Robust Feature Learning for Remaining Useful Life Estimation Using Siamese Neural Networks

1st Gurkan Aydemir
Electrical and Electronics Engineering
Bursa Technical University
Bursa, Turkey
gurkan.aydemir@btu.edu.tr

2rd Kamran Paynabar
H. Milton Stewart School of
Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA, USA
kamran.paynabar@isye.gatech.edu

3nd Burak Acar
VAVlab
Electrical & Electronics Engineering
Bogazici University
Istanbul, Turkey
acarbu@boun.edu.tr

***Abstract*—Remaining useful life (RUL) estimation is very critical for planning the maintenance of machinery in various industries. Deep learning models have gained popularity as the key tools in estimating RUL. However, raw sensor measurements are affected by multiple factors beyond the degradation level, which is the primary goal of RUL estimation. This adversely affects model training, especially in the lack of sufficient data. To address the aforementioned issues, we propose to train an artificial neural network (ANN) that captures the smooth and monotonous degradation process. A siamese neural network (SNN) model is used to train the ANN so as to minimize feature variation across consecutive time instances while maximizing it across changes in health conditions. The effectiveness of the smooth features in RUL estimation is illustrated using turbofan engine degradation simulation data set and degradation image stream data set that are collected from rotating engines. We further discuss that the new features can be individually used to assess the health condition of the machinery without an RUL estimator.**

***Index Terms*—Deep learning, Siamese Neural Networks, Remaining useful life estimation, Multiple operating conditions, Image prognostics, Predictive maintenance, Feature extraction.**

I. Introduction

Estimating remaining useful life (RUL) which is defined as the remaining time until a required repair/replacement of a machine or component, is the central problem in predictive maintenance [1]. RUL estimation methods can be classified into three main classes, namely physics-based, knowledge-based and data-driven methods [2]. In recent years, the data-driven models dominate the RUL estimation studies as they do not require the domain knowledge or physical model of the system of interest [3]. Deep learning is becoming dominant among the data-driven models thanks to the increasing availability of computational power and sensor data [4].

In general, deep learning models ignore the domain knowledge and model the RUL estimator as a black box from input to output. This approach may affect the performance. For instance, the performance of the suggested models on multiple operating condition data is far worse than that under a single operating condition set-up since the underlying smooth and monotonous degradation process is suppressed by varying operating conditions in the raw sensor signals. [5]–[8]. As a trivial approach, increasing the complexity of the models, at the cost of harder training (due to computation load and need for larger data sets), may potentially increase the accuracy of RUL estimation under multiple operating conditions [9], [10]. Although these studies reported improvement in RUL estimations under varying operating conditions, the performance was significantly inferior to non-varying conditions.

On the other hand, the redundancy in the multivariate raw sensor signals may decrease the RUL estimation accuracy of deep learning models. For instance, image and profile data are commonly employed for industrial prognostics [13], [14]. However, with the scarcity of training data, the deep learning models may fail to fully utilize the multivariate data in these problems. The authors proposed two different architectures to decrease the dimension and achieve a better RUL estimation performance [15]. Although a better RUL estimation accuracy than the classical methods is reported, the models are problem-specific and generalizability is not verified.

This paper proposes to train a feature extractor neural network using Siamese Neural Network (SNN) architecture that robustly captures the underlying smooth and monotonous degradation process. The **robust sensor features (RSF)** are shown to improve the RUL estimation under complex setups such as machinery with varying operating conditions and multivariate sensor data. The efficiency of the proposed method is demonstrated using a popular benchmark simulation data set and infrared degradation image streams from rotating machinery [16], [17]. Two separate deep learning models, namely DCNN and LSTM networks, are adopted for RUL estimation to demonstrate the performance of the RSF by the feature extractor that is trained using SNN, in comparison with the raw features for the simulation data. LSTM networks with the RSF are also utilized for the RUL estimation in the degradation image stream data.

II. Smooth and Monotonous (Robust) Feature Extraction Methodology

Two different neural network models are trained as feature extractors from raw features using an SNN architecture. The first SNN model includes two identical MLP branches that are employed for the unidimensional multi-sensor data. The

second SNN model has a pair of identical CNNs that extract the features from multivariate sensor data. Pairs of raw features are fed into the SNN, with twin branches, which are trained so as to minimize the difference in the output (features) for consecutive time windows and to maximize the difference between time windows from the start and end of individual recordings which are known to span the complete lifetime of individual machines. One of the twin networks in the SNN is used as the feature extractor.

A. Unidimensional Multi-Sensor Data

SNNs are the special types of neural networks that contain twin sub-networks (with identical weights) connected through an output layer, which may simply be a loss function. Various types of ANNs such as LSTM, CNN, and MLPs, can be utilized as sister networks. In general, multiple sensor measurements are used to assess the degradation level of industrial systems. An MLP network that consists of two hidden layers with hyperbolic tangent activation is preferred for the twin branches (so as the feature extractor) for the unidimensional multi-sensor data since there is no predefined correlation between the different unidimensional sensor measurements. The number of output features is suggested to be fixed to the number of input sensor measurements to achieve the same degree of freedom.

The training data is prepared in pairs where each pair is labeled/tagged for supervised training. The pairs are fed into twin networks and the specially designed loss functions compute a loss over the outputs of twin networks. The twin networks are trained using a contrastive loss function as described in Section II-C.

To achieve a better convergence, all the inputs (raw features) are normalized to the $[0, 1]$ range using min-max normalization [18]. The network's hyperparameters, such as the number of neurons in the hidden layers, can be empirically optimized using a validation set or by splitting the training set into training and validation sets in the absence. The model with the parameter weights which provides the best performance on the validation set is selected and one of the twin networks is used as the RSF extractor.

B. Multivariate Sensor Data

In most of the multivariate degradation data, such as image and profile data, there are spatial or temporal correlations between the neighbor values in the individual sensor measurements. A convolutional SNN architecture can be used to exploit the correlations in the multivariate data. Parameter sharing through space or time decreases the number of trainable parameters and prevents overfitting in a lack of sufficient data. A 2-D CNN architecture is proposed to extract the RSF from 2-D degradation data for generalizability. The SNN architecture consists of three convolutional layers with the maximum pooling layers, a dense layer, and a loss layer. The obtained features are flattened after the third convolutional layer and, hence 2-D input is mapped into a 1-D feature vector by the CNN feature extractor.

The network is trained using the pairs where each pair is labeled/tagged for supervised training as described in Section II-A. The input values are normalized to $[0, 1]$ range using min-max normalization. The network's hyperparameters, such as the number of kernels in convolutional layers and maximum pooling layers, can be empirically optimized using a validation set. The model with the parameter weights which provides the best performance on the validation set is selected and one of the twin CNNs is used as the RSF extractor.

C. Feature Extractor Training with Contrastive Loss

SNNs were first proposed to learn a similarity metric [19], [20]. The known sample is given to the first sister network and the sample that is wanted to be verified is given to the second one. The loss function measures the similarity of this pair. It is desired that the sister networks generate representations that have a low loss value for genuine, i.e., *like* pairs, and a high loss value for fake i.e., *unlike* ones. The training of SNN requires like and unlike pairs which should be created from the individual degradation sensor recordings. In our case, the *like* and *unlike* pairs refer to data from the same and different health conditions, respectively.

For the like pairs, it is assumed that the health status of a system at consecutive time instances remains unchanged in a degradation sensor recording, hence the successive recordings are used as the like pairs. For the unlike pairs, it is assumed that very early measurements are from the healthy period and the very late measurements are from the faulty period. Hence pairs of recordings, one from the assumed healthy period and one from the faulty period, are utilized to generate the unlike pairs. We used an equal number of recordings from both periods and used all pairs of these two sets as, unlike training pairs.

Another critical issue is the loss function that is used in the training of the SNN model. Let $(\mathbf{x}_i^1, \mathbf{x}_i^2)$ be the i^{th} pair of raw features and $f(.; \mathbf{W})$ be the function that transforms raw features to the new features, $\mathbf{r}_i^1$ and $\mathbf{r}_i^2$, as,

$$\mathbf{r}_i^1 = f(\mathbf{x}_i^1; \mathbf{W}), \tag{1}$$

$$\mathbf{r}_i^2 = f(\mathbf{x}_i^2; \mathbf{W}), \tag{2}$$

where $\mathbf{W}$ are the parameters to be learned. The contrastive loss is defined as,

$$L_i(y_i, \mathbf{r}_i^1, \mathbf{r}_i^2) = (1 - y_i) d(\mathbf{r}_i^1, \mathbf{r}_i^2) + y_i \max(0, m - d(\mathbf{r}_i^1, \mathbf{r}_i^2)) \tag{3}$$

where y_i is binary indicator which is equal to 0 for the like pairs and 1 for the unlike pairs, $d(\mathbf{r}_i^1, \mathbf{r}_i^2) = ||(\mathbf{r}_i^1 - \mathbf{r}_i^2)||_2^2$ and m is the margin. Minimization of L corresponds to minimization/maximization of the distance between like/unlike pairs. The margin m limits the intra-distance of unlike pairs and affects only the range of the extracted features. The loss is minimized with respect to $\mathbf{W}$ using the AdaGrad algorithm [21].

D. RUL Estimation Using the RSF

The RSF can be used to assess the degradation of the industrial machinery. One of the sister networks in the Siamese

models is used as the feature extractor after the training. The RUL estimation scheme is illustrated in Fig. 1.

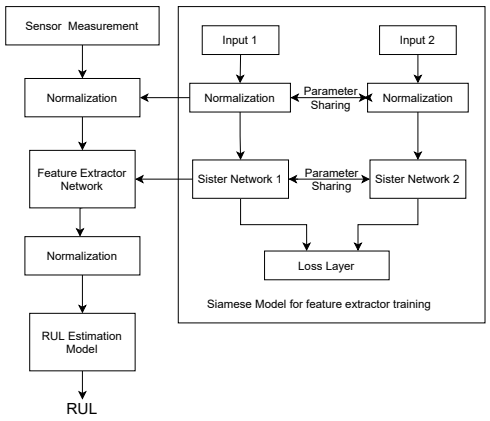

Fig. 1: The general architecture of RUL estimation using the proposed smooth and monotone feature extraction method. It should be remarked that the RSF are also normalized before estimating RUL.

Any RUL estimation model can be used together with the RSF set, however, for the illustrative purposes only two deep learning models, namely DCNN and LSTM networks, are adopted for the RUL estimation. The individual sensor measurements are normalized and then fed into the feature extraction block. The RSF are also normalized to $[0, 1]$ using min-max normalization for a better convergence of the RUL estimation model. Then, the RUL estimation model estimates the RUL by using the normalized RSF. RUL estimators are trained using a root mean square error as loss function since the RUL estimation is a regression task. Adam optimizer is employed for the minimization of loss [22]. The hyperparameters of RUL estimators, i.e., number of layers in the models, number of neurons, and activation functions, can be optimized experimentally using a validation set as in the optimization of the feature extractors' hyperparameters.

III. Experimental Results

The efficiency of the proposed feature extraction method is assessed using unidimensional multi-sensor simulation data and multivariate case study data. The simulation data is NASA C-MAPSS data which is commonly used as a benchmark data set for RUL estimation [23]. The case study data is a degradation image stream data that is collected from rotating machines [17].

A. Simulation Experiments

The RSF are compared with conventional features, which are the raw sensor measurements in the given data set, using two popular deep RUL estimation architectures, namely DCNN and LSTM [6], [7]. The RUL estimation results are reported in terms of root mean square estimation error (RMSE) and the score function that punishes the negative errors (i.e. the true RUL is less than the estimated RUL) more than the positive errors as,

$$\text{Score} = \begin{cases} \sum_{i=1}^{n} e^{-\left(\frac{\hat{\mathrm{T}}_i - \mathrm{T}_i}{a_1}\right)} - 1 \ , \ (\hat{\mathrm{T}}_i, -\mathrm{T}_i) < 0 \\ \sum_{i=1}^{n} e^{\left(\frac{\hat{\mathrm{T}}_i - \mathrm{T}_i}{a_2}\right)} - 1 \ , \ (\hat{\mathrm{T}}_i - \mathrm{T}_i) \geq 0 \end{cases} \tag{4}$$

where T is the true RUL, $\hat{\mathrm{T}}$ is the estimated RUL, free-parameters are chosen as $\alpha_1 = 10$ and $\alpha_2 = 13$ [16]. Lower scores indicate better performance.

NASA C-MAPSS is a tool that is designed to simulate the degradation of large turbofan engines [23]. RUL is defined in terms of cycles. Multiple operating conditions-single fault mode (MCSF) and multiple operating conditions-multiple fault modes (MCMF) subsets of the data set are used for the experiments. Each subset is divided into training and test sets. The training set includes the sensor recordings of engines from a healthy condition to failure. The sensor measurements in test engines are available until an arbitrary cycle and RUL labels are given at this cycle in terms of the number of cycles that remains to the failure. The number of available training and test samples are 260 and 259 for MCSF, 248 and 249 for MCMF.

The feature extractor network's hyperparameters are empirically optimized using a validation set formed by randomly selecting 10% of the training set. Both input layers have 21 neurons since there are 21 sensor measurements. The margin value $m = 1$ is used for the training. The learning rate and the number of epochs are set to 0.01, and 100, respectively, and no early-stopping is employed. The minimum loss in SNN training, with twin feature extractor MLP networks, on the validation set is achieved with 64 and 128 neurons in the two hidden layers of the MLP twins. The output layer provides the almost same performance for the layer size of 14 to 25. The network underfits for an output layer that has less than 14 neurons and overfits for the one that has more than 25 neurons. The best performance is obtained using an output layer with 21 neurons. The Keras library is used for implementation and simulations are run on GPU at Google™Colaboratory [24].

We trained and used identical LSTM and DCNN RUL estimators with raw features and RSF separately, to comparatively assess their RUL estimation performance. The former model includes two LSTM layers that have 32 and 64 neurons, respectively, with two fully connected layers that have 16 neurons with hyperbolic tangent activation. The output layer is a linearly activated layer that regresses the estimated RUL value. The hyperparameters were optimized based on the test performance with raw features. Identical hyperparameters are used in experiments with the RSF. The DCNN model does not have inherent memory, hence a sliding window approach is used to capture temporal information. The input of the DCNN model is created as a 2D matrix with dimensions of L (the size of the sliding window) by 21 (the number of features). Window lengths are 20 and 15 for MCSF and MCMF data sets because the shortest test sequences include only 20 and 15 cycles, respectively. The DCNN architecture consists of four identical convolutional layers with ten 10×1 kernels

TABLE I: The RUL estimation RMSE (mean±std) and score (mean±std) for DCNN, LSTM, hybrid [10] and biLSTM [9] models with single and multiple faults, under varying operating conditions.

		DCNN Raw Features	DCNN RSF	LSTM Raw Features	LSTM RSF	Hybrid Model [10] Raw Features	biLSTM Model [9] Raw Features
MCSF	**RMSE**	26.82 ± 1.23	18.28 ± 0.52	21.78 ± 2.41	$\mathbf{13.07 \pm 1.04}$	15.24 ± 2.65	25.11 ± 2.44
	Score	11286 ± 3675	1966 ± 239	3694 ± 2067	$\mathbf{776 \pm 134}$	1282 ± 260	4785 ± 380
MCMF	**RMSE**	27.65 ± 1.69	20.73 ± 0.29	25.54 ± 3.14	$\mathbf{15.17 \pm 0.90}$	18.16 ± 2.17	29.12 ± 3.44
	Score	12644 ± 6428	3134 ± 571	11527 ± 12341	$\mathbf{1020 \pm 135}$	1527 ± 322	6224 ± 450

and an extra convolutional layer with 3×1 kernel. Rectified linear unit (ReLU) activation is used in convolutional layers. The extracted $2D$ feature array is flattened and fed into a fully connected layer with 100 neurons. The output layer with linear activation regresses the estimated RUL.

The experiments from the training of the MLP feature extractor to the RUL estimation are repeated 10 times to report the confidence intervals. Keras Python library is used for implementation and runs on Google Colaboratory [24].

Table I provides a comparison of the performance of LSTM and DCNN RUL estimators with raw and extracted features, together with the state-of-art hybrid and biLSTM models [9], [10]. The RUL RMSE and estimation scores were improved for MCSF and MCMF data sets both with DCNN and LSTM RUL estimator. The LSTM RUL estimator with the RSF performs better than both the hybrid and biLSTM models reported in the literature for the same data set. In addition to the improvement in mean RMSE, its standard deviation across the test engines also showed a clear decrease, indicating a more robust prediction and higher reliability. The LSTM RUL estimator outperformed both the hybrid and biLSTM models, both in terms of the mean and standard deviation of scores.

B. Case Study

The performance of the smooth in time feature extraction method is demonstrated in a case study data that is collected from a rotational machine set-up [17]. Only a limited number of image degradation data samples are obtained from the bearings from a brand new state to failure, because of the high cost of accelerated degradation tests. Four degradation streams that consist of 375, 611, 827, and 1478 images, respectively, are available. To increase the number of samples for training and testing, the original image streams are resampled. 284 streams of length between 17 and 55 are generated [13], [15].

Leave-one-out cross-validation is used for the experiments and a total of 284 experiments are conducted. The time-to-failure (TTF) estimation performance is evaluated in terms of absolute percentage error. Tensor regression and two other deep learning models are provided as benchmark RUL estimators [13], [15].

Images in the degradation streams are two-dimensional arrays with dimensions of $40, 20$. In the experiments, a convolutional SNN architecture is adopted to decrease the number of trainable parameters since there are spatial correlations between the neighbor pixels in the individual images. The kernel sizes in the convolutional layers are fixed to 3×3 and the number of kernels is $8, 16, 16$, respectively. Maximum pooling layers are employed with a pooling size of 2×2. The dense layer has 20 hidden neurons and the contrastive loss that is described in Section II is used. Dimension of the input images is decreased from 800 to 20 with one of the CNN twins and features that capture the masked degradation are extracted.

The RSF exhibit smooth and monotonous increasing/decreasing trends from the starting point to the failure. Values of the features at the starting and failure points are similar. Therefore, it is claimed that the features themselves can be used to assess the health status and estimate the RUL of the monitored rotating machine. The lowest(highest) one from the features that have an increasing (decreasing) trend is chosen as the monitoring statistics. The average of the values of this feature before the last 3 cycles in the training samples is set as a threshold to prevent failure. This threshold provides an alarm in the last $2-5$th cycles for all test samples before the exact failure with an average of 3.2 cycles.

We trained and used only the LSTM network to estimate TTF from the degradation streams since the LSTM network performs better than DCNN in the simulation experiments. The network includes two LSTM layers with 128 neurons and two fully connected layers that have $64, 32$ neurons with hyperbolic tangent activation. The output layer is a linearly activated layer that regresses the estimated TTF value. A weighted squared error loss function is used in the training which is calculated as follows:

$$C = \sum_{k=t_b}^{t_f} |k(TTF_k - \widehat{TTF_k})|^2 \tag{5}$$

where t_b, t_f are burn-in time and failure time, respectively. The amount of information increases as time progresses, so as the degradation, hence the late estimation errors are punished more. Moreover, the TTF estimations start after a burn-in period (3 in the experiments) to capture some time information from the individual images. Keras Python library is used for the implementation on Google Colaboratory [24].

Fig. 2 provides a comparison of the performance of the LSTM RUL estimator with RSF, together with the tensor regression method and the state-of-art deep learning models, namely convolutional LSTM and LSTM with autoencoder designed features in terms of mean absolute prediction error and with respect to observation percentiles [13], [15]. The

mean absolute prediction error of LSTM with the RSF is 0.060 on average which is lower than the errors of convolutional LSTM (0.072) and LSTM with autoencoder-designed features 0.064 and slightly higher than the tensor regression's error (0.058). Although the LSTM RUL estimator with the RSF performs worse in lower percentiles, its performance is very close to the benchmark models at 50% observation percentiles and superior to them for the higher percentiles, i.e., it provides a better RUL estimation accuracy at the points that are close to the failure and more critical for the maintenance planning.

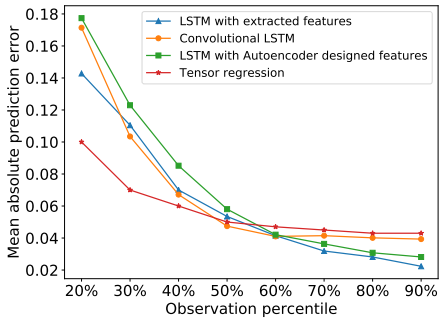

Fig. 2: The performance comparison of LSTM RUL estimator with RSF, together with tensor regression and the state-of-art deep learning based RUL estimators, namely convolutional LSTM and LSTM with autoencoder designed features

IV. Conclusion

A novel deep learning-based approach is proposed in this paper to extract features that are smooth in time. The RSF extractor is trained using an SNN architecture. The effectiveness of the RSF in RUL estimation is illustrated using the C-MAPSS turbofan engine degradation data set and degradation image stream data set that are collected from rotating engines. The proposed model assumes that the measurements are recorded periodically. The performance of the method under non-periodic recordings, i.e., missing data, needs to be assessed. Further, in the lack of periodic recordings, the training approach needs to be revised. In that case, it is possible to build training pairs by annotating them with a continuous *likeness* parameter that (eg. linearly) decreases with increasing time difference between the measurements. Extensions of the proposed approach to the machinery with the non-periodic data is a prominent future research direction. Moreover, the feature extractor can be further trained jointly with the RUL estimator. Therefore, this is left for future research.

References

[1] B. Bagheri, S. Yang, H.-A. Kao, and J. Lee, "Cyber-physical Systems Architecture for Self-Aware Machines in Industry 4.0 Environment," IFAC-PapersOnLine, vol. 48, no. 3, pp. 1622–1627, Jan. 2015.

[2] Y. Peng, M. Dong, and M. J. Zuo, "Current status of machine prognostics in condition-based maintenance: a review," The International Journal of Advanced Manufacturing Technology, vol. 50, no. 1-4, pp. 297–313, Sep. 2010.

[3] F. Ahmadzadeh and J. Lundberg, "Remaining useful life estimation: review," International Journal of System Assurance Engineering and Management, vol. 5, no. 4, pp. 461–474, Dec. 2014.

[4] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," Mechanical Systems and Signal Processing, vol. 115, pp. 213–237, Jan. 2019.

[5] G. Sateesh Babu, P. Zhao, and X.-L. Li, "Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life," in Database Systems for Advanced Applications, ser. Lecture Notes in Computer Science, S. B. Navathe, W. Wu, S. Shekhar, X. Du, X. S. Wang, and H. Xiong, Eds. Dallas, TX, USA: Springer International Publishing, 2016, pp. 214–228.

[6] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," Reliability Engineering & System Safety, vol. 172, pp. 1–11, Apr. 2018.

[7] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long Short-Term Memory Network for Remaining Useful Life estimation," in 2017 IEEE International Conference on Prognostics and Health Management (ICPHM), Dallas, TX, USA, Jun. 2017, pp. 88–95.

[8] Y. Liao, L. Zhang, and C. Liu, "Uncertainty Prediction of Remaining Useful Life Using Long Short-Term Memory Network Based on Bootstrap Method," in 2018 IEEE International Conference on Prognostics and Health Management (ICPHM), Seattle, Washington, USA, Jun. 2018, pp. 1–8.

[9] C.-G. Huang, H.-Z. Huang, and Y.-F. Li, "A Bidirectional LSTM Prognostics Method Under Multiple Operational Conditions," IEEE Transactions on Industrial Electronics, vol. 66, no. 11, pp. 8792–8802, Nov. 2019.

[10] A. Al-Dulaimi, S. Zabihi, A. Asif, and A. Mohammadi, "A multimodal and hybrid deep neural network model for Remaining Useful Life estimation," Computers in Industry, vol. 108, pp. 186–196, Jun. 2019.

[11] S. Al-Dahidi, F. Di Maio, P. Baraldi, and E. Zio, "Remaining useful life estimation in heterogeneous fleets working under variable operating conditions," Reliability Engineering & System Safety, vol. 156, pp. 109–124, Dec. 2016.

[12] N. Li, Y. Lei, T. Yan, N. Li, and T. Han, "A Wiener-Process-Model-Based Method for Remaining Useful Life Prediction Considering Unit-to-Unit Variability," IEEE Transactions on Industrial Electronics, vol. 66, no. 3, pp. 2092–2101, Mar. 2019.

[13] X. Fang, K. Paynabar, and N. Gebraeel, "Image-Based Prognostics Using Penalized Tensor Regression," Technometrics, vol. 0, no. ja, pp. 1–41, Nov. 2018.

[14] Z. Wang, S. Zeng, J. Guo, and T. Qin, "Remaining capacity estimation of lithium-ion batteries based on the constant voltage charging profile," PLOS ONE, vol. 13, no. 7, pp. 1–22, Jul. 2018, publisher: Public Library of Science.

[15] G. Aydemir and K. Paynabar, "Image-Based Prognostics Using Deep Learning Approach," IEEE Transactions on Industrial Informatics, vol. 16, no. 9, pp. 5956–5964, Sep. 2020.

[16] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in 2008 International Conference on Prognostics and Health Management, Denver, CO, USA, Oct. 2008, pp. 1–9.

[17] N. Gebraeel, A. Elwany, and J. Pan, "Residual Life Predictions in the Absence of Prior Degradation Knowledge," IEEE Transactions on Reliability, vol. 58, no. 1, pp. 106–117, Mar. 2009.

[18] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, Nov. 2016. [Online]. Available: http://www.deeplearningbook.org

[19] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in Proceedings of the 6th International Conference on Neural Information Processing Systems, ser. NIPS'93. Denver, Colorado: Morgan Kaufmann Publishers Inc., Nov. 1993, pp. 737–744.

[20] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a Similarity Metric Discriminatively, with Application to Face Verification," in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1. San Diego, CA, USA: IEEE, 2005, pp. 539–546.

[21] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," Journal of Machine Learning Research, vol. 12, no. Jul, pp. 2121–2159, 2011.

[22] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," International Conference on Learning Representations, 12 2014.

[23] A. Saxena and K. Goebel. (2008, Jan.) Turbofan engine degradation simulation data set. NASA Ames Prognostics Data Repository. [Online]. Available: http://ti.arc.nasa.gov/project/prognostic-data-repository

[24] F. Chollet et al., "Keras," https://keras.io, 2015.