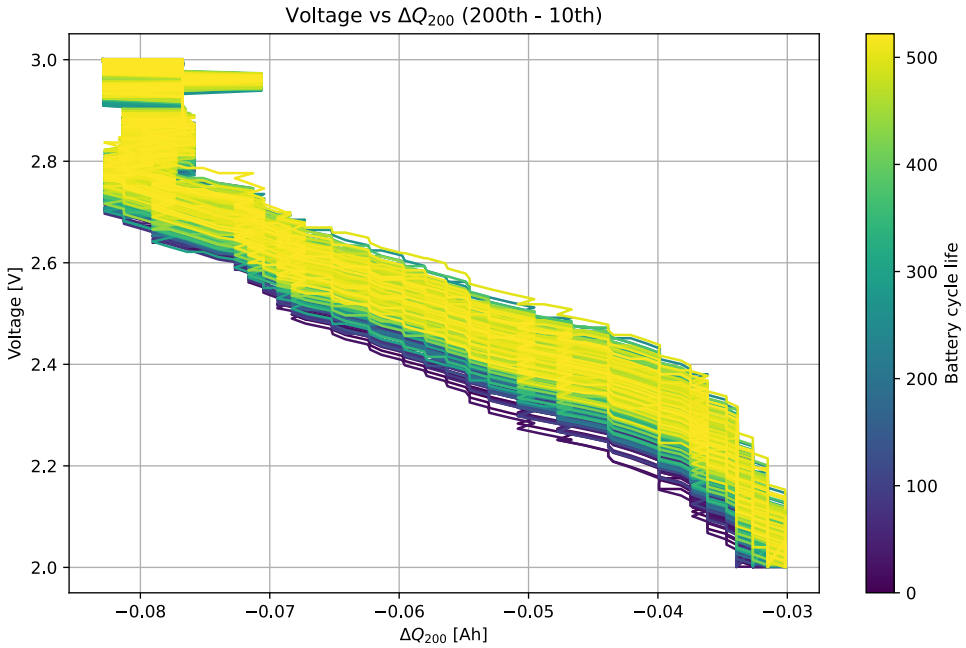


Figure 2.4: Discharging capacity as a function of voltage for single battery from the range 2.0V to 3.0V

ception of cycles 1 through 10, using the 11th cycle as the reference point. The resulting feature space for both Ftr1, Ftr2 and Ftr3 exhibits an almost linear distribution over the battery's lifespan, as illustrated for cycle 200 of all batteries in Figure 2.6. The final feature (Ftr4) is calculated as the common logarithm of the sum of the average temperatures across all cycles.

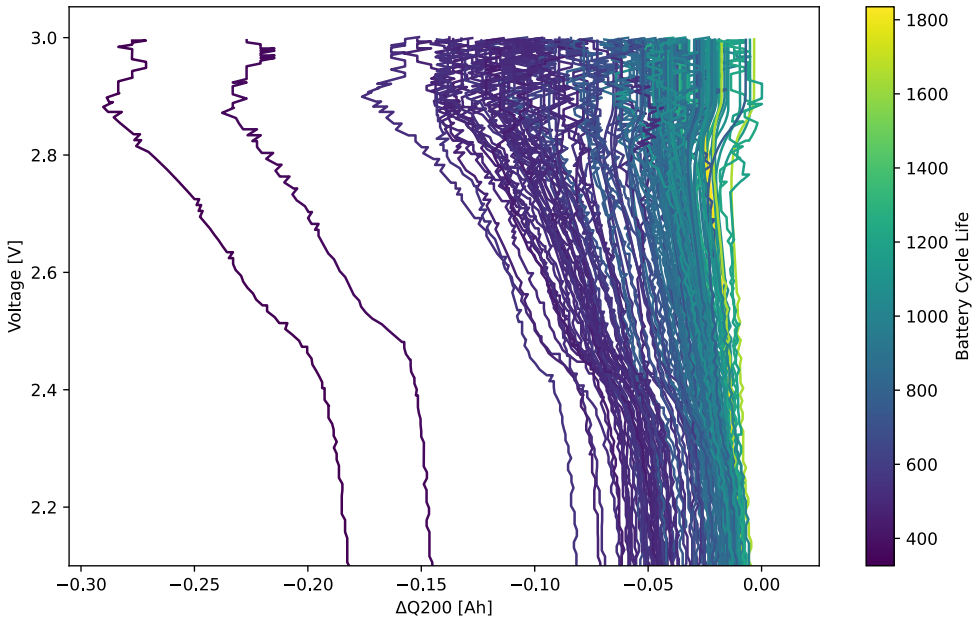


Figure 2.5: Variation in discharge capacity curves between the reference cycle 10 and cycle 200 for all batteries, within the voltage range of 2.0V to 3.0V.

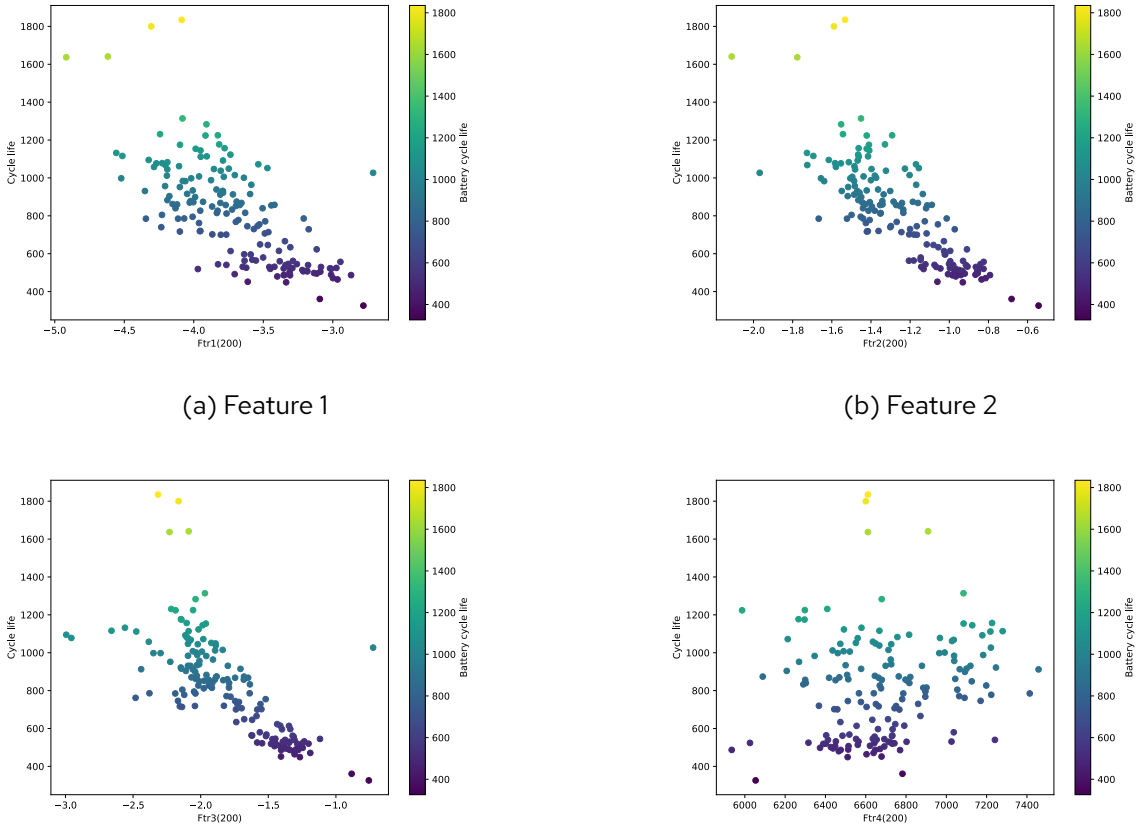
$$
\begin{cases}
Ftr1(k) = \log\left(\left|\dfrac{1}{p-1}\displaystyle\sum_{i=1}^{p}\left(\Delta Q_{k,i}(V) - \overline{\Delta Q_k(V)}\right)^2\right|\right) \\
Ftr2(k) = \log\left(|\min(\Delta Q_k(V))|\right) \\
Ftr3(k) = \log\left(|\max(\Delta Q_k(V))|\right) \\
Ftr4(k) = \log\left(\displaystyle\sum_{cycle=1}^{k}\overline{T_{cycle}}\right) \\
\Delta Q_k(V) = Q_k(V) - Q_{10}(V) \\
\overline{\Delta Q_k(V)} = \dfrac{1}{p_c}\displaystyle\sum_{i=11}^{p_c}\Delta Q_{k,i}(V)
\end{cases}
\tag{2.1}
$$

where:

- $Q_{10}(V)$: the discharge capacity at cycle 10th as a function of voltage
- $Q_k(V)$: the discharge capacity at cycle k as a function of the voltage.
- $\Delta Q_k(V)$: is the difference between $Q_{10}(V)$ and $Q_k(V)$ for cycle k.
- $\overline{\Delta Q}_k(V)$: represents the average value of $\Delta Q_k(V)$ for cycle k.

- p_c: Indicates the number of data points in the curve.
- T_{cycle}: represents an average temperature in cycle i-th.

Figure 2.6 show the distribution of Ftr1, Ftr2, Ftr3 and Ftr4 at cycle 200 for all batteries



(a) Feature 1

(b) Feature 2

(c) Feature 3

(d) Feature 4

Figure 2.6: The distribution of 4 features in cycle 200 for all batteries

2.2 Custom-Built Dense Neural Network.

2.2.1 Dense Neural Network Model

A Dense Neural Network (DNN), also known as a fully connected network, is "an artificial neural network where "each neuron in one layer connects to every neuron in the next" [22]. This structure enables the network to capture "complex patterns by fine-tuning the weights between neurons throughout the training process" [23]. DNNs consist of an input layer, one or more hidden layers, and an output layer which is shown in Figure 2.7.

DNNs is flexible [24, 25] and has the ability to model complex relationships [26, 27, 28, 29] within sequential data. Unlike traditional models, DNNs can capture non-linear patterns and interactions between time-dependent variables [30], making them suitable for handling the intricate dynamics of time series data. By stacking multiple layers, DNNs can learn hierarchical features, improving their predictive accuracy. Additionally, DNNs are adaptable to various types of time series [31, 32, 33, 34], whether univariate or multivariate, and can integrate external variables, enhancing forecasting performance in diverse applications. Several researchers have demonstrated the remarkable power of DNNs in forecasting time series data, highlighting their effectiveness in capturing complex patterns and improving prediction accuracy across various domains. The document [35] explores the application of an ensemble of Mixture Density Neural Networks (MDNNs) for short-term wind speed and power forecasting. By leveraging MDNNs, the model captures uncertainties in predictions, offering probabilistic forecasts. Another group of researchers [36] applied DNNs for forecasting stock prices and demonstrated remarkable results.

The figure 2.8 shows the function of DNNs. Every neuron in one layer is "connected to every neuron in the subsequent layer" [39]. The process of solving optimization problem using DNNs involves several key steps [40, 41, 42], including input processing, matrix multiplication, bias addition, and the application of an activation function. First, consider an input vector **IN** with three elements:

$$\mathbf{IN} = \begin{bmatrix} \text{IN1} \\ \text{IN2} \\ \text{IN3} \end{bmatrix} \tag{2.2}$$

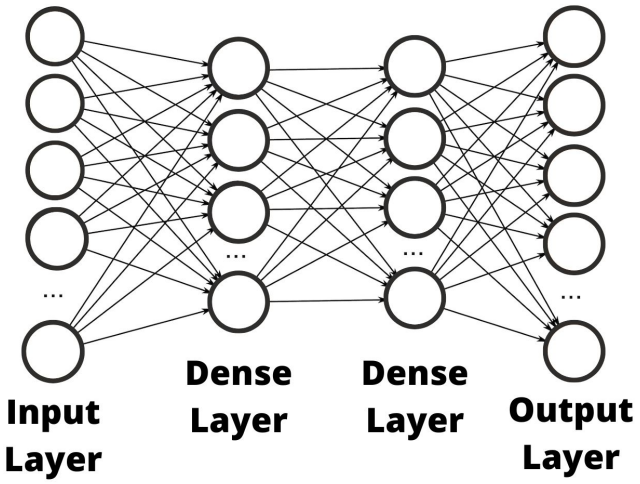


Figure 2.7: Dense Layers [37]

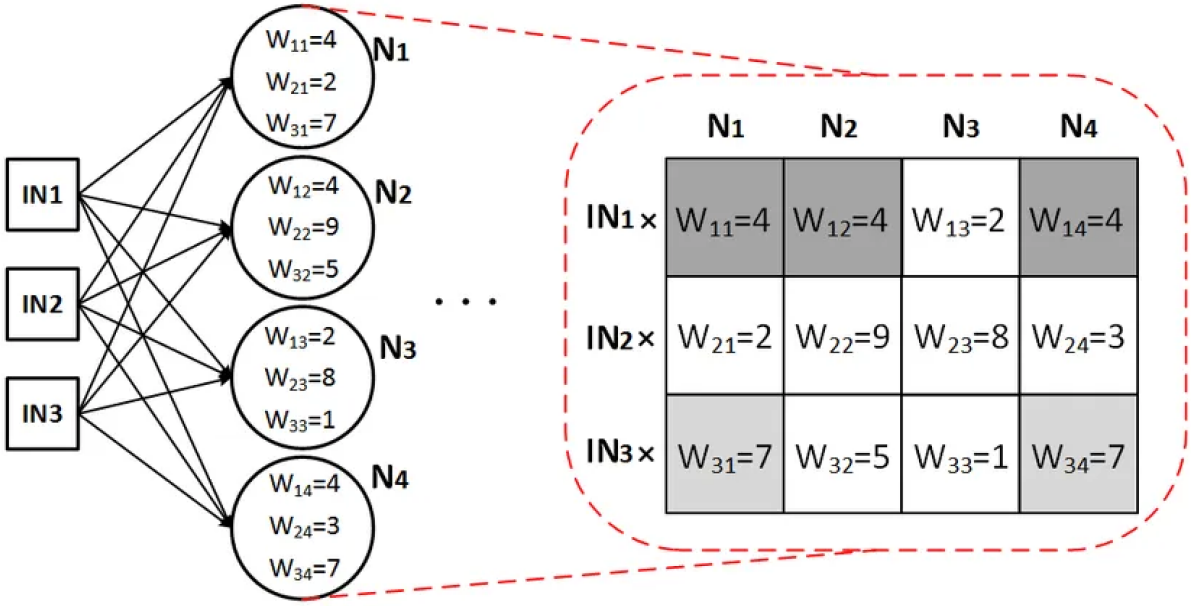


Figure 2.8: Dense Layers [38]

and a weight matrix W [43] that connects the input layer to a hidden layer with four neurons:

$$W = \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \end{bmatrix} \tag{2.3}$$

Each neuron in the hidden layer also has an associated bias. The bias vector $\mathbf{b_b}$ [44] is given as:

$$\mathbf{b_b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \tag{2.4}$$

The first step in processing the input through the dense layer is to perform a matrix multiplication between the input vector **IN** and the weight matrix W. This operation computes the weighted sum of the inputs for each neuron in the hidden layer:

$$\mathbf{Z} = \mathbf{IN} \times W = \begin{bmatrix} \text{IN1} & \text{IN2} & \text{IN3} \end{bmatrix} \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \end{bmatrix} \tag{2.5}$$

This results in a vector **Z** that represents the linear combination of inputs and weights before applying the bias and activation:

$$\mathbf{Z} = \begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \end{bmatrix} = \begin{bmatrix} \text{IN1} \times W_{11} + \text{IN2} \times W_{21} + \text{IN3} \times W_{31} \\ \text{IN1} \times W_{12} + \text{IN2} \times W_{22} + \text{IN3} \times W_{32} \\ \text{IN1} \times W_{13} + \text{IN2} \times W_{23} + \text{IN3} \times W_{33} \\ \text{IN1} \times W_{14} + \text{IN2} \times W_{24} + \text{IN3} \times W_{34} \end{bmatrix} \tag{2.6}$$

Next, the bias vector $\mathbf{b_b}$ is added to the result of the matrix multiplication. This step adjusts the weighted sum by shifting it before applying the activation function:

$$\mathbf{Z} = \mathbf{Z} + \mathbf{b_b} = \begin{bmatrix} Z_1 + b_1 \\ Z_2 + b_2 \\ Z_3 + b_3 \\ Z_4 + b_4 \end{bmatrix} \tag{2.7}$$

Finally, an activation function $\sigma(\cdot)$ is applied to each element of the vector **Z**. The activation function introduces non-linearity into the network, allowing it to solve complex problems:

$$\mathbf{N} = \sigma(\mathbf{Z}) = \begin{bmatrix} \sigma(Z_1 + b_1) \\ \sigma(Z_2 + b_2) \\ \sigma(Z_3 + b_3) \\ \sigma(Z_4 + b_4) \end{bmatrix} \tag{2.8}$$

Common activation functions include the sigmoid function, ReLU (Rectified Linear Unit), eLU (Exponential Linear Unit), and Tanh (Hyperbolic Tangent), with the choice depending on the

problem the network is intended to address.

The resulting vector **N** represents the output of the dense layer:

$$\mathbf{N} = \begin{bmatrix} N1 \\ N2 \\ N3 \\ N4 \end{bmatrix} = \begin{bmatrix} \sigma(\text{IN1} \times W_{11} + \text{IN2} \times W_{21} + \text{IN3} \times W_{31} + b_1) \\ \sigma(\text{IN1} \times W_{12} + \text{IN2} \times W_{22} + \text{IN3} \times W_{32} + b_2) \\ \sigma(\text{IN1} \times W_{13} + \text{IN2} \times W_{23} + \text{IN3} \times W_{33} + b_3) \\ \sigma(\text{IN1} \times W_{14} + \text{IN2} \times W_{24} + \text{IN3} \times W_{34} + b_4) \end{bmatrix} \tag{2.9}$$

The DNN solves problems by learning the appropriate weights W_{ij} and biases b_j during the training process. Initially, the weights and biases are randomly initialized. The input data is passed through the dense layers, generating an output.

During the training phase, the network adjusts the weights and biases to minimize the discrepancy between the predicted output and the actual target value, based on a loss function [45]. This adjustment is performed through backpropagation, where the gradients of the loss function with respect to each weight and bias are calculated [46]. The parameters are then updated using optimization algorithms, such as gradient descent.

As the network iteratively adjusts these parameters, it gradually improves its accuracy in predicting the correct output. Over time, the DNN learns to map input data to the desired output, effectively solving the problem it was trained on.

2.2.2 Build The Architecture of Model 1

The model's input includes four features (mentioned above), and its output is the predicted SOH. The model comprises five layers, each contributing uniquely to the network's capacity to learn and generalize from data. At each time step k, four input features Ftr1(k), Ftr2(k), Ftr3(k), and Ftr4(k) are collected and flattened into a single input vector. This vector is passed through a series of fully connected dense layers with 256, 128, and 64 units, respectively. The final output layer comprises a single neuron that produces the estimated SOH. By stacking multiple dense layers, the model can learn progressively more abstract representations of the input features, ultimately improving the accuracy of the SOH prediction. To enhance the network's performance, the following hyperparameters are selected:

- **Activation Functions:** The Exponential Linear Unit (ELU) activation function is employed in all dense layers except the output layer. ELU is advantageous over other activation functions like ReLU because it has a negative saturation region, which can help the network learn representations more robustly and prevent dead neurons during training. ELU also helps the model to converge faster and improve generalization.

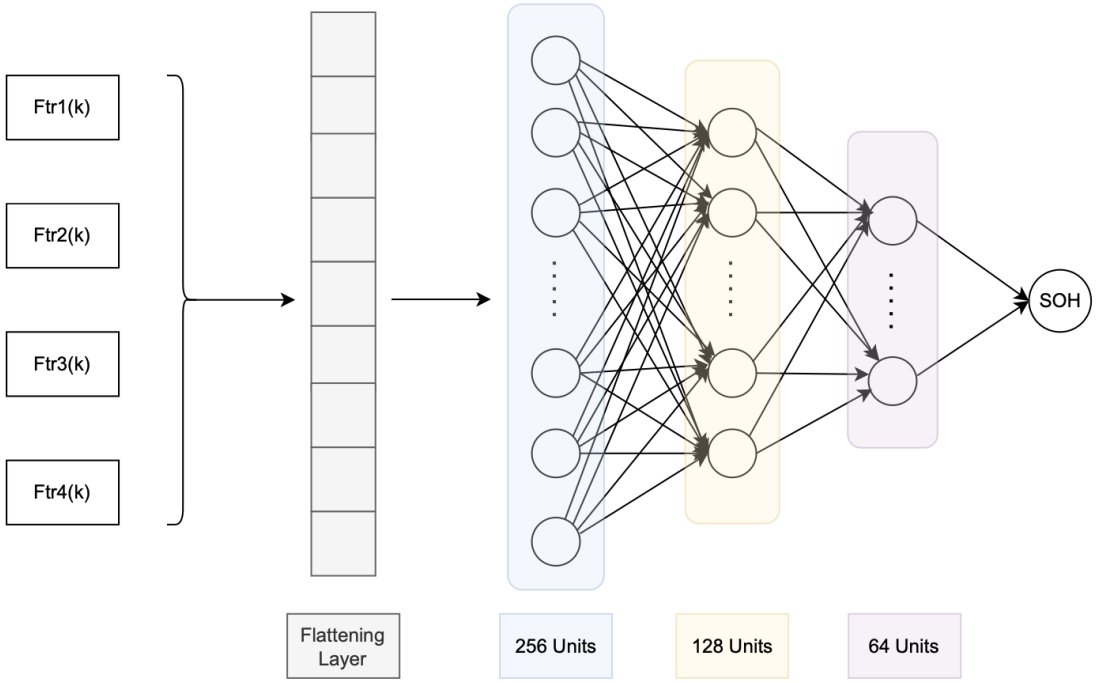


Figure 2.9: The Architecture of Model 1

- **Regularization:** L2 regularization is applied to each dense layer with a coefficient of 0.0001. This regularization method introduces "a penalty to the loss function that is proportional to the square of the weights' magnitudes" [47]. The primary goal of L2 regularization is to prevent overfitting by discouraging the network from relying too heavily on any single feature or set of features, leading to better generalization on unseen data.

- **Model Compilation and Training:** The model is compiled with the `adam` optimizer, which is an adaptive learning rate optimization algorithm. Adam is widely used due to its computational efficiency and ability to perform well on a wide range of problems without requiring much hyperparameter tuning.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.10}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.11}$$

y_i denotes the true value of the i-th data point.

$\hat{y}_i$ represents the corresponding predicted value for that data point.

- **The loss function:** Mean Squared Error (MSE) is used as the loss function, which is widely applied in regression tasks. MSE computes the average of the squared deviations between the predicted and actual values, giving more weight to larger discrepancies. The model's performance is further assessed using Mean Absolute Error (MAE), which computes "the average of the absolute differences between predictions and actual values" [48]. Unlike MSE, MAE is more interpretable, as it represents the average error in the same units as the target variable.

The model is generated to receive the input of 200 cycles (with 4 features) to predict the SOH of 500 cycles (start from cycle 11 to 511). The model is trained for 500 epochs with a batch size of 1. A batch size of 1 implies that the model updates its weights after every single training example, which is commonly referred to as stochastic gradient descent. This approach can lead to more frequent updates and potentially faster convergence. The validation data (*X_test*, *Y_test*) is used to monitor the model's performance on unseen data during training. This neural network model is designed for a regression task, utilizing a combination of dense layers, ELU activation, L2 regularization, and the Adam optimizer. The progressive reduction in the number of units across layers reflects a common strategy to reduce model complexity as it approaches the output layer, aiding in generalization. The choice of hyperparameters, such as the regularization coefficient, batch size, and number of epochs, is critical to the model's performance and its ability to generalize to new data.

3 Result and Comparation

The dataset, consisting of 167 batteries, was manually split into training and test sets with a 90:10 ratio to ensure a balanced and representative evaluation. The test set was selected and fixed, comprising a total of 16 datasets. These datasets were chosen to represent a diverse range of batteries, spanning from those with low cycle counts to those with high cycle counts. The details of the test set are provided in Table 3.1.

The training process and results is illustrated in Figures 3.1.

Label	Cycle	Label	Cycle	Label	Cycle
B1c41	1049	B2c26	498	B9c2	785
B2c4	471	B2c27	495	B9c40	1123
B2c11	505	B2c40	521	B9c41	998
B2c13	520	B3c11	816	B9c44	1078
B2c14	509	B3c13	856		
B2c18	618	B3c36	922		

Table 3.1: Cycle data for various labels

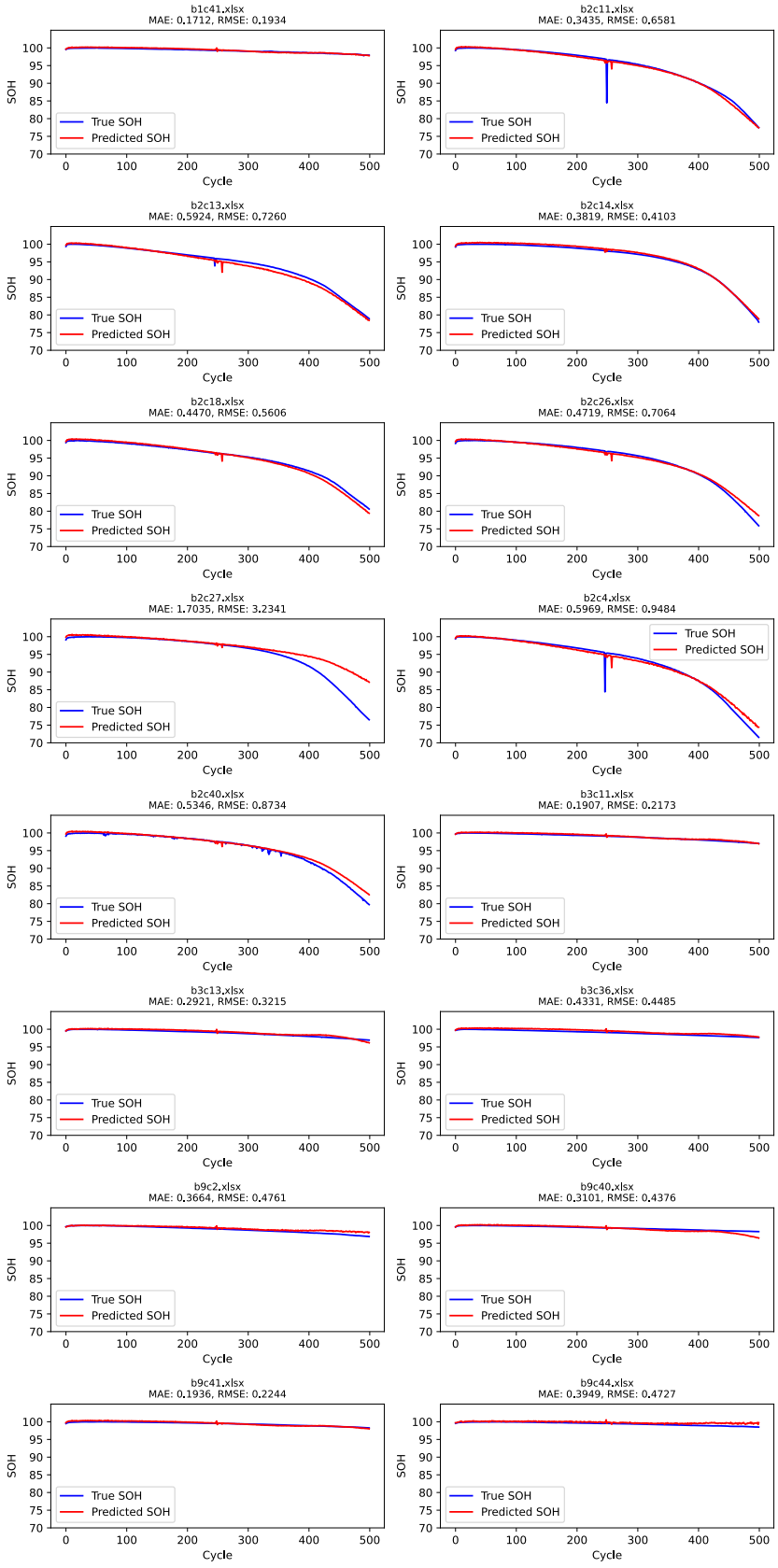


Figure 3.1: The result of model 1

References

[1] G. Dong, Z. Zhu, Y. Lou, J. Yu, L. Wu, and J. Wei, "Optimal charging of lithium-ion battery using distributionally robust model predictive control with wasserstein metric," *IEEE Transactions on Industrial Informatics*, 2024.

[2] H. E. Perez, X. Hu, S. Dey, and S. J. Moura, "Optimal charging of li-ion batteries with coupled electro-thermal-aging dynamics," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 9, pp. 7761–7770, 2017.

[3] J. Du and Y. Sun, "The influence of high power charging on the lithium battery based on constant and pulse current charging strategies," in *2020 IEEE Vehicle Power and Propulsion Conference (VPPC)*, 2020, pp. 1–7.

[4] A. B. Malla and H. Myneni, "Analysis of different charging methods of batteries for ev applications with charge equalization," in *2023 IEEE IAS Global Conference on Renewable Energy and Hydrogen Technologies (GlobConHT)*, 2023, pp. 1–6.

[5] M. Gaglani, R. Rai, and S. Das, "Implementation of multilevel battery charging scheme for lithium-ion batteries," in *2019 National Power Electronics Conference (NPEC)*, 2019, pp. 1–6.

[6] A. R. Zarif, A. Kazemi, and Y. Mafinejad, "A novel li-ion battery charge management system for applying in renewable energies based on pulse charge method," in *2024 9th International Conference on Technology and Energy Management (ICTEM)*, 2024, pp. 1–6.

[7] T. Gao, R. Liu, and K. Hua, "Dispatching strategy optimization for orderly charging and discharging of electric vehicle battery charging and swapping station," in *2015 5th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies (DRPT)*, 2015, pp. 2640–2645.

[8] K. H. Chen and Z. D. Ding, "Lithium-ion battery lifespan estimation for hybrid electric vehicle," in *The 27th Chinese Control and Decision Conference (2015 CCDC)*, 2015, pp. 5602–5605.

[9] J. Wang, P. Liu, J. Hicks-Garner, E. Sherman, S. Soukiazian, M. Verbrugge, H. Tataria, J. Musser, and P. Finamore, "Cycle-life model for graphite-lifepo4 cells," *Journal*

of Power Sources, vol. 196, no. 8, pp. 3942–3948, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378775310021269

[10] T. Imai and H. Yamaguchi, "Dynamic battery charging voltage optimization for the longer battery runtime and lifespan," in *2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE)*, 2013, pp. 219–223.

[11] D. Lipko, I. Yamnenko, A. Manzhelii, and O. Bondarenko, "Possibilities and challenges of partially using a charge-discharge cycle of battery to increase its resource," in *2023 IEEE 4th KhPI Week on Advanced Technology (KhPIWeek)*, 2023, pp. 1–5.

[12] A. Kruekaew, P. Khemchan, S. Waengdeesorn, A. Sirisawat, A. Norasarn, and S. Khunchai, "Residential energy management system to reduce electricity costs considering the lifespan of energy storage and electric vehicle batteries," in *2023 7th International Conference on Information Technology (InCIT)*, 2023, pp. 220–225.

[13] B. Couraud, S. Norbu, M. Andoni, V. Robu, H. Gharavi, and D. Flynn, "Optimal residential battery scheduling with asset lifespan consideration," in *2020 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*, 2020, pp. 630–634.

[14] M. Nassary, M. Orabi, M. Ghoneima, A. El Aroudi, and M. K. El-Nemr, "A high current ripple ev battery charger utilizing capacitor-less cuk converter," in *2019 IEEE Conference on Power Electronics and Renewable Energy (CPERE)*, 2019, pp. 232–237.

[15] H. D. Misalkar, A. W. Burange, and U. V. Nikam, "Increasing lifespan and achieving energy efficiency of wireless sensor network," in *2016 International Conference on Information Communication and Embedded Systems (ICICES)*, 2016, pp. 1–5.

[16] P. Wongdet and B. Marungsri, "Hybrid energy storage system in standalone dc microgrid with ramp rate limitation for extending the lifespan of battery," in *2018 International Electrical Engineering Congress (iEECON)*, 2018, pp. 1–4.

[17] N. A. Azis, E. Joelianto, and A. Widyotriatmo, "State of charge (soc) and state of health (soh) estimation of lithium-ion battery using dual extended kalman filter based on polynomial battery model," in *2019 6th International Conference on Instrumentation, Control, and Automation (ICA)*, 2019, pp. 88–93.

[18] A. Sedighfar and M. R. Moniri, "Battery state of charge and state of health estimation for vrla batteries using kalman filter and neural networks," in *2018 5th International Conference on Electrical and Electronic Engineering (ICEEE)*, 2018, pp. 41–46.

[19] A. Singh, C. Feltner, J. Peck, and K. I. Kuhn, "Data driven prediction of battery cycle life before capacity degradation," *arXiv preprint arXiv:2110.09687*, 2021.

[20] E. Petkovski, I. Marri, L. Cristaldi, and M. Faifer, "State of health estimation procedure for lithium-ion batteries using partial discharge data and support vector regression," *Energies*, vol. 17, p. 206, 12 2023.

[21] M. A. Hoque, M. Siekkinen, J. Koo, and S. Tarkoma, "Full charge capacity and charging diagnosis of smartphone batteries," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3042–3055, 2017.

[22] I. to Artificial Intelligence, "Mlp - multi-layer perceptron," https://www.introtoartificialintelligence.com/MLP, 2024, accessed: 20-Sep-2024.

[23] F. Library, "Quantum machine learning: Dense layer," https://library.fiveable.me/key-terms/quantum-machine-learning/dense-layer, 2024, accessed: 20-Sep-2024.

[24] W. Sun, A. Zhou, S. Stuijk, R. Wijnhoven, A. O. Nelson, h. Li, and H. Corporaal, "Dominosearch: Find layer-wise fine-grained n:m sparse schemes from dense neural networks," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 20 721–20 732. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/ad68473a64305626a27c32a5408552d7-Paper.pdf

[25] Z. Luo, Z. Sun, W. Zhou, Z. Wu, and S. ichiro Kamata, "Constructing infinite deep neural networks with flexible expressiveness while training," *Neurocomputing*, vol. 487, pp. 257–268, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221016842

[26] R. Saleem, B. Yuan, F. Kurugollu, A. Anjum, and L. Liu, "Explaining deep neural networks: A survey on the global interpretation methods," *Neurocomputing*, vol. 513, pp. 165–180, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222012218

[27] EITCA Academy. (2024) What are the advantages of using deep neural networks over linear models for complex datasets? Accessed: 2024-09-21. [Online]. Available: https://eitca.org/artificial-intelligence/eitc-ai-gcml-google-cloud-machine-learning/first-steps-in-machine-learning/deep-neural-networks-and-estimators/examination-review-deep-neural-networks-and-estimators/what-are-the-advantages-of-using-deep-neural-networks-over-linear-models-for-complex-datasets/

[28] J. Jo, B. Kwak, B. Lee, and S. Yoon, "Flexible dual-branched message-passing neural network for a molecular property prediction," *ACS Omega*, vol. 7, no. 5, pp. 4234–4244, Jan 27 2022.

[29] S. Ha and H. Jeong, "Unraveling hidden interactions in complex systems with deep learning," *Scientific Reports*, vol. 11, no. 1, p. 12804, 2021. [Online]. Available: https://doi.org/10.1038/s41598-021-91878-w

[30] S. Vijay and T. S. Gujral, "Non-linear deep neural network for rapid and accurate prediction of phenotypic responses to kinase inhibitors," *iScience*, vol. 23, no. 5, p. 101129, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S258900422030314X

[31] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3126908.3126964

[32] Y. Liu, Y. Wang, X. Yang, and L. Zhang, "Short-term travel time prediction by deep learning: A comparison of different lstm-dnn models," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–8.

[33] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at microsoft," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 8604–8608.

[34] J. Wei, Y. Zhang, Z. Zhou, Z. Li, and M. A. Al Faruque, "Leaky dnn: Stealing deep-learning model secret with gpu context-switching side-channel," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2020, pp. 125–137.

[35] Z. Men, E. Yee, F.-S. Lien, D. Wen, and Y. Chen, "Short-term wind speed and power forecasting using an ensemble of mixture density neural networks," *Renewable Energy*, vol. 87, pp. 203–211, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148115303645

[36] S. Jain, R. Gupta, and A. A. Moghe, "Stock price prediction on daily stock data using deep neural networks," in *2018 International conference on advanced computation and telecommunication (ICACAT)*. IEEE, 2018, pp. 1–13.

[37] PySource, "Flatten and dense layers | computer vision with keras p.6," Oct. 2022, accessed: 2024-08-13. [Online]. Available: https://pysource.com/2022/10/07/flatten-and-dense-layers-computer-vision-with-keras-p-6/

[38] Coinmonks, "The mathematics of neural network," 2024, accessed: 2024-08-13. [Online]. Available: https://medium.com/coinmonks/the-mathematics-of-neural-network-60a112dd3e05

[39] DeepAI. (2019) Feed forward neural network definition. Accessed: 2024-09-21. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network

[40] T. Zhao, X. Pan, M. Chen, and S. H. Low, "Ensuring dnn solution feasibility for optimization problems with convex constraints and its application to dc optimal power flow problems," 2023. [Online]. Available: https://arxiv.org/abs/2112.08091

[41] J. R. Caldwell, R. A. Watson, C. Thies, and J. D. Knowles, "Deep optimisation: Solving combinatorial optimisation problems using deep neural networks," 2018. [Online]. Available: https://arxiv.org/abs/1811.00784

[42] D. Wu and A. Lisser, "A deep learning approach for solving linear programming problems," *Neurocomputing*, vol. 520, pp. 15–24, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222014412

[43] Wikipedia contributors. (2018) Weighing matrix. Accessed: 2024-09-21. [Online]. Available: https://en.wikipedia.org/wiki/Weighing_matrix

[44] DeepAI. (2018) Bias vector definition. Accessed: 2024-09-21. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/bias-vector

[45] B. I. of Analytics, "Neural networks demystified: A beginner's guide to deep learning," https://bostoninstituteofanalytics.org/blog/neural-networks-demystified-a-beginners-guide-to-deep-learning/, 2024, accessed: 20-Sep-2024.

[46] V. Vendittoli, W. Polini, M. S. J. Walter, and S. Geißelsöder, "Using bayesian regularized artificial neural networks to predict the tensile strength of additively manufactured polylactic acid parts," *Applied Sciences*, vol. 14, no. 8, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/8/3184

[47] T. Schneppat, "L2 regularization (ridge)," https://schneppat.com/l2-regularization_ridge.html, 2024, accessed: 20-Sep-2024.

[48] M. A. Khasawneh and A. Awasthi, "Intelligent meta-heuristic-based optimization of traffic light timing using artificial intelligence techniques," *Electronics*, vol. 12, no. 24, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/24/4968

[49] Wikipedia contributors, "Long short-term memory — wikipedia, the free encyclopedia," 2025, [Online; accessed 1-February-2025]. [Online]. Available: https://en.wikipedia.org/wiki/Long_short-term_memory

[50] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 12 2014.

[51] Wikipedia contributors, "Gated recurrent unit — wikipedia, the free encyclopedia," 2025, [Online; accessed 1-February-2025]. [Online]. Available: https://en.wikipedia.org/wiki/Gated_recurrent_unit

List of Abbreviations

Abbreviation	Significance
BBO	Biogeography Based Optimization
BEC	Battery Equivalent Circuits
CC-CV	Constant Current - Constant Voltage
DNN	Dense Neural Network
EKF	Extended Kalman Filter
ELU	Exponential Linear Unit
emf	Electromotive Force
Ftr1,Ftr2,Ftr3,Ftr4	Feature 1, Feature 2, Feature 3, Feature 4
HPPC	Hybrid Pulse Power Characterization
KF	Kalman Filter
LDWPSO	Linearly Decreasing Inertia Weight
LFP	Lithium-Ion Phosphate
MDNNs	Mixture Density Neural Networks
MAE	Mean Average Error
MSE	Mean Square Error
OCV	Open Circuit Voltage
RMSE	Root Mean Square Error
RLS	Recursive Least Squares
RTLS	Recursive Total Least Squares
RUL	Remaining Useful Life
SOC	State of Charge
SOH	State of Health

List of Symbols

Variable	Significance	Base Unit
$Q_{10}(V)$	Discharge capacity at reference cycle 10 as a function of voltage	Ah
$Q_k(V)$	Discharge capacity at cycle k as a function of voltage	Ah
$\Delta Q_k(V)$	Difference between $Q_{10}(V)$ and $Q_k(V)$ for cycle k	Ah
$\overline{\Delta Q_k(V)}$	Average value of $\Delta Q_k(V)$ for cycle k	Ah
p_c	Number of points in the curve	
IN	Input Vector	
W	Weight Matrix	
b_b	Bias Vector	
Z	Matrix Multiplication	
$\sigma(\cdot)$	Activation function	
N	Nonlinear matrix	
y_i	Actual value of the i-th data point	
$\hat{y}_i$	Predicted value of the i-th data point	
f_t	Forget gate activation vector	
i_t	Input gate activation vector	
$\tilde{C}_t$	Candidate cell state vector	
C_t	Cell state vector at time step t	
C_{t-1}	Cell state vector at time step $t-1$	
o_t	Output gate activation vector	
h_t	Hidden state vector at time step t	
h_{t-1}	Hidden state vector at time step $t-1$	
x_t	Input vector at time step t	
tanh	Hyperbolic tangent activation function	
U	Weight matrices for recurrent connections	
V	Weight matrices for peephole connections (if applicable)	
b_f	Bias vector for the forget gate	
b_i	Bias vector for the input gate	
b_o	Bias vector for the output gate	