PHYSICS GUIDED NEURAL NETWORK BASED STATE-OF-CHARGE ESTIMATOR FOR LITHIUM-ION BATTERIES

Fedi SALHI

Yildiz Technical University, Turkey

ORCID ID: https://orcid.org/0000-0002-0118-7861

Erhan AKDOGAN

Yildiz Technical University, Turkey

ORCID ID: https://orcid.org/0000-0003-1223-2725

ABSTRACT

In light of the growing demands of reliable substitutions to vulnerable conventional energy sources, electrical energy manifests as a potent surrogate that has started to prevail in a variety of industries and applications. The effectiveness and usability of this energy is highly associated with the development of sophisticated storage systems, a topic that has been increasingly attracting the attention of scientists and engineers. Among all technologies, batteries are the most preponderant solutions for electrical energy storage, in the form of electro-chemical potential energy. Particularly, Lithium-ion (Li-ion) batteries, being characterized by a high energy density and a low self-discharge rate, are considered one of the most effective solutions for this purpose. The State-of-Charge (SOC) represents a crucial parameter estimated by Battery Management Systems (BMS) used in energy and power calculations. Hence, an accurate SOC estimation is indispensable to prolonging the battery life and preventing hazardous usage conditions. Strenuous effort has been put into the development and enhancement of SOC estimation techniques including model-based methods and data-driven approaches. However, data-driven estimators are prone to be inconsistent with physics laws, which make their deployment in real-life applications riskier. In this work, we leverage the latest advancements in machine learning to develop a novel technique to train a Physics-Guided-Neural-Network-based SOC estimator for Li-ion batteries. In a nutshell, the methodology consists of introducing physics constraints during the training phase by designing a cost function that combines both physical and data losses, based on an energy approach. The proposed algorithm is validated using open data of Panasonic 18650PF Li-ion battery, by training and testing physics-guided and non-physics guided Convolutional Neural Network models, referred to as PGCNN and N-PGCNN models, respectively. Results showed that the proposed methodology outperformed the classic paradigm in both accuracy and physics consistency based on Mean Absolute Percentage Error (MAPE) and Physics Consistency Index (PCI) metrics, for large and small time-windows, which validates the proposed approach as a surrogate method for Neural Networks (NN) based SOC estimation.

Keywords: Physics-Guided-Neural-Network, State-of-Charge Estimation, Lithium-Ion Batteries, Data-Driven Modeling

1. INTRODUCTION

Batteries are powering every aspect of our modern life, from cell phones and computers, to electrical vehicles and drones. The notable presence of batteries in several areas required the development of sophisticated electronic systems and algorithms to ensure a faultless functioning of these systems. A Battery Management System (BMS) is an electronic device that provides a safe operation of rechargeable batteries. The primary functions of BMS include the protection of both the application user and the battery pack itself by monitoring the battery state, detecting unsafe conditions, and preventing undesired outputs and hazardous conditions. Another important function of BMS is to optimize the battery performance and service life by specifying the most convenient energy and power limits that maximize the battery's life. The fulfillment of these requirements necessitates an accurate estimation of SOC, which is a crucial parameter that signifies the current and the available charge of a battery [2]. The SOC is defined in the literature by the ratio of the available amount of charge $Q_{available}$, to the maximum amount of charge of the battery Q_{rated}, as mentioned in [3].

Despite the uncomplicated definition of SOC, it is challenging to provide an accurate estimation of this parameter due to the non-linear nature of Li-ion battery characteristics and complexity of the electro-chemical reactions [4]. A significant effort has been spent by researchers to tackle these challenges with different methods and approaches. Model-based methods are the most dominant estimators in industrial applications. Data-driven algorithms, on the other hand, are gaining more and more attention in the academia due to the fast-paced development of hardware platforms which is unlocking the potential of these algorithms that require a relatively high computational power.

The model-based approach, also known as white-box modeling, is considered to be a valid classic paradigm to solve engineering problems [5] that requires a set of physical and mathematical rules that describe the system behavior or the modeled process [6]. In the literature, we distinguish different methods for model-based SOC estimation. These methods generally use either the electrical or chemical behaviors of the cell or a combination of both. The Electrochemical Model (EM) takes into consideration the porous electrode theory, explained in details in [7]. The Equivalent Circuit Model (ECM) is based on the electrical properties of the battery cell, represented by the combination of various electrical elements. There are several electrical models proposed in the literature including the Thevenin model, the Rint model, and the RC model [8]. Obviously, the ECM requires to be parameterized in order to include the cell behavior that changes depending on the battery type and chemicals of the cell [9]. The Electrochemical Impedance Model (EIM) is similar to the EC model, but with inclusion of Z_{arc} and Warburg element [10]. The model-based SOC estimation approach often involves state estimation algorithms and adaptive filters into the estimation process including Unscented Kalman Filter [11], Adaptive Extended Kalman Filter [12], Adaptive Unscented Kalman Filter [13], Particle Filter [14], H∞Filter [15], Luenberger Observer [16], Proportion Integration (PI) Observer [17], and Sliding-mode Observer [18].

Data-driven SOC estimation paradigm avoids the necessity to have a deep understanding of the internals to model the battery cell. This technique is accompanied with simplifications of the modeling procedure, and reduction of the complexity usually associated with model-based SOC estimation that often necessitate the construction and the verification of a robust set of rules describing the system. The simplicity of date-driven modeling consists in mapping the inputs (typically the measured internal temperature, current, and voltage) with the output (SOC) using a relationship that does not involve battery models and estimators or observers. A variety of data-driven models have been used for SOC estimation. Shijie et. al [19] achieved 3.8% of SOC average error by splitting the classic 1-vector input structure into three vectors

corresponding to three modes: Idle mode, charging mode, and discharging mode. Ephren et. al [20] used a Deep Feedforward NN to directly map the battery measurements to the SOC prediction. The model was trained and tested for different ambient temperatures between -20 and $-25℃$. The proposed model achieved a Mean Absolute Error (MAE) lower than 2.17%. In [21], Bin et. al used a GRU model to for the SOC estimation problem. An Ensemble Optimizer, based on Nadam and AdaMAx algorithms, was used to find the optimal parameters that minimize the cost function. The suggested model provided a reduced training duration with increased prediction accuracy. In [22], Juan et al. used an SVM model to estimate the SOC of a lithium iron manganese phosphate ($LiFeMnPO_4$) battery cell based on current, voltage, and temperature measurements. Results showed that the proposed model maintained a high accuracy for constant or variable current charge/discharge profile.

Although the data-driven SOC estimation methods offer a variety of advantages including the modeling simplicity requiring only basic battery knowledge and the facility of training and deployment, they still suffer from some limitations. In particular, there are no constraints to prevent these models from delivering predictions inconsistent with physics laws. For instance, during a discharge cycle, the available energy should decrease between two consecutive SOC estimations. Hence, SOC estimations that lead to increased energy in the battery pack during a discharge phase are considered as physics inconsistent. In this work, we utilize this property to deliver a data-driven Li-ion SOC estimator, based on a Convolutional Neural Network (CNN), that outperforms the classic paradigm by designing a cost function that combines data and physical losses. Data-driven models incorporating physics knowledge into the learning process are known in the literature by Physics-Guided Neural Networks (PGNN). The term was coined for the first time by Arka Daw et al. in [23], in which they presented a novel framework that can be used in scientific problems that incorporates physics knowledge into NNs, and has been used since then in several works [24][25][26][27][28][29][30] for a variety of engineering problems.

The contribution of this work to the literature consists of the introduction of a new methodology that improves the classic data-driven Li-ion SOC estimation paradigm by embedding physics knowledge to the NN learning process. Although this novel approach is validated using Li-ion battery data, it can be applied to other types of batteries with a convenient hyper-parameter tuning. This approach should be also valid for other NN architectures. Results showed that the proposed model, referred to in this work as PGCNN (Physics Guided Convolutional Neural Network), outperformed the Non-Physics Guided Model, referred to as N-PGCNN, achieving lower MAPE and PCI.

2. METHOD AND MATERIALS

2.1 Panasonic 18650PF Lithium-Ion Battery Data

Panasonic 18650PF Li-ion Battery Data is used to train and test the PGCNN and N-PGCNN models. This dataset was mainly collected for research purposes and it was intended to be used for NN and Kalman Filter SOC estimation algorithms test and comparison or for battery models development. As stated in the original description, the dataset consists of different tests, all of which were conducted using a new $2.9Ah$ Panasonic 18650PF cell in a $0.226\,cm^3$ ($8\,ft^3$) thermal chamber with a $25A$, $18V$ Universal Battery Tester (UBT) and other test equipment manufactured by Digatron. The dataset was collected using a Panasonic NCR18650PF Lithium-Ion battery cell [31]. This type of batteries is characterized by a high energy and power density, and can be used in power assisted bicycles, 2-way radios, medical devices and robotics. The

specifications of the battery and the test equipment [32] are summarized in Table 1 and Table 2, respectively.

Table 1. NCR18650PF Lithium-Ion Battery Specifications

Maximum Capacity	2900 mAh
Nominal Voltage	3.6 V
Minimum/Maximum Voltage	2.5/4.2 V
Charge Temperature	0 to + 45°C
Discharge Temperature	−20 to + 60°C
Storage Temperature	−20 to + 50°C
Energy Capacity	577 Wh/l
Weigh (Maximum)	48.0 g

Table 2. Test Equipment Specifications

Cell Cycler Manufacturer	Digatron
Test Channel	25 A, 0 − 18 V Channel
Voltage / Current Accuracy	±0.1%
Data Acquisition Rate	10 Hz
Thermal Chamber	Cincinatti Sub Zero ZP-8
Thermal Chamber Size	0.226 cm^3 (8 ft^3)
Thermal Chamber Temperature Accuracy	±0.5°C

2.2 Structure of N-PGCNN and PGCNN Models

The CNN structure has been chosen to demonstrate the proposed methodology for SOC estimation. This architecture of NNs is widely used in image-based deep learning applications including object detection and image classification. However, in this work, we exploit the properties of CNN and its capability to learn the relationship between input features from one side, and the temporal pattern from the other side. Chamber and battery temperatures, voltage, and current measurements are the models' inputs. Figure 1 shows the analogy between image and multi time series input tensors for 2-D CNN architecture.

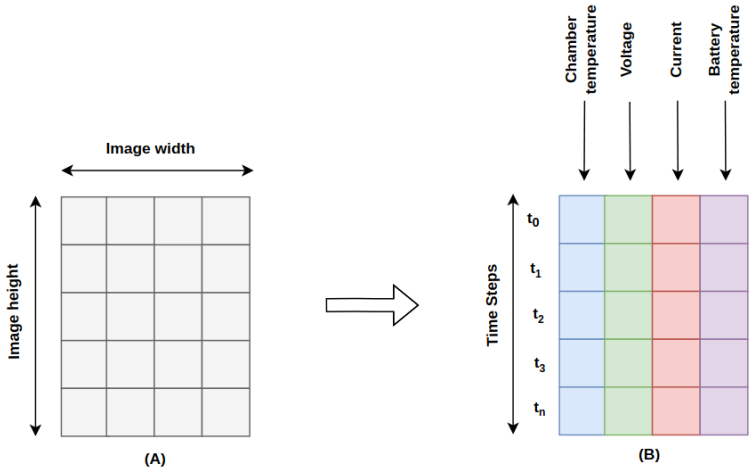


Figure 1. Analogy Between Image and Multi Time Series Input Tensors for 2-D CNN Architecture.

(A) is an input image (gray-scale/binary) for 2-D CNN architecture. (B), which is the input of our 2-D CNN models, has the same dimensions of the gray-scale/binary image (A). (B)

consists of 4 time series representing chamber and battery temperatures, voltage, and current data. In this work, different time steps are utilized to train and test the CNN models

The CNN architecture consists of a sequence of layers that perform specific operations on input tensors without reducing their dimensions. The main building blocks of CNN architecture are Convolutional, Pooling, and Fully-Connected layers. Table 3 shows the sequence of layers constructing the N-PGCNN and PGCNN models, and their tuned hyper-parameters. The activation functions, including Leaky ReLU, ReLU and Sigmoid, are only used with the Convolutional and Fully-Connected layers. This structure led to a satisfactory convergence with the designed cost function.

Table 3. N-PGCNN and PGCNN Models Structure

Layer	Layer Type	Activation Function	Parameters
Layer 1	2-D Convolutional	Leaky ReLU	Num. Of filters: 10 Filter size: 21x4 Zero-Padding
Layer 2	2-D Max Pooling	No Activation	Filter size 2x2 No Padding
Layer 3	Batch Normalization	No Activation	Norm. axis: Depth axis Momentum: 0.99 Epsilon: 0.001 Centering Scaling
Layer 4	2-D Convolutional	Leaky ReLU	Num. of filters: 5 Filter size: 7 ×4 Stride: 1 ×1 Zero-Padding
Layer 5	2-D Max Pooling	No Activation	Kernel size: 2x2 No Padding
Layer 6	Batch Normalization	No Activation	Normalization axis: Depth axis Momentum: 0.99 Epsilon: 0.001 Centering Scaling
Layer 7	Flatten	No Activation	None
Layer 8	Fully-Connected	ReLU	Num. of units: 15
Layer 9	Fully-Connected	Sigmoid	Num. of units: 1

2.3 Train Algorithm for N-PGCNN and PGCNN Models

A common CNN structure (described in Section 2.2) is used for N-PGCNN and PGCNN models. However, for the PGCNN models, physics knowledge is introduced to the learning process. NNs are generally confined to the data they are exposed to, during the training phase, and do not necessarily deliver physics consistent predictions. To overcome this limitation, we designed a hybrid loss function that guides the NN during the learning phase to increase the physics consistency. We will also demonstrate later that this approach delivers more accurate predictions and increases the model reliability. A simplified train algorithm for the PGCNN and N-PGCNN models is given in Figure 2. The symbols and abbreviations used in the algorithm are explained in Table 4.

Algorithm 1 Train Physics Guided Neural Network

$k, s = 0$

while $k < EPOCHS$ **do**

  for $s = 0$ to SPE **do**

    $B_{train} \leftarrow BG_{train}()$

    $SOC_{pred} \leftarrow CNN(B_{train})$

    $L_{data} \leftarrow MAPE(SOC_{target}, SOC_{pred})$

    $SOC_{Phy_0} \leftarrow CNN(IN_{Phy_0})$

    $SOC_{Phy_{100}} \leftarrow CNN(IN_{Phy_{100}})$

    $E_{Phy_0} \leftarrow Q.\Delta SOC \odot V_{Phy_0}$

    $E_{Phy_{100}} \leftarrow Q.\Delta SOC \odot V_{Phy_{100}}$

    $L_{Phy} \leftarrow \lambda.Mean(ReLU(E_{Phy_{100}} - E_{Phy_0}))$

    $L_{total} \leftarrow L_{data} + L_{Phy}$

    $P \leftarrow P_{updated}$

  end for

  $k \leftarrow k + 1$

end while

Figure 2. Simplified Train Algorithm for N-PGCNN (λ=0) PGCNN (λ≠0)

Table 4. Symbols and Abbreviations Used in Algorithm 1 (Figure 2)

B_{train}	Train Batch
BG_{train}	Train Batch Generator
CNN	Convolutional Neural Network
$CNN(X)$	Inference of X input tensor
$EPOCHS$	Number of train iterations
E_{Phy_0}	Energy vector corresponding to IN_{Phy_0}
$E_{Phy_{100}}$	Energy vector corresponding to $IN_{Phy_{100}}$
IN_{Phy_0}	First input tensor for physical loss regularization
$IN_{Phy_{100}}$	Second input tensor for physical loss regularization
k	Train iteration index
L_{data}	Loss calculated from the model prediction, data loss
L_{Phy}	Physical loss
L_{total}	Total loss
$MAPE$	Mean Absolute Percentage Error
P	Trainable parameters of the CNN model
$P_{updated}$	Trainable parameters updated after back propagation update
Q	Battery rated capacity
SOC_{pred}	State-of-Charge prediction
SOC_{Phy_0}	State-of-Charge predictions inferred from IN_{Phy_0}
$SOC_{Phy_{100}}$	State-of-Charge predictions inferred from $IN_{Phy_{100}}$
SOC_{target}	State-of-Charge ground truth
SPE	Steps Per Epochs, the number of batches in each epoch (iteration)
V_{Phy_0}	Voltage vector corresponding to IN_{Phy_0}
$V_{Phy_{100}}$	Voltage vector corresponding to $IN_{Phy_{100}}$

ΔSOC	The change in State-of-Charge
λ	Physical regularization rate
$\odot$	Element-wise product operator

In the training algorithm, the total loss L_{total} consists of two parts: the physical loss L_{Phy} and data loss L_{data} . In this work, the "data loss" term is used to describe the difference between the ground truth and the model predictions, without including any physical regularization. This difference is expressed as $MAPE$, as shown in Equation 1, in which the SOC ground truth and predictions as referred to as SOC_{target} and SOC_{pred} , respectively.

$$L_d = \frac{1}{n}\sum_{i=0}^{n}\left|\frac{SOC_{target_i} - SOC_{pred_i}}{SOC_{target_i}}\right| \cdot 100 \qquad (1)$$

We introduce a physical regularization to the cost function to include physical knowledge into the learning process of the CNN. The physical loss L_{Phy} is a function of the difference of two energy vectors E_{Phy_0} and $E_{Phy_{100}}$. These vectors are calculated from the model prediction vectors SOC_{Phy_0} and $SOC_{Phy_{100}}$ which are the inference results of two physical regularization input tensor batches $IN_{Phy_0}$and $IN_{Phy_{100}}$, respectively, as shown in Equations 3, 4, 5, and 6. The total loss is the sum of the data loss and physical loss as described in Equation 2. Note that the "cost" and "loss" terms in this work are used interchangeably and mean exactly the same thing.

$$L_{total} = MAPE\left(SOC_{target}, SOC_{pred}\right) + \lambda \cdot Mean\left(ReLU\left(E_{Phy_{100}} - E_{Phy_0}\right)\right) \quad (2)$$

$$SOC_{Phy_0} = CNN\left(IN_{Phy_0}\right) \qquad (3)$$

$$SOC_{Phy_{100}} = CNN\left(IN_{Phy_{100}}\right) \qquad (4)$$

$$E_{Phy0} \approx Q \cdot \Delta SOC_{Phy0} \odot V_{Phy0} \qquad (5)$$

$$E_{Phy100} \approx Q \cdot \Delta SOC_{Phy100} \odot V_{Phy100} \qquad (6)$$

Equations 5 and 6 give an approximation of the battery available energy, in which Q is the battery rated capacity scalar value, V_{Phy_0} and $V_{Phy_{100}}$ are the voltage vectors of the input tensors IN_{Phy_0} and $IN_{Phy_{100}}$, respectively. The exact energy formula is given in Equation 7, in which OCV is the open circuit voltage of the battery.

$$E(t) = Q \cdot \int_{SOC_{min}}^{SOC_t} OCV(\epsilon)d\epsilon \approx Q \cdot V \cdot \Delta SOC \qquad (7)$$

The input tensors IN_{Phy_0} and $IN_{Phy_{100}}$ are two 2-D input vectors extracted from the train data set and each of them consists of 10000 data rows (time steps). These vectors are identical, but shifted by 100 time steps i.e., $IN_{Phy_0}[k+100] = IN_{Phy_{100}}[k] \;\; \forall\, k \in [0,9899]$, as shown in Figure 3.

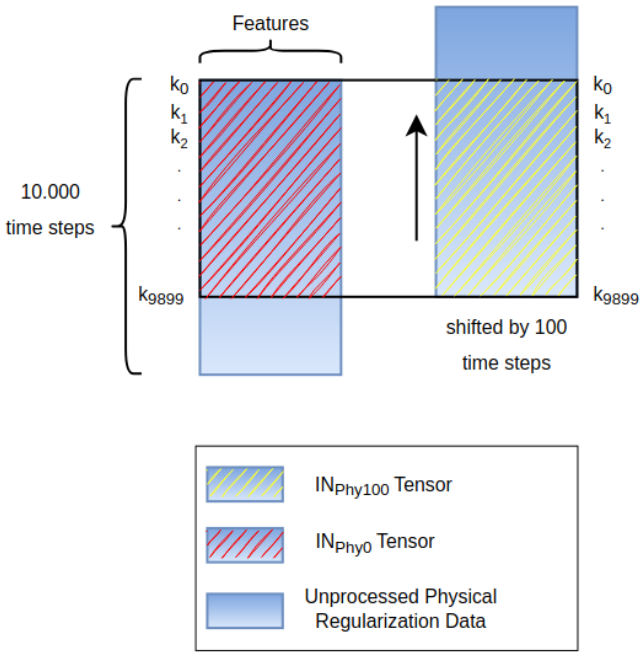


Figure 3. IN_{Phy_0} and $IN_{Phy_{100}}$ Input Tensors Used for Physical Loss Regularization

k indicates the time step in both tensors. The second tensor $IN_{Phy_{100}}$ is identical to the first one IN_{Phy_0}, but with a 100-time-step shift. This means that $IN_{Phy_0}[k+100] = IN_{Phy_{100}}[k] \; \forall \, k \in [0,9899]$

Given the fact that the data belong to discharge cycle, the condition in Equation 8 should be verified, as the battery remain energy decreases with time:

$$E_{Phy_{100}}[k] < E_{Phy_0}[k] \; \forall \, k \in [0,9899] \qquad (8)$$

Then, from Equations 5 and 6 we can write:

$$Q \cdot \left(SOC_{Phy_{100}}[k] - SOC_{min}\right) \cdot V_{Phy_{100}}[k] < Q \cdot \left(SOC_{Phy_0}[k] - SOC_{min}\right) \cdot V_{Phy_0}[k] \; \forall \, k \in [0,9899] \quad (10)$$

Supposing that the minimum SOC value is zero ($SOC_{min} = 0$), by substituting the SOC estimations in Equation 10 by their expressions in Equations 3 and 4, we obtain the condition that should be verified for CNN inferences denoted by $CNN(IN_{Phy_0})$ and $CNN(IN_{Phy_{100}})$:

$$Q \cdot \left(CNN(IN_{Phy_{100}})[k]\right) \cdot V_{Phy_{100}} < Q \cdot \left(CNN(IN_{Phy_0})[k]\right) \cdot V_{Phy_0} \; \forall \, k \in [0,9899] \quad (11)$$

As we all know, the $ReLU$ function, is defined by:

$$ReLU(x) = \begin{cases} x & \forall x > 0 \\ 0 & \forall x \leq 0 \end{cases} \qquad (12)$$

The $ReLU$ function defined in Equation 12 and used in Equation 2, returns a non-zero value for SOC predictions that violate the condition 11 i.e., when $E_{Phy_{100}}[k] > E_{Phy_0}[k] \; \forall \, k \in [0,9899]$. In this case, a physical loss, regularized by λ parameter, is added to the total loss function to minimize physics inconsistency in the CNN inferences. A simplified implementation of the combined loss function in Python programming language is given in Figure 4, and Figure 5 shows a visualization of the loss function from the model TensorBoard (a TensorFlow [33] framework visualization tool) graph.

```python
from keras import backend as K
import numpy as np
from keras.losses import mean_absolute_percentage_error

def combined_loss(parameters):
    """
    Computes the combined loss
    Args:
        parameters:    list containing the energy difference
                       and the physical regularization constant
                       lambda. energy_difference
                       = compute_energy(CNN(IN_Phy100)) -
                         compute_energy(CNN(IN_Phy0))
    Returns:
        combined loss = data loss + physical loss
    """
    energy_difference , lam = parameters
    energy_difference_np    = np.array(energy_difference ,
                                       dtype=np.float32 )

    def loss(y_true , y_pred):
        return mean_absolute_percentage_error(y_true , y_pred) + \
        lam * K.mean(K.relu(energy_difference_np))

        return loss
```

Figure 4. Simplified Combined Loss Function Implementation in Python Programming

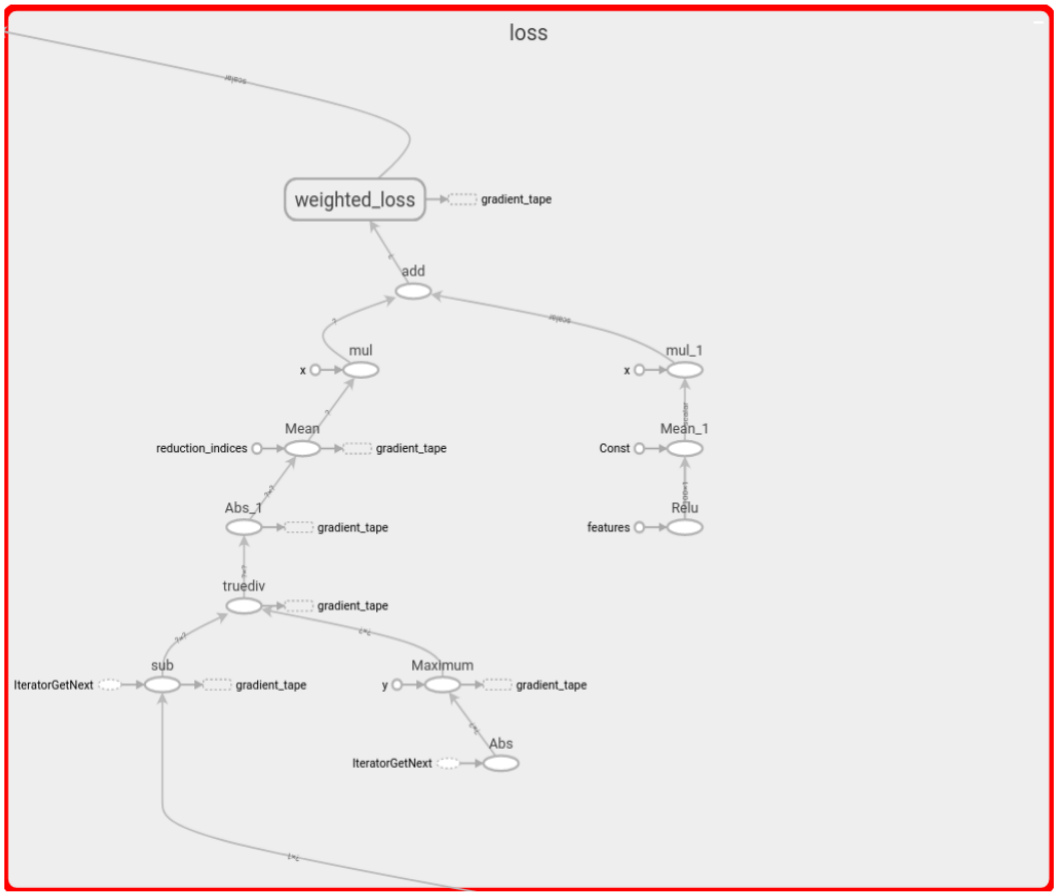


Figure 5. Visualization of the Combined Loss Function in TensorBoard Environment

The right and left branches of the loss function corresponds to the physical and the data losses, respectively. The two losses are added together and the output result is transferred to the model optimizer. To avoid dividing by zero, a small epsilon is added to the denominator of the MAPE function

3. RESULTS

PGCNN ($\lambda \neq 0$) and N-PGCNN ($\lambda = 0$) models are tested with different time windows and physical loss regularization λ values. The time window corresponds to the number of time steps i.e., data rows in the input tensors. λ rate regulates the amount of physical loss to add to the total cost function. The models are mainly evaluated on two metrics: MAPE and PCI. The first metric, MAPE, gives an idea on the significance of the prediction errors compared to the targets. The second metric, PCI, is used in this work as an indication of the physics consistency of tested models. The test procedure consists of feeding the models with 100 input batches, each contains 1000 input tensors with 100 or 25 time steps. These batches are randomly chosen from the test data set, which is separate from the train and validation data, and unused during the training phase.

The MAPE value is computed for each model using Equation 13, in which the $total\ inputs$ variable is equal to $number\ of\ batches\ \times\ batch\ size$.

$$MAPE(\%) = \frac{1}{total\ inputs}\sum_{i=0}^{total\ inputs}\left|\frac{SOC_{target_i} - SOC_{pred_i}}{SOC_{target_i}}\right| \times 100 \qquad (13)$$

The second evaluation metric, PCI (Equation 14), compares models in term of physics consistency. Generally, models that deliver SOC predictions leading to increased energy in the battery during a discharge cycle are considered non consistent with physics and vice versa. The PCI is used to quantify this statement and compare the N-PGCNN and PGCNN models in this regard.

$$PCI = \frac{1}{total\ batches}\sum_{i=0}^{total\ batches} L_{Phy_i} \qquad (14)$$

L_{Phy_i} in Equation 14, is the physical loss corresponding to the i^{th} input batch and defined by:

$$L_{Phy_i} = \frac{1}{batch\ size}\sum_{i=0}^{batch\ size} ReLU\left(E_{Phy_{100}}[k] - E_{Phy_0}[k]\right) \qquad (15)$$

$E_{Phy_{100}}$ and E_{Phy_0} in Equation 15, are energy vectors computed from $SOC_{Phy_{100}}$ and SOC_{Phy_0} model predictions, respectively.

As we can see from Equations 14 and 15, higher PCI value corresponds to lower physics consistency and vice versa.

Table 5 and Figures 6-9 show the test results of the N-PGCNN and PGCNN models. The physical regularization rates λ of the N-PGCNN models are zero. Their cost function does not contain a physical loss and only consists of data loss, as is the case in classic NNs paradigm. In contrast, the PGCNN models introduce a physical regularization to the cost function that guides the NN during the learning phase and leads to better prediction accuracy.

Table 5. N-PGCNN and PGCNN Models Test Results

Models	Time Window	λ	MAPE (%)	PCI
N-PGCNN	100	0	3.990	0.133
	25	0	8.385	0.156
PGCNN	100	1	3.015	0.098
	25	0.1	5.394	0.126



Figure 6. SOC Predictions and Absolute Error of PGCNN Model (λ=1, time window = 100) with Random Test Data



Figure 7. SOC Predictions and Absolute Error of N-PGCNN Model (λ=0, time window = 100) with Random Test Data

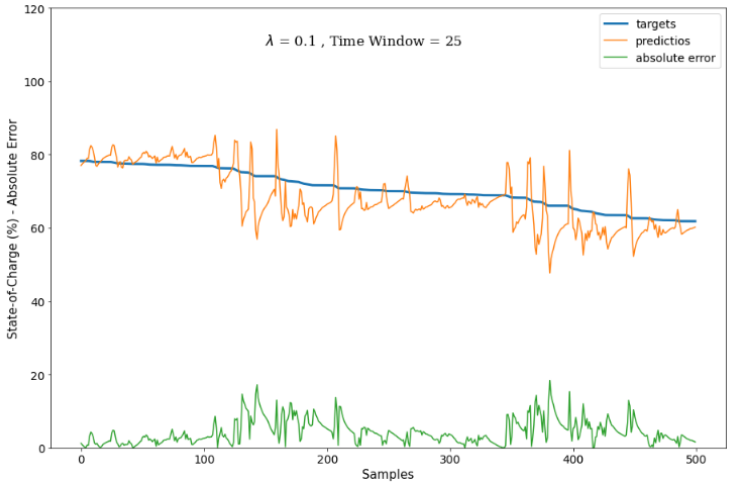


Figure 8. SOC Predictions and Absolute Error of PGCNN Model (λ=0.1, time window = 25) with Random Test Data

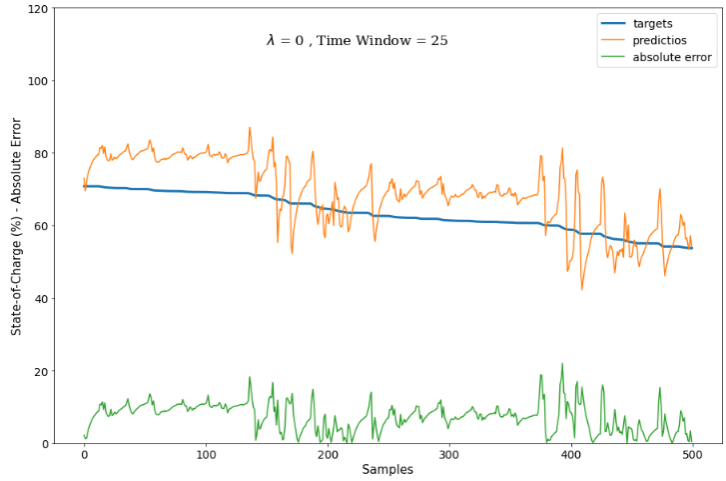


Figure 9. SOC Predictions and Absolute Error of N-PGCNN Model (λ=0, time window = 25) with Random Test Data

Overall, models with larger time window achieved lower MAPE and PCI results which means that the increase of time steps in the input tensor enhanced the performance of all models regardless of the introduction of physical knowledge into the learning process. The N-PGCNN model with large time window (100 time-steps) scored a 4.395% lower MAPE and 0.023 lower PCI compared with the small-time-window (25 time-steps) N-PGNN model. This improvement was also recorded for the PGCNN models with a 2.379% and 0.028 decrease in MAPE and PCI, respectively.

The comparison of N-PGCNN and PGCNN models with the same time window shows that the regularization of the cost function with a physical loss led to better results in both accuracy and physics consistency. For the large time window, the MAPE decreased from 3.990% to 3.015% on the test data set, with a regularization rate $\lambda = 1$. This enhancement was also observed with the small time window, from 8.385% to 5.394% MAPE, with $\lambda = 0.1$.

The proposed methodology also achieved an improvement in the physics consistency translated in lower PCI results. The 100-time-step PGCNN scored 0.098 PCI, compared to 0.133 for the N-PGCNN model with the same time window. In addition, the PCI metric shows and

improvement in the physics consistency in small-time-window models from 0.156 to 0.126 for N-PGCNN and PGCNN models, respectively.

4. DISCUSSION

In this study, a novel technique for SOC estimation is proposed. The main concept is to design and train a CNN model with physics constraints based on an energy approach for Li-ion batteries. Physics knowledge is incorporated into the learning process of the model by designing a hybrid cost function that combines physical and data losses, to improve the classic NN based data-driven paradigm in both accuracy and physics consistency. A publicly available data of Panasonic 18650PF Lithium-ion battery is utilized to validate the new approach by training and testing physics guided and non-physics guided models, in this work, referred to as PGCNN and N-PGCNN, respectively, with different time windows. Models' comparison was conducted based on MAPE and PCI. Results showed that the PGCNN outperformed the N-PGCNN in both accuracy and physics consistency for large and small time windows, which validates the proposed approach as a surrogate paradigm for NN based data-driven SOC estimation.

Future works may include the application of the proposed methodology to different NN architectures, and the deployment of these models to real-time battery management systems.

REFERENCES

1. P. Kollmeyer, Panasonic 18650pf li-ion battery data, Jun. 2018. DOI: 10.17632/ wykht8y7tg . 1. [Online]. Available: https : / / data . mendeley . com / datasets/wykht8y7tg/1
2. M. A. Hannan, M. S. H. Lipu, A. Hussain, M. H. Saad, A. Ayob, “Neural network approach for estimating state of charge of lithium-ion battery using backtracking search algorithm,” Ieee Access, vol. 6, pp. 10 069–10 079, 2018
3. D. N. How, M. Hannan, M. H. Lipu, P. J. Ker, “State of charge estimation for lithium-ion batteries using model-based and data-driven methods: A review,” Ieee Access, vol. 7, pp. 136 116–136 136, 2019
4. M. A. Hannan, M. S. H. Lipu, A. Hussain, M. H. Saad, A. Ayob, “Neural network approach for estimating state of charge of lithium-ion battery using backtracking search algorithm,” Ieee Access, vol. 6, pp. 10 069–10 079, 2018.
5. [5] S. Yin, X. Li, H. Gao, O. Kaynak, “Data-based techniques focused on modern industry: An overview,” IEEE Transactions on Industrial Electronics, vol. 62, no. 1, pp. 657–667, 2014.
6. S. Yin, S. X. Ding, X. Xie, H. Luo, “A review on basic data-driven approaches for industrial process monitoring,” IEEE Transactions on Industrial Electronics, vol. 61, no. 11, pp. 6418–6428, 2014.
7. M. Doyle, T. F. Fuller, J. Newman, “Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell,” Journal of the Electrochemi- cal society, vol. 140, no. 6, p. 1526, 1993.
8. H. He, R. Xiong, J. Fan, “Evaluation of lithium-ion battery equivalent circuit models for state of charge estimation by an experimental approach,” energies, vol. 4, no. 4, pp. 582–598, 2011.
9. Y. Hu, S. Yurkovich, Y. Guezennec, B. Yurkovich, “Electro-thermal battery model identification for automotive applications,” Journal of Power Sources, vol. 196, no. 1, pp. 449–457, 2011.
10. J. Xu, C. C. Mi, B. Cao, J. Cao, “A new method to estimate the state of charge of lithium ion batteries based on the battery impedance model,” Journal of power sources, vol. 233, pp. 277–284, 2013

11. X. Wu, X. Li, J. Du, "State of charge estimation of lithium-ion batteries over wide temperature range using unscented kalman filter," Ieee Access, vol. 6, pp. 41 993–42 003, 2018

12. Z. Chen, X. Li, J. Shen, W. Yan, R. Xiao, "A novel state of charge estimation algorithm for lithium-ion battery packs of electric vehicles," Energies, vol. 9, no. 9, p. 710, 2016.

13. S. Peng, C. Chen, H. Shi, Z. Yao, "State of charge estimation of battery energy storage systems based on adaptive unscented kalman filter with a noise statistics estimator," Ieee Access, vol. 5, pp. 13 202–13 212, 2017

14. J. Wang, L. Zhang, J. Mao, J. Zhou, D. Xu, "Fractional order equivalent circuit model and soc estimation of supercapacitors for use in hess," IEEE Access, vol. 7, pp. 52 565–52 572, 2019

15. Q. Yu, R. Xiong, C. Lin, "Online estimation of state-of-charge based on the h infinity and unscented kalman filters for lithium ion batteries," Energy Procedia, vol. 105, pp. 2791–2796, 2017.

16. B. Wang, Z. Liu, S. E. Li, S. J. Moura, H. Peng, "State-of-charge estimation for lithium-ion batteries based on a nonlinear fractional model," IEEE Transactions on Control Systems Technology, vol. 25, no. 1, pp. 3–11, 2016

17. L. Zheng, L. Zhang, J. Zhu, G. Wang, J. Jiang, "Co-estimation of state-of-charge, capacity and resistance for lithium-ion batteries based on a high-fidelity electrochemical model," Applied Energy, vol. 180, pp. 424–434, 2016

18. J. Du, Z. Liu, Y. Wang, C. Wen, "An adaptive sliding mode observer for lithium-ion battery state of charge and state of health estimation in electric vehicles," Control Engineering Practice, vol. 54, pp. 81–90, 2016

19. S. Tong, J. H. Lacap, J. W. Park, "Battery state of charge estimation using a load classifying neural network," Journal of Energy Storage, vol. 7, pp. 236– 243, 2016

20. E. Chemali, P. J. Kollmeyer, M. Preindl, A. Emadi, "State-of-charge estimation of li-ion batteries using deep neural networks: A machine learning approach," Journal of Power Sources, vol. 400, pp. 242–255, 2018

21. B. Xiao, Y. Liu, B. Xiao, "Accurate state-of-charge estimation approach for lithium-ion batteries by gated recurrent unit with ensemble optimizer," Ieee Access, vol. 7, pp. 54 192–54 202, 2019.

22. J. C. Álvarez Antón, P. J. García Nieto, C. Blanco Viejo, J. A. Vilán Vilán, "Support vector machines used to estimate the battery state of charge," IEEE Transactions on Power Electronics, vol. 28, no. 12, pp. 5919–5926, 2013. DOI: 10.1109/ TPEL.2013.2243918.

23. A. Karpatne, W. Watkins, J. Read, V. Kumar, "Physics-guided neural networks (pgnn): An application in lake temperature modeling," arXiv preprint arXiv:1710.11431, 2017

24. J. Wang, Y. Li, R. Zhao, R. X. Gao, "Physics guided neural network for machining tool wear prediction," Journal of Manufacturing Systems, vol. 57, pp. 298–310, 2020.

25. R. Zhang, Y. Liu, H. Sun, "Physics-guided convolutional neural network (phycnn) for data-driven seismic response modeling," Engineering Structures, vol. 215, p. 110 704, 2020.

26. X. Hu, H. Hu, S. Verma, Z.-L. Zhang, "Physics-guided deep neural networks for power

27. S. Pawar, O. San, B. Aksoylu, A. Rasheed, T. Kvamsdal, "Physics guided machine learning using simplified theories," Physics of Fluids, vol. 33, no. 1, p. 011 701, 2021.

28. Y. Yu, H. Yao, Y. Liu, "Structural dynamics simulation using a novel physics-guided machine learning method," Engineering Applications of Artificial Intelligence, vol. 96, p. 103 947, 2020.

29. J. N. Kani, A. H. Elsheikh, "Dr-rnn: A deep residual recurrent neural network for model reduction," arXiv preprint arXiv:1709.00939, 2017.

30. F. G. Pascual, W. Q. Meeker, "Estimating fatigue curves with the random fatigue-limit model," Technometrics, vol. 41, no. 4, pp. 277–289, 1999

31. Lithium cell tester mct series, Panasonic. [Online]. Available: https://actec.dk/media/documents/70FC46554038.pdf (visited on 01/15/2022).

32. E. Chemali, P. J. Kollmeyer, M. Preindl, R. Ahmed, A. Emadi, "Long short-term memory networks for accurate state-of-charge estimation of li-ion batteries," IEEE Transactions on Industrial Electronics, vol. 65, no. 8, pp. 6730–6739, 2017.

33. Martín Abadi et al., TensorFlow: Large-scale machine learning on heterogeneous systems, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/