IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 25, 2019, accepted May 28, 2019, date of publication May 31, 2019, date of current version June 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2920297

Validation of Data-Driven Labeling Approaches Using a Novel Deep Network Structure for Remaining Useful Life Predictions

ANDRÉ LISTOU ELLEFSEN[1], SERGEY USHAKOV[2], VILMAR ÆSØY[1], AND HOUXIANG ZHANG[1], (Senior Member, IEEE)

[1]Mechatronics Laboratory, Department of Ocean Operations and Civil Engineering, Norwegian University of Science and Technology Ålesund, 6009 Ålesund, Norway
[2]Department of Marine Technology, Norwegian University of Science and Technology, 7491 Trondheim, Norway

Corresponding author: Andre Listou Ellefsen (andre.ellefsen@ntnu.no)

This work was supported in part by the Department of Ocean Operations and Civil Engineering, Norwegian University of Science and Technology, under Project 90329106, and in part by the Research Council of Norway under Grant 280703.

ABSTRACT Today, most research studies that aim to predict the remaining useful life (RUL) of industrial components based on deep learning techniques are using piecewise linear (PwL) run-to-failure targets to model the degradation process. However, this PwL degradation model assumes a constant initial RUL value in which only time is needed to model normal operating conditions. Thus, it ignores the entire diagnostics aspect. To provide high and reliable RUL prediction accuracy, a prognostics algorithm must incorporate diagnostics information. This paper will provide the Prognostics and Health Management Community an empirical study that validates the PwL degradation model against other, more recent data-driven labeling approaches. We compare three different data-driven labeling approaches for RUL predictions. First, an unsupervised reconstruction-based fault detection algorithm is used to provide valuable diagnostics information. Then, optimized initial RUL values are calculated based on this information. Finally, these values are used to construct PwL, descriptive statistics, and anomaly score function run-to-failure targets for subset FD001 in the popular and publicly available C-MAPSS data set. A deep network structure is proposed and trained on the three different run-to-failure targets in order to predict the RUL. During the training process, a genetic algorithm approach is used to tune a selected search space of hyper-parameters. The results suggest that the network trained on PwL run-to-failure targets with the optimized initial RUL values performs the best and provides the most reliable RUL prediction accuracy. This network also outperforms the most robust results in the literature.

INDEX TERMS Data-driven labeling approaches, deep learning, fault detection, prognostics and health management, remaining useful life.

I. INTRODUCTION

Data-driven Prognostics and Health Management (PHM) applications use algorithms built on sensor measurements to perform fault detection, condition assessment, and remaining useful life (RUL) predictions [1]. Prognostics algorithms predict the progression of faults. Thus, the associated RUL predictions tend to achieve the ideal maintenance policy through predictions of the available time until failure after a fault is detected within the component [2]. In this way, PHM applications have the potential to prevent failures before they occur, and hence, considerably increase operational availability, reliability, and life expectancy of industrial systems.

During the last three years, state-of-the-art deep learning (DL) techniques have outperformed traditional data-driven prognostics algorithms in RUL predictions for engine degradation [3]–[5]. Researchers have typically used the publicly available Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) data set, produced and provided by NASA [6], to train and evaluate the proposed DL approaches. The C-MAPSS data set consists of numerous time series of aircraft gas turbine engines where the engines

The associate editor coordinating the review of this manuscript and approving it for publication was Dong Wang.

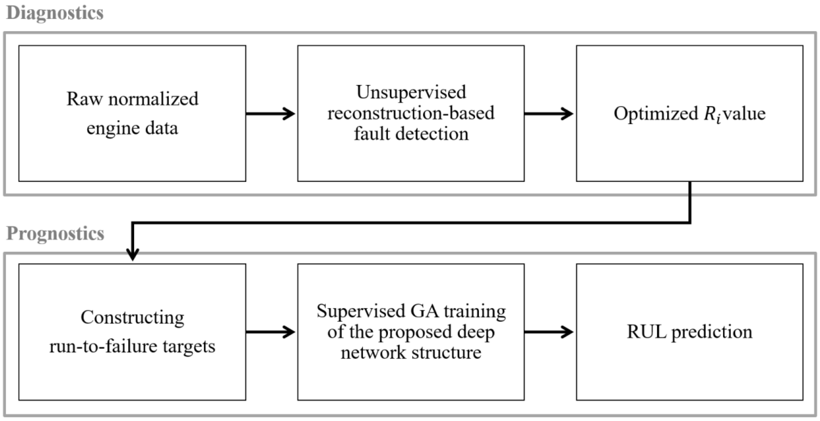

Diagnostics
Raw normalized engine data
Unsupervised reconstruction-based fault detection
Optimized $R_i$value
Prognostics
Constructing run-to-failure targets
Supervised GA training of the proposed deep network structure
RUL prediction

FIGURE 1. **An overview of the complete training structure.**

are subjected to a varying number of time steps and different degrees of degradation. Within the PHM research field, the C-MAPSS data set is acknowledged as the benchmark data set for data-driven prognostics algorithms.

Today, DL techniques that aim to predict RUL still depend on large amounts of run-to-failure targets in order to model the degradation process in the supervised training procedure. Hence, most studies construct run-to-failure targets based on the piece-wise linear (PwL) degradation model, which Heimes [7] proposed in 2008. This degradation model assumes a constant initial RUL (R_i) value when the engines operate in normal conditions. Then, the model degrades linearly until failure after the engines are subjected to a fault, namely, after the fault time step. A subsequent assumption is that all engines utilize the same constant R_i value. In other words, the constructed run-to-failure targets depend on the total number of time steps in each engine and not on the actual degradation process. By the latter assumption, the entire diagnostics aspect is ignored. In real-life PHM applications, any supervised prognostics algorithm should depend on an accurate fault detection algorithm in order to construct reliable run-to-failure targets. Then, the prognostics algorithm is able to model the true degradation process and potentially achieve higher and more reliable RUL prediction accuracy. Therefore, it would be highly beneficial for the PHM community to possess a study that validates the PwL degradation model against other and more recent data-driven labeling approaches.

The objective of this paper is to make a thorough comparison of three different data-driven labeling approaches, based on accurate fault detection, for RUL predictions. First, raw normalized engine data will act as the input for an unsupervised reconstruction-based fault detection algorithm in order to predict the fault time step for each engine [8]. Next, an optimized R_i value for each engine can be obtained. These values are then used to construct PwL, descriptive statistics (DS) [9], in order to model degradation by finding some consistency in the phenomenon leading to failure, and anomaly score function (ASF), which is obtained from the unsupervised reconstruction-based fault detection algorithm, run-to-failure targets for subset FD001 in the C-MAPSS data set. Additionally, this paper proposes a deep network structure for RUL predictions, which will be trained on the three different data-driven labeling approaches. A Genetic Algorithm (GA) approach [5] will also be used to tune hyper-parameters during the supervised training process since each labeling approach requires different values of hyper-parameters within the deep network structure in order to perform with the highest RUL prediction accuracy possible. A flow chart of the complete training structure, where the final RUL prediction incorporates valuable diagnostics information is shown in Figure 1. Finally, the proposed deep network structure trained on the run-to-failure targets with the highest RUL prediction accuracy will be compared to the most robust results in the literature. This is done to demonstrate that prognostics algorithms achieve higher RUL prediction accuracy when trained on run-to-failure targets based on accurate fault detection. This study's main contributions are as follows:

- A comprehensive comparison between PwL, DS, and ASF run-to-failure targets with optimized R_i values is conducted.
- A deep network structure for RUL predictions is proposed.
- The network trained on PwL run-to-failure targets with optimized R_i values outperforms both the networks trained on DS and ASF run-to-failure targets, as well as, the most robust results in the literature with respect to RUL predictions on subset FD001 in the C-MAPSS data set.

The overall organization of the paper is as follows. Section II introduces recent and related work on subset FD001. Section III introduces the necessary background on Feed-forward Neural Network (FNN), Convolutional Neural Network (CNN), Long-Short Term Memory (LSTM), and the proposed deep network structure. The experimental study is elaborated in Section IV. Section V, considers important experimental results and discussions. Finally, Section VI concludes the paper and provides directions for future work.

II. RELATED WORK

Subset FD001 in the C-MAPSS data set has been frequently used to evaluate most DL approaches proposed for RUL predictions in recent years. In data-driven PHM applications, time series data is the standard input format. The LSTM [10] is a well-established DL technique that essentially was designed to process time series data. Zheng *et al.* [11] stacked two LSTM layers, two FNN layers, and a final output layer in order to provide RUL predictions. The proposed approach achieved higher RUL prediction accuracy compared to the Hidden Markov Model and a traditional Recurrent Neural Network (RNN).

A Deep Belief Network (DBN) [12] consists of stacked Restricted Boltzmann Machines (RBMs). Zhang *et al.* [3] proposed a multiple objective evolutionary ensemble learning frameworks for the DBN training process. Consequently, the proposed approach constructs multiple DBNs of varying accuracy and diversity before the evolved DBNs are combined to perform RUL predictions. The proposed approach outperformed several traditional machine learning algorithms, such as Support Vector Machine and Multilayer Perceptron.

During the past decade, CNNs have outperformed more traditional approaches in several domains, including object recognition [13] and face recognition [14]. However, CNNs have also more recently performed excellently on prognostics problems. Li *et al.* [4] proposed a new CNN approach in order to provide RUL predictions. In this approach, all convolution operations are performed in one dimension. Thus, the CNN extracts and learns low-level to high-level representations of each raw sensor measurement from the very start rather than learning the spatial relationship between the sensor measurements and then extracting prognostics information.

Yoon *et al.* [15] used a semi-supervised learning approach to predict the RUL. Their approach included an embedding network obtained from a Variational Autoencoder (VAE) followed by an RNN which was trained based on the latent space defined by the VAE. However, the main goal of this study was to show high RUL prediction accuracy with limited run-to-failure targets in the training procedure.

Ellefsen *et al.* [5] also used a semi-supervised learning approach to predict the RUL. An initial RBM layer was used as an unsupervised pre-training stage in order to initialize the weights in a region near a good local minimum before supervised fine-tuning of the whole network was conducted. The remaining layers of their network consisted of two LSTM layers, one FNN layer, and a final output layer to perform RUL predictions. Additionally, a GA approach was used to tune a big search space of hyper-parameters.

All above-mentioned studies utilize the PwL degradation model with the same constant R_i value for all engines. Even though the constant R_i value varies among different studies, the diagnostics aspect is ignored in these studies. However, one study uses a different degradation model to predict the RUL. Malhotra *et al.* [16] used an LSTM encoder-decoder (LSTM-ED) approach to reconstruct the engines. A reconstruction error was then used to compute a health index (HI) curve for both the training and test set. Then, the HI curves were subjected to normalization and linear regression. Finally, RUL estimations were performed by matching the HI curves. Similar to [16], this study also utilizes a reconstruction error at each time step for each engine to construct an ASF [8]. The ASF will both be used to predict an optimized R_i value for each engine and to create run-to-failure targets as one of the data-driven labeling approaches compared in this study.

III. BACKGROUND

This section will introduce the necessary background on the proposed deep network. First, FNN and the main DL techniques, 1D CNN and LSTM, are defined. Finally, the proposed deep network structure is elaborated.

A. FEED-FORWARD NEURAL NETWORK

FNNs form the basis of the DL techniques used in this study. The objective of this network is to approximate a function f^* by mapping an input $\boldsymbol{x}$ to a target y, that is, $y = f^*(\boldsymbol{x})$. An FNN defines a mapping $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ and learns the value of the parameters $\boldsymbol{\theta}$ (weights and biases) through the back-propagation algorithm [17]. FNNs are typically called networks since they are represented by stacking several layers [18]. Each unit in layer l computes its own activation value:

$$a_j^l = \sigma(z_j^l) \tag{1}$$

where σ is the activation function and the argument is the weighted sum

$$z_j^l = b_j^l + \sum_k w_{jk}^l a_k^{l-1} \tag{2}$$

of the output a_k^{l-1} from unit k in the previous layer $l-1$. b_j^l is the bias and w_{jk}^l are the weight factors. In the first hidden layer $l = 1$, the input is $a_j^0 = x_j$, where $x_j, j = 1 \ldots n$, are the inputs to the FNN. As each layer is fully connected, the weighted sum of the outputs of layer $l - 1$ is over all units k.

B. CONVOLUTIONAL NEURAL NETWORK

CNNs are a specialized kind of FNNs designed for processing multiple arrays of 1D, 2D, or 3D grid-like topology data [18]. Examples of a 1D, 2D, and 3D grid are time series data where each feature is considered as a 1D grid of time steps at regular time intervals, image data is considered as a 2D grid of pixels, and video or volumetric images, respectively. Regardless of the input data, 1D, 2D, and 3D CNNs share the same key advantages, including convolution operations, shared weights, pooling, and the use of many layers [19]. However, the main difference is how the kernel (filter) slides across the data, namely, how the convolution operation is performed.

Today, sensor data is the most common data type format for data-driven PHM applications [2]. Subset FD001 contains

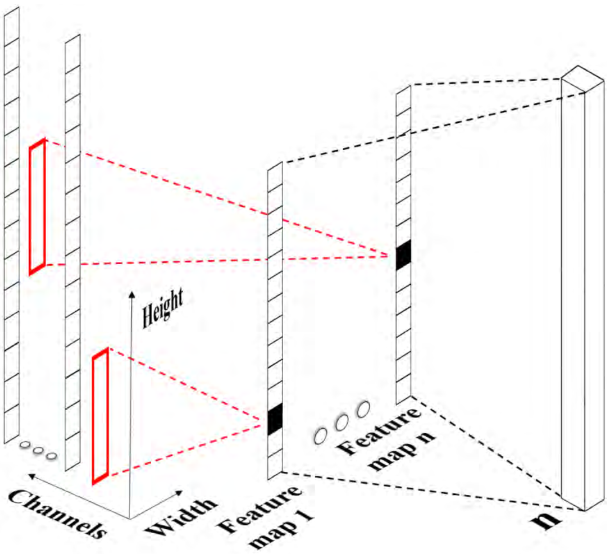


FIGURE 2. An illustration of the 1D convolution operation for multivariate time series data. The red rectangles represent 1D kernels.

several shorter time series of the overall data, where each time series (engine) is subjected to several sensor measurements. The spatial relationship between the sensor measurements is not of great importance [4]. Therefore, 1D CNN is highly suitable and will be used in this study. With respect to mathematical understanding, the convolution operation is typically denoted with an asterisk, and hence, the discrete 1D convolution operation can be defined as [18]:

$$\boldsymbol{s}(t) = (\boldsymbol{x} * \boldsymbol{k})(t) = \sum_{a} \boldsymbol{x}(t-a)\boldsymbol{k}(a) \tag{3}$$

where $\boldsymbol{x} = [x_1 \ldots x_t]$ is a 1D input vector of time steps t, and $\boldsymbol{k}$ is a 1D kernel. The kernel is defined by its height k_h and slides through the whole input vector with a stride equal to one in 1D CNNs. The complete output, $\boldsymbol{s}(t)$, is usually referred to as the feature map. Figure 2 illustrates the 1D convolution operation for multivariate time series data. The height equals the number of time steps, the width is equal to one, and the amount of channels (depth) equals the number of input features. Due to the relatively low input dimension in FD001, pooling will not be used in this study. Like FNNs, CNNs are also trained by the back-propagation algorithm, but the reduced number of parameters and shared weights improve the training efficiency. It should also be noted that CNNs are capable of handling raw normalized input data. Hence, data pre-processing is rare.

C. LONG-SHORT TERM MEMORY

In recent times, the original LSTM [10] has been subjected to adjustments by [20]–[22], and the literature refers to this as the "vanilla LSTM." This study utilizes "vanilla LSTM" with no peephole connections. The LSTM introduces a memory cell that controls the information flow in and out of the cell. Hence, the memory cell is able to maintain its state over time, such that it learns long-term dependencies, and this feature is its superior strength compared to traditional RNNs. The memory cell consists of three non-linear gating units that control and protect the cell state, $\boldsymbol{S}_t$ [23]:

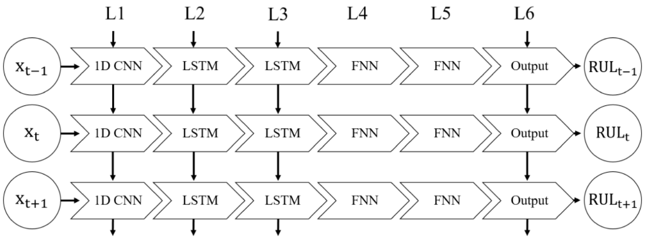


FIGURE 3. The proposed deep network structure.

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{R}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \tag{4}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{R}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \tag{5}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{R}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \tag{6}$$

where σ is the logistic sigmoid gate activation function, $\sigma(x) = \frac{1}{1+e^{-x}}$, which provides a scaled value between 0 and 1. $\boldsymbol{W}$ is the input weight, $\boldsymbol{R}$ is the recurrent weight, and $\boldsymbol{b}$ is the bias weight. The new candidate state values, $\tilde{\boldsymbol{S}}_t$, are created by the tanh layer:

$$\tilde{\boldsymbol{S}}_t = tanh(\boldsymbol{W}_s \boldsymbol{x}_t + \boldsymbol{R}_s \boldsymbol{h}_{t-1} + \boldsymbol{b}_s) \tag{7}$$

The previous cell state, $\boldsymbol{S}_{t-1}$, is updated into the new cell state, $\boldsymbol{S}_t$, by

$$\boldsymbol{S}_t = \boldsymbol{f}_t \otimes \boldsymbol{S}_{t-1} + \boldsymbol{i}_t \otimes \tilde{\boldsymbol{S}}_t \tag{8}$$

where $\otimes$ indicates element-wise multiplication of two vectors. First, $\boldsymbol{f}_t$ decides which historical information the memory cell should forget. Next, $\boldsymbol{i}_t$ determines what new information in $\tilde{\boldsymbol{S}}_t$ the memory cell will input and store in $\boldsymbol{S}_t$. Finally, $\boldsymbol{o}_t$ decides which parts of $\boldsymbol{S}_t$ the memory cell will output:

$$\boldsymbol{h}_t = \boldsymbol{o}_t \otimes tanh(\boldsymbol{S}_t) \tag{9}$$

Through these steps, the LSTM has the power to remove or add information to $\boldsymbol{S}_t$, which makes it extremely fit to process time series data. Like FNNs and CNNs, the LSTM is trained by the back-propagation algorithm.

D. THE PROPOSED DEEP NETWORK STRUCTURE

The proposed deep network structure is shown in Figure 3. In the first layer (L1), a 1D CNN will be utilized to extract and learn low-level temporal features from each sensor measurement individually [4]. These features might contain important degradation information which will then be used to form more complex patterns within the next layers. In both the second and the third layer (L2 and L3), an LSTM layer is used to reveal hidden information and learn long-term dependencies within the features obtained from L1 [5], [11]. Next, an FNN layer is used in both the fourth (L4) and the fifth (L5) layers in order to map all extracted features. In addition, the well-proven regularization technique dropout [24] is applied to L5. Dropout randomly drops units during training. In this way, dropout approximately connects an exponential number of different structures. Thus, the network learns to make generalized representations of the input data, which will prevent the network from extracting the same degradation features repeatedly. In the final layer (L6), a time distributed, fully connected output layer is attached to handle error calculations and perform RUL predictions.

IV. EXPERIMENTAL STUDY

In the following experimental study, all experiments are run on NVIDIA GeForce GTX 1060 6 GB and the Microsoft Windows 10 operating system. The programming language is Java 8 and the deep learning library is "deeplearning4j" (DL4J) version 1.0.0-SNAPSHOT [25]. It should be noted that the DL techniques included in the proposed deep network structure are optimized by the NVIDIA CUDA Deep Neural Network library (cuDNN) [26]. cuDNN is a GPU-accelerated library of primitives for DL techniques. In DL4J, time series data has the following input shape: [miniBatchSize, inputSize, timeSeriesLength], where miniBatchSize is the number of time series in a mini batch, input size is the number of columns, and timeSeriesLength is the total number of time steps in the mini batch. If time series in a mini-batch have variable time step length, the shorter time series are padded with zeros such that the time step lengths are equal to the longest among them. Consequently, mask arrays are used during training. These additional arrays record whether a time step is really present, or whether it is just padding.

A. SUBSET FD001 IN THE BENCHMARK C-MAPSS DATA SET

Subset FD001 consists of 100 time series from aircraft gas turbine engines in both the training and test set. Each engine starts with different degrees of initial wear and manufacturing variation. These initial degradation mechanics are unknown to the public. All engines operate in normal condition at the start, then begin to degrade at an unknown time step during the time series. The degradation in the training set grows in magnitude, namely with increasing acceleration, until failure. The degradation in the test set, however, ends sometime prior to failure. Accordingly, true RUL targets are provided at the last time step for each engine in the test set. The data is contaminated with sensor noise and subset FD001 includes 24 input features: three operational sensor settings and 21 sensor measurements. Please see [27] for a detailed description of each input feature. Table 1 summarizes subset FD001.

TABLE 1. Subset FD001 in the C-MAPSS data set [6].

FD001	
Engines in training set	100
Total number of time steps in training set	20,631
Engines in test set	100
Total number of time steps in test set	13,096
True RUL targets in test set	100
Operating conditions	1
Fault conditions	1

B. PERFORMANCE EVALUATIONS

The scoring function (S) provided in [27] and the root mean square error ($RMSE$) are used in this study as performance evaluations for the test set:

$$S = \begin{cases} \sum_{i=1}^{n} e^{(-\frac{d_i}{13})} - 1, & \text{for} \quad d_i < 0 \\ \sum_{i=1}^{n} e^{(-\frac{d_i}{10})} - 1, & \text{for} \quad d_i \geq 0 \end{cases} \tag{10}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_i^2} \tag{11}$$

where n is the total number of true RUL targets in the test set and $d_i = RUL_{predicted,i} - RUL_{true,i}$. In both performance evaluations, the main objective is to achieve the smallest value possible, that is, when $d_i = 0$. The $RMSE$ gives equal penalty to early and late RUL predictions, namely, when $d_i < 0$ and $d_i > 0$, respectively. In S, however, the penalty for late RUL predictions is larger. This is because late RUL predictions are prone to system failures in real-life PHM applications as maintenance operations will be scheduled too late. On the other hand, early predictions pose less risk to system failures since maintenance operations will be scheduled too early.

Previously, both hold-out and k-fold cross-validation have been used for hyper-parameter tuning on subset FD001 [5], [11]. However, in this study, the total number of time steps in the training set is considered large enough to utilize a hold-out approach, that is, splitting the total training set into 80 engines for training and 20 engines for cross-validation, randomly. In addition to S and $RMSE$, the root mean square error horizon ($RMSE_{hz}$) is used in this study as a performance evaluation for both the training set and the cross-validation set:

$$RMSE_{hz} = \sqrt{\frac{1}{m}\sum_{j=1}^{m} d_j^2} \tag{12}$$

where m is the total number of constructed run-to-failure targets in both the training set and cross-validation set, and $d_j = RUL_{predicted,j} - RUL_{target,j}$. The $RMSE_{hz}$ will be used to compare the true overall prognostics accuracy of the different labeling approaches. The prognostics horizon is a critical measurement designed to evaluate the different labeling approaches with respect to both inherent uncertainties with the degradation process and potential flaws with the constructed run-to-failure targets.

C. DIAGNOSTICS - DETECTING THE FAULT TIME STEP

Ellefsen et al. [8] used an unsupervised reconstruction-based fault detection algorithm for maritime components. Their proposed algorithm is also used in this work in order to predict the fault time step for each engine in FD001. First, a VAE, with three hidden layers and corresponding hidden units (28,14,7) in the encoder and three hidden layers with corresponding hidden units (7,14,28) in the decoder, is trained on normal operating data in an unsupervised manner. It should be noted that the selection process of the hidden units, $h1$, $h2$, and $h3$, is based on the following experience-based formula:

$$h1 = \mathbb{Z}(24 \cdot 1.2) \quad h2 = \mathbb{Z}\left(\frac{h1}{2}\right) \quad h3 = \mathbb{Z}\left(\frac{h2}{2}\right)$$

where 24 is the number of input features in FD001. The initial 25% of each engine is considered normal operating data. Then, the algorithm estimates a raw anomaly score function (ASF) by calculating a reconstruction error, the mean square error (MSE), at each time step for each engine:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} ||\hat{y}_i - y_i||^2 \tag{13}$$

where n is the number of input features, and $\hat{y}_i$ and y_i are the i_{th} predicted and target feature measurement, respectively. Next, the algorithm creates three sliding windows of length w in order to smooth the ASF:

$$w = \frac{T_t}{p} \tag{14}$$

where T_t is the total number of time steps in each engine and p is a tune-able parameter. First, the three windows slide across the raw ASF for each time step. A distance equal to w is used between each sliding window. In order to remove a certain amount of noise in the raw ASF, the average reconstruction error is calculated in the three windows. Since p decides the length of w, it also decides the amount of smoothing performed on the raw ASF. Thus, p should be tuned carefully based on the amount of noise in the raw ASF. In this work, $p = 30$ is used for all engines in order keep the same percentage level, that is $(1/30) \cdot 100 = 3.33\%$, on T_t. This p value will not smooth the raw ASF too much, and hence, keep important degradation trends. Second, the velocity between windows 1 and 2 and between windows 2 and 3 are calculated. Finally, the acceleration between the two velocities is estimated. Please see [8] for a more detailed explanation of the algorithm.

Compared to the data sets used in [8], the nature of degradation is somewhat different in FD001. In this data set, the degradation grows with increasing acceleration until failure. Thus, the highest acceleration, which is used as the fault criterion in [8], is not suitable for FD001. Therefore, an alternative approach for predicting the fault time step $\hat{f}_t$ is used in this study. First, the highest acceleration in normal operating data a_{nod} is calculated for each engine. a_{nod} is equivalent to the maximum increase in deviation between the normal operating sensor measurements. Then, a dynamic acceleration threshold, $a_{Th} = 1.15 \cdot a_{nod}$, is used as the fault criterion in the remaining data for each engine. In this work, the value of 1.15 is based on trial an error. However, this value is a critical parameter and should be tuned carefully for other applications. This value will depend on the nature of degradation. Finally, $\hat{f}_t$ is estimated when the acceleration increases a_{Th}. Thus, the algorithm aims to detect the initial time step where one or several sensor measurements have started to deviate from the normal operating data rapidly. Table 2 shows T_t, $\hat{f}_t$, and the corresponding R_i for each engine in FD001.

TABLE 2. Total time step length T_t, predicted fault time step $\hat{f}_t$, and corresponding initial RUL value R_i for each engine in FD001.

Train set								Cross-validation set			
Engine	T_t	$\hat{f}_t$	R_i	Engine	T_t	$\hat{f}_t$	R_i	Engine	T_t	$\hat{f}_t$	R_i
1	192	63	129	53	195	99	96	2	287	171	116
3	179	47	132	54	257	93	164	5	269	141	128
4	189	62	127	55	193	77	116	7	259	124	135
6	188	86	102	57	137	47	90	13	163	69	94
8	150	53	97	58	147	94	53	21	194	98	96
9	201	86	115	59	231	117	114	31	234	149	85
10	222	70	152	60	172	85	87	33	200	110	90
11	240	120	120	61	185	104	81	42	196	103	93
12	170	94	76	62	180	99	81	46	256	91	165
14	180	76	104	63	174	84	90	47	214	129	85
15	207	86	121	64	283	140	143	50	198	79	119
16	209	72	137	66	201	83	118	52	213	79	134
17	276	123	153	67	312	130	182	56	275	90	185
18	195	71	124	69	362	245	117	65	153	59	94
19	158	39	119	71	208	96	112	68	199	71	128
20	234	104	130	72	213	83	130	70	137	84	53
22	202	103	99	73	213	104	109	74	166	75	91
23	168	94	74	75	229	94	135	81	239	100	139
24	147	60	87	76	210	147	63	93	155	57	98
25	230	129	101	77	154	38	116	94	258	101	157
26	199	101	98	78	231	91	140				
27	155	85	70	79	199	114	85				
28	165	72	93	80	185	58	127				
29	163	100	63	82	214	106	108				
30	194	116	78	83	293	176	117				
32	191	121	70	84	267	135	132				
34	194	66	128	85	188	88	100				
35	181	111	70	86	278	119	159				
36	158	52	106	87	178	105	73				
37	170	77	93	88	213	80	133				
38	194	114	80	89	217	105	112				
39	128	58	70	90	154	73	81				
40	188	79	109	91	135	46	89				
41	216	88	128	92	340	167	173				
43	207	98	109	95	283	114	169				
44	192	131	61	96	336	204	132				
45	158	73	85	97	202	66	136				
48	231	81	150	98	156	68	88				
49	215	75	140	99	185	74	111				
51	213	92	121	100	200	71	129				

D. DATA-DRIVEN LABELING APPROACHES

This study compares three different data-driven labeling approaches for constructing run-to-failure targets. The optimized R_i values in Table 2 are used to construct run-to-failure targets based on the PwL degradation model, DS, and on the raw ASF obtained from the anomaly detector in Section IV-C.

1) PIECE-WISE LINEAR

In the original PwL degradation model by Heimes [7], all engines in the training and cross-validation sets utilize the same R_i value when the engines operate in normal condition. The major limitation of this assumption is that the fault time step for each engine depends on T_t and not on the actual degradation pattern. Actually, each engine has an individual degradation pattern [5]. Therefore, the PwL degradation model used in this study utilizes an optimized R_i value for each engine. These R_i values are dependent on the actual degradation pattern in each engine. Algorithm 1 shows the procedure on how to construct PwL run-to-failure targets for engine i.

2) DESCRIPTIVE STATISTICS

DS [9] aims to find some consistency in the phenomenon leading to failure. In other words, there are typical values of

Algorithm 1 Algorithm for Constructing Piece-Wise Linear Run-to-Failure Targets for Engine i

```
Input: $T_t, \hat{f}_t, R_i$
Output: $PwL_i$
    for $t := 0$ to $T_t$ do
        if $(t \leq \hat{f}_t)$ then
            $PwL_i \leftarrow R_i$
        else
            $PwL_i \leftarrow (T_t - t)$
        end if
    end for
    return $PwL_i$
```

the sensor measurements at the failure time step (F) for each engine in both the training set and cross-validation set. Previous research has proven that sensors 2, 3, 4, 7, 11, 12, and 15 are subjected to a clear degradation trend and that they are contaminated with less noise than the remaining sensors [28]. This sensor selection process is of high importance for the degradation precision of the subsequent constructed run-to-failure targets. First, the mean values of F in the selected sensors are calculated:

$$\begin{aligned} E(X(F)) &= \left[E(x^2(F)), \ldots, E(x^{15}(F))\right] \\ &= \left[\frac{1}{m}\sum_{i\in I} x_i^2(F_i), \ldots, \frac{1}{m}\sum_{i\in I} x_i^{15}(F_i)\right] \\ &= \left[E^2, \ldots, E^{15}\right] \end{aligned} \tag{15}$$

where m is the number of failures, I is the set of engines that experienced a failure, F_i is the failure time step of engine i, and $E(X(F))$ is the vector of mean values observed at each failure time step. Second, the mean values are used to construct run-to-failure targets at any time t up until failure for engine i:

$$\begin{aligned} Y_i(t) &= X_i(t) - E(X(F)) \\ &= \left[\left(x_i^2(t) - E^2\right)^2 + \cdots + \left(x_i^{15}(t) - E^{15}\right)^2\right]^{\frac{1}{2}} \end{aligned} \tag{16}$$

where $Y_i(t)$ is the raw run-to-failure targets. Third, the raw run-to-failure targets are scaled according to the R_i value obtained from Table 2 for each engine:

$$DS_i(t) = \frac{R_i \cdot (Y_i(t) - Y_i(T_t))}{Y_i(t_1) - Y_i(T_t)} \tag{17}$$

where $Y_i(t)$ is the current raw run-to-failure target, $Y_i(T_t)$ is the last raw run-to-failure target, and $Y_i(t_1)$ is the first raw run-to-failure target. Finally, polynomial regression is performed on $DS_i(t)$ in order to remove noise. It should be noted that the polynomial regression used in this study performs a QR decomposition of the underlying Vandermonde matrix and the degree of the polynomial is 2. Figure 4 compares the raw DS targets and DS targets with polynomial regression.

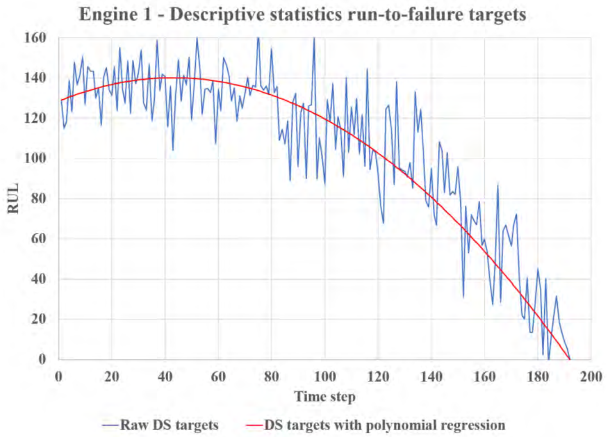


FIGURE 4. Comparison between raw DS targets and DS targets with polynomial regression for engine 1.

Algorithm 2 Algorithm for Constructing a Smooth Version of the Anomaly Score Function for Each Engine i

```
Input: $ASF_i(t)$, $w_s$, $T_t$
Output: $ASF_i(t)_s$
    $w_s \leftarrow T_t \ / \ 1$
    Creating one sliding window $SW$ of length $w_s$ which
    slides across $ASF_i(t)$ for each time step $t$.
    for $t := 0$ to $T_t$ do
        $SW \leftarrow ASF_i(t)$
        $SW_{sum} \leftarrow 0$
        for $s := 0$ to $w_s$ do
            $SW_{sum} += SW(s)$
        end for
        $ASF_i(t)_s \leftarrow \frac{SW_{sum}}{w_s}$
    end for
    return $ASF_i(t)_s$
```

3) ANOMALY SCORE FUNCTION

First, the raw ASF for each engine $ASF_i(t)_r$ is scaled according to the R_i value obtained from Table 2 for each engine:

$$ASF_i(t) = \frac{R_i \cdot (ASF_i(t)_r - ASF_i(T_t)_r)}{ASF_i(t_1)_r - ASF_i(T_t)_r} \tag{18}$$

where $ASF_i(t)_r$ is the current raw run-to-failure target, $ASF_i(T_t)_r$ is the last raw run-to-failure target, and $ASF_i(t_1)_r$ is the first raw run-to-failure target. Finally, in order to remove noise and make a smooth version, an additional sliding window SW of length $w_s = T_t/1$ is created. This sliding window slides across $ASF_i(t)$ for each time step t. Algorithm 2 shows the procedure on how to construct the smooth anomaly score function $ASF_i(t)_s$ for engine i. Figure 5 compares the raw ASF targets and the smooth ASF targets.

4) SELECTED DATA-DRIVEN LABELING APPROACHES

In the following experiments, the PwL, the DS with polynomial regression, and the smooth ASF targets will be

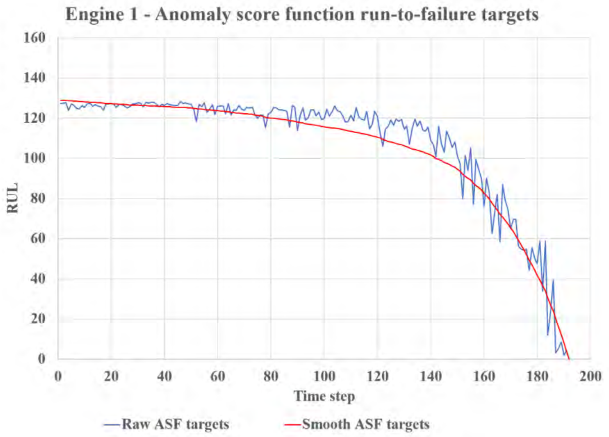


FIGURE 5. Comparison between raw ASF targets and smooth ASF targets for engine 1.

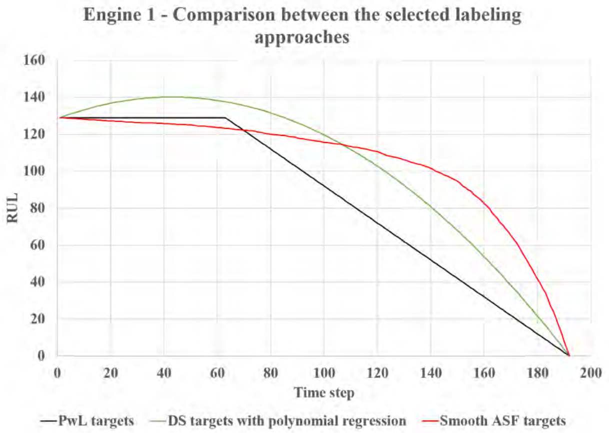


FIGURE 6. Comparison between the selected data-driven labeling approaches for engine 1.

used as supervised run-to-failure training targets for subset FD001. Figure 6 compares the selected data-driven labeling approaches.

E. DATA AUGMENTATION AND NORMALIZATION

Each input measurement x_n in the training set is normalized with zero mean and unit variance (z-score) normalization:

$$\hat{x}_n = \frac{x_n - \mu}{\sigma} \tag{19}$$

where μ and σ is the mean and the corresponding standard deviation of the population, respectively. Then, the normalization statistics obtained from the training set are applied to both the cross-validation set and the test set. Additionally, to reduce overfitting, random white Gaussian noise, g, is added to each $\hat{x}_n$ in each engine in the training set. P_{signal} and P_{noise} are the average power of the signal and the noise, respectively, and defined as follows:

$$P_{signal} = \frac{1}{T_t}\sum_{t=1}^{T_t}\left(\sqrt{\frac{1}{n}(\hat{x}_1^2 + \cdots + \hat{x}_n^2)}\right)_t \tag{20}$$

$$P_{noise} = \frac{1}{T_t}\sum_{t=1}^{T_t}\left(\sqrt{\frac{1}{n}\left((\hat{x}_1+g)^2+\ldots+(\hat{x}_n+g)^2\right)}\right)_t \tag{21}$$

where T_t is the total time step length of each engine and n is the number of input features. Then, the signal-to-noise-ratio (SNR) can be defined as:

$$SNR(\%) = \frac{P_{signal}}{P_{noise}} \cdot 100 \tag{22}$$

In all experiments, 95% SNR is applied to the training set.

F. NETWORK CONFIGURATION AND TRAINING

Deep networks introduce several hyper-parameters, which are both challenging and time-consuming to optimize in the training procedure. Additionally, the proposed deep network structure requires different values of hyper-parameters for each labeling approach in order to perform with the highest RUL prediction accuracy possible. Thus, the proposed GA approach in [5] will also be used in this study in order to optimize the hyper-parameters for the networks trained on the three labeling approaches in an efficient manner.

The GA is a metaheuristic inspired by the natural selection process [29]. It is an effective algorithm for finding a near-optimal solution in a big search space, in this case, a big search space of hyper-parameters. However, in order to slightly reduce the search space, the networks will use some joint-hyper parameters which previously have shown great results on subset FD001 [4], [5]. Stochastic gradient descent (SGD) is the selected optimization algorithm and adaptive moment estimation (Adam) is the learning rate method [30]. To better preserve the low-level temporal features obtained from the 1D CNN layer, the learning rate in L1 is $l_r = 5 \cdot 10^{-5}$, while the learning rate in the remaining layers is $l_r = 1 \cdot 10^{-5}$. Xavier weight initialization [31] is applied to all layers. The rectified linear unit activation function [32] is used in both 1D CNN and FNN layers. However, in the LSTM layers, the tanh activation function is used in order to push the input and output values between -1 and 1. The mini-batch size is five engines, as previously optimized in [5]. The selected joint hyper-parameters are summarized in Table 3.

Table 4 shows the hyper-parameters which the GA approach optimized for each of the three networks. n is the number of hidden units in each layer, k_h is the kernel height in L1, and p is the dropout retaining probability of each unit in L5. A p value of 1.0 is functionally equivalent to zero dropout, namely, 100% probability of retaining each hidden unit. First, the GA approach selects random values of each hyper-parameter. One such set of random hyper-parameters is called an individual and a set of individuals is called a population. Each individual in the population is trained on the training set and evaluated on the cross-validation set. The $RMSE_{hz}$, equation 12, is the selected objective function. To prevent overfitting, early stopping is applied to monitor the

TABLE 3. Joint hyper-parameters.

Hyper-parameter	Method/value
Optimization algorithm	SGD
l_r method	Adam
l_r L1	$5 \cdot 10^{-5}$
l_r remaining layers	$1 \cdot 10^{-5}$
Weight initialization	Xavier
Activation function	ReLU (tanh in LSTM)
Mini-batch	5 engines

TABLE 4. Selected hyper-parameters in the GA approach.

Hyper-parameter	Values
n L1	32, 48, 64
n L2	128, 192, 256
n L3	64, 96, 128
n L4	64, 96, 128
n L5	16, 32, 48
k_h L1	4, 6, 8, 10
p L5	0.5, 0.6, 0.7, 0.8
$l2$ regularization	$1 \cdot 10^{-4}$, $1 \cdot 10^{-5}$, $1 \cdot 10^{-6}$

TABLE 5. Parameters of the GA approach.

Parameter	Value
Population size	30
Nr of Elite	1
Mutation Rate	0.5
Mutation Gain	0.3
Evolution iterations	4

performance during the training process of each individual. If the number of epochs with no reduction on $RMSE_{hz}$ on the cross-validation set exceeds four, the training process is terminated. Then, the network, in the epoch with the lowest $RMSE_{hz}$, is saved.

To limit the time consumed during the optimization process, the population size is restricted to 30 individuals. The best individual from the population is then kept and used as the parent for the next generation of hyper-parameters. Additionally, some random mutation is performed after the crossover for increasing the exploration of the algorithm. The population is evolved four times. This results in an average training time of 13.33 hours for each labeling approach, where each individual trained for 80 epochs on average with an average training time per epoch of 5 seconds. The parameters of the GA approach are shown in Table 5. In the end, the top five GA individuals for each labeling approach are evaluated on the test set where both $RMSE$ and S are calculated. The GA individuals with the best result on the test set for each labeling approach are shown in Table 6 and the corresponding RUL prediction accuracy are shown in Table 7.

V. EXPERIMENTAL RESULTS AND DISCUSSIONS

The aim of this paper is to make a thorough comparison of three different data-driven labeling approaches for RUL predictions. The degradation significance within each of the constructed run-to-failure targets is extremely important for the RUL prediction performance of the proposed deep network structure. First, the GA optimized networks, as seen in Table 6, for the three labeling approaches are compared with three different performance evaluations on both the training set and the cross-validation set. Finally, the network with the highest RUL prediction accuracy on the test set is compared to the most robust results in the literature.

A. COMPARISON BETWEEN THE DATA-DRIVEN LABELING APPROACHES

The $RMSE_{hz}$ accuracy is considered an important performance indicator since it evaluates how accurately the networks are able to model the true overall degradation process in both the training set and cross-validation set. In addition, high $RMSE_{hz}$ accuracy is critical in order to achieve reliable confidence intervals for the corresponding RUL prediction in real-life PHM applications. As shown in Table 7, the network trained on PwL targets outperforms both the networks trained on DS and ASF targets with respect to the $RMSE_{hz}$ accuracy.

Both the $RMSE$ and S accuracy are important performance indicators since high and reliable RUL prediction accuracy at the very end of the engines lifetime have great significance for real-life PHM applications. Thus, $RMSE$ and S are only calculated at the last time step for each engine. It should be noted that both $RMSE$ and S is the overall accuracy of all engines. In other words, the overall accuracy of 80 engines in the training set, 20 engines in the cross-validation set, and 100 engines in the test set. Additionally, to prevent overfitting, both dropout and random white Gaussian noise will reduce the accuracy on the training set compared to the accuracy on the cross-validation set. As shown in Table 7, the networks trained on PwL and DS targets perform with satisfactory $RMSE$ and S accuracy. The network trained on ASF targets, however, performs with unacceptable $RMSE$ and S accuracy. This is mainly because the run-to-failure targets decrease with increasing acceleration until failure. Thus, the network struggles to predict the failure ASF target for each engine, that is, when RUL $= 0$ in both the training set and the cross-validation set. This also indicates that the predicted ASF targets are prone to late RUL predictions, namely, when $RUL_{predicted} - RUL_{true} > 0$. This reflects the extremely low S accuracy. Late RUL predictions could cause serious system failures in real-life PHM applications as maintenance operations will be scheduled too late.

In Figure 7, engines 2, 21, 52, and 70 in the cross-validation set are randomly selected for comparison. As previously mentioned, all three labeling approaches utilize an optimized R_i value for each engine. The high variance in R_i between engines in a mini-batch makes it difficult for the networks to predict the run-to-failure targets when the engines are operating in normal condition. Additionally, each engine in a mini-batch has different T_t. Thus, the shorter engines are padded with zeros such that all T_t are equal. Accordingly,

TABLE 6. GA individuals.

Labeling approach	Layer index	DL technique	nIn	nOut	Params	Total params	k_h L1	p L5	$l2$ regularization
PwL	1	1D CNN	24	32	4640	516,289	6	0.8	$1 \cdot 10^{-4}$
	2	LSTM	32	256	295,936				
	3	LSTM	256	128	197,120				
	4	FNN	128	128	16,512				
	5	FNN	128	16	2064				
	6	Output	16	1	17				
DS	1	1D CNN	24	32	4640	397,313	6	0.8	$1 \cdot 10^{-5}$
	2	LSTM	32	256	295,936				
	3	LSTM	256	64	82,176				
	4	FNN	64	128	8320				
	5	FNN	128	48	6192				
	6	Output	48	1	49				
ASF	1	1D CNN	24	64	6208	487,041	4	0.5	$1 \cdot 10^{-5}$
	2	LSTM	64	256	328,704				
	3	LSTM	256	96	135,552				
	4	FNN	96	128	12,416				
	5	FNN	128	32	4128				
	6	Output	32	1	33				

TABLE 7. The RUL prediction accuracy on subset FD001 for the three data-driven labeling approaches.

Labeling approach	Data set	S	$RMSE$	$RMSE_{hz}$
PwL	Training set	215.32	12.63	20.20
	Cross-validation set	23.69	7.84	25.03
	Test set	185.54	12.08	–
DS	Training set	307.29	13.97	22.34
	Cross-validation set	22.19	7.51	28.76
	Test set	348.85	14.75	–
ASF	Training set	1250.76	23.29	22.35
	Cross-validation set	68.62	14.60	29.77
	Test set	5305.78	29.85	–

TABLE 8. *S* and *RMSE* comparison with the literature on the test set of subset FD001.

Author & Refs.	Year	Approach	S	$RMSE$
Ramasso [33]	2014	RULCLIPPER	216	13.27
Malhotra et al. [16]	2016	LSTM-ED	256	12.81
Zheng et al. [11]	2017	LSTM + FNN	338	16.14
Zhang et al. [3]	2017	MODBNE	334	15.04
Yoon et al. [15]	2017	VAE + RNN	419	14.80
Li et al. [4]	2018	CNN + FNN	274	12.61
Ellefsen et al. [5]	2019	RBM + LSTM + FNN	231	12.56
Ellefsen et al.	2019	1D CNN + LSTM + FNN	**186**	**12.08**

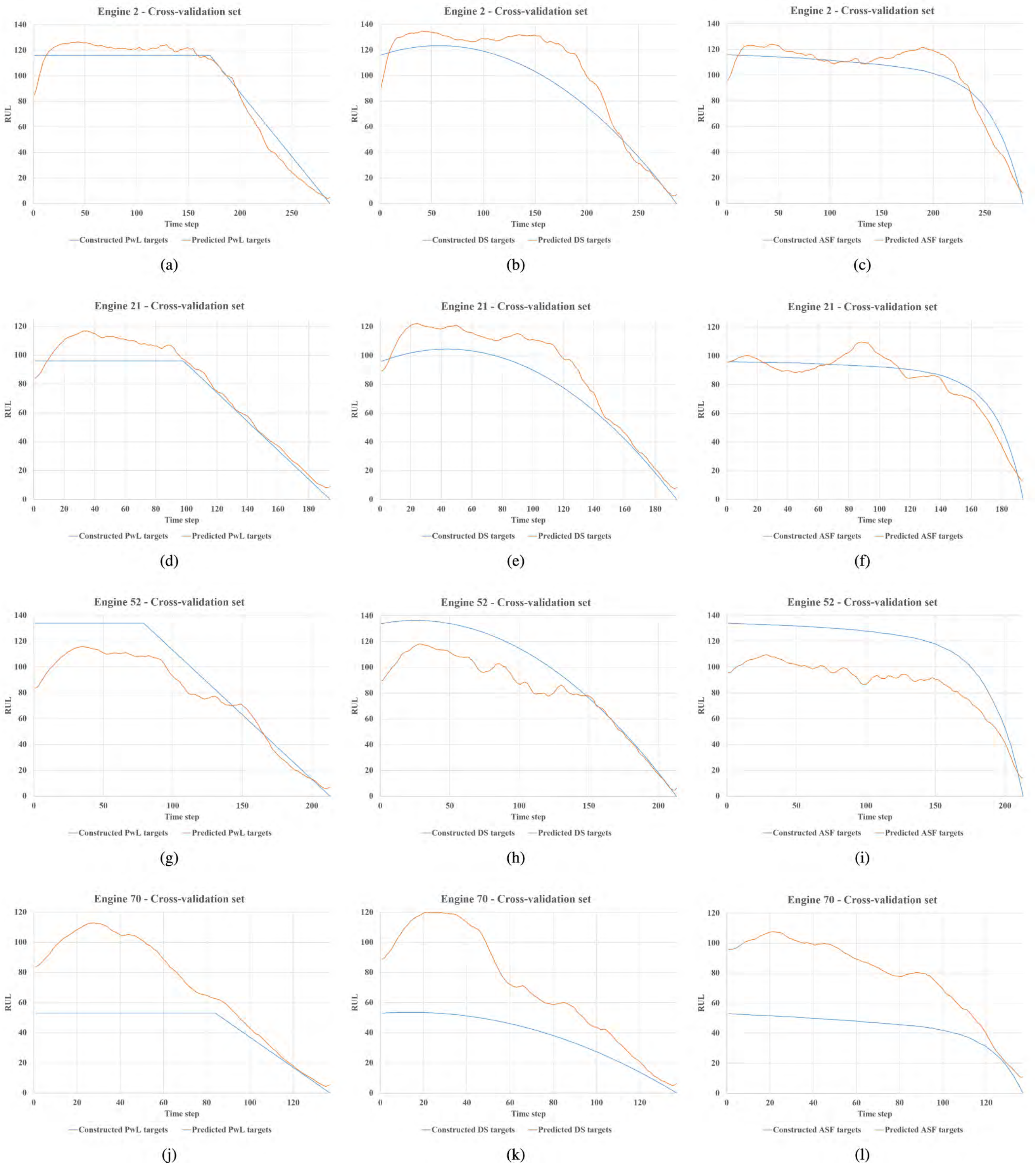
mask arrays are used during the training process in order not to include the padded zeros in the performance evaluations. These masking arrays consist of the same value for each engine. The values are 82.6, 88.2, and 95.5, for the networks trained on PwL, DS, and ASF targets, respectively. Each network starts to predict based on its masking array value so that they do not start predicting on zero for each engine. Thus, this predicting approach is not optimal for the engines that are utilizing a R_i value either lower or higher than the masking array value. This is illustrated in Figure 7.

Nevertheless, the optimized R_i values are based on the degradation process rather than the number of time steps. Hence, the network trained on PwL targets predicts $RMSE_{hz}$, $RMSE$, and S with high accuracy after the predicted fault time step, that is, in the faulty degradation data of the engines lifetime. Thus, the optimized R_i values enable this network to generalize well on data never seen before, namely, the test set. Based on the superior results on the test set, the PwL degradation model is able to construct the most reliable run-to-failure targets for RUL predictions. PwL targets are also highly suitable if the RUL is to be considered as a time-based index, e.g., if the RUL decreases by one and the time step increases by one. This could be highly relevant for real-life PHM applications.

B. COMPARISON WITH THE LITERATURE

The network trained on PwL targets with optimized R_i values was able to generalize well, and hence, performed the highest RUL prediction accuracy on the test set. Thus, this network is compared with the literature. The authors have tried to include the most robust and recent results for comparison. That's why the well-known RULCLIPPER is also included. The RULCLIPPER does not utilize any DL techniques to make RUL predictions. Instead, it predicts the RUL based on imprecise health indicators modeled by planar polygons and similarity-based reasoning [33].

In Table 6, the selected studies are arranged in descending order based on the year they are published. As opposed to [33], the remaining studies utilize prognostics algorithms based on DL techniques to predict the RUL. However, most of these studies do not incorporate diagnostics information since the algorithms are trained on PwL run-to-failure targets with the same R_i value for all engines. On the other hand, the proposed deep network in this study is trained on PwL run-to-failure targets with optimized R_i values for each engine. Thus, the network takes into account the diagnostics aspect before making any RUL predictions. The high generalization towards the test set indicates that the optimized R_i values enable the network to model the true degradation process within subset FD001. To the best of the authors' knowledge, the proposed deep network, when trained on PwL run-to-failure targets with optimized R_i values, provides higher RUL prediction accuracy on subset FD001 than any in the literature.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

FIGURE 7. Cross-validation set comparison. (a) Engine 2 - PwL targets. (b) Engine 2 - DS targets. (c) Engine 2 - ASF targets. (d) Engine 21 - PwL targets. (e) Engine 21 - DS targets. (f) Engine 21 - ASF targets. (g) Engine 52 - PwL targets. (h) Engine 52 - DS targets. (i) Engine 52 - ASF targets. (j) Engine 70 - PwL targets. (k) Engine 70 - DS targets. (l) Engine 70 - ASF targets.

VI. CONCLUSION AND FUTURE WORK

This paper has compared three different data-driven labeling approaches for constructing run-to-failure targets. Additionally, a deep network structure has been proposed for RUL predictions. The experiments are performed on subset FD001 in the publicly available C-MAPSS data set. Most research studies that aim to predict the RUL based on DL approaches are still using the PwL degradation model to construct run-to-failure targets. This model assumes a constant R_i value that only depends on time to model normal operating conditions. Hence, it neglects the entire diagnostics aspect. As illustrated in this study, any supervised prognostics algorithm should consider the diagnostics aspect before making any RUL predictions to achieve higher and more reliable accuracy. Thus, an unsupervised reconstruction-based fault detection algorithm has been used in this study to predict the fault time step for each engine. Then, an optimized R_i value for each engine was obtained. These R_i values were then used in the construction process of PwL, DS, and ASF run-to-failure targets. Finally, the proposed deep network structure was trained on the three different constructed run-to-failure targets. Additionally, a GA approach was used to tune the search space of hyper-parameters.

The network trained on PwL run-to-failure targets with optimized R_i values outperformed both the networks trained on DS and ASF run-to-failure targets with respect to RUL predictions. Additionally, this network outperformed the most robust results in the literature. The optimized R_i values are based on the individual degradation process in each engine. Hence, the network predicts $RMSE_{hz}$, $RMSE$, and S with high accuracy in the faulty degradation data of the engine's lifetime. The optimized R_i values enable the network to generalize well on data never seen before. The strong generalization indicates that the network is able to model the true degradation processes within the data set before making any RUL predictions. In other words, the diagnostics aspect is incorporated.

In this work, it was also discovered that the high variance in R_i between engines in a mini-batch made it difficult for the networks to predict the run-to-failure targets when the engines were operating in normal condition. To solve this issue we propose the following. First, a_{Th} can be further optimized in a more generic way for each engine. Second, the utilization of bigger (more parameters) and possibly deeper (more layers) networks. Finally, more training data with more engines with similar degradation processes, namely, with similar R_i values, would be favorable. Future work will address these issues.

Subset FD001 only contains one fault mode and one operating condition. If, however, several operating conditions were introduced in the data set, the unsupervised reconstruction-based fault detection algorithm could face some problems since the sensor measurements might differ strongly between different time steps with different operating conditions. This issue will also be explored in future work.

ACKNOWLEDGMENT

The authors would like to thank Digital Twins For Vessel Life Cycle Service (DigiTwin).

REFERENCES

[1] P. W. Kalgren, C. S. Byington, M. J. Roemer, and M. J. Watson, "Defining PHM, A lexical evolution of maintenance and logistics," in *Proc. IEEE Autotestcon*, Sep. 2006, pp. 353–358.

[2] A. L. Ellefsen, V. Æsøy, S. Ushakov, and H. Zhang, "A comprehensive survey of prognostics and health management based on deep learning for autonomous ships," *IEEE Trans. Rel.*, vol. 68, no. 2, pp. 720–740, 2019. doi: 10.1109/TR.2019.2907402.

[3] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan, "Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2306–2318, Oct. 2017.

[4] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Rel. Eng. Syst. Saf.*, vol. 172, pp. 1–11, Apr. 2018.

[5] A. L. Ellefsen, E. Bjørlykhaug, V. Æsøy, S. Ushakov, and H. Zhang, "Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture," *Rel. Eng. Syst. Saf.*, vol. 183, pp. 240–251, Mar. 2019.

[6] A. Saxena and K. Goebel. Turbofan engine degradation simulation data set. NASA Ames Prognostics Data Repository, NASA Ames Research Center, Moffett Field, CA, USA. [Online]. Available: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

[7] F. O. Heimes, "Recurrent neural networks for remaining useful life estimation," in *Proc. Int. Conf. Prognostics Health Manage. (PHM)*, Oct. 2008, pp. 1–6.

[8] A. L. Ellefsen, E. Bjørlykhaug, V. Æsøy, and H. Zhang, "An unsupervised reconstruction-based fault detection algorithm for maritime components," *IEEE Access*, vol. 7, pp. 16101–16109, 2019.

[9] K. Le Son, A. Barros, M. Fouladirad, E. Levrat, and B. Iung, "Remaining useful life estimation based on probabilistic model," in *Proc. 17th ISSAT Int. Conf. Rel. Qual. Design*, 2011, pp. 10–25.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long short-term memory network for remaining useful life estimation," in *Proc. IEEE Int. Conf. Prognostics Health Manage. (ICPHM)*, Jun. 2017, pp. 88–95.

[12] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105.

[14] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proc. IEEE Conf. CVPR*, Jun. 2014, pp. 1701–1708.

[15] A. S. Yoon, T. Lee, Y. Lim, D. Jung, P. Kang, D. Kim, K. Park, and Y. Choi, "Semi-supervised learning with deep generative models for asset failure prediction," *CoRR*, vol. abs/1709.00845, pp. 1–9, Sep. 2017.

[16] P. Malhotra, V. Tv, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Multi-sensor prognostics using an unsupervised health index based on LSTM encoder-decoder," in *Proc. Workshop Mach. Learn. Prognostic Health Manage.*, 2016, pp. 1–10.

[17] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 2012, pp. 9–48.

[18] I. Goodfellow, Y. Bengio, A. Courville, and F. Bach, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[20] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," in *Proc. 9th Int. Conf. Artif. Neural Netw. (ICANN)*, vol. 2, Sep. 1999, pp. 850–855.

[21] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, no. 5, pp. 602–610, 2005.

[22] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 3, Jul. 2000, pp. 189–194.

[23] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[25] Apache Software Foundation License 2.0. (2018). *Eclipse Deeplearning4j Development Team, Deeplearning4j: Open-Source Distributed Deep Learning for the JVM*. [Online]. Available: http://deeplearning4j.org

[26] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cuDNN: Efficient primitives for deep learning," 2014, *arXiv:1410.0759*. [Online]. Available: https://arxiv.org/abs/1410.0759

[27] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *Proc. Int. Conf. Prognostics Health Manage.*, Oct. 2008, pp. 1–9.

[28] T. Wang, J. Yu, D. Siegel, and J. Lee, "A similarity-based prognostics approach for remaining useful life estimation of engineered systems," in *Proc. Int. Conf. Prognostics Health Manage. (PHM)*, Oct. 2008, pp. 1–6.

[29] V. Roberge, M. Tarbouchi, and G. Labontè, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 132–141, Feb. 2013.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980

[31] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[32] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," *J. Mach. Learn. Res.*, vol. 15, no. 4, pp. 315–323, 2011.

[33] E. Ramasso, "Investigating computational geometry for failure prognostics," *Int. J. Prognostics Health Manage.*, vol. 5, no. 1, p. 005, 2014.

ANDRÉ LISTOU ELLEFSEN received the master's degree in subsea technology from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 2016. He is currently pursuing the Ph.D. degree with NTNU, Ålesund, Norway, as part of the Mechatronics Laboratory, Department of Ocean Operations and Civil Engineering. His current research interests include artificial intelligence, deep learning, decision support, predictive maintenance, prognostics and health management, and digital twins.

SERGEY USHAKOV received the Ph.D. degree from the Department of Marine Technology, Norwegian University of Science and Technology, in 2012 with a focus on the measurement and characterization of particulate matter emissions from marine diesel engines, where he rejoined, in 2016, as a Professor in marine machinery. For several years, he was with MARINTEK (currently SINTEF Ocean) within the fields of marine diesel engine emission characterization and emission reduction technologies covering both volatile and non-volatile exhaust emissions. During this work, he was involved in a number of bigger and smaller research projects, where accumulated substantial experience with experimental work both in laboratory and on board of different vessels. The current research focus is environmentally friendly shipping as well as the improvement of marine diesel engines' efficiency, especially emphasizing the experimental part of this work.

VILMAR ÆSØY graduated from NTNU, in 1989, and continued his research on natural gas fueled marine engines at NTNU/MARINTEK, in 1997. In 1996, he received the Ph.D. degree for his research on natural gas ignition and combustion through experimental investigations and numerical simulations. From 1989 to 1997, he was involved in several large R&D projects developing gas fueled engines and fuel injection systems for the diesel engine manufacturers, Wärtsilä and Bergen Diesel (Roll-Royce). From 1998 to 2002, he was an R&D Manager for Rolls-Royce Marine Deck Machinery. Since 2002, he has been employed in teaching with the Aalesund University College, developing and teaching courses in marine product and systems design on bachelor's and master's level. In 2010, he received the Green Ship Machinery Professorship. His special research interest is within the field of energy and environmental technology, with a focus on combustion engines and the need for more environmental friendly and energy-efficient systems.

HOUXIANG ZHANG (M'04–SM'12) received the Ph.D. degree in mechanical and electronic engineering from the Robotics Institute, Beihang University, in 2003, and the Habilitation degree in informatics from the University of Hamburg, Germany, in 2011. Since 2004, he has been a Postdoctoral Fellow and a Senior Researcher with the Department of Informatics, Faculty of Mathematics, Informatics and Natural Sciences, Institute of Technical Aspects of Multimodal Systems (TAMS), University of Hamburg. He joined Norwegian University of Science and Technology, Norway, in 2011, where he is currently a Professor in mechatronics. He has involved in two main research areas: 1) biological robots and modular robotics, especially on biological locomotion control, and 2) virtual prototyping in demanding marine operation. He has applied for and coordinated more than 20 projects supported by the Norwegian Research Council (NFR), the German Research Council (DFG), and the industry. In these areas, he has published over 160 journal and conference papers as an author or a coauthor. He has received four best paper awards and four finalist awards for the best conference paper at the International conference on Robotics and Automation.