LiFe-net: Data-driven Modelling of Time-dependent Temperatures and Charging Statistics Of Tesla's LiFePo4 EV Battery

Anonymous Author(s)
Affiliation
Address
email

Abstract

Modelling the temperature of Electric Vehicle (EV) batteries is a fundamental task of EV manufacturing. Extreme temperatures in the battery packs can affect their longevity and power output. Although theoretical models exist for describing heat transfer in battery packs, they are computationally expensive to simulate. Furthermore, it is difficult to acquire data measurements from within the battery cell. In this work, we propose a data-driven surrogate model (LiFe-net) that uses readily accessible driving diagnostics for battery temperature estimation to overcome these limitations. This model incorporates Neural Operators with a traditional numerical integration scheme to estimate the temperature evolution. Moreover, we propose two further variations of the baseline model: LiFe-net trained with a regulariser and LiFe-net trained with time stability loss. We compared these models in terms of generalization error on test data. The results showed that LiFe-net trained with time stability loss outperforms the other two models and can estimate the temperature evolution on unseen data with a relative error of 2.77 % on average.

1 Introduction

As a result of ongoing attempts to decrease the dependency on fossil fuels, the public interest in electric vehicles (EVs) has been steadily growing for the past decade. Consequently, the need for efficient batteries has also increased. In this work, we examine the Lithium Iron Phosphate ($LiFePO_4$) battery used in the Tesla Model 3. $LiFePO_4$ batteries are a category of Lithium-ion (Li-ion) batteries which are widely used due to their high energy density, small size, and wide operating temperature range [1], [2], [3]. However, they are very sensitive to temperature variations [4]. At high temperatures, Li-ion batteries degrade rapidly, while the low temperatures cause a reduction in power and energy output [5]. Therefore, monitoring temperature evolution is crucial for EV manufacturers. Theoretically, heat transfer inside Li-ion batteries can be described by Equation 1 [1],[6].

$$mc_{cell}\frac{\mathrm{d}T_{cell}}{\mathrm{d}t} = I^2R + T_{cell}\Delta S\frac{I}{nF} + Ah(T_{cell} - T_{amb}), \tag{1}$$

The first part represents the heat generated by the battery cell's internal resistance. The second term denotes the entropy change during discharge. Finally, the last term is the heat transferred to the ambience by convection [1]. Traditional numerical methods for solving such problems require a new simulation for every new instance of the ODE and hence are computationally expensive. Therefore,

Submitted to 35th DLDE Workshop in the Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

there is a potential for alternative data-driven models that are fast, flexible, and robust in predicting battery temperature. Motivated by this potential, we make following contributions in this work:

1. We propose a data-driven approach for learning time and environmental condition-dependent neural operator that predict the temperature evolution of a $LiFePO_4$ EV battery.
2. We introduce regularisation strategies to improve the robustness of the neural operator [7].
3. The connection of neural operators with a surrogate model allows for in-situ prediction of charging statistics.
4. Good generalisation is verified in terms of unseen driving data.

2 Methodology

The experimental setup for acquiring the battery temperature data is shown in Fig.3 in Appendix A. The temperature probe of an Inkbird IBS-TH1 was installed on the floor of the cabin of a Tesla Model 3 with a CATL $LiFePo_4$ battery under the driver's seat. This setup also offers insights into the electrochemical system such as maximum charging power. We also collected different driving diagnostics from Tesla API, including the car's speed, power consumption, outside temperature, and battery level.

2.1 Neural Operator

We model the baseline model of predicting battery temperature evolution in the form of Ordinary Differential Equations (ODEs) as shown in Equation 2 :

$$\begin{aligned} \frac{\mathrm{d}u(t,\mathbf{E})}{\mathrm{d}t} = f(t, u(t,\mathbf{E})) \approx NN_\theta(t,\mathbf{E}), \qquad t \in [0,T] \\ u(t_0,\mathbf{E}_0) = u_0 \end{aligned} \tag{2}$$

where u(t, $\mathbf{E}$) is our hidden solution of temperature, t is relative time (duration of each drive session starting from t_0 = 0 to final time T), $\mathbf{E}$ is the vector of environmental parameters, while u_0 and $\mathbf{E}_0$ are the initial temperature and initial environmental parameters, respectively. We represent the unknown right-hand side $f(t, u(t,\mathbf{E})$ with a neural network NN_θ (multilayer perceptron (MLP)) as shown in Equation 2. It takes relative time and environmental parameters (power, speed, battery level, outside temperature, current battery temperature) as inputs and outputs the estimation of the neural operator at discrete time steps. The neural network is then trained by minimizing the mean squared error (MSE) between the prediction of our network and the ground-truth values for the differential operator calculated by forward differences (FD) as shown in Equation 3. Thus, we learn the neural operator that describes the underlying physics of time-dependent temperature evolution in a supervised manner.

$$\mathcal{L}_{NO} = \sum_{i=0}^{N-1} \left\| \frac{u(t_{i+1},\mathbf{E}_{i+1}) - u(t_i,\mathbf{E}_i)}{t_{i+1} - t_i} - NN_\theta(t_i,\mathbf{E}_i) \right\|_2^2 \tag{3}$$

The trained neural operator is then used in Euler Forward numerical scheme to iteratively estimate the complete temperature evolution during inference with the time-step h as shown in Equation 4:

$$u(t_{i+1},\mathbf{E}_{i+1}) = u(t_i,\mathbf{E}_i) + h * NN_\theta(t_i,\mathbf{E}_i) \qquad for \quad i = 0,\ldots,N-1 \tag{4}$$

2.2 Regularisation

To explore further improvements to the baseline model, we introduce a regularizer $\mathcal{L}_{smooth}$ minimizing the norm of the partial derivative of NN_θ with respect to its input parameters to improve the smoothness of the prediction (see Equation 5).

$$\mathcal{L}_{smooth} = \frac{1}{M} \sum_{j=1}^{M} \left\| \frac{\mathrm{d}NN_\theta(t,\mathbf{E})}{\mathrm{d}E_j} \right\|_2^2 \tag{5}$$

where M is the total number of environmental parameters, E_j is one of the environmental parameters considered. Consequently, the total loss function is $\mathcal{L}_{reg} = \mathcal{L}_{NO} + \lambda \mathcal{L}_{smooth}$ where λ is the weighting factor for the regularisation term. This model will be denoted as regularised LiFe-net.

2.3 Neural Operator with Time Stability Loss

In this subsection we incorporate the temperature evolution into the training objective in terms of a time-stability loss $\mathcal{L}_{TS}$, by introducing the Euler forward method into the training loop. It allows us to recover the complete time evolution of the temperature for each drive during training and do a multi-time-step optimization between the predicted and the ground-truth temperatures at once (see Equation 6). This enforces local smoothness like $\mathcal{L}_{reg}$ as well as consistency with our training data.

$$\mathcal{L}_{TS} = \sum_{i=1}^{N-1} \left\| u_{gt}^{i} - (NN_{\theta}(t_{i-1}, \mathbf{E}_{i-1}) * h + u_{pred}^{i-1}) \right\|_2^2 \tag{6}$$

where u_{gt}^{i} and u_{pred}^{i} are the ground-truth and predicted temperatures at time step i, respectively. N is the number of time steps for the particular drive session. This model will be denoted as LiFe-net (time-stability) in this article. The training pipeline of this model is illustrated in Fig.1

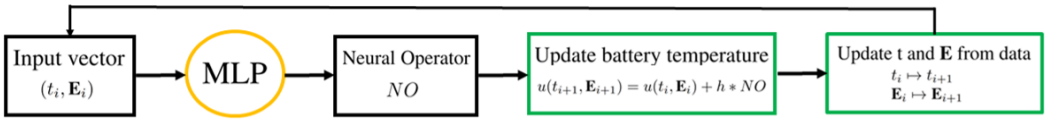


Figure 1: Training pipeline of LiFe-net (time stability). See Algorithm 1 and 2

.

2.4 Surrogate Model of Charging Statistics

The correlation of battery temperature to charging speed c_p and charging time t_{charge} is modeled by a linear regression model, viz.

$$\begin{aligned} c_p &= c_{soc} C_{bat} + c_T u_i(t_i, E_i) + o \\ t_{charge} &= c_{soc}^{s} C_{bat}^{s} + c_{soc}^{e} C_{bat}^{e} + c_T^{t} u_i(t_i, E_i) + o^t \end{aligned} \tag{7}$$

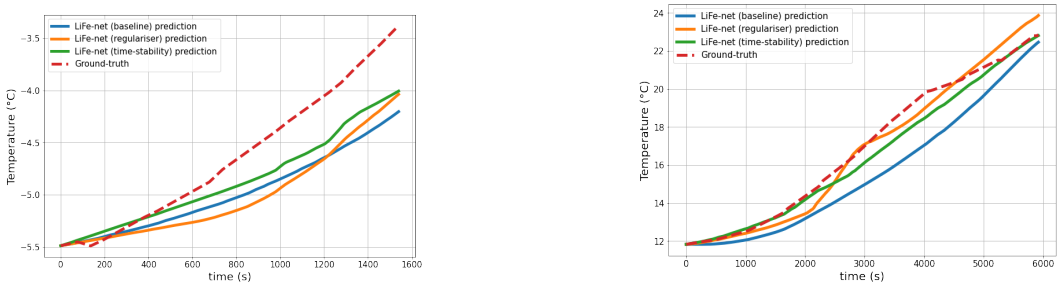
with c_{soc} explaining the contribution of state-of-charge C_{bat} and c_T scaling the contribution of battery temperature $u_i(t_i, E_i)$. Charging time is also modeled by coefficients c_{soc}^{s}, c_{soc}^{e} corresponding to state-of-charge at the beginning C_{bat}^{s} and end C_{bat}^{e} of the charging session. c_T^{t} models the contribution of battery temperature while o, o^t introduce offsets to the predicted charging time and power. The connection of this surrogate model with our neural operator now allows for advanced driver's assistance in deciding whether to heat the battery or not (see Fig. 4 in Appendix A).

3 Results and Discussion

The full dataset consists of 445 000 data measurements collected during 99 drives under different environmental conditions. The data was split with respect to individual drives leaving 92 drives for training and 7 for testing. Hence, the split for the training and test dataset is 80% and 20%, respectively. Table 2 summarizes the range for parameters used. All models were trained on the Taurus HPC system at TU Dresden using Tesla V100-SXM2-32GB GPU. Furthermore, we validate the performance of the models on the test data using MSE, MAE and relative error between predicted and ground-truth temperature values. The corresponding dataset and the implementation code are available at the following GitHub repository: `https://github.com/Jeyhun1/LiFe-net.git`

3.1 Prediction accuracy

We performed hyperparameter optimization of all discussed models using Weights and Biases library [8]. The results summarised in Table 3 show that regularised LiFe-net with λ parameter value



(a) (b)

Figure 2: (a) Time-stability loss is improving the accuracy at the interface of our training data range (here: winter). (b) All models perform well on test data that is well represented by our training data.

of 0.1 outperforms the baseline model where $\lambda = 0$. Furthermore, Tables 4 and 5 (See Appendix A) display MSE for the different number of hidden layers and different numbers of neurons per layer for both regularised LiFe-net and time-stability LiFe-net, respectively. Finally, Table 1 summarises the prediction accuracy of best performing models for each method considered in this work averaged over seven test data. Furthermore, comparison plots of all three models on individual test datasets are shown in Fig. 2. The results show that the LiFe-net model trained with the time stability loss function performs the best, especially on data at the interface of the training data range (see Figure 2a). This can be explained by the multi-time step optimisation during training which the other two methods lack. However, this comes at the cost of considerably slower training time (see Table 1).

3.2 Learning Charging Statistics from Temperature Data

The coefficients of our surrogate model explaining charging statistics as of Equation 7 have been estimated by solving the corresponding least-squares objective function by normal equations. The regression model of c_p got a $R^2 = 0.726$ while t_{charge} was explained even better since $R^2 = 0.884$. The coefficients can be found in Table 6. It can be observed that the state-of-charge at start of charging session is important for predicting both peak charging power ($c_{soc} = -0.9557$, $p = 0.015$) as well as -time ($c^s_{soc} = -0.6213$, $p = 0.018$). The reciprocal relationship implies that lower state-of-charge results in higher maximum charging power and therefore fast charging times since electrons can be transferred easier into the battery. Electrochemical processes also react faster in warmer environmental conditions which are reflected by the significant contribution of c_T and $c_T^{\Delta_c}$.

Table 1: Prediction accuracy of LiFe-net averaged on 7 test data.

Model	MAE	MSE	Relative error (%)	time per epoch (s)
LiFe-net (baseline)	0.7068	0.9342	4.6452	**104.89**
LiFe-net ($\mathcal{L}_{reg}$)	0.5445	0.6312	3.6223	123.46
LiFe-net ($\mathcal{L}_{TS}$)	**0.3845**	**0.2692**	**2.7715**	1151.25

4 Conclusion and broader impact

In this work, we introduced three variations of a Neural Operator that is learning the temperature evolution of a temperature probe attached to the surface of an EV battery depending on environmental conditions. We demonstrated that the LiFe-net trained with time-stability loss where we optimized the solution operator in a multi-timestep fashion is the most accurate. This information is then passed into a surrogate model that relates the measured temperature to charging statistics such as charging time and maximum charging power. Hence, this model enables EV manufacturers to get real-time never-before accessible data regarding battery sensitivities to numerous environmental conditions, which will give insights into designing more efficient battery systems in the future.

References

[1] Nur Hazima Faezaa Ismail, Siti Fauziah Toha, Nor Aziah Mohd Azubir, Nizam Hanis Md Ishak, Mohd Khair Hassan, and Babul Salam KSM Ibrahim. Simplified heat generation model for lithium ion battery used in electric vehicle. *IOP Conference Series: Materials Science and Engineering*, 53:012014, dec 2013.

[2] Ziye Ling, Fangxian Wang, Xiaoming Fang, Xuenong Gao, and Zhengguo Zhang. A hybrid thermal management system for lithium ion batteries combining phase change materials with forced-air cooling. *Applied Energy*, 148(C):403–409, 2015.

[3] Manh-Kien Tran, Satyam Panchal, Vedang Chauhan, Niku Brahmbhatt, Anosh Mevawalla, Roydon Fraser, and Michael Fowler. Python-based scikit-learn machine learning models for thermal and electrical performance prediction of high-capacity lithium-ion battery. *International Journal of Energy Research*, 46(2):786–794, 2022.

[4] Scott Mathewson. Experimental measurements of lifepo4 battery thermal characteristics, 2014.

[5] C R Pals, J Newman, and CA Univ. of California, Berkeley. Thermal modeling of the lithium/polymer battery. 1: Discharge behavior of a single cell. *Journal of the Electrochemical Society*, 142(10), 10 1995.

[6] Mojtaba Shadman Rad, Dmitry Danilov, Morteza Baghalha, Mohammad Kazemeini, and Peter Notten. Thermal modeling of cylindrical lifepo4 batteries. *Journal of Modern Physics*, 4:1–7, 07 2013.

[7] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces, 2021.

[8] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [N/A]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Section 3
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See section 3
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See section 3

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

Figure 3: Installation of the sensor on the floor of the cabin.

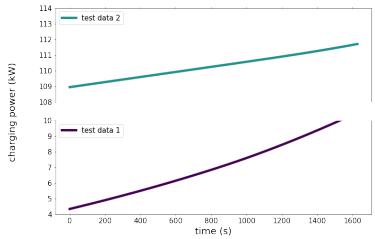


Figure 4: The connection of LiFe-net with our surrogate model now allows us to monitor and predict evolution of peak charging power during drives. Peak charging power is strongly reduced in winter as can be seen in test data 1 (purple).

Table 2: The value range of parameters in training and test dataset.

Parameter	Training data	Test data
power (kW)	[-68, 166]	[-67, 152]
speed (km/h)	[0, 167]	[0, 166]
battery level (%)	[10, 100]	[22, 95]
outside temperature (°C)	[-9.5, 35.5]	[-10, 29.75]
battery temperature (°C)	[-5.79, 33.9]	[-5.49, 33.85]

Table 3: MSE for regularised LiFe-net with different λ values

λ	0	1e-6	1e-4	0.01	0.1	1	10	100	100	10000
MSE	0.9342	0.934	0.9313	0.9268	**0.8349**	0.9317	0.9076	1.233	3.165	4.56

Table 4: MSE for different regularised LiFe-net architectures with λ=0.1

Hidden size	Number of hidden layers			
	2	4	6	8
10	0.7578	0.7787	0.7929	1.267
20	0.6981	1.132	0.8298	0.9033
50	0.7632	0.827	0.636	**0.6312**
100	0.6325	0.8349	0.7817	0.8284

Table 5: MSE for different LiFe-net (time-stability) architectures

Hidden size	Number of hidden layers			
	2	4	6	8
10	0.4734	1.631	1.51	0.8112
20	0.7086	0.6756	0.5135	0.3114
50	0.5767	0.5748	0.5943	0.4667
100	0.6005	0.5296	0.4614	**0.2692**

Table 6: Coefficients of Surrogate Model with * denoting significant coefficients at $p < 0.5$.

c_{soc}	c_T	o
-0.6109^*	3.7923^*	39.7772^*

c^s_{soc}	c^e_{soc}	c^t_T	o^t
-0.4505^*	0.7690^*	-0.6267^*	-6.9412

Algorithm 1 Training of LiFe-net with time stability loss

```
for epoch in 0, 1, ..., N do
    for M number of batches do
        Set initial condition for env.parameters in 'input'
        for j in 0, 1, ..., T.shape − 1 do
            NO = mlp(input[j])
            u_pred[j+1]=u_pred[j]+NO * step_size[j]
            Update t, u_pred and env. parameters in the 'input'
        end for
        loss = MSE(u_pred, u_gt)
        loss.backward()
        optimizer.step()
    end for
end for
```

Algorithm 2 Inference phase of LiFe-net

```
Set initial condition for env.parameters in 'input'
for i in 0, 1, ..., T.shape − 1 do
    NO = model(input[j])
    u_pred[i+1]=u_pred[i]+NO * step_size[i]
    Update t, u_pred and env. parameters in the 'input'
end for
```