Evaluation of 1D CNN Autoencoders for Lithium-ion Battery Condition Assessment Using Synthetic Data

Christopher J. Valant[1], Jay D. Wheaton[2], Michael G. Thurston[3], Sean P. McConky[4], and Nenad G. Nenadic[5]

[1,2,3,4,5] *Rochester Institute of Technology, Rochester, NY, 14623, USA*
cxvgis@rit.edu
jdwgis@rit.edu
mgtasp@rit.edu
spm9605@rit.edu
nxnasp@rit.edu

ABSTRACT

To access ground truth degradation information, we simulated charge and discharge cycles of automotive lithium ion batteries in their healthy and degrading states and used this information to determine performance of an autoencoder-based anomaly detector. The simulated degradation mechanism was an abrupt increase in the battery's rate of time-dependent capacity fade. The neural network topology was based on one-dimensional convolutional layers. The decision-support system, based on the sequential probability ratio test, interpreted the anomaly generated by the autoencoder. Detection time and time to failure were the metrics used for performance evaluation. Anomaly detection was evaluated on five different simulated progressions of damage to examine the effects of driving profile randomness on performance of the anomaly detector.

1. INTRODUCTION

The layers of capability of Prognostics Health Monitoring (PHM) in ascending order are: anomaly detection, diagnostics, and prognostics. This manuscript is chiefly concerned with the first capbility - anomaly detection. An early successful demonstration of an autoencoder neural network for vibration data provided the first layer of Prognostics Health Monitoring (PHM) capability – anomaly detection (Japkowicz, Myers, Gluck, et al., 1995). Since then, there has been a revolution in training deep neural networks, which facilitated training of deeper, more expressive neural network models and demonstrated its performance on many important machine learning tasks (LeCun, Bengio, & Hinton, 2015). The significant advantage of the deep learning approach in PHM is that it can be trained on abundantly available normal data (Yan & Yu, 2015), whereas classical machine learning models require a more statistically large data set (Bishop, 2013), with a significant number of instances of failure, which are expensive to collect. By the time of this study, autoencoders have become the first choice method for anomaly detection in PHM (Eklund, 2018).

PHM systems can evolve in time, growing their capabilities from anomaly detection, towards diagnostics and prognostics (Sikorska, Hodkiewicz, & Ma, 2011; Bussey, Nenadic, Ardis, & Thurston, 2014). As more failure data becomes available, representation learning features of deep learning can be exploited to learn better Condition Indicators (CIs). The experiments with real world data have demonstrated the potential of the deep learning models to effectively detect anomalies. However, because the ground truth of failure is typically not accessible without significant feature engineering (see, e.g., the fundamental axioms of structural health monitoring, and Axiom IVa in particular (Worden, Farrar, Manson, & Park, 2007)), we explored the effectiveness of deep learning models using simulated failures. The selection of the physical system and its model that generated synthetic data was based on the following criteria: the system had to be well-researched in PHM community (to provide familiarity with the nature of the solution and facilitate intuitive interpretations), it had to be nonlinear (to avoid trivial solutions), with information contained in multiple time scales, and to have existing published models (to leverage known results and reduce the debugging time). A lithium ion battery system quickly emerged from this criteria as a good candidate since the system was well-researched in the PHM community (see e.g. (Saha & Goebel, 2008; Olivares, Munoz, Orchard, & Silva, 2013)).

The simulated battery system enabled the study of a 1D Convolutional Neural Network (CNN) autoencoder, built to detect anomalous behavior where ground truth knowledge of how the system degraded up to failure was known. Degrada-

tion was set in the form of battery cell capacities. A driving profile in the form of power was input into the simulation, where alternating cycles of charging and driving with regenerative braking were present.

2. METHOD

To better evaluate performance of an autodencoder as an anomaly detector, we decided to build a simulation model because real failures of high-severity, low-frequency type (the type that is of interest to PHM), are expensive and the ground truth is often not accessible. The first step was to select the system to simulate. As stated above, the requirements were that the model be complex and nonlinear, with multiple temporal scales. With nonlinear relationships between series resistance and state-of-charge (SOC), open-circuit voltage and SOC, a couple of minutes time constant associated with relaxation and charge/discharge cycle on the order of hours, the lithium ion battery had the desired characteristics. Section 3 describes the customized simulation.

Because the focus was to assess anomaly detection capability of an autoencoder-based model, the model was trained on normal data to learn the internal representation. The details of data preprocessing and model development are provided in Section 4.

We built a decision-support layer to operate on the error signal produced by the autoencoder (the difference between its input and output) to assess the performance of the model with respect to its capability to operate as an anomaly detector. Section 5 describes the decision support and the metrics used for the assessment in detail.

Finally, in Section 6, we compared performances of multiple simulated failures to understand the sensitivity of the model to the rate of degradation and to the randomness of the driving cycle.

3. BATTERY MODEL SIMULATION

3.1. Simulation environment

Standard Modelica libraries, with Open Modelica interface OMEdit were used for simulation (Fritzson et al., 2006). The battery model was based on the Electrical Energy Storage (EES) library (Einhorn et al., 2011). The EES library was chosen because it was readily available, has been referenced in literature, and provided non-trivial examples that served as the starting point for our battery simulations. We started with the example *AdvancedStackCycling* which includes a three cell battery stack, charger, vehicle driving cycle, battery management system, and observable states. An example simulation output of one cycle including the voltage, current, SOC are shown in Figure 1. This includes a Constant Current (CC) - Constant Voltage (CV) charge cycle followed by a rest period before entering a drive cycle with regenerative braking. Ultimately, the battery behavior for the purpose of anomaly detection is represented by battery stack voltage and current waveforms.

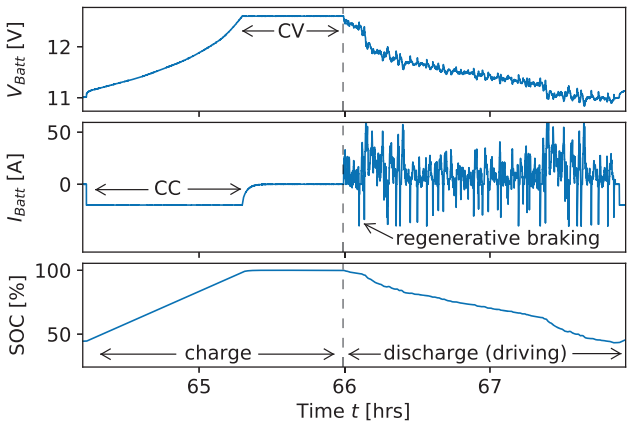

Figure 1. A closer view of charge up and driving with regenerative braking.

3.2. Extending a published simulation library

Simulating faults required extensions of the EES's *AdvancedStackCycling* simulation, including updated constants along with protection against over-voltage during regeneration, variable time-based capacity aging, variable resistance with SOC, and measurement noise. The extended top level block diagram is shown in Figure 2 with modifications of the original example explained in turn.

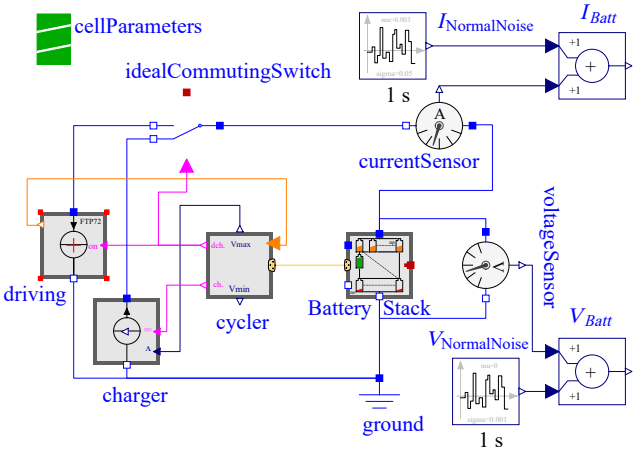

Figure 2. Top level Modelica stack cycling block.

3.2.1. Over-voltage protection

Over-voltage during the driving cycle was a consequence of the way the simulation model extrapolates the drive cycle

power in a Modelica CombiTimeTable. With periodic extrapolation, the time-series drive cycle was continuously repeated to ensure a value existed for the entire simulation time. Since the table was time-based rather than event based, each driving cycle started at a different point on the drive cycle, not necessarily at the beginning. This can result in the driving cycle starting at a high regeneration period (charging) with a fully charged battery. Without protection, we observed voltage spikes above the maximum cell voltage in certain instances. This was mitigated by adding a two-second moving average on the voltage to compare with a set threshold voltage 1% below V_{max}. If this threshold was exceeded, current was set to zero to represent a regeneration safety limit in a vehicle. This resulted in a slightly different simulation profile for each driving cycle, adding complexity to the model.

3.2.2. Capacity aging

In addition to voltage protection, complexity was added by including multiple linear capacity aging rates to represent healthy and faulty capacity fade over the lifetime of the battery. Capacity aging was defined as a time-dependent adjustment to the battery capacity as shown in Eq. (1) with the aging factor K defined in Eq. (2). For the simulations, normal capacity aging was set to reach our failure criteria, 70% of the initial capacity C_0, in approximately 3.1 years, that is C_f=70%. This typical time-based aging is shown as the healthy slope in Figure 3, interrupted by the accelerated slope due to a simulated fault. Low-rate degradation always accompanies normal battery operation, therefore we labeled this region as *healthy*. This region was used to train the autoencoder (see Section 4). The slope was adjusted by the damage factor m_d in the faulty region which started at the onset of failure point t_{onset}, set at 62.5 hours in our simulations. As listed in Section 6, Table 2, multiple slopes above and below the baseline of $m_d = 400$ were considered in our simulation to get a variety of datasets for testing the autoencoder model.

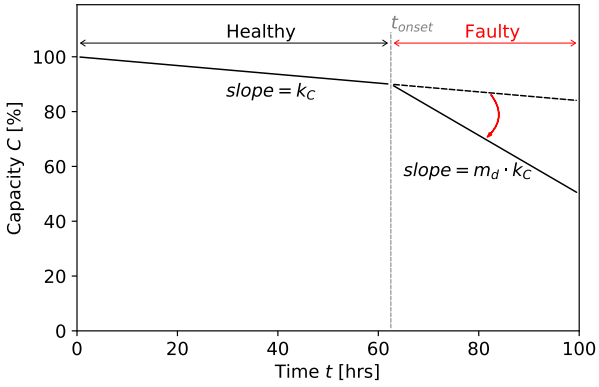

Figure 3. Capacity aging for healthy and faulty conditions.

$$C = C_0 + K(t) \cdot t \tag{1}$$

$$K(t) = \begin{cases} k_C, & t \leq t_{onset} \\ m_d \cdot k_C, & t > t_{onset} \end{cases} \tag{2}$$

3.2.3. Variable resistance with SOC

Batteries typically exhibit higher resistance at a lower state of charge. This behavior was implemented in the simulations as shown in Figure 4. A linearly decreasing resistance occurs until SOC reaches 50% and becomes a constant value. While this functional dependence was not quantitatively based on a specific battery behavior, it provided a qualitative model that was more accurate than assuming a constant resistance. In addition to the dependence on the state of charge, the cell resistance also depends on the capacity. The capacity fade is accompanied by in an increase in the cell resistance, which affects the relationship between the terminal battery voltage and the battery current. While the implementation of the functional dependence $R_{Batt}(C)$ seemed straightforward in principle, the attempts to extend the physics-based model to include $R_{Batt}(C)$ caused problems with the numerical integration of the overall dynamical model. Unrealistic oscillations observed in the resulting simulations prevented the extension of the model in this manner.

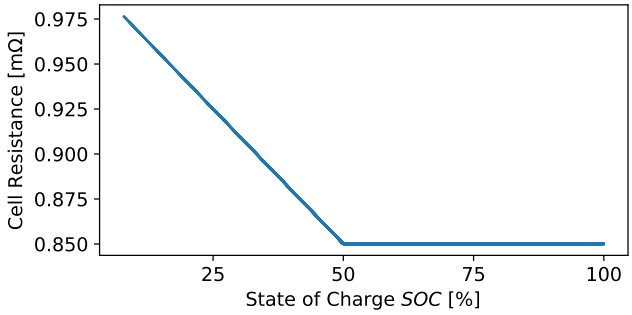

Figure 4. Cell resistance variation with SOC.

3.2.4. Measurement noise

The final level of complexity was added as noise on the stack voltage and current measurement. This is representative of measurement error and noise of the monitoring equipment. Normal noise was added to stack current with $\mu_I = 0.003$, $\sigma_I = 0.05$, and a sample period of one second. Normal noise was added to the stack voltage with $\mu_V = 0$, $\sigma_V = 0.001$, and a sample period of one second.

3.3. Modified discharge cycle

The EES library provides an FTP72 discharge profile in terms of electrical power (kW) for the Urban Dynamometer Driving Cycle which was used as the basis for the simulation,

hereafter referred to as the Urban Driving Cycle. The Urban Driving Cycle profile was extended to obtain more depth of discharge before recharging because of the limitations of the physics-based model: as stated above, the battery model did not include $R_{Batt}(C)$ dependence and the simulations did not reveal the changes in the voltage-current relationship for shallow depths of discharge. Consequently, deeper depths of discharge were needed to allow the autoencoder to observe a change in the voltage-current relationship over a longer period of time. An autoencoder trained on experimental data should detect changes even for shallow depths of discharge.

The modification of the discharge cycle was accomplished by widening all positive power (discharge) values from one second duration to five second duration for each instance. Any value equal or below zero power (charge) remained a one second duration. As a result, the total driving cycle duration for our Modified Driving Cycle simulation was 4482 seconds instead of 1370 seconds for the Urban Driving Cycle. The driving profiles are shown in Figure 5. Additionally, the power was reduced with a gain of 0.025 for the driving profile in Modelica because the battery stack being simulated consisted of only three cells and could not realistically power these loads.

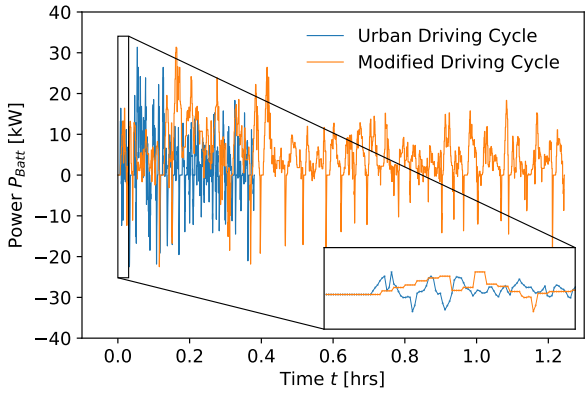

Figure 5. Battery power load profile for urban driving cycle with modification.

3.4. Design of multiple simulations

Multiple simulations were performed to create data with variations in the drive cycle and level of capacity degradation for testing repeatability of the autoencoder. The Modelica EES implementation featured two separate cycles: one for producing discharge power and the other for controlling the state of usage (whether it was in driving or in charging mode). In general, these two periods were non-commensurate and produced some level of pseudo-random driving cycles, which behaved more realistic than fully deterministic cycles. With this inherent randomness, we were able to create different driving patterns with time offsets (shifts in the horizontal axis)

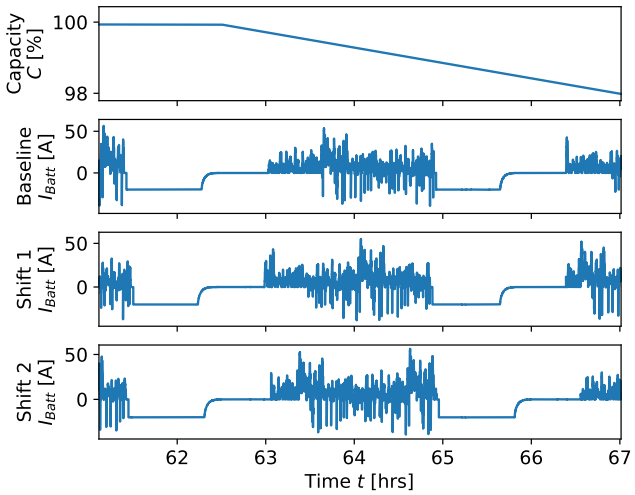

Figure 6. Comparison of drive cycle variations due to a time shift from the baseline profile.

of the Modified Driving Cycle shown in Figure 5, keeping all other battery parameters the same as the baseline. As explained in Section 3.2.1, periodic repetition of the driving profile was utilized for continuity at all time steps, therefore each shifted profile was different and no two cycles were identical. To illustrate the behavior of the simulation, Figure 6 shows the variability of battery current I_{Batt} that resulted from time shifts of the baseline drive profile, referred to as *Shift 1* and *Shift 2*. Since fault detection was our main focus, we also performed simulations with two levels of capacity fade by adjusting damage factor m_d. The simulation with a less severe capacity fade compared to the baseline was labeled *Slower* while the quicker capacity fade was labeled *Faster*. The method for adjusting the capacity is explained in Section 3.2.2.

4. AUTOENCODER-BASED ANOMALY DETECTOR

4.1. Data preparation

The simulated data contained thirty-six signals of various internal states of the battery model. The autoencoder model was concerned only with two that were practically observable: terminal battery voltage V_{Batt} and current I_{Batt}. A constant sampling rate of 1Hz was used for the two signals in the data. Data was first split into healthy and faulty sections. The battery was considered healthy up to where the change in the slope of cell capacity degradation occurred. After which, the battery was considered faulty until 70% of its cell's capacity had degraded, at which point failure was designated.

4.1.1. Segmenting samples

The convolutional model took input data of size (256, 2), i.e., 256 seconds by 2 signals, voltage and current. For

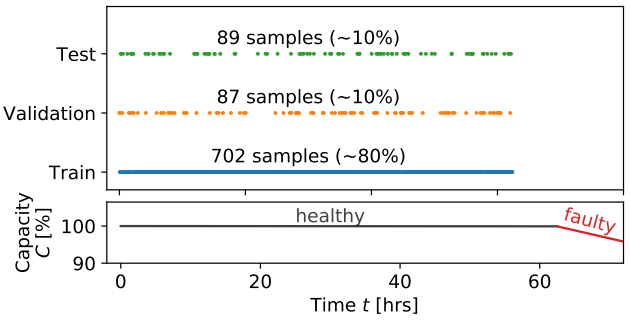

Figure 7. Split of healthy data: training, validation, and test (baseline simulation).

example, 100,000 data points in time would be segmented into 390 samples of 256 seconds, with 160 points discarded (100,000 = 390×256 + 160). This allowed 4 to 5 minutes of operation for each training sample, which corresponded to 1 to 2 time constants of a lithium ion battery relaxation (Nenadic, Bussey, Ardis, & Thurston, 2014; Valant, Gaustad, & Nenadic, 2019), so that the model could learn dynamical relations in the data. The value of 256 was then chosen for accelerating computations on the Graphical Processing Units (GPUs). After samples were created, healthy data was split into training, validation, and testing for the model.

4.1.2. Healthy data split

Healthy data samples were randomly split into training, validation, and testing, as shown in Figure 7. Approximately 80% of the healthy data was used to train the autoencoder. About 10% was used for validation during training, and another 10% was used for further testing after training was finished. This resulted in 702 training samples, 87 validation samples, and 89 testing samples. Any length of time could have been simulated, but the resulting 702 training samples were sufficient to achieve a reasonably accurate model. The details of the model accuracy are provided in Section 4.2

4.1.3. Scaling

Data was scaled into the 0 to 1 range using a min/max scaler. It was important to segment the data and split it into training and validation prior to scaling so that only training data was fit for scaling to avoid *data snooping* (Abu-Mostafa, Magdon-Ismail, & Lin, 2012).

4.2. Model development

The motivation for 1D CNNs was their compactness and ability to better exploit computational advantages of GPUs compared to Multi-Layer Perceptrons (MLPs). As shown below, these networks employ Finite Impulse Response (FIR) filters, whose parameters are computed during the training process, which makes them very flexible models, because many physical systems have been approximated as linear time invariant systems and modeled with filters. The nonlinear activation functions have the ability to extend traditional linear modeling. In our experiments, GPUs were able to moderately accelerate training of MLP-based models by cutting the training time in half; by contrast, the GPU acceleration in training of 1D CNN models of similar complexity (to that of their MLP counterparts) was about a tenfold increase.

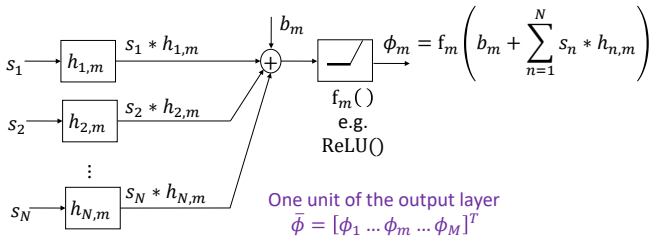

Figure 8. CNN 1D unit cell.

The basic cell of a 1D CNN layer, shown in Figure 8, is described as follows. A hidden layer output vector $\overline{\phi}$ of a 1D CNN cell consists of M hidden units ϕ_m

$$\overline{\phi} = [\phi_1 \ldots \phi_m \ldots \phi_M]^T, \tag{3}$$

where an individual unit is the result of a nonlinear transformation applied on an affine transformation of the N differently filtered input signal s

$$\phi_m = \mathrm{f_m}\left(b_m + \sum_{n=1}^{N} s_n * h_{n,m}\right) \tag{4}$$

$\mathrm{f_m}$ is the activation function (e.g. $\mathrm{ReLU}(z) \triangleq \max(z, 0)$) that performs the nonlinear transformation, $h_{n,m}$ are the coefficients of the N digital filters, and b_m is the offset. Symbol $*$ denotes convolution and each digital filter corresponds to

$$\begin{aligned} s_n[k] * h_{n,m}[k] &= \sum_{i=-\infty}^{\infty} s[k-i] h_{n,m}[i] \\ &= \sum_{i=0}^{k} s[k-i] h_{n,m}[i], \end{aligned} \tag{5}$$

where the limits of the last sum were reduced on the account of the causality of the signal and the filter. In the present case $N = 2$, with $s_1 = V_{Batt}$ and $s_2 = I_{Batt}$.

The overall model topology is depicted in Figure 9: the encoding consisted of alternating convolutional and max pooling layers, whereas the decoding consisted of alternating upsampling and convolutional layers.

The model was implemented in Keras (Chollet et al., 2015; Chollet, 2017), a deep learning framework within Tensorflow (Abadi et al., 2015), which includes an implementation of

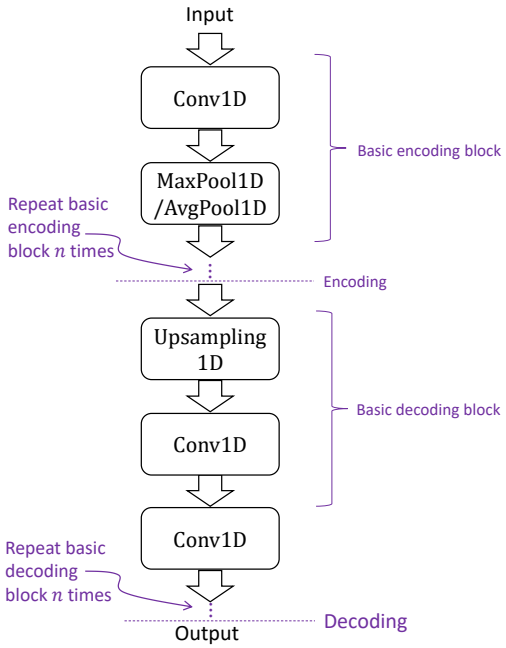

Figure 9. Topology of a 1D CNN autoencoder.

1D CNN layer. The implementation code is provided in the appendix.

Several iterations, after exploring model variations, viz. the number of layers, the width of the layers, and the filter size employed by the convolutional layers, arrived at a model summarized in Table 1. The convolutional layers employed 10% dropout for regularization (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). The optimizer was ADAM (Kingma & Ba, 2014), with the learning rate of 10^{-3}, and the Mean Squared Error (MSE) loss function. In addition, Mean Absolute Error (MAE) was used as a metric. The model was trained, validated, and tested on normal data (see Figure 7) and evaluated on the faulty interval. An additional restriction in this implementation was that only

Table 1. The summary of autoencoder model by layers.

Layer	Number of signals	Signal length	Number of parameters	Activation
Input	2	256	0	–
Convolutional 1D	40	256	2,600	ReLU
Max-Pooling 1D	40	128	0	–
Convolutional 1D	20	128	25,620	ReLU
Max-Pooling 1D	20	64	0	–
Convolutional 1D	4	64	2,564	ReLU
Up-Sampling 1D	4	128	0	–
Convolutional 1D	20	128	2,580	ReLU
Up-Sampling 1D	20	256	0	–
Convolutional 1D	40	128	3,240	ReLU
Convolutional 1D	2	256	82	Linear
Total params			36,686	

Dropout 10% and kernel size 32 for all convolutional layers

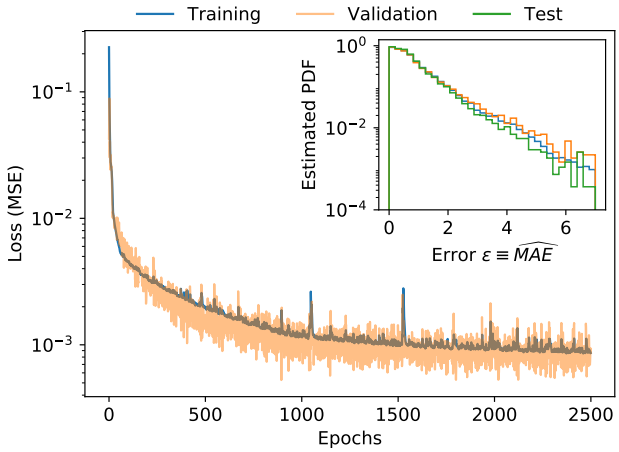

Figure 10. Loss during training with inset plot showing error histograms for train, validation, and test.

discharge data – the data associated with driving – was used for model training, validation, test, and evaluation. This decision was made after observing that the step change in signals associated with the onset of charge cycle was slow to train and resulted in larger errors compared to the rest of the data. Because these recurring large errors were lowering the sensitivity of the early anomaly detector prototypes, the data associated with charging of the battery was removed from the training.

Figure 10 shows loss during training for both training and validation samples. The training loss was greater than the validation loss because dropout turned off random neurons when evaluating training data. No neurons were turned off during evaluation of validation data resulting in a smaller loss, which is a typical manifestation of dropout regularization. The inset plot displays the scaled histograms, to estimate Probability Density Functions (PDFs), of prediction error for training, validation, and test data, with the logarithmic y-scale to better show the tails of the distributions. The three histograms were virtually indistinguishable (except from the end of their tails), suggesting good generalization of the trained model.

Figure 11 shows the model evaluation for the entire baseline simulation of the battery degradation; the top two subplots are the input signals and the outputs of the autoencoder (battery voltage $V_{Batt}(t)$ and current $I_{Batt}(t)$), the third subplot is the error, and the fourth subplot is the ground truth of the degradation – capacity $C(t)$. Note that the local peaks of the error signal increased in both height and width as capacity faded. The error was higher near the end of a driving cycle, as further shown in Figure 12, which zooms into two driving cycles. Recall that the depth of discharge was only slightly over 50% (see Figure 1). The capacity degradation of the simulated battery was observable from the terminal port only at relatively high depths of discharge, where the relationship

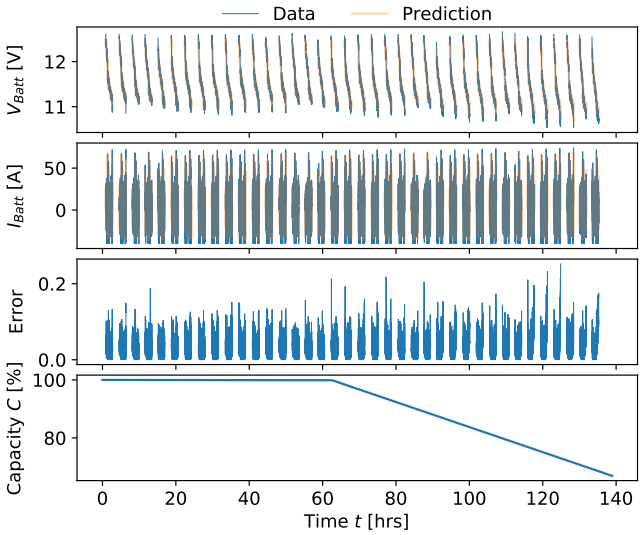

Figure 11. Model evaluation over *Baseline* simulation.

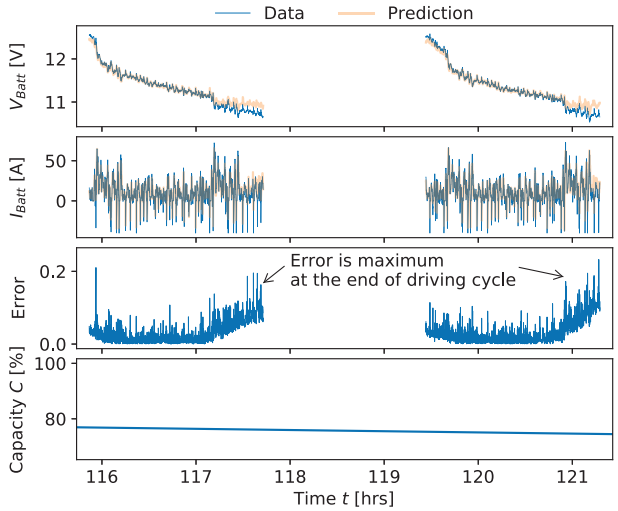

Figure 12. Model evaluation zoomed over two driving cycles.

between voltage and current started to change. A close-up of the battery and current signals in Figure 12 also provides a better visual display of the agreement between the inputs and outputs of the autoencoder.

5. DECISION SUPPORT

The error signal from an autoencoder can be used directly by an analyst, but a decision-support layer is typically employed to interpret the error signal and facilitate further actions. While a simple decision support layer is a filter-and-threshold, this study used the Sequential Probability Ratio Test (SPRT). The SPRT was developed for quality control (Wald, 1947) and first used in PHM for monitoring of nuclear plants (Gross & Humenik, 1991) and later for monitoring of

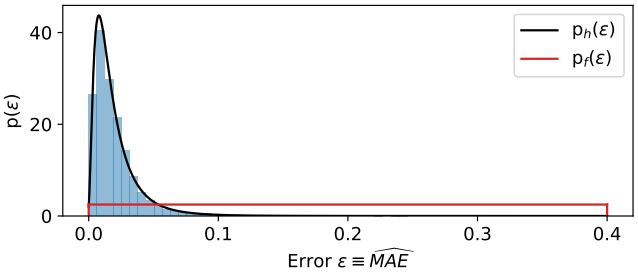

Figure 13. Error distribution of healthy data (histogram and fitted lognormal distribution) and assumed faulty distribution

software failures in servers (Cassidy, Gross, & Malekpour, 2002).

In the present implementation the SPRT computed the log-likelihood ratio of the error metric - MAE:

$$\ell = \sum_{i=1}^{N_s} \ln \mathrm{p_f}(\epsilon_i) - \sum_{i=1}^{N_s} \ln \mathrm{p_h}(\epsilon_i) \tag{6}$$

where ϵ_i was a sample MAE, N_s is the number of error samples considered, and $\mathrm{p_h}$ and $\mathrm{p_f}$ were the PDFs associated with distributions of healthy and faulty data, respectively.

The decision D has three states $\{Healthy, Faulty, Need\ more\ data\}$ and the decision process was carried out using the following logic in

$$D = \begin{cases} Faulty, & A \leq \ell \\ Need\ more\ data, & A \leq \ell \leq B \\ Healthy, & \ell \leq B \end{cases} \tag{7}$$

where A and B are the design parameters.

The probability of the healthy state was readily estimated from the histogram of MAE associated with training data (see Figure 13). A log-normal distribution was selected to model $\mathrm{p_h}$ because the histogram of MAE resembled the prototypical shape of the log-normal PDF given by Eq. (8)

$$\mathrm{p_h}(\epsilon|\mu_L, \sigma_L) = \frac{1}{\epsilon\sigma_L\sqrt{2\pi}} \exp\left(-\frac{\ln(\epsilon) - \mu_L}{2\sigma_L^2}\right), \tag{8}$$

where μ_L and σ_L are the parameters that were fitted to the error data.

The probability distribution associated with faulty data was less straight forward because there could be different faults associated with different failure modes and the decision-support layer should aim to catch all of them. Our design choice of uniform distribution

$$\mathrm{p_f}(\epsilon|\epsilon_{\max}) = \begin{cases} \frac{1}{\epsilon_{\max}}, & 0 \leq \epsilon \leq \epsilon_{\max} \\ 0, & \text{elsewhere} \end{cases} \tag{9}$$

where $\epsilon_{\max}$ was a hyperparameter, was loosely based on

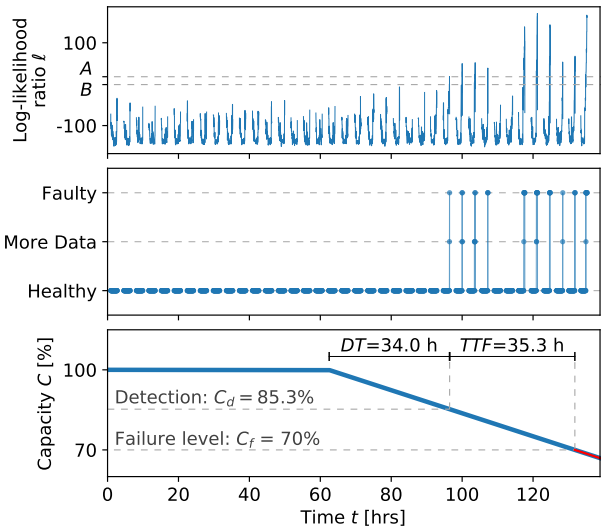

Figure 14. Anomaly interpretation (baseline case)

Laplace's principle of insufficient reason and $\epsilon_{\max}$=0.4 was heuristically chosen to extend considerably beyond the tail of $\mathrm{p_h}$.

Three hyperparameters of the decision support layer, viz. the number of samples N_s, the lower limit B and the upper limit A had to be selected. The limits were set based on the values of ℓ during training, with some margin for unseen cycles (see Figure 14) to be $B = -1$ and $A = 18$. The number of cycles was selected so that it corresponded to a couple of minutes of data, specifically $N_s = 128$.

Two metrics were used to evaluate the performance of the overall system: 1) Detection Time (DT) computed as the time difference between the onset of the accelerated degradation and the first detection of the anomaly and 2) Time To Failure (TTF) as the time difference between the first detection and the degradation level that was designated as the failure by the system specification C_f = 70%. Note that the battery is still operational at this level, but the level that the system deems unacceptable. For DT, smaller values indicated better performance, while for TTF, larger values indicated better performance. In addition to these two metrics, capacity of the battery at the time of the first anomaly detection, denoted by C_d, was also considered.

Figure 14 shows the performance of the decision-support layer that operated on the autoencoder error, shown in the third subplot from the top of Figure 12. The top subplot shows the log-likelihood ratio ℓ, the middle subplot shows the output of the decision layer D, and the bottom subplot provides the ground truth – capacity fade C. The log-likelihood ratio ℓ had peaks at the end of driving cycles. The grid lines indicate the SPRT thresholds A and B in the figure. The subsequent peaks were generally increasing both in height and in width, although not strictly monotonically. A few driving cycles with less depth of discharge did not have large peaks. The capacity subplot indicated the system-designed failure level of degradation and the two metrics – DT and TTF. C_d and C_f are indicated in the bottom subplot.

6. DETECTION PERFORMANCE

Four simulations, in addition to the baseline, were run to assess the anomaly detection performance of the autoencoder. Two of the additional simulations varied the damage factor, or the multiplicative factor of the normal capacity fade; one was degrading faster and the other slower, compared to the baseline. Their labels were *Slower* and *Faster*. The other two additional simulations varied the simulation time offset parameter to affect the random pattern of slightly more or less driving, which corresponded to slightly deeper or less deep depths of discharge. These two simulations were dubbed *Shift 1* and *Shift 2*.

Table 2 lists the five different simulations, with their characteristics (damage factor and offset time) and the performance metrics (DT and TTF). Capacity associated with the first anomaly detection was also indicated. As expected, there was some variations in performance among these five simulations. *Baseline* simulation had similar performance as *Shift 1* and *Shift 2*, the former performing a little worse and the latter performing a little better. The anomaly detector took longer time to detect the anomaly in the *Slower* simulation compared to *Baseline*, but C_d was higher. Under these circuimstances, it would be reasonable to expect that TTF was also longer, but it was not. The reason for the shorter than expected TTF was due to a couple of deeper than normal discharges that just happen to take place in that simulation.

The dashed lines indicating the SPRT thresholds A and B in Figure 14 were set considerably higher than the observed peaks. The margin was intentionally set to avoid false detection. After applying the anomaly detection on five different simulations, we revisited these thresholds. Figure 15 shows the peaks of ℓ during normal operation that occurred at the end of driving cycles in the form of a scaled histogram, which was then fitted to the normal distribution, also shown in the graph. The upper SPRT threshold A is indicated in the graph as well. The set threshold level was conservative, $\simeq$4.8 standard deviations away from the estimated mean. This design choice, given the observed simulated data, virtually assures zero false alarms. A less conservative threshold would be able to detect anomalies sooner, but would also increase the potential for Type II error.

7. CONCLUSION

The objective of the study was to obtain ground truth of the damage through simulation of a physical system and to use this ground truth to improve the assessment of the perfor-

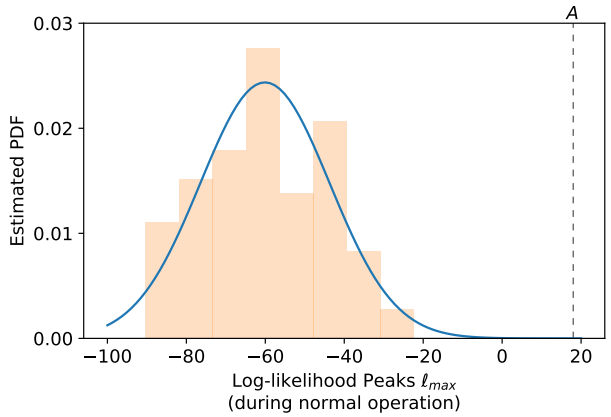

Figure 15. Estimated distribution of log-likelihood peaks for the five simulations with the fit to normal distribution.

Table 2. CNN performance on several simulation variations.

Simulation Labels	Damage Factor m_d	Offset t [s]	DT [hrs]	TTF [hrs]	C_d [%]
Baseline	400	0	34.0	35.3	85.3
Slower	350	0	45.0	31.4	87.8
Faster	450	0	30.1	20.3	82.1
Shift 1	400	1,500	34.2	31.1	85.2
Shift 2	400	3,500	27.1	42.4	88.2

mance of autoencoder-based anomaly detectors. We simulated normal aging and accelerated aging in a lithium ion battery system. A published battery model in Modelica was modified to simulate cell capacity degradation. The autoencoder was based on 1D CNN layers, which seemed particularly well-suited for simulating time-domain signals, and to our knowledge have not received significant attention. An SPRT-based decision-support layer was employed to interpret the error signals from the autoencoder. We showed that the autoencoder's prediction error increased as cell capacities faded during faulty operation, and quantified these anomalies using two metrics; detection horizon and remaining useful lifetime. Five variations of the battery simulations were generated in order to verify detection results.

Future work will focus on modeling faulty operation rather than healthy operation, expanding the battery simulation, and extracting condition indicators from the encodings of the autoencoder model. The battery simulation could be expanded to more cells that better match empirical data, resistance as a function of capacity fade, thermal effects, and an improved battery management system. A more advanced simulation with these features should provide nuances for the autoencoder model to interpret. If condition indicators on the advanced model can be found, they can inform damage assessment and diagnostics. Furthermore, developing the diagnostic level of PHM could lead to the prognostics level through forecasting these condition indicators.

ACKNOWLEDGMENT

This material is based upon work supported by the Office of Naval Research under Award No. N00014-17-1-2726

DISCLAIMER

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research.

NOMENCLATURE

t	Simulation time
I_{Batt}	Battery current
V_{Batt}	Battery voltage
SOC	Battery state-of-charge
C	Battery cell capacity
C_0	Initial battery cell capacity C
C_f	Failure level
C_d	Capacity associated with the first detection
K	Time-dependent slope of capacity fade
k_C	Normal capacity fade
t_{onset}	Onset of accelerated battery degradation
m_d	Multiplicative factor for faulty capacity fade
μ_I	Mean of current measurement noise
σ_I	Standard dev. of current measurement noise
μ_V	Mean of voltage measurement noise
σ_V	Standard dev. of voltage measurement noise
s	General input signal of a 1D CNN cell
N	Number of input signals of a 1D CNN cell
ϕ_k	k^{th} output of a 1D CNN cell
$\overline{\phi}$	Outputs of a 1D CNN cell arranged as a vector
M	Number of outputs of a 1D CNN cell
l	Log-likelihood ratio
$\mathrm{p_f}$	Probability density function for faulty data
$\mathrm{p_h}$	Probability density function for healthy data
ϵ	Sample mean average error
D	Decision
N_s	Number of samples in SPRT
A, B	Design parameters
μ_L	Log-normal distribution mean
σ_L	Log-normal distribution standard deviation

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... others (2015). Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*.

Abu-Mostafa, Y. S., Magdon-Ismail, M., & Lin, H.-T.

(2012). Learning from data. In (p. 173-177). AML-Book New York, NY, USA.

Bishop, C. M. (2013). Model-based machine learning. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *371*. Retrieved from https://doi.org/10.1098/rsta.2012.0222

Bussey, H. E., Nenadic, N. G., Ardis, P. A., & Thurston, M. G. (2014). Case study: Models for detecting low oil pressure anomalies on commercial vehicles. In *Annual conference of the prognostics and health management society 2014* (p. 13). PHM Society, http://www.phmsociety.org/node/1466/.

Cassidy, K. J., Gross, K. C., & Malekpour, A. (2002). Advanced pattern recognition for detection of complex software aging phenomena in online transaction processing servers. In *Proceedings international conference on dependable systems and networks* (pp. 478–482).

Chollet, F. (2017). *Deep learning with python.* Manning Publications Company.

Chollet, F., et al. (2015). Keras: Deep learning library for theano and tensorflow. *URL: https://keras. io/k*.

Einhorn, M., Conte, F. V., Kral, C., Niklas, C., Popp, H., & Fleig, J. (2011, 06). A modelica library for simulation of elecric energy storages.
doi: 10.3384/ecp11063436

Eklund, N. (2018). Anomaly detection tutorial. In *Proceedings of the annual conference of the prognostics and health management society.*

Fritzson, P., Aronsson, P., Pop, A., Lundvall, H., Nystrom, K., Saldamli, L., ... Sandholm, A. (2006). Openmodelica-a free open-source environment for system modeling, simulation, and teaching. In *2006 ieee conference on computer aided control system design, 2006 ieee international conference on control applications, 2006 ieee international symposium on intelligent control* (pp. 1588–1595).

Gross, K. C., & Humenik, K. E. (1991). Sequential probability ratio test for nuclear plant component surveillance. *Nuclear Technology*, *93*(2), 131–137.

Japkowicz, N., Myers, C., Gluck, M., et al. (1995). A novelty detection approach to classification. In *Ijcai* (Vol. 1, pp. 518–523).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436.

Nenadic, N. G., Bussey, H. E., Ardis, P. A., & Thurston, M. G. (2014). Estimation of state-of-charge and capacity of used lithium-ion cells. *Int J Progn Health Manag*, *5*, 12.

Olivares, B. E., Munoz, M. A. C., Orchard, M. E., & Silva, J. F. (2013). Particle-filtering-based prognosis framework for energy storage devices with a statistical characterization of state-of-health regeneration phenomena. *IEEE Transactions on Instrumentation and Measurement*, *62*(2), 364–376.

Saha, B., & Goebel, K. (2008). Uncertainty management for diagnostics and prognostics of batteries using bayesian techniques. In *2008 ieee aerospace conference* (pp. 1–8).

Sikorska, J. Z., Hodkiewicz, M., & Ma, L. (2011). Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing*, *25*(5), 1803-1836.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929–1958.

Valant, C., Gaustad, G., & Nenadic, N. (2019). Characterizing large-scale, electric-vehicle lithium ion transportation batteries for secondary uses in grid applications. *Batteries*, *5*(1), 8.

Wald, A. (1947). *Sequential analysis*. John Wiley & Sons.

Worden, K., Farrar, C. R., Manson, G., & Park, G. (2007). The fundamental axioms of structural health monitoring. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *463*(2082), 1639–1664.

Yan, W., & Yu, L. (2015). On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach. In *Proceedings of the annual conference of the prognostics and health management society.*

BIOGRAPHIES

Christopher J. Valant received his B.S. in Physics from Rochester Institute of Technology (RIT), where he is currently working towards an M.S. in Data Science. He joined the Golisano Institute of Sustainability (GIS) in 2015. His research interests include applied machine learning, evolutionary algorithms, computer vision, simulations, optimization and programming.

Jay D. Wheaton received his B.S./M.S. in Mechanical Engineering from Rochester Institute of Technology in 2012 where he is currently working towards an MBA in Innovation Management. He joined the Golisano Institute of Sustainability in 2017 as a pollution prevention engineer before transitioning to a PHM role. He has broad industry experience as a design engineer focused on new product development and advanced manufacturing techniques in the fields of RF communications and vibration/motion control. His research interests include system modeling, machine learning, fatigue analysis, and bio-medical devices.

```
from tensorflow.keras import Input, optimizers, losses
from tensorflow.keras import metrics as metric
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout, Conv1D, MaxPooling1D, UpSampling1D
dropout_rate = 0.1
# --- Input ---
input_data = Input(shape=(sample_size, num_signals))
# --- Encode ---
encoded = Conv1D(filters=40, kernel_size=32, activation='relu', padding='same')(input_data)
encoded = MaxPooling1D(pool_size=2)(encoded)
encoded = Dropout(rate=dropout_rate)(encoded)
encoded = Conv1D(filters=20, kernel_size=32, activation='relu', padding='same')(encoded)
encoded = MaxPooling1D(pool_size=2)(encoded)
encoded = Dropout(rate=dropout_rate)(encoded)
# --- Encodings ---
encoded = Conv1D(filters=4, kernel_size=32, activation='relu', padding='same')(encoded)
encoded = Dropout(rate=dropout_rate)(encoded)
# --- Decode ---
decoded = UpSampling1D(size=2)(encoded)
decoded = Conv1D(filters=20, kernel_size=32, activation='relu', padding='same')(decoded)
decoded = Dropout(rate=dropout_rate)(decoded)
decoded = UpSampling1D(size=2)(decoded)
decoded = Conv1D(filters=40, kernel_size=32, activation='relu', padding='same')(decoded)
decoded = Dropout(rate=dropout_rate)(decoded)
# --- Output ---
output_layer = Conv1D(filters=2, kernel_size=1, activation='linear', padding='same')(decoded)
autoencoder = Model(input_data, output_layer)
lr = 1e-3 # learning rate
autoencoder.compile(optimizer = optimizers.Adam(lr=lr), loss = losses.MSE, metrics = [metric.MAE,])
```

Figure 16. Our Keras implementation of the model.

Michael G. Thurston received his B.S. and M.S. in Mechanical Engineering from Rochester Institute of Technology (Rochester, NY, USA) in 1988, and his Ph.D. in Mechanical and Aerospace Engineering from the University of Buffalo (Buffalo, NY, USA) in 1998. He is the Technical Director and Research Associate Professor at the Center of Integrated Manufacturing Studies at Rochester Institute of Technology. He formerly held positions in air conditioning system development at General Motors and Delphi, and as a Researcher at the Applied Research Laboratory at Penn State University. He holds 7 patents in the areas of air conditioning and asset health monitoring. His research interests include: sustainable design and production, condition based maintenance and prognostics, and asset health management. He is a member of the Society of Automotive Engineers, and was awarded the Boss Kettering Award for product innovation by Delphi.

Sean P. McConky received his B.S. in Industrial Engineering from Rochester Institute of Technology (Rochester, NY, USA) in 1999. He joined the Center for Integrated Manufacturing Studies (CIMS) at Rochester Institute of Technology in 2000, where he is currently a Senior Staff Engineer. His research interests include condition based maintenance, asset health monitoring, additive manufacturing, and remanufacturing. He holds one patent for dynamic display systems related to asset health monitoring. He was a member of the team awarded the Defense Manufacturing Excellence Award for work on lifecycle logistics support tools.

Nenad G. Nenadic received his B.S. in Electrical Engineering from University of Novi Sad (Novi Sad, Serbia) in 1996 and his MS and Ph.D. in Electrical and Computer Engineering from University of Rochester (Rochester, NY, USA) in 1998 and 2001, respectively. He joined Kionix Inc. in 2001, where he worked on development of microelectromechanical inertial sensors. Since 2005, he has been with Center for Integrated Manufacturing Studies (CIMS) at Rochester Institute of Technology, where he is currently a Research Associate Professor. His research interest include design, analysis, and monitoring of electromechanical devices and systems. He has two patents in electromechanical design. He co-authored a textbook *Electromechanics and MEMS* and is a member of IEEE.

APPENDIX

Figure 16 lists the Keras implementation of the model.