Reliability Engineering and System Safety 183 (2019) 240–251

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture

André Listou Ellefsen*,[a], Emil Bjørlykhaug[a], Vilmar Æsøy[a], Sergey Ushakov[b], Houxiang Zhang[a]

[a] Department of Ocean Operations and Civil Engineering, Norwegian University of Science and Technology, Aalesund 6009, Norway
[b] Department of Marine Technology, Norwegian University of Science and Technology, Trondheim 7491, Norway

ARTICLE INFO

Keywords:
C-MAPSS
Deep learning
Genetic algorithm
Prognostics and health management
Remaining useful life
Semi-supervised learning

ABSTRACT

In recent years, research has proposed several deep learning (DL) approaches to providing reliable remaining useful life (RUL) predictions in Prognostics and Health Management (PHM) applications. Although supervised DL techniques, such as Convolutional Neural Network and Long-Short Term Memory, have outperformed traditional prognosis algorithms, they are still dependent on large labeled training datasets. With respect to real-life PHM applications, high-quality labeled training data might be both challenging and time-consuming to acquire. Alternatively, unsupervised DL techniques introduce an initial pre-training stage to extract degradation related features from raw unlabeled training data automatically. Thus, the combination of unsupervised and supervised (semi-supervised) learning has the potential to provide high RUL prediction accuracy even with reduced amounts of labeled training data. This paper investigates the effect of unsupervised pre-training in RUL predictions utilizing a semi-supervised setup. Additionally, a Genetic Algorithm (GA) approach is applied in order to tune the diverse amount of hyper-parameters in the training procedure. The advantages of the proposed semi-supervised setup have been verified on the popular C-MAPSS dataset. The experimental study, compares this approach to purely supervised training, both when the training data is completely labeled and when the labeled training data is reduced, and to the most robust results in the literature. The results suggest that unsupervised pre-training is a promising feature in RUL predictions subjected to multiple operating conditions and fault modes.

1. Introduction

The remaining useful life (RUL) is a technical term used to describe the progression of faults in Prognostics and Health Management (PHM) applications [1]. Prognosis algorithms tend ideally to achieve the ideal maintenance policy through predictions of the available time before a failure occurs within a component or sub-component, that is RUL [2]. In this way, RUL predictions have the potential to prevent critical failures, and hence, becomes an important measurement to achieve the ultimate goal of zero-downtime performance in industrial systems. However, traditional prognosis algorithms suffer from a decreased capacity to process the increased complexity in today's sequential data with accuracy.

Recently, deep learning (DL) has emerged as a potent area to process highly non-linear and varying sequential data with minimal human input within the PHM domain [3]. Today, DL is an extremely active sub-field of machine learning. With increased processing power and continuous developments in graphics processors, DL has the potential to improve prediction tasks as the computational burden reduces significantly [4]. However, deep architectures introduce many diverse hyper-parameters, which are challenging to optimize in the training process. Thus, this study proposes a Genetic Algorithm (GA) approach in order to optimize the hyper-parameters in an efficient manner.

DL techniques, such as Convolutional Neural Network (CNN) and Long-Short Term Memory (LSTM), have shown rapid developments and outperformed traditional prognosis algorithms in RUL predictions for turbofan engine degradation [5–7]. DL techniques predict the RUL without any prior knowledge of engine degradation mechanics. Thus, data analysts today apply their knowledge about the RUL prediction problem to the selection and design of DL techniques, rather than to feature engineering. However, both CNN and LSTM depend on purely supervised learning. In other words, they require large labeled training datasets in the training procedure. Thus, the RUL prediction accuracy strongly depends on the quality of the constructed run-to-failure

* Corresponding author.
E-mail addresses: andre.ellefsen@ntnu.no (A. Listou Ellefsen), emil.bjorlykhaug@ntnu.no (E. Bjørlykhaug), vilmar.aesoy@ntnu.no (V. Æsøy), sergey.ushakov@ntnu.no (S. Ushakov), hozh@ntnu.no (H. Zhang).

https://doi.org/10.1016/j.ress.2018.11.027
Received 18 June 2018; Received in revised form 16 November 2018; Accepted 24 November 2018
Available online 26 November 2018

training data labels.

In contrast, unsupervised DL techniques introduce an initial pre-training stage to extract high-level abstract features from raw unlabeled training data automatically. Thus, the combination of unsupervised and supervised (semi-supervised) learning has the potential for even higher RUL prediction accuracy since the weights are initialized in a region near a good local minimum before supervised fine-tuning is conducted to minimize the global training objective [8].

More advanced and recent activation functions [9], learning rate methods [10], regularization techniques [11], and weight initializations [12,13] have indeed reduced the need for unsupervised pre-training in a variety of domains when the training data is completely labeled. Nevertheless, in real-life PHM applications, high-quality run-to-failure labeled training data is not easily obtained, especially from new equipment. However, unsupervised pre-training in semi-supervised setups has the potential to perform with high RUL prediction accuracy even with reduced amounts of labeled training data in the fine-tuning procedure. Additionally, most data collected in real-life PHM applications is subjected to several operating conditions and fault modes. This increases the inherent degradation complexity, which makes it more difficult for the prognosis algorithm to discover clear trends in the input data directly. To cope with this issue, the initial unsupervised pre-training stage can be utilized. Unsupervised pre-training extracts more degradation related features before supervised fine-tuning, and hence, has the potential to support the whole architecture to better understand the underlying degradation phenomena.

The aim of this paper is to show the effect of unsupervised pre-training in RUL predictions utilizing a semi-supervised setup. The results are verified on the four different simulated turbofan engine degradation datasets in the publicly available Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset, produced and provided by NASA [14]. This study's main contributions are as follows:

- The GA approach effectively tunes hyper-parameters in deep architectures.
- Semi-supervised learning improves the RUL prediction accuracy compared to supervised learning in multivariate time series data with several operating conditions and fault modes when the training data is completely labeled.
- Semi-supervised learning performs higher RUL prediction accuracy compared to supervised learning when the labeled training data in the fine-tuning procedure is reduced.

The overall organization of the paper is as follows. Section 2 introduces recent and related work on the C-MAPSS dataset. Section 3 introduces the necessary background on GAs and the proposed semi-supervised setup. The experimental approach, results, and discussions are considered in Section 4. Finally, Section 5 concludes and closes the paper and provides directions for future work.

2. Related work

The C-MAPSS dataset has been extensively used to evaluate several DL approaches to RUL predictions. This section reviews the most recent studies applied on the C-MAPSS dataset. The selected studies either utilize a Convolutional Neural Network (CNN), a Deep Belief Network (DBN) or Long-Short Term Memory (LSTM) in the proposed deep architecture.

In most PHM applications, sequential data is a standard format of the input data, for example pressure and temperature time series data. LSTM is a well-established DL technique to process sequential data. The original LSTM [15] was developed after the early 1990s, when researchers discovered a vanishing and exploding gradient issue in traditional Recurrent Neural Networks (RNNs) [16]. This issue confirmed that traditional RNNs had difficulty learning long-term dependencies. To cope with this issue, the LSTM introduces a memory cell that regulates the information flow in and out of the cell. Consequently, the memory cell is able to preserve its state over long durations, that is learning long-term dependencies that may influence future predictions. Yuan et al. proposed an LSTM approach for several different faults [17]. The proposed approach was compared with traditional RNN, Gated Recurrent Unit LSTM (GRU-LSTM) and AdaBoost-LSTM. It showed improved performance in all cases. Another LSTM approach was provided by Zheng et al. [6]. The proposed approach provides RUL predictions using two LSTM layers, two Feed-forward Neural Network (FNN) layers, and an output layer. The LSTM layers were able to reveal hidden patterns in the C-MAPSS dataset and achieved higher accuracy compared to the Hidden Markov Model or traditional RNN. A similar study was provided by Wu et al. [18]. In this study, an LSTM was combined with a dynamic difference method in order to extract new features from several operating conditions before the training procedure. These features contain important degradation information, which improves the LSTM to better control the underlying physical process. The proposed approach showed enhanced performance compared to traditional RNN and GRU-LSTM.

Although CNNs have performed excellently on 2D and 3D grid-structured topology data, such as object recognition [20] and face recognition [21], respectively, CNNs can also be applied to 1D grid-structured topology sequential data in PHM applications. Babu et al. proposed a novel CNN approach for RUL predictions [5]. This CNN approach includes two layers with convolution and average-pooling steps, and a final FNN layer to perform RUL predictions. The proposed approach indicated improved accuracy compared to the Multilayer Perceptron (MLP), Support Vector Machine (SVM), and Relevance Vector Machine. More recently, [7] takes a CNN approach. In this study, Li et al. achieved even higher accuracy on the C-MAPSS dataset compared to both the LSTM approach in [6] and the CNN approach in [5]. They employed the recently developed, proven regularization technique "dropout" [11] and the adaptive learning rate method "adam" [10].

Hinton et al. introduced the greedy layer-wise unsupervised learning algorithm in 2006, designing it for DBNs [22]. A DBN consists of stacked Restricted Boltzmann Machines (RBMs) where the hidden layer in the previous RBM will serve as the input layer for the current RBM. The algorithm performs an initial unsupervised pre-training stage to learn internal representations from the input data automatically. Next, supervised fine-tuning is performed to minimize the training objective. Zhang et al. have proposed a multiobjective DBN ensemble approach [19]. This approach combines a multiobjective evolutionary ensemble learning framework with the DBN training process. Accordingly, the proposed approach creates multiple DBNs of varying accuracy and diversity before the evolved DBNs are combined to perform RUL predictions. The combined DBNs are optimized through differential evolution where the average training error is the single objective. The proposed approach outperformed several traditional machine learning algorithms, such as SVM and MLP. The recent studies are summarized in Table 1.

These studies all utilize a completely labeled run-to-failure training dataset in the training procedure. However, in real-life PHM scenarios, most data accumulated is unstructured and unlabeled from the start.

Table 1
Recent DL approaches proposed for RUL predictions on the C-MAPSS dataset [14] (the years between 2016 and 2018).

Author & Refs.	Year	Approach
Li et al. [7]	2018	CNN + FNN
Wu et al. [18]	2018	LSTM
Zheng et al. [6]	2017	LSTM + FNN
Yuan et al. [17]	2016	LSTM
Zhang et al. [19]	2016	MODBNE
Babu et al. [5]	2016	CNN + FNN

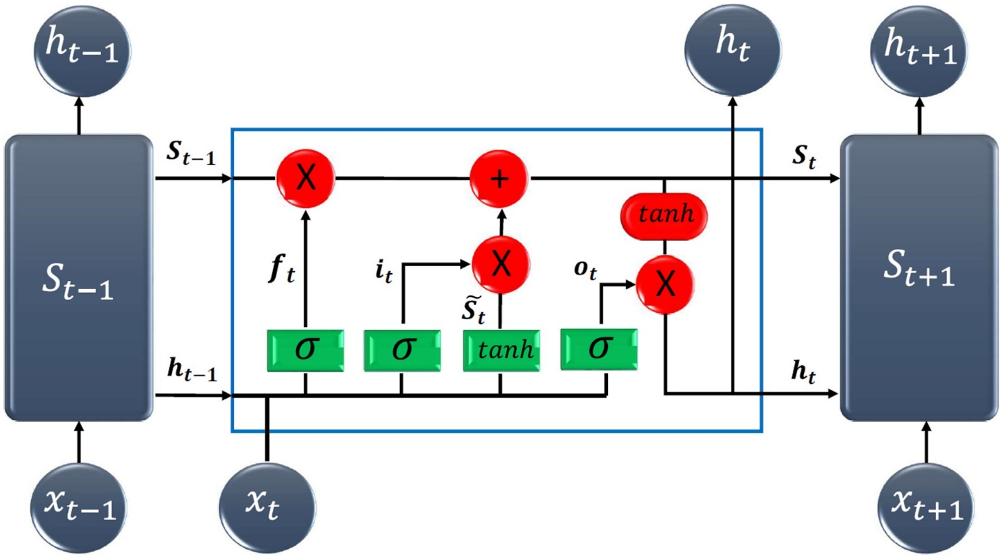

Fig. 1. Vanilla LSTM, adopted from Olah [28]. The blue rectangle represents the memory cell. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Valuable domain knowledge is required to construct run-to-failure data labels. This is both a time-consuming and challenging process. Thus, this study will investigate the effect of unsupervised pre-training in a semi-supervised setup both with reduced and completely labeled training datasets.

3. Proposed semi-supervised setup

This section will introduce the necessary background on the proposed semi-supervised setup. First, the main DL techniques included, RBM and LSTM, are defined. Next, the proposed deep architecture structure as well as the GA approach for hyper-parameter tuning are elaborated.

3.1. *Restricted Boltzmann machine*

RBMs were originally developed using binary stochastic visible units, *v*, in the input layer and binary stochastic hidden units, *h*, in the hidden layer [23]. However, in real-value data applications, like the C-MAPSS dataset, linear Gaussian units replace the binary visible units and rectified linear units replace the binary hidden units [24]. RBMs are symmetrical bipartite graphs since the visible and hidden units are fully connected and units in the same layer have zero connections.

RBMs are energy-based models with the joint probability distribution specified by their energy function [25]:

$$P(v,\,h) = \frac{1}{Z}e^{-E(v,h)} \tag{1}$$

where Z is the partition function that ensures that the distribution is normalized:

$$Z = \sum_{v}\sum_{h} e^{-E(v,h)} \tag{2}$$

The energy function for RBMs with Gaussian visible units is given by:

$$E(v,\,h) = \sum_{i=1}^{V}\frac{(v_i - b_i)^2}{2\gamma_i^2} - \sum_{j=1}^{H} c_j h_j - \sum_{i=1}^{V}\sum_{j=1}^{H}\frac{v_i}{\gamma_i}h_j w_{ij} \tag{3}$$

where w_{ij} denotes the weight between the visible unit v_i and hidden unit h_j, b_i and c_j represents the bias terms, V and H expresses the numbers of visible and hidden units, respectively, and γ_i is the standard deviation of v_i. As recommended by Hinton [25], zero mean and unit variance normalization should be applied to the input data. Contrastive divergence is used to train RBMs:

$$\Delta w_{ij} = \epsilon(\langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{recon}) \tag{4}$$

where ϵ is the learning rate. First, the data distribution samples visible units based on hidden units. Then, the input data is reconstructed, generated by Gibbs sampling, which samples hidden units based on visible units. This process continues until the parameters converge, that is, the hidden layer approximates the input layer. In this way, RBMs are able to model data distributions without any label knowledge. Typically, after the pre-training stage, the reconstruction part of the RBM is omitted and the pre-trained weights facilitate a subsequent supervised fine-tuning procedure.

3.2. *Long-Short term memory*

Modifications by Gers et al. [26] have been included in the original LSTM, and researchers generally refer to this LSTM setup as the "vanilla LSTM." Although several variants of the vanilla LSTM have been proposed, Greff et al. have shown that none of the variants can improve the vanilla LSTM significantly [27]. Thus, the proposed semi-supervised setup uses the vanilla LSTM.

The memory cell, as illustrated in Fig. 1, consists of three non-linear gating units that protect and regulate the cell state, S_t [28]:

$$f_t = \sigma(W_f x_t + R_f h_{t-1} + b_f) \tag{5}$$

$$i_t = \sigma(W_i x_t + R_i h_{t-1} + b_i) \tag{6}$$

$$o_t = \sigma(W_o x_t + R_o h_{t-1} + b_o) \tag{7}$$

where σ is the sigmoid gate activation function in order to obtain a scaled value between 0 and 1, W is the input weight, R is the recurrent weight, and b is the bias weight.

The new candidate state values, $\tilde{S}_t$, are created by the tanh layer:

$$\tilde{S}_t = tanh(W_s x_t + R_s h_{t-1} + b_s) \tag{8}$$

The previous cell state, S_{t-1}, is updated into the new cell state, S_t, by:

$$S_t = f_t \otimes S_{t-1} + i_t \otimes \tilde{S}_t \tag{9}$$

where $\otimes$ denotes element-wise multiplication of two vectors. First, the forget layer, f_t, determines which historical information the memory cell removes from S_t. Then, the input layer, i_t, decides what new

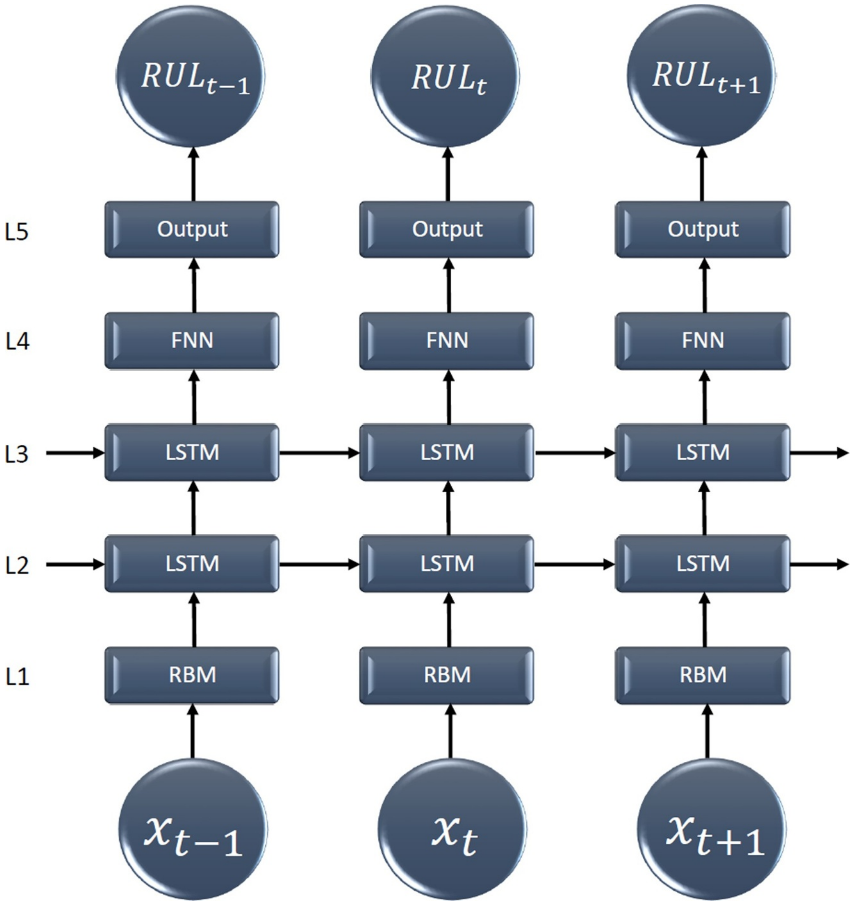

Fig. 2. The proposed semi-supervised deep architecture structure.

information in $\tilde{S}_t$ the memory cell will update and store in S_t.

The output layer, o_t, determines which parts of S_t the memory cell will output. S_t is filtered in order to push the values between -1 and 1:

$$h_t = o_t \otimes \ tanh\,(S_t) \tag{10}$$

Through these steps, the vanilla LSTM has the ability to remove or add information to S_t.

3.3. The proposed deep architecture structure and the genetic algorithm approach

The proposed semi-supervised deep architecture structure is shown in Fig. 2. In the first layer (L1), a RBM will be utilized as an unsupervised pre-training stage in order to learn abstract features from raw unlabeled input data automatically. These features might contain important degradation information, and hence, initialize the weights in a region near a good local minimum before supervised fine-tuning of the whole architecture is conducted. 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 in sequential data with multiple operating and fault conditions [6]. Next, an FNN layer is used in the fourth layer (L4) in order to map all extracted features. In the final layer (L5), a time distributed fully connected output layer is attached to handle error calculations and perform RUL predictions.

The GA is a metaheuristic inspired by the natural selection found in nature [29]. It is a powerful tool for finding a near-optimal solution in a big search space. In this work, a GA approach is proposed to tune hyper-parameters. First, the GA approach selects random hyper-parameters for the proposed semi-supervised deep architecture within a given search space. One such set of random hyper-parameters is called an individual and a set of individuals is called a population. Next, the accuracy of each of the individuals in the population are evaluated by training networks with the individuals hyper-parameters. The best results from the training are then kept and used as parents for the next generation of hyper-parameters. Additionally, some random mutation is performed after the crossover for increasing the exploration of the algorithm.

4. Experimental study

In the following experimental study, the proposed semi-supervised deep architecture will be compared to recent studies in the literature as well as purely supervised training. The latter comparison will be performed with and without the initial pre-training stage utilizing the proposed semi-supervised deep architecture when the labeled training data in the fine-tuning procedure is reduced. Experiments are performed on the four subsets provided in the benchmark C-MAPSS dataset: FD001, FD002, FD003, and FD004. All experiments are run on NIVIDIA GeForce GTX 1060 6 GB and the Microsoft Windows 10 operating system. The programming language is Java 8 with deep learning library "deeplearning4j" (DL4J) version 0.9.1 [30].

4.1. The benchmark C-MAPSS dataset and performance evaluation

The C-MAPSS dataset is divided into four subsets, as shown in Table 2, and each subset is further divided into training and test sets of multiple multivariate time series. Each time series is from a different aircraft gas turbine engine and starts with different degrees of initial wear and manufacturing variation, which is unknown to the data analyzer. All engines operate in normal condition at the start, then begin to degrade at some point during the time series. The degradation in the training sets grows in magnitude until failure, while the

Table 2
The C-MAPSS dataset [14].

Dataset	FD001	FD002	FD003	FD004
Time series training set	100	260	100	249
Time series test set	100	259	100	248
Operating conditions	1	6	1	6
Fault conditions	1	1	2	2

degradation in the test sets ends sometime prior to failure, that is the RUL. That is, the last time step for each engine in the test sets provides the true RUL targets. Thus, the main objective is to predict the correct RUL value for each engine in the test sets. The four subsets vary in operating and fault conditions and the data is contaminated with sensor noise. Each subset includes 26 columns: engine number, time step, three operational sensor settings, and 21 sensor measurements. See [14,31] for a deeper understanding of the C-MAPSS dataset.

The scoring function (S) provided in Saxena et al. [31] and the root mean square error ($RMSE$) are used in this study to evaluate the performance of the proposed semi-supervised setup:

$$S = \begin{cases} \sum_{i=1}^{n} e^{(-\frac{d_i}{13})} - 1, & \text{for} \quad d_i < 0 \\ \sum_{i=1}^{n} e^{(-\frac{d_i}{10})} - 1, & \text{for} \quad d_i \geq 0 \end{cases} \tag{11}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_i^2} \tag{12}$$

where n is the total number of true RUL targets in the respective test set and $d_i = RUL_{predicted} - RUL_{true}$. As shown in Fig. 3, the $RMSE$ gives equal penalty to early and late predictions. In the asymmetric scoring function, however, the penalty for late predictions is larger. Late predictions could cause serious system failures in real-life PHM applications as the maintenance procedure will be scheduled too late. On the other hand, early predictions pose less risk since the maintenance procedure will be scheduled too early, and hence, there is still time to perform maintenance. Nevertheless, the main objective is to achieve the smallest value possible for both S and $RMSE$, that is, when $d_i = 0$.

Only evaluating performance at the last time step for each engine in the test sets has both advantages and disadvantages. High and reliable RUL prediction accuracy at the very end of components and sub-components lifetime have of course great industrial significance, as this period is critical for PHM applications. However, this evaluation approach could hide the true overall prognostics accuracy as the prognostics horizon of the algorithm is not considered. The prognostics horizon is critical in order to achieve trustworthy confidence intervals for the corresponding RUL prediction. These confidence intervals are important due to both inherent uncertainties with the degradation process and potential flaws in the prognosis algorithm [32].

Table 3
Genes in the GA approach.

Gene	Hyper-parameter	Values
1	R_c	115, 120, 125, 130, 135, 140
2	Learning rate RBM layer	10^{-1}, 10^{-2}, 10^{-3}
3	Learning rate remaining layers	10^{-2}, 10^{-3}, 10^{-4}
4	L2 Regularization	10^{-4}, 10^{-5}, 10^{-6}
5	miniBatch	5, 10
6	n L1	32, 64, 128
7	n L2	32, 64, 128
8	n L3	32, 64, 128
9	n L4	8, 16
10	p L2	0.5, 0.6, 0.7, 0.8, 0.9
11	p L3	0.5, 0.6, 0.7, 0.8, 0.9
12	p L4	0.5, 0.6, 0.7, 0.8, 0.9
13	I/O activation function LSTM	sigmoid, tanh
14	Activation function FNN	sigmoid, tanh

Table 4
Parameters of the GA approach.

Parameter	Value
Population size	20
Nr of elite	3
Mutation rate	0.5
Mutation gain	0.3
Evolution iterations	3

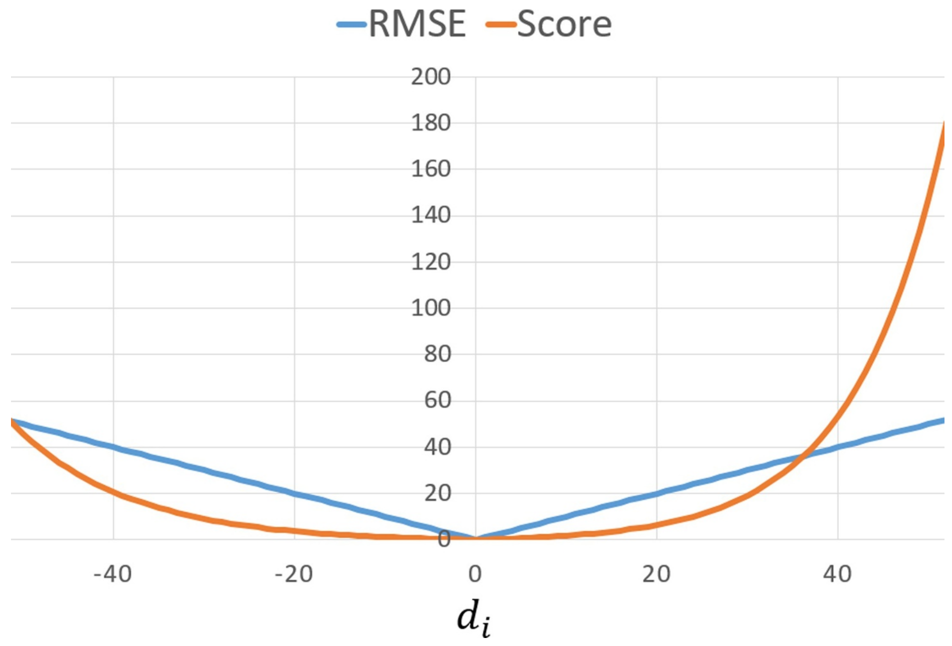

Fig. 3. Simple illustration of the scoring function vs. RMSE, where $d_i = RUL_{predicted} - RUL_{true}$.

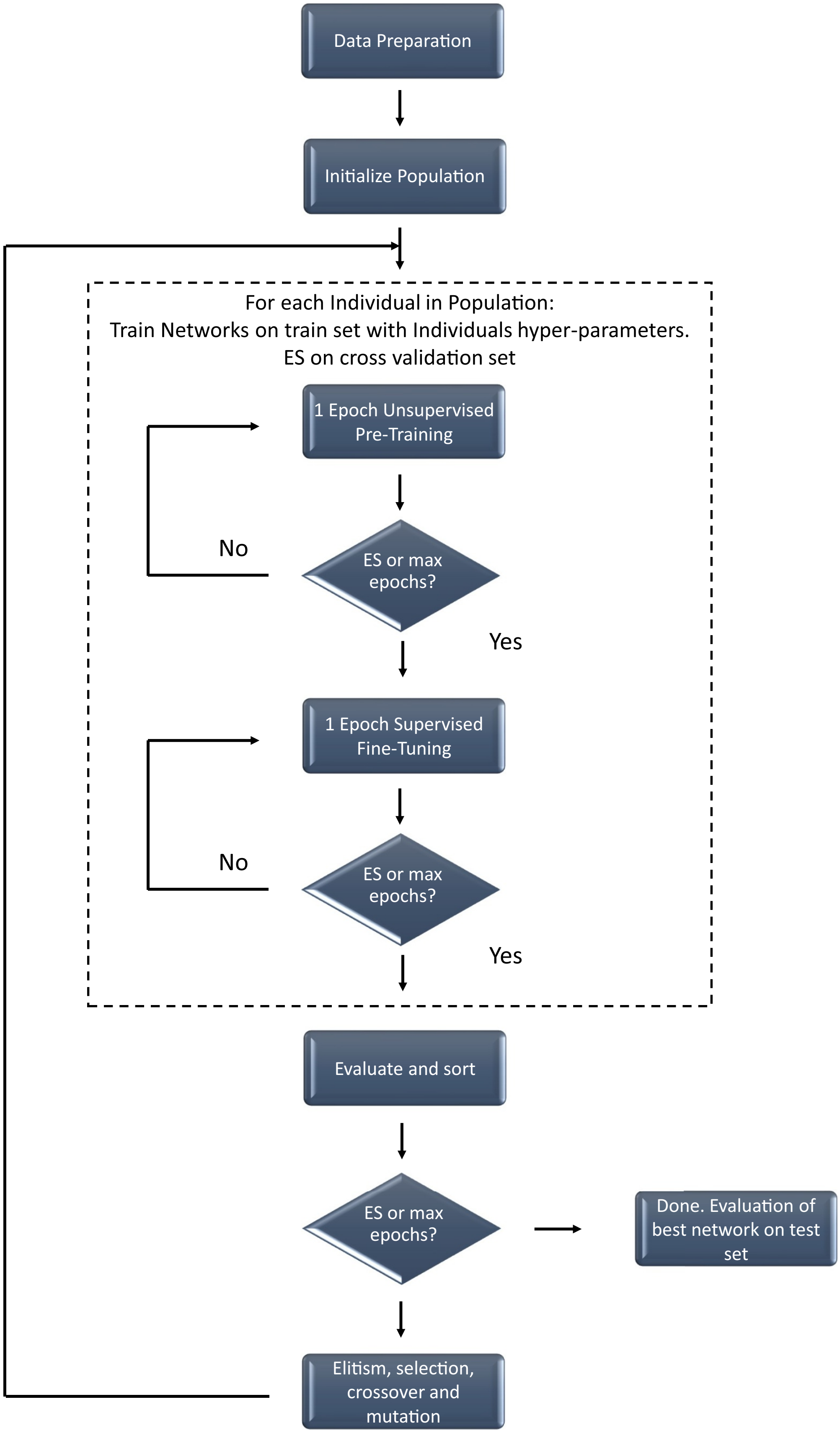
Data Preparation
Initialize Population
For each Individual in Population:
Train Networks on train set with Individuals hyper-parameters.
ES on cross validation set
1 Epoch Unsupervised Pre-Training
ES or max epochs?
No
Yes
1 Epoch Supervised Fine-Tuning
ES or max epochs?
No
Yes
Evaluate and sort
ES or max epochs?
Done. Evaluation of best network on test set
Elitism, selection, crossover and mutation

Fig. 4. Flowchart of the GA approach.

Table 5
GA individuals.

FD001	Layer index	DL technique	nIn	nOut	Dropout	Activation function
	1	RBM	14	64	1.0	ReLU
	2	LSTM	64	64	0.9	Sigmoid
	3	LSTM	64	64	0.6	Sigmoid
	4	FNN	64	8	0.6	Sigmoid
	5	Output	8	1	1.0	Identity
	R_c	Learning rate RBM layer	Learning rate remaining layers	L2 regularization	mini batch size	*RMSE* cross-validation set
	115	10^{-2}	10^{-3}	10^{-6}	5	8.49
FD002	Layer index	DL technique	nIn	nOut	Dropout	Activation function
	1	RBM	24	64	1.0	ReLU
	2	LSTM	64	128	0.7	Sigmoid
	3	LSTM	128	32	0.8	Sigmoid
	4	FNN	32	8	0.6	Sigmoid
	5	Output	8	1	1.0	Identity
	R_c	Learning rate RBM layer	Learning rate remaining layers	L2 regularization	mini batch size	*RMSE* cross-validation set
	135	10^{-2}	10^{-3}	10^{-5}	10	9.60
FD003	Layer index	DL technique	nIn	nOut	Dropout	Activation function
	1	RBM	14	32	1.0	ReLU
	2	LSTM	32	128	0.9	Sigmoid
	3	LSTM	128	64	0.9	Sigmoid
	4	FNN	64	8	0.9	Sigmoid
	5	Output	8	1	1.0	Identity
	R_c	Learning rate RBM layer	Learning rate remaining layers	L2 regularization	mini batch size	*RMSE* cross-validation set
	125	10^{-2}	10^{-3}	10^{-6}	5	8.59
FD004	Layer index	DL technique	nIn	nOut	Dropout	Activation function
	1	RBM	24	64	1.0	ReLU
	2	LSTM	64	128	0.8	Sigmoid
	3	LSTM	128	32	0.7	Sigmoid
	4	FNN	32	8	0.6	Sigmoid
	5	Output	8	1	1.0	Identity
	R_c	Learning rate RBM layer	Learning rate remaining layers	L2 regularization	mini batch size	*RMSE* cross-validation set
	135	10^{-2}	10^{-3}	10^{-5}	10	10.45

Table 6
The proposed semi-supervised deep architecture with and without unsupervised pre-training on subset FD004 when the labeled training data is reduced from 100% to 10%. Improvement $= (1 - \frac{\text{Semi} - \text{supervised}}{\text{Supervised}})$.

RMSE	100%	80%	60%	40%	20%	10%
Semi-supervised with 100% training features in the pre-training stage	22.66	23.04	24.07	25.46	30.26	34.19
Supervised	23.62	23.45	24.14	26.40	30.27	34.90
Improvement	4.06%	1.75%	0.29%	3.56%	0.03%	2.03%
S	100%	80%	60%	40%	20%	10%
Semi-supervised with 100% training features in the pre-training stage	2840	3175	3576	5522	9562	22,476
Supervised	3234	3427	3650	6536	15,612	27,138
Improvement	12.18%	7.35%	2.03%	15.51%	38.75%	17.18%
Average training time per epoch (s)	100%	80%	60%	40%	20%	10%
Pre-training stage	7.08	7.08	7.08	7.08	7.08	7.08
Fine-tuning procedure	34.14	28.97	22.39	15.2	9.74	5.93

4.2. Data preparation

4.2.1. Masking and padding

The DL4J library provides a “CSVSequenceRecordReader” to handle time series data. It reads time series data, where each time series is defined in its own file. Each line in the files represents one time step. Consequently, each time series (engine) in the four training sets are split into their own file. The input training data has the following shape: [miniBatchSize, inputSize, timeSeriesLength], where miniBatchSize is the number of engines in the mini batch, input size is the number of columns, and timeSeriesLength is the total number of time steps in the mini batch. The engines have variable time step lengths, and hence, the shorter engines in a mini batch are padded with zeros such that the time step lengths are equal to the longest among them. Accordingly, mask arrays are used during training. These additional arrays record whether a time step is actually present, or whether it is just padding. In all performance evaluations, mask arrays are considered.

4.2.2. Feature selection

Sensor 1, 5, 6, 10, 16, 18, and 19 in subset FD001 and FD003 exhibit constant sensor measurements throughout the engineâ;;s lifetime. Constant sensor measurements does not provide any useful degradation

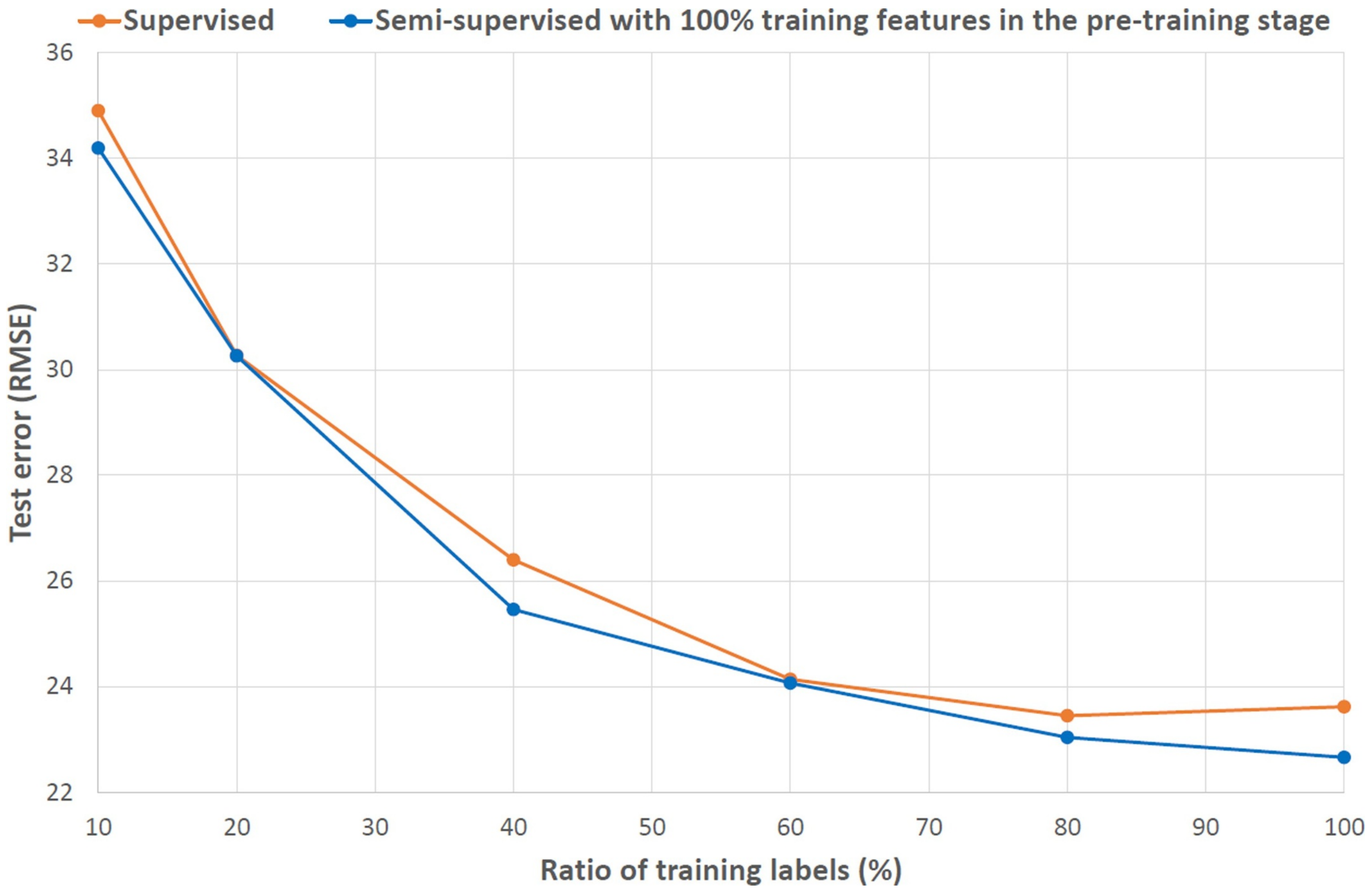

Fig. 5. *RMSE* comparison on subset FD004 when the labeled training data is reduced from 100% to 10%.

information regarding RUL predictions [19,33]. In addition, subset FD001 and FD003 are subjected to a single operating condition [5]. Hence, the three operational settings are excluded. Accordingly, sensor 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, and 21 are used as the input features for subset FD001 and FD003.

Subset FD002 and FD004 are more complex due to six operating conditions [18]. Six operating conditions make it challenging for the prognosis algorithm to detect clear degradation patterns in the input data directly. However, two LSTM layers were able to find hidden patterns in Zheng et al. [6]. Additionally, the initial unsupervised pre-training stage is able to capture hierarchically statistical patterns before the supervised fine-tuning procedure. Consequently, these patterns will enable the whole architecture to cope with the complexity inherent in degradation. Thus, all three operational sensor settings and all sensor measurements are used as the input features for subset FD002 and FD004.

4.2.3. RUL targets

True RUL targets are only provided at the last time step for each engine in the test sets. In order to construct labels for every time step for each engine in the training sets, Heimes et al. [33] used an MLP function estimator to show that it is reasonable to estimate RUL as a constant value when the engines operate in normal condition. Based on their experiments, a degradation model was proposed with a constant RUL value (R_c) of 130 and a minimum value of 0. This piece-wise linear RUL target function is still the most common approach in the literature [5–7,18,19]. However, R_c varies among the different studies. For this study, the GA approach is used to test different R_c since it has a notable impact on the experimental performance for the different subsets in the C-MAPSS dataset.

4.2.4. Data normalization

All input features and labels are normalized with zero mean unit variance (z-score) normalization:

$$z = \frac{x - \mu}{\sigma} \tag{13}$$

where μ is the mean and σ is the corresponding standard deviation.

4.3. Deep architecture configuration and training

In the initial RBM layer, a rectified linear unit (ReLU) is used as the activation function as ReLUs improve the performance of RBMs compared to the tanh activation function [9]. Stochastic gradient descent is the selected optimization algorithm and adaptive moment estimation (Adam) [10] is the learning rate method applied to the deep architecture. Recently, Adam has shown great results on the C-MAPSS dataset [7,18]. To better preserve the information in the pre-trained weights, the learning rate in the initial RBM layer is one order of magnitude higher than the learning rate in the remaining layers. ReLU weight initialization [13] is applied to the RBM layer while Xavier weight initialization [12] is applied to the remaining layers in the proposed semi-supervised deep architecture.

Truncated backpropagation through time (TBPTT) is used in this study due to a large amount of time steps in the training sets. TBPTT performs more frequent parameter updates compared to standard backpropagation through time. This both reduces computational complexity and improves learning of temporal dependencies [34]. The forward and backward passes are set to 100 time steps, as the shortest time series in the C-MAPSS dataset contains 128 time steps.

In the training procedure, each complete training subset is split into a training set and a cross-validation set. In subset FD001 and FD003,

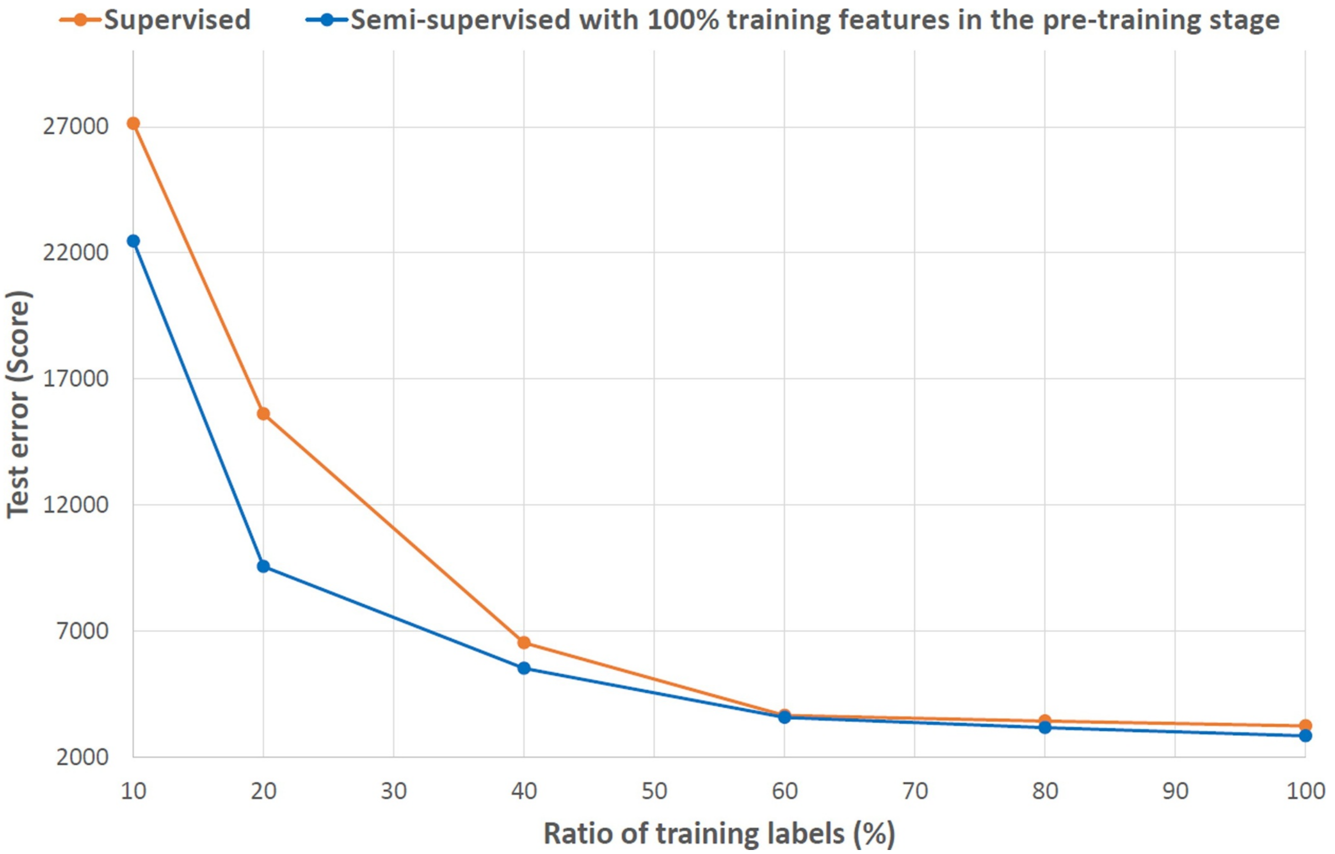

Fig. 6. *S* comparison on subset FD004 when the labeled training data is reduced from 100% to 10%.

20% of the total engines in the complete training subsets are randomly selected for cross-validation. The remaining 80% are designated as the training sets. Due to an increased complete training subset size in subset FD002 and FD004, 10% of the total engines are randomly selected for cross-validation while the remaining 90% are designated as the training sets.

Table 3shows all the hyper-parameters which the GA approach needs to optimize for each subset. The recent and well-proven regularization technique dropout [11] is applied to the deep architecture. Dropout introduces the hyper-parameter, *p*, which randomly drops units during training. In this way, dropout approximately combines an exponential number of different architectures. Thus, the deep architecture learns to make generalized representations of the input data, which enhances the feature extraction ability. In Table 3, *n* and *p* refer to the number of hidden units and the probability of retaining each hidden unit in the coupled hidden layer L, respectively. A *p* value of 1.0 is functionally equivalent to zero dropout, i.e. 100% probability of retaining each hidden unit. A typical value for *p* used in the literature is 0.5 [7,18]. However, *p* depends on *n*. In this study, the GA approach is able to test different values of *n* in both L1, L2, L3, and L4, and hence, it is also able to test different values of *p* in the range from 0.5 to 0.9. As Patterson and Gibson [35] recommend, to preserve important features in the input data, dropout is disabled in the first layer, L1. Additionally, dropout is not used in the output layer, L5. It should be noted that dropout is only applied to the non-recurrent connections in the LSTM layers.

The GA approach is run once for each subset. It trains a diverse number of individuals on the training sets and evaluates the *RMSE*, Eq. 12, on the cross-validation set as its objective function. In this way, the GA approach optimizes the hyper-parameters for each subset. To limit the time consumed during the optimization process, the population size is restricted to 20 individuals and the population is evolved three times with the selected GA parameters as shown in Table 4. This results in an average training time of 60 hours for each subset. However, the training time will reduce significantly along with future developments in GPUs. Additionally, to prevent overfitting, early stopping (ES) is applied to monitor the performance during the training process of each individual. In the unsupervised pre-training stage, ES is used to monitor the reconstruction error on the training set. If the number of epochs with no improvement exceeds nine, the unsupervised pre-training procedure is terminated. In the fine-tuning procedure, ES is used to monitor the RMSE accuracy on the cross-validation set. If the number of epochs with no improvement exceeds four, the fine-tuning procedure is terminated. Finally, the top five GA individuals for each subset are evaluated on the test sets where both *RMSE* and *S* are calculated. A complete flowchart of the GA approach is shown in Fig. 4 and the best GA individuals for each subset are shown in Table 5. In Table 5, nIn and nOut represents the number of input and output (hidden) units for each layer, respectively.

4.4. Experimental results and discussions

The aim of this paper is to show increased RUL prediction accuracy in multivariate time series data subjected to several operating conditions and fault modes utilizing a semi-supervised setup. The experiments conducted in this study shows the effect of unsupervised pre-training both when the training data is completely labeled and when the labeled training data in the fine-tuning procedure is reduced.

4.4.1. The effect of unsupervised pre-training in RUL predictions

Subset FD004 is chosen for this experiment due to the complexity inherent in its six operating conditions and two fault modes. As shown in Table 6, semi-supervised learning provides higher RUL prediction accuracy compared to supervised learning when the training data is 100% labeled. This indicates that the unsupervised pre-training stage initializes the weights using a more suitable local minimum than

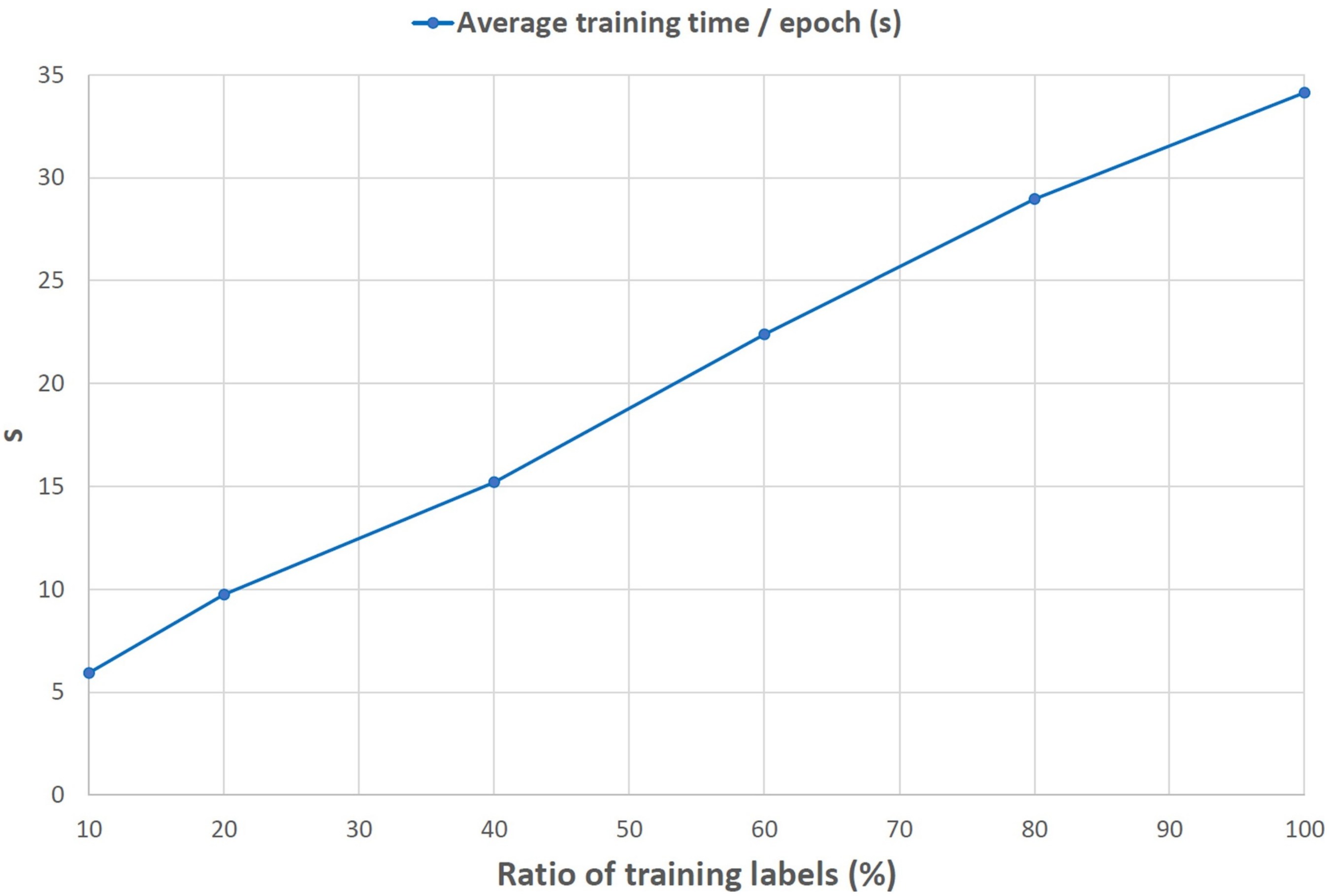

Fig. 7. Average training time in seconds per epoch in the fine-tuning procedure when the labeled training data is reduced from 100% to 10%.

weights that are randomly initialized. Consequently, unsupervised pre-training supports better comprehension of the inherent degradation complexity in the whole architecture.

In real-life PHM scenarios, high-quality labeled training data is hard to acquire. To address this problem, this study has performed an experiment where only reduced parts of the training data in subset FD004 contains labels. The labels in the training set are randomly reduced into fractions of 20%, 40%, 60%, 80%, and 90%, respectively. To minimize any selection bias, the random selection process is repeated five times for each fraction. Each random selection is then trained on the training set and evaluated on the test set where *RMSE* and *S* are calculated. Finally, the top three performance results are averaged as shown in Table 6. It should be noted that a similar experiment, which has made interesting and valuable results using a variational autoencoder (VAE), is conducted on subset FD001 in [36].

To show the effect of unsupervised pre-training, the proposed deep architecture is trained with and without the initial pre-training stage. In the initial pre-training stage, the proposed deep architecture is trained with 100% training features. The ES procedure is used to monitor the performance. As shown in Figs. 5 and 6, the proposed semi-supervised deep architecture provides the overall highest RUL prediction accuracy when trained with the initial unsupervised pre-training stage. It should be noted that the proposed deep architecture, when trained in a purely supervised manner, also provides satisfactory RUL prediction accuracy, especially when more than 60% of the training labels are included. This proves that recent weight initializations and regularization techniques, such as Xavier and dropout, have indeed reduced the need for unsupervised pre-training. Dropout in particular improves the feature extraction ability by approximately combining several different architectures in the fine-tuning procedure. However, the improvement of utilizing semi-supervised learning is noticeable when more than 40% of the training labels are removed, as shown in Table 6.

Additionally, as shown in Fig. 7, the average training time per epoch will almost linearly decrease with decreasing training labels, e.g. 15.2 s training time at 40% labels, which is 15.2 s/34.14 s= 44.5% training time per epoch compared to 100% labels. Also, as seen in Figs. 5 and 6, the RUL prediction accuracy is satisfactory when more than 60% training labels are included. Depending on the reliability and safety requirements of the application, the trade-off of reduced RUL prediction accuracy might be acceptable if the training time is critical.

4.4.2. *Comparison with the literature*

Studies that have reported results on all four subsets in the C-MAPSS dataset have been selected for comparison. Although the initial *Rc* values are somewhat different, the results are still comparable. As shown in Tables 7 and 8, the proposed semi-supervised deep architecture has achieved promising results compared to the recent studies when the training data is completely labeled. The CNN approach in Li et al. [7] achieved slightly higher *RMSE* prediction accuracy on subset FD002. However, the proposed semi-supervised deep architecture indicates substantially improved *S* prediction accuracy on all subsets. Consequently, the proposed semi-supervised deep architecture reduces the

Table 7
RMSE comparison with the literature on the C-MAPSS dataset.

DL approach & refs.	FD001	FD002	FD003	FD004
CNN + FNN [5]	18.45	30.29	19.82	29.16
LSTM + FNN [6]	16.14	24.49	16.18	28.17
MODBNE [19]	15.04	25.05	12.51	28.66
CNN + FNN [7]	12.61	**22.36**	12.64	23.31
Proposed semi-supervised setup	**12.56**	22.73	**12.10**	**22.66**

Table 8
Score function comparison with the literature on the C-MAPSS dataset.

DL approach & Refs.	FD001	FD002	FD003	FD004
CNN + FNN [5]	1287	13,570	1596	7886
LSTM + FNN [6]	338	4450	852	5550
MODBNE [19]	334	5585	422	6558
CNN + FNN [7]	274	10,412	284	12,466
Proposed semi-supervised setup	**231**	**3366**	**251**	**2840**

average number of late predictions across the test sets considerably. This is because the unsupervised pre-training stage extracts more degradation related features before supervised fine-tuning. Thus, this stage supports the whole architecture to better understand the underlying degradation trends. Late predictions impose a serious threat to reliability and safety in real-life PHM applications as the maintenance procedure will be scheduled too late. Therefore, semi-supervised learning is a promising approach in RUL predictions tasks both subjected to a single and multiple operating conditions and fault modes.

5. Conclusion and future work

This paper has investigated the effect of unsupervised pre-training in RUL predictions utilizing a semi-supervised setup. The experiments are performed on the publicly available C-MAPSS dataset. Additionally, a GA approach was proposed to tune the number of diverse hyper-parameters in deep architectures. Combining all the hyper-parameters in Table 3 results in a total of 8 748 000 combinations. Although, the GA approach only used 20 individuals and three evolutions, it was able to optimize hyper-parameters for each subset in the C-MAPSS dataset effectively. This is a promising approach compared to using a time consuming, exhaustive search. However, the average training time of 60 hours for each subset will be further optimized in future work.

In the experimental study, the proposed semi-supervised setup is compared to purely supervised training as well as recent studies in the literature. The proposed semi-supervised setup achieved promising RUL prediction accuracy with both completely and reduced amounts of labeled training data. Hence, unsupervised pre-training is indeed a promising feature in real-life PHM applications subjected to multiple operating conditions and fault modes, as large amounts of high-quality labeled training data might be both challenging and time-consuming to acquire. Unsupervised pre-training supports the deep architecture to improve our understanding of the inherent complexity by extracting more features that contain important degradation information.

In this study, an RBM was utilized as the initial unsupervised pre-training stage. However, RBM is a rather old, unsupervised DL technique. Today, more powerful unsupervised DL techniques are available. For instance, the VAE [36,37] seems promising. The VAE models the underlying probability distribution of the training data using variational inference. It is possible to extend to a wide range of model architectures, and this is one of its key advantages compared to RBM, which requires careful model design to maintain tractability [38].

In RUL predictions based on data-driven approaches, such as DL, the accuracy strongly depends on the quality of the constructed run-to-failure training data labels. This study confirms that R_c has a notable impact on the RUL prediction accuracy for each subset. Nevertheless, the piece-wise linear degradation model used in this study is considered a major limitation as each engine in each subset has, in fact, an individual degradation pattern. Recently, the VAE has been used for unsupervised reconstruction based anomaly detection by applying a reconstruction error as an anomaly score [39]. Thus, in future work, the VAE will also be used in order to create an unsupervised fault detector to optimize R_c for each engine in each subset in the C-MAPSS dataset.

Normally, tanh is used as the input and output (I/O) activation function in LSTMs. However, in this study it was discovered that sigmoid performed better than tanh as the LSTM I/O activation function in combination with the initial RBM layer with ReLU as the activation function. A novel rectified LSTM I/O activation function would be a positive contribution to be included in future work.

Acknowledgment

This work was supported by the Norwegian University of Science and Technology within the Department of Ocean Operations and Civil Engineering under project no. 90329106 and funded by the Research Council of Norway, grant no. 245613/O30. The authors would like to thank Digital Twins For Vessel Life Cycle Service (DigiTwin) NFR 280703.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.ress.2018.11.027

References

[1] Kalgren PW, Byington CS, Roemer MJ, Watson MJ. Defining phm, a lexical evolution of maintenance and logistics. 2006 IEEE Autotestcon. 2006. p. 353–8. https://doi.org/10.1109/AUTEST.2006.283685.

[2] Peng Y, Wang Y, Zi Y. Switching state-space degradation model with recursive filter/smoother for prognostics of remaining useful life. IEEE Trans Ind Inf 2018. https://doi.org/10.1109/TII.2018.2810284. 1–1

[3] Zhao G, Zhang G, Ge Q, Liu X. Research advances in fault diagnosis and prognostic based on deep learning. Prognostics and System Health Management Conference (PHM-Chengdu). IEEE; 2016. p. 1–6. https://doi.org/10.1109/PHM.2016.7819786.

[4] Chen XW, Lin X. Big data deep learning: challenges and perspectives. IEEE Access 2014;2:514–25. https://doi.org/10.1109/ACCESS.2014.2325029.

[5] Sateesh Babu G, Zhao P, Li X-L. Deep convolutional neural network based regression approach for estimation of remaining useful life. Cham: Springer International Publishing; 2016. p. 214–28. https://doi.org/10.1007/978-3-319-32025-0_14. ISBN 978-3-319-32025-0

[6] Zheng S, Ristovski K, Farahat A, Gupta C. Long short-term memory network for remaining useful life estimation. 2017 IEEE International Conference on Prognostics and Health Management (ICPHM). IEEE; 2017. p. 88–95.

[7] Li X, Ding Q, Sun J-Q. Remaining useful life estimation in prognostics using deep convolution neural networks. Reliab Eng Syst Saf 2018;172:1–11.

[8] Erhan D, Manzagol P-A, Bengio Y, Bengio S, Vincent P. The difficulty of training deep architectures and the effect of unsupervised pre-training. Artificial Intelligence and Statistics. 2009. p. 153–60.

[9] Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. In: Gordon G, Dunson D, DudÃk M, editors. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics; vol. 15 of Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR; 2011. p. 315–23.

[10] Kingma D.P., Ba J.. Adam: a method for stochastic optimization. arXiv:141269802014;.

[11] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15(1):1929–58.

[12] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. Proceedings of the 13th International Conference on Artificial Intelligence and Statistics. 2010. p. 249–56.

[13] He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. Proceedings of the IEEE international conference on computer vision. 2015. p. 1026–34.

[14] Saxena A., Goebel K.. Turbofan engine degradation simulation data set. NASA Ames Prognostics Data Repository (https://tiarcnasagov/tech/dash/groups/pcoe/prognostic-data-repository/), NASA Ames Research Center, Moffett Field, CA2008;.

[15] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

[16] Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Netw 1994;5(2):157–66. https://doi.org/10.1109/72.279181.

[17] Yuan M, Wu Y, Lin L. Fault diagnosis and remaining useful life estimation of aero engine using lstm neural network. IEEE International Conference on Aircraft Utility Systems (AUS). IEEE; 2016. p. 135–40.

[18] Wu Y, Yuan M, Dong S, Lin L, Liu Y. Remaining useful life estimation of engineered systems using vanilla lstm neural networks. Neurocomputing 2018;275:167–79.

[19] Zhang C, Lim P, Qin AK, Tan KC. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. IEEE Trans Neural Netw Learn Syst 2017;28(10):2306–18. https://doi.org/10.1109/TNNLS.2016.2582798.

[20] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ, editors. Advances in Neural Information Processing Systems 25. Curran Associates, Inc.; 2012. p. 1097–105.

[21] Taigman Y, Yang M, Ranzato M, Wolf L. Deepface: closing the gap to human-level performance in face verification. 2014 IEEE Conference on Computer Vision and

Pattern Recognition. 2014. p. 1701–8. https://doi.org/10.1109/CVPR.2014.220.

[22] Hinton GE, Osindero S, Teh Y-W. A fast learning algorithm for deep belief nets. Neural Comput 2006;18(7):1527–54. https://doi.org/10.1162/neco.2006.18.7.1527.

[23] Freund Y, Haussler D. Unsupervised learning of distributions on binary vectors using two layer networks. In: Moody JE, Hanson SJ, Lippmann RP, editors. Advances in Neural Information Processing Systems 4. Morgan-Kaufmann; 1992. p. 912–9.

[24] Nair V, Hinton GE. Rectified linear units improve restricted Boltzmann machines. Proceedings of the 27th international conference on machine learning (ICML-10). 2010. p. 807–14.

[25] Hinton GE. A practical guide to training restricted Boltzmann machines. Neural Networks: Tricks of the Trade. Springer; 2012. p. 599–619.

[26] Gers FA, Schmidhuber JA, Cummins FA. Learning to forget: continual prediction with lstm. Neural Comput 2000;12(10):2451–71. https://doi.org/10.1162/089976600300015015.

[27] Greff K, Srivastava RK, Koutník J, Steunebrink BR, Schmidhuber J. Lstm: a search space odyssey. IEEE Trans Neural Netw Learn Syst 2017;28(10):2222–32. https://doi.org/10.1109/TNNLS.2016.2582924.

[28] Olah C. Understanding lstm networks. 2015. 2015. URL http://colahgithubio/posts/2015-08-Understanding-LSTMs/img/LSTM3-chainpng

[29] Roberge V, Tarbouchi M, Labonté G. Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning. IEEE Trans Ind Inf 2013;9(1):132–41.

[30] Eclipse deeplearning4j development team, deeplearning4j: open-source distributed deep learning for the jvm. Apache Software Foundation License 20 http://deeplearning4jorg 2018;.

[31] Saxena A, Goebel K, Simon D, Eklund N. Damage propagation modeling for aircraft engine run-to-failure simulation. 2008 International Conference on Prognostics and Health Management. IEEE; 2008. p. 1–9.

[32] Sikorska JZ, Hodkiewicz M, Ma L. Prognostic modelling options for remaining useful life estimation by industry. Mech Syst Signal Process 2011;25(5):1803–36. https://doi.org/10.1016/j.ymssp.2010.11.018.

[33] Heimes FO. Recurrent neural networks for remaining useful life estimation. International Conference on Prognostics and Health Management, PHM 2008.. IEEE; 2008. p. 1–6.

[34] Sutskever I. Training recurrent neural networks. Toronto, Ont, Canada: University of Toronto; 2013.

[35] Patterson J, Gibson A. Deep learning: a practitioner's approach. "O'Reilly Media, Inc."; 2017.

[36] Yoon AS, Lee T, Lim Y, Jung D, Kang P, Kim D, et al. Semi-supervised learning with deep generative models for asset failure prediction. CoRR 2017. abs/1709.00845

[37] Kingma D.P., Welling M.. Auto-encoding variational bayes. arXiv:13126114 2013;.

[38] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016. http://www.deeplearningbook.org

[39] Park D, Hoshi Y, Kemp CC. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. IEEE Rob Autom Lett 2018;3(3):1544–51.