Improving Semi-Supervised Learning for Remaining Useful Lifetime Estimation Through Self-Supervision

Tilman Krokotsch[1], Mirko Knaak[2], and Clemens Gühmann[3]

[1,3] *Chair of Electronic Measurement and Diagnostic Technology, Technische Universität Berlin*
tilman.krokotsch@tu-berlin.de
clemens.guehmann@tu-berlin.de

[1, 2] *Thermodynamics & Power Systems, Power Train & Power Engineering, IAV GmbH*
tilman.krokotsch@iav.de
mirko.knaak@iav.de

ABSTRACT

RUL estimation plays a vital role in effectively scheduling maintenance operations. Unfortunately, it suffers from a severe data imbalance where data from machines near their end of life is rare. Additionally, the data produced by a machine can only be labeled after the machine failed. Both of these points make using data-driven methods for RUL estimation difficult. Semi-Supervised Learning (SSL) can incorporate the unlabeled data produced by machines that did not yet fail into data-driven methods. Previous work on SSL evaluated approaches under unrealistic conditions where the data near failure was still available. Even so, only moderate improvements were made. This paper defines more realistic evaluation conditions and proposes a novel SSL approach based on self-supervised pre-training. The method can outperform two competing approaches from the literature and the supervised baseline on the NASA Commercial Modular Aero-Propulsion System Simulation dataset.

1. INTRODUCTION

Predictive Maintenance (PDM) is one of the core pillars of Industry 4.0 and enables more cost-effective operation of machinery. While early approaches to PDM focused on hand-crafted, physical models and heuristics, nowadays data-driven methods are on the rise. Fueled by massive amounts of data provided by an increasing number of sensors, data-driven PDM makes hand-crafting physical models less necessary. Due to the advent of deep learning, data-driven models can ingest even more data without the need for specialized feature engineering. Nevertheless, PDM suffers from a server data imbalance as data from healthy machines is far more ubiquitous than data from degraded or faulty ones. This makes training effective data-driven models a challenging task.

Remaining Useful Lifetime (RUL) estimation, as a sub-field of PDM, is defined as "the length from the current time to the end of the useful life". (Si, Wang, Hu, & Zhou, 2011) It plays a vital role in effectively scheduling maintenance operations. Unfortunately, labeling data for RUL estimation is only possible after a machine fails, making it hard to acquire enough labeled data for conventional, supervised approaches to work. On the other hand, large amounts of unlabeled data are available from machines that did not yet fail. SSL can be a possible solution for this problem.

It makes it possible to distill knowledge from large amounts of unlabeled data, thus lowering the amount of labeled data needed to achieve good performance. Specifically, SSL aims to learn a conditional distribution $P(y|x)$ where $x \sim P(X)$ are the available features (i.e. sensor readings) and $y \sim P(Y)$ are the labels (i.e. the RUL). The learning algorithm has access to the set of labeled training data $D_L = \{(x_1, y_1), \ldots, (x_i, y_i)\}$ and the mutually exclusive set of unlabeled data $D_U = \{x_{i+1}, \ldots, x_{i+j}\}$.

Recent works, e.g. (Listou Ellefsen, Bjørlykhaug, Æsøy, Ushakov, & Zhang, 2019; Yoon et al., 2017), have shown promising results using different SSL methods for RUL estimation. More specifically, they were able to modestly reduce the test Root Mean Squared Error (RMSE) and RUL-Score on subsets of the NASA Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset (Saxena & Goebel, 2008) using as few as 1% of the labeled data. Although these findings are leading in the right direction, there are some shortcomings this paper wants to address. First, the previous work only evaluates their SSL approaches on one of the four subsets of the C-MAPSS dataset. As these subsets are relatively small (compared to other deep learning data sets),

https://doi.org/10.36001/IJPHM.2022.v13i1.3096

at least all of the subsets should be investigated to see if the performance improvements are universal. Secondly, the time series of the unlabeled data were used as-is. In our previous work (Krokotsch, Knaak, & Gühmann, 2020), we have shown that unlabeled time series data for RUL estimation should not contain time steps at the point of failure. If they do, the whole time series could be labeled trivially, making the use of SSL unnecessary. Therefore, the previous studies on SSL for RUL estimation may produce overly optimistic results, as the approaches have access to more data near failure than could be considered realistically. This *grade of degradation* of the unlabeled data, i.e. how far from failure the machines in it are, is of significant importance for SSL.

As previous work produced only modest improvements, we propose a novel SSL method based on self-supervised pre-training. Self-supervision has been proven useful for pre-training deep neural networks on unlabeled data (Doersch, Gupta, & Efros, 2015; Gidaris, Singh, & Komodakis, 2018; Devlin, Chang, Lee, & Toutanova, 2019). Our pre-training task is to estimate the time between two time steps in a time series as a proxy for the difference in RUL. Afterward, the pre-trained network is fine-tuned on labeled data. We will conduct experiments on all four subsets of C-MAPSS comparing our approach and two common SSL approaches, i.e. Restricted Boltzmann Machines (RBMs) and Autoencoders (AEs), to a supervised baseline. This will serve to answer our following research questions:

- Do findings of previous SSL studies on RUL estimation hold when taking the *grade of degradation* of the unlabeled data into account?
- Can self-supervised pre-training improve on other pre-training-based SSL approaches for RUL estimation?

The remaining paper is structured as follows. First, we will lay out the related work on RUL estimation with deep neural networks, Semi-Supervised Learning and self-supervised learning in section 2. In section 3, we will describe our network architecture, our semi-supervised learning approach, and our self-supervised pre-training. Afterward, we explain our experimental setup, including the data, performance metrics, and competing approaches, as well as the evaluation procedure and hyperparameter selection. Section 5 is concerned with the presentation and discussion of our results, while section 6 concludes this paper and gives an outlook on future work. The code to reproduce the results of this paper is available on GitHub: `www.github.com/tilman151/self-supervised-ssl`

2. RELATED WORK

This section gives an overview of the current state of the literature. First, we will discuss RUL estimation with Deep Neural Networks (DNNs). Second, we will investigate Semi-Supervised Learning and self-supervised learning in the scope of RUL estimation.

2.1. Remaining Useful Lifetime Estimation

RUL estimation is often treated as a regression problem. Recent work focuses mainly on DNNs because they work on the raw data and do not require hand-crafted features. Obviously, the early works were focuses on shallow networks and Multi-Layer Perceptrons (MLPs) (Gebraeel, Lawley, Liu, & Parmeshwaran, 2004). Later ones settled on Convolutional Neural Networks (CNNs) and LSTM as network architectures.

Long Short Term Memory Networks (LSTMs) are Recurrent Neural Networks (RNNs) and a natural fit for the time series data seen in RUL applications. They process one time step at a time with the help of an internal cell state derived from all previously seen time steps. (Zheng, Ristovski, Farahat, & Gupta, 2017) used LSTMs on three benchmark datasets and found them working best compared to MLPs and CNNs. Meanwhile, (Wu, Yuan, Dong, Lin, & Liu, 2018) compared LSTMs against vanilla RNNs and GRUs. They declared LSTMs superior, as well.

CNNs on the other hand seem better suited for image data than time series. But, using 1d-convolution instead of 2d we can use it for RUL estimation, too. In a comparison, (Bai, Kolter, & Koltun, 2018) found CNNs equal to LSTMs in performance, even though they are faster in inference and training. First attempts at RUL estimation from (Sateesh Babu, Zhao, & Li, 2016) were still worse than LSTMs. They were still using 2d-convolution and when (Li, Ding, & Sun, 2018) switched to 1d, they were able to surpass LSTMs altogether. (Zhu, Chen, & Peng, 2019) combined features from multiple hidden layers with their multi-scale CNN which did better on their dataset than traditional CNNs. Instead of using raw time series data, they transformed it to the frequency domain and used that as the input of their network. (Jiang, Lee, & Zeng, 2020) resorted to combining both network types and report better performance.

Next to all networks were trained against Mean Squared Error (MSE) (Zheng et al., 2017; Wu et al., 2018; Sateesh Babu et al., 2016; Zhu et al., 2019; Jiang et al., 2020). Only (Li et al., 2018) used RMSE.

2.2. Semi-Supervised Learning

SSL can be divided into several sub-fields (van Engelen & Hoos, 2020). The most common distinction is the goal of the training process itself. While transductive methods are only concerned with providing the labels for the unlabeled training data, inductive methods yield a model that can be used on unseen data. We will focus on the inductive methods, as for PDM applications we need to apply the trained model on unseen data after training. Even though there are many diverse approaches to SSL (e.g. S3VMs), we will narrow our

perspective to the ones applicable to DNNs.

Wrapper methods offer some of the oldest SSL approaches. They rely on producing pseudo-labels for the unlabeled portion of the data and then using it in conjunction with the labeled data to train supervised. Self-training is one of the most basic methods and was published in 1995 by (Yarowsky, 1995). First, a model trained only on the labeled data provides the pseudo-labels for the unlabeled data. Afterward, a final model is trained on the combined labeled and pseudo-labeled data.

Pre-training-based methods rely on unsupervised learning. The DNN or a part of it is trained with an unsupervised learning method and taken as the initialization for a supervised learning stage. The literature provides examples for several unsupervised approaches, most commonly AEs, used for pre-training. (Cheng, Zhou, Ma, Wu, & Yuan, 2019) used deep autoencoders for semi-supervised machine translation. (Kingma, Rezende, Mohamed, & Welling, 2014) used variational autoencoders for semi-supervised image classification.

There are several deep learning methods for SSL that directly incorporate an unsupervised component into their loss. Examples are Ladder Networks (Rasmus, Valpola, Honkala, Berglund, & Raiko, 2015) which incorporate an autoencoder reconstruction loss, or Pseudo-ensembles (Bachman, Alsharif, & Precup, 2014) which use a consistency loss between the outputs of a parent network and perturbed child networks for the unlabeled data. The survey of (van Engelen & Hoos, 2020) gives an excellent overview of the previously mentioned methods.

There are a few papers on SSL for RUL estimation. (Listou Ellefsen et al., 2019) used a RBM for pre-training on the NASA C-MAPSS dataset. (Yoon et al., 2017) pre-trained their network on the same dataset with a variational autoencoder. (He, Dai, Lu, & Mou, 2018) used ladder networks for RUL estimation of centrifugal pumps. Unfortunately, the field of SSL suffers from a multitude of evaluation setups, which makes comparing approaches difficult (Oliver, Odena, Raffel, Cubuk, & Goodfellow, 2018). SSL for RUL estimation is not excluded from this issue, as many papers report results only on parts of their datasets, or omit critical information like the amount of available labeled data. We aim to take these pitfalls into account in this work.

2.3. Self-Supervised Learning

Like many methods in deep learning, self-supervised learning first succeeded in computer vision and spread to different fields from there. The aim was to learn features that are beneficial for solving common tasks (image classification, object detection, etc.) in the absence of labeled data. For this, a so-called pre-text task is defined to train a neural network, which is then used as a feature extractor for the real task. For example, (Doersch et al., 2015) predicted the position of an image patch, cut from a larger image, in relation to another patch from the same image. The trained network was then able to perform object detection. Another example is (Gidaris et al., 2018), who used predicting image rotation as a pre-text task for classification. Approaches like these were able to produce state-of-the-art results in a semi-supervised regime (large amount of unlabeled, small amount of labeled data) but could not outperform supervised approaches trained on large-scale datasets.

Self-supervised learning gained prominence as a pre-training technique in natural language processing through (Devlin et al., 2019). Their model, named BERT, was pre-trained on the pre-text task of predicting missing words in sentences sampled from a 3.3M word dataset, including the whole of the English Wikipedia. The model was then able to produce state-of-the-art results on eleven benchmark tasks by training a single layer on top of the pre-trained network.

Metric learning is another possible pre-text task that can be used supervised and self-supervised. It aims to learn a distance or similarity metric between pairs of data points. The recent work of (Musgrave, Belongie, & Lim, 2020) gives an excellent overview of popular methods but shows that newer, more complex approaches perform only as good as older, simple ones. This leads us to the conclusion that even simple metric learning methods could be used as a self-supervised pre-text task, as well.

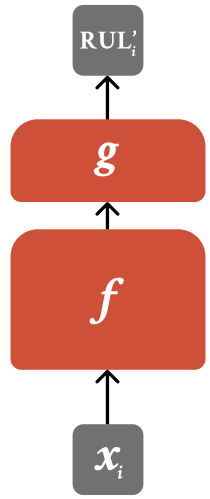
There is, to our knowledge, still a lack of work in self-supervised methods for multivariate time series data as found in PDM applications. Recently, (Franceschi, Dieuleveut, & Jaggi, 2019) proposed a metric-learning-based pre-text task for time series. They trained siamese networks (Baldi & Chauvin, 1993; Bromley et al., 1993) to predict the similarity of two time series snippets with a triplet loss. Their pre-trained network outperformed dynamic time warping in an unsupervised regime, other deep learning approaches in a semi-supervised regime and yield competitive results when compared to supervised state-of-the-art approaches. This hints at self-supervised learning as a promising direction for pre-training on time series data.

3. METHODS

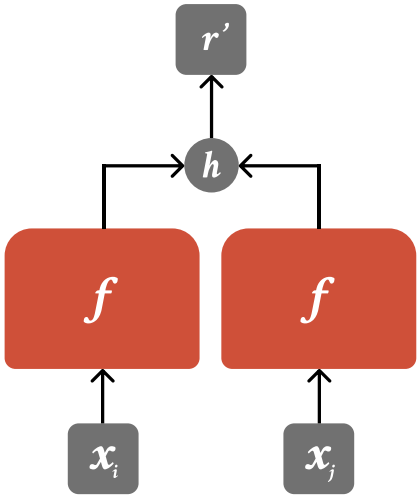
In this section, we will describe the neural network we use for RUL estimation, SSL via pre-training, and our novel self-supervised pre-training technique.

3.1. RUL Estimation Network

In general, DNNs for regression follow a common architecture, as seen figure 1a. They consist of two networks: the feature extractor f and a regression head g. Networks for RUL estimation are no different, estimating the RUL value of



(a) RUL Estimation Network



(b) Self-Supervised Siamese Network

Figure 1. **Overview of the networks used in this work:** Depicted are the feature extractor f, the regression head g, and the distance function h. The supervised RUL estimation network takes a time frame x_i as its input and predicts the RUL value RUL'_i. The siamese network produces a pair of embeddings from the inputs x_i and x_j. The distance $h(f(x_i), f(x_j))$ is r', the predicted relative RUL. The trainable parts of the networks are depicted in red.

a sample x as:

$$\text{RUL}' = g(f(x)) \tag{1}$$

As seen in section 2.1, the feature extractor can take the form of a CNN, LSTM or any other network architecture. The regression head, on the other hand, almost always takes the form of a simple linear layer ($ax + b$) or a MLP.

We expect the network input $x^{(k)}_{(i-w):i}$, or x_i for short, to be a time frame of size w from a multivariate time series k ending at time step i. Therefore, we use a simple 1d-CNN feature extractor with a fully-connected layer as the regression head. The feature extractor consists of multiple 1d-CNN layers with Dropout, Batch Normalization (Ioffe & Szegedy, 2015), and a ReLU activation function, followed by a single linear layer also with Batch Normalization and ReLU. The complete architecture is depicted in figure 2. We selected a CNN because they are faster to train than e.g. LSTMs and have less trainable parameters than a MLP with similar performance. In this work, we will focus on this feature extractor only, as we are mainly concerned with the influence of pre-training it on unlabeled data. Furthermore, previous works imply that differences in performance between different extractor architectures for RUL estimation are marginal (Li et al., 2018).

We train the network by mini-batch gradient descent with a RMSE loss:

$$\mathcal{L}_{\text{RMSE}} = \sqrt{\frac{1}{|X|}\sum_{i=1}^{|X|}(\text{RUL}_i - \text{RUL}'_i)^2} \tag{2}$$

where X is a (mini-)batch of samples $\{x_1, ..., x_{|X|}\}$, RUL_i the RUL value associated with x_i and $\text{RUL}'_i = g(f(x_i))$.

3.2. SSL through Pre-Training

SSL is a machine learning regime that includes labeled and unlabeled data into the training process. In this paper, we focus on the two-stage approach of combining unsupervised pre-training with supervised fine-tuning. First, the feature extractor f is trained using a method that does not require labels on the features of the labeled and unlabeled data. Afterward, the pre-trained feature extractor is used as the starting point for the conventional, supervised training as described in the previous section. The regression head g is still initialized randomly.

This procedure aims for the feature extractor to learn useful features from the unsupervised data that are beneficial to the supervised training. For this to work, it is necessary to assume that the marginal data distribution $P(X)$ contains information about the conditional distribution $P(\text{RUL}|X)$ we want to learn (van Engelen & Hoos, 2020). It follows that we have to assume that the labeled and unlabeled data were generated by the same distribution P.

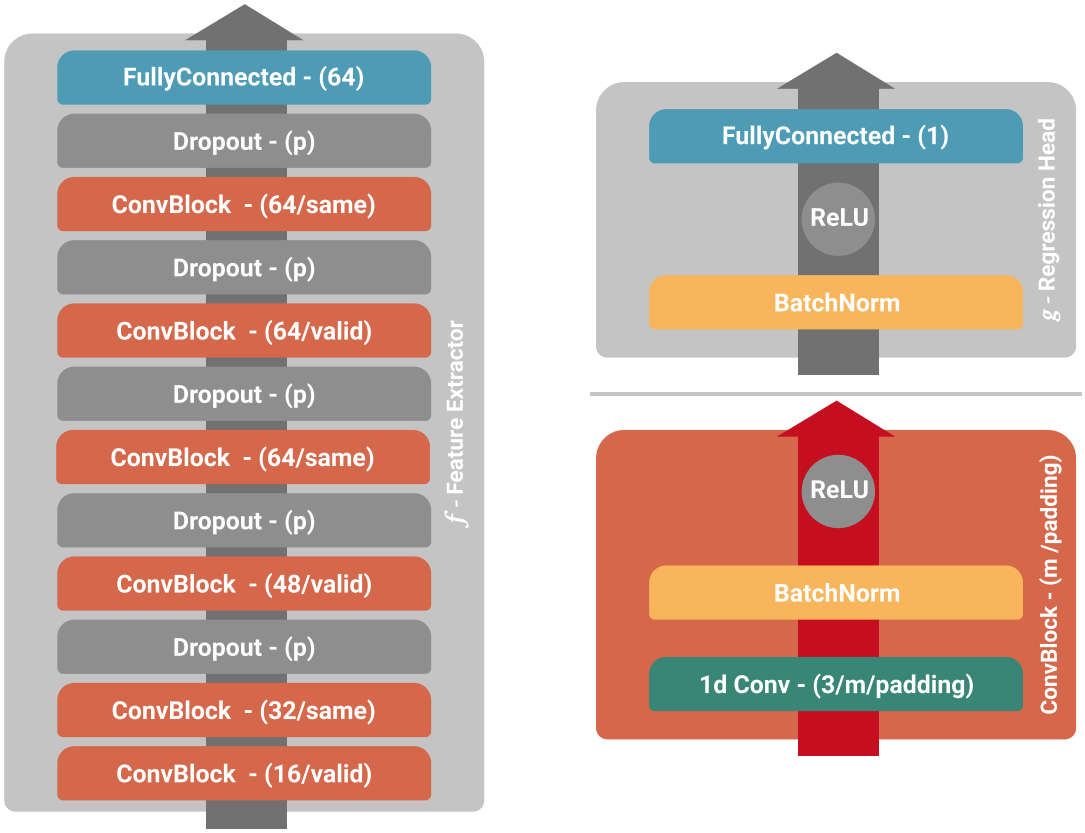


Figure 2. **Architecture of the used Networks:** The feature extractor f consists of six $ConvBlocks$. These blocks are stacks of 1d-convolutions of kernel size three, m filters, and either $same$ or $valid$ convolution padding with zeros, followed by Batch Normalization and a Rectified Linear Unit (ReLU) activation function. The even layers use valid padding and the odd layers use same padding. This makes it possible to use more $ConvBlocks$ with valid padding only as same padding does not reduce the size of the input. In between the $ConvBlocks$, we place 2d-dropout that drop whole time steps from their input. The probability of dropping a time step is p. The last layer of the feature extractor is a fully-connected layer reducing its input to a latent dimension of 64. The regression head g is a Batch Normalization layer followed by a ReLU activation function and a fully-connected regression layer with a single unit.

3.3. Self-Supervised Pre-Training

In this section, we lay out the intuitive motivation behind our proposed self-supervised pre-training through analyzing the latent space of trained networks. Afterward we derive a pre-text task from this intuition and devise a training regime.

3.3.1. Motivation

To be able to design a suitable pre-text task, one has to understand the supervised training process and its shortcomings. The architecture of our neural network suggests the following division of labor between the feature extractor f and the regression head g. The regression head consists mostly of affine operations and a single non-linearity in form of the ReLU activation function. It can, therefore, learn only a nearly linear mapping between the outputs of the feature extractor and the RUL. Consequently, an optimal feature extractor needs to linearize the relationship between its output and the RUL to match the capabilities of the regression head. In other words, the feature extractor maps a non-linear degradation process to a linear one, while the regression head maps this linear degradation process to RUL.

To verify this assumption, we will look at the features learned by the feature extractor f trained with different amounts of labeled data. One of them is trained with many labeled data points and achieves a low validation error, while the other is trained on only a few labeled data points and achieves a much higher validation error. Unfortunately, high-dimensional data, such as the extracted features, is impossible to understand for humans. Therefore, we project it into two dimensions by using Uniform Manifold Approximation and Projection (UMAP) (McInnes, Healy, Saul, & Grossberger, 2018), and plot the features in a scatter plot. UMAP is a reasonable choice for visualizing the learned features, as we do not need to make assumptions about it as with, e.g., principal component analysis.

Figure 3a depicts the features extracted from the validation data of C-MAPSS subset FD004 after training on all available labeled data. The network achieves a validation MSE loss of 19.3. We can observe a snake-like structure in the features that broadens on the left side and narrows on the right side. Through coloring the data points according to their associated RUL value, we can see that the left end of the snake corresponds to high and the right side to low RUL values. Furthermore, no discernible sub-structures are apparent which shows that the network does not differentiate between time frames from different time series and the same RUL. The clear gradient of RUL values inside the snake-like structure hints at a linear relationship between RUL and the learned features.

Figure 3b depicts the features extracted by a feature extractor trained only on three labeled time series, which corresponds to 2% of the available labeled data. The network achieves a validation loss of 31.8. Again, we can observe the snake-like structure but this time the high-RUL end is much more feathered out. Additionally, the RUL values are not as clearly separated as in the previous figure. The noisier gradient of RUL values can be interpreted as a failure of the feature extractor to linearize the relationship between RUL and its features.

Comparing the two plots supports our theory about the nature of the features learned by the feature extractor. A suitable pre-text task should result in a feature extractor that effectively takes care of the non-linearity of the degradation process. We can, furthermore, conclude via transitivity that if the relationship between RUL and learned features is linear, the relationship between two learned feature vectors is linear as well.

3.3.2. Definition of the Pre-Text Task

To derive a pre-text task from the intuition gained in the previous section, we need to formalize it. We want to train a feature extractor f so that the extracted features for each data point can be mapped to a RUL by a (nearly) linear function h (in the supervised case, the regression head):

$$\text{RUL} = h(f(x)) \tag{3}$$

For this derivation, we will assume that h is a completely linear function. With this assumption, we can make the following statement:

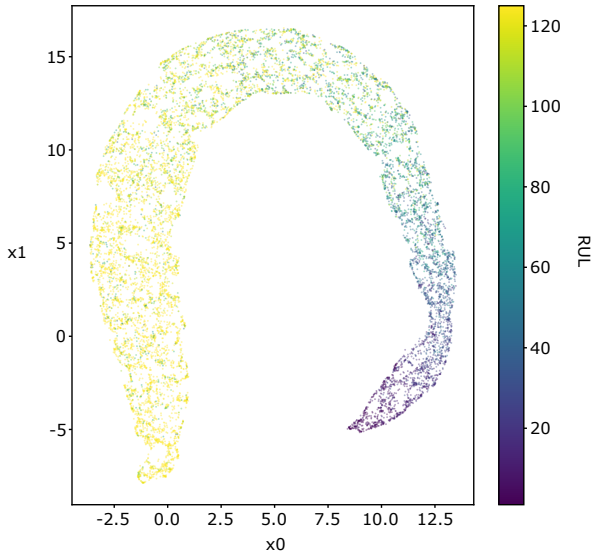
$$\begin{aligned} \text{RUL}_i - \text{RUL}_j &= h(f(x_i) - h(f(x_j)) && (4)\\ \text{RUL}_i - \text{RUL}_j &= h(f(x_i) - f(x_j)) + c && (5)\\ r &= h(f(x_i) - f(x_j)) + c && (6) \end{aligned}$$

where c is a constant equal to the bias term of h. The difference in RUL of two data points has to be proportional to the difference between their extracted features. In consequence, we should be able to train an optimal feature extractor by letting it predict the difference in RUL of two data points for a given linear function h. We will hereafter refer to this difference as the relative RUL r between two data points.

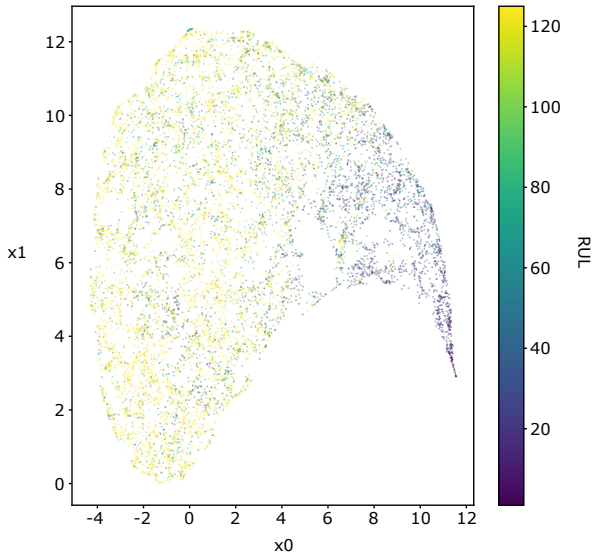
Because the pre-text task needs to work with unlabeled data, we cannot calculate r directly from the ground truth RUL. We need to derive r from the data itself, which is why our method falls under the category of self-supervised learning. By definition, RUL can be calculated as:

$$\text{RUL}_i^{(k)} = |k| - i \tag{7}$$

where k is a time series, i is the current time step, and $|k|$ is the length of the time series. The relative RUL r of two data points i and j from the time series k can therefore be



(a) All labeled time series



(b) Three labeled time series

Figure 3. **Features produces by trained feature extractors:** The depicted features are UMAP projections of the output of a feature extractor f. The first plot shows an f trained on all labeled data from FD004, while the second one shows an f trained on only three time series. Both were trained for 10 epochs. The color indicates the RUL value associated with the embedding. Both (a) and (b) were generated from the validation data of FD004. The random seed for both training runs was fixed.

calculated as:

$$r = \text{RUL}_i^{(k)} - \text{RUL}_j^{(k)} \quad (8)$$
$$r = (|k| - i) - (|k| - j) \quad (9)$$
$$r = j - i \quad (10)$$

This eliminates the length of the time series from the equation and makes it possible to use data from machines that did not yet fail. Combining equation 10 and 6 we get the following formulation of the pre-text task:

$$j - i = h(f(x_i^{(k)}) - f(x_j^{(k)})) \quad (11)$$

We can ignore the constant c from equation 6 here, as it is not dependent on f and therefore does not influence the optimization process. Alternatively, we can restrict h to have a bias term equal to zero.

Unfortunately, the literature for C-MAPSS (Heimes, 2008) does not adhere to the RUL definition of equation 7. Instead, they declare a maximum value RUL_{MAX} which results in a piece-wise linear RUL function:

$$\text{RUL}_i^{(k)} = \max\left(\text{RUL}_{\text{MAX}},\ |k| - i\right) \quad (12)$$

This prohibits the step from equation 9 to 10. Nevertheless, equation 10 is still an acceptable approximation of r, if we assume that $i < j$ and $i - j \leq \text{RUL}_{\text{MAX}},\ \forall i, j$. Additionally, we can now normalize r to the range of $[0, 1]$ by dividing it by RUL_{MAX}. The final pre-text task for the C-MAPSS dataset is then:

$$h(f(x_i^{(k)}) - f(x_j^{(k)})) \approx \frac{j - i}{\text{RUL}_{\text{MAX}}} \quad (13)$$
$$i < j,\ i - j \leq \text{RUL}_{\text{MAX}} \quad \forall i, j$$

3.3.3. Training Procedure

The goal of our pre-training is training a neural network on estimating the target value r from equation 13. Afterward, we extract the feature extractor f from the pre-trained network to initialize the RUL estimation network. We realize this goal by training siamese networks (Bromley et al., 1993; Baldi & Chauvin, 1993) with metric learning. Metric learning is concerned with learning a distance metric of data points, in our case the relative RUL r. Siamese networks take the form of the feature extractor f and a distance function $h(a, b)$ operating on two samples, x_i and x_j, so that:

$$r' = h(f(x_i), f(x_j)) \quad (14)$$

where r' is the predicted value of r. To be consistent with equation 13, h needs to be a linear function. We, instead, decide to use a function defined as:

$$h(a, b) = \left|\left| \frac{a}{||a||} - \frac{b}{||b||} \right|\right|^2 \quad (15)$$

where $||\cdot||$ is the euclidean norm. Our reason is that the euclidean norm behaves linear in the desired domain and has the added benefit of being symmetric. Dividing the embeddings by their norm restricts the embedding space to a hyper ball, which often is found to be useful for metric learning (Musgrave et al., 2020).

Figure 1b depicts the schematic structure of the siamese networks. We use MSE as a loss function with mini-batch gradient descent to train the siamese networks:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{|X|} \sum_{k=1}^{|X|} (r_k - r'_k) \tag{16}$$

For each (mini-)batch X, we sample $|X|$ time series from the training data, uniformly sample pairs of time frames $x_i^{(k)}$ and $x_j^{(k)}$, and calculate r_k according to equation 13. Because both samples are time frames from the same time series, a difference between i and j that is much smaller than the length of the time frame results in significant overlap between the samples. Preliminary experiments have shown that this can destabilize the pre-training. Therefore, we introduce a minimum distance for sampling pairs, which is regarded as a hyperparameter. We used the Adam optimizer (Kingma & Ba, 2014) for training with the default values of 0.9 and 0.99 for β_1 and β_2.

4. EXPERIMENTAL DESIGN

This section describes the data set used for our experiments, which performance metrics were used, and how the evaluation was conducted.

4.1. Data

We evaluate the effect of our pre-training technique on the publicly available NASA C-MAPSS dataset (Saxena & Goebel, 2008). It is a dataset of simulated aero engines commonly used to benchmark RUL estimation algorithms. Each engine starts its simulation with a random level of initial wear that is still considered as healthy. At a random time during the simulation, a fault is inserted into the engine. The simulation ends when the engine is no longer functional. For each engine, the time series of 3 operation parameters and 21 sensor readings are recorded. The sensor readings are subjected to additive sensor noise.

The dataset contains four subsets (FD001 - FD004) of different operating conditions and possible fault modes. Engines in FD001 and FD002 experience only high-pressure compressor (HPC) degradation, while the ones in FD003 and FD004 can additionally experience fan degradation. Further, in FD001 and FD003 are engines run under only one operating condition, while engines from FD002 and FD004 vary between six. Each subset is split into training and test data by the dataset authors. The training data contains multivariate time series data of aero engines up until the time step they failed. Each time series can be considered as a different engine of the same type. The test data's time series stop at a random time step before failure, for which the true RUL label is provided. Table 1 summarizes the details of the dataset. We construct one validation set from each training set by taking 20% of the time series.

Dataset	**FD001**	**FD002**	**FD003**	**FD004**
# Training Engines	100	260	100	249
# Test Engines	100	259	100	248
# Operation Conditions	1	6	1	6
# Failure Modes	1	1	2	2

Table 1. C-MAPSS Dataset

For pre-processing, we follow (Li et al., 2018) and select 14 of the 21 sensor channels with the indices 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, and 21 as the input of the network. The features are scaled by channel, using a min-max scaling calculated on each subsets' training set. We then use a sliding window with a step size of one to obtain frames of unified length in the time dimension. We use a window size of 30, 20, 30, and 15 for the subsets 1-4. These sizes were determined by the length shortest time series in each subsets test set. The RUL labels are calculated by a piece-wise RUL function (Heimes, 2008) with a maximum value RUL_{MAX} of 125.

4.2. Data Scenarios

As described in the introduction, unlabeled RUL data cannot contain the point of failure as we could simply compute the labels otherwise. It follows that we have to truncate the time series before failure when studying SSL, as the model would have access to more failure data than normally available otherwise. How early we truncate the time series represents the *grade of degradation* of the engine that the time series was collected from. Previous work on SSL for RUL estimation varied only the amount of labeled data available, i.e. how many engines had already failed. Our experimental design includes varying the *grade of degradation* for the unlabeled data, too.

Adapting our previous work (Krokotsch et al., 2020), we impose data scenarios on each subset. In our case, a data scenario is characterized by the *number of failed engines* and *grade of degradation* of the unlabeled ones. Both factors are interpreted as percentages. A data scenario of *number of engines* at $n\%$ would mean that only $n\%$ of the machines in the subset are used as labeled data for training. The rest is assumed to have not yet failed and is used as unlabeled data. This limits the amount of machine-to-machine variance that is covered by the labeled data. A data scenario of *grade of degradation* at $n\%$ would mean that only the first $n\%$ of time steps of each unlabeled time series is available during training, as seen in figure 4. This effectively limits the amount of available data near failure.

We use five different *grades of degradation* of 40%, 60%, 70%,

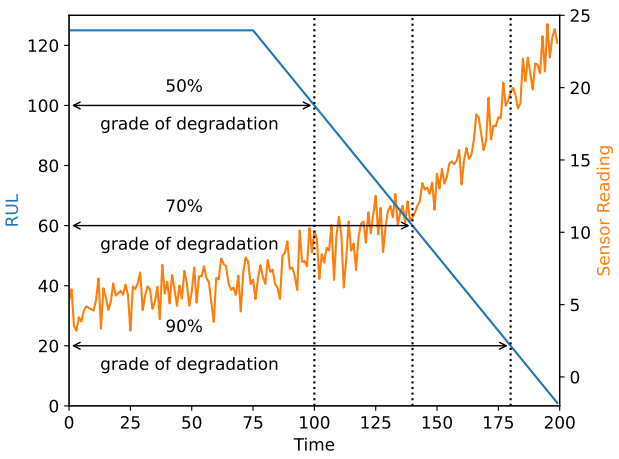


Figure 4. **Grade of Degradation:** The plot depicts the time series of the RUL and an exemplary sensor reading of a fictitious engine in the C-MAPSS dataset. The arrows indicate which part of the time series is used for a given grade of degradation.

80% and 90% for our evaluation. A *grades of degradation* of 100% would not make sense, as the unlabeled machines would have failed already and could be used as labeled data. The lower limit of percentages is chosen due to the piece-wise linear degradation model. Using fewer than 40% of the time steps would mean that the unlabeled data contains next to no data with a RUL of less than RUL_{MAX}. This means that the unlabeled data would come from completely healthy machines and may add no benefit to training.

The number of failed engines is set at 2%, 10%, 20%, 40%, 100%. Using 100% of failed machines means that no unlabeled data is available and pre-training is conducted on the labeled data only. We decided on these percentages because preliminary experiments did not show any degradation in performance for more than 40% compared to using 100% of the engines. The lower limit of 2% is chosen because it results in at least one failed engine for each subset.

4.3. Performance Metrics

We employ two common performance metrics from the field of PDM. The first is the RMSE, as described in equation (2) over the test set. The second is the *RUL-Score*, first proposed in the PHM 2008 Data Challenge (Heimes, 2008). It is calculated as follows:

$$s_i = \begin{cases} e^{-\frac{\Delta\text{RUL}_i}{13}} - 1 & \text{for } \Delta\text{RUL}_i < 0 \\ e^{\frac{\Delta\text{RUL}_i}{10}} - 1 & \text{for } \Delta\text{RUL}_i \geq 0 \end{cases} \tag{17}$$

where ΔRUL_i is the difference between predicted and true RUL for sample x_i. This metric penalizes overestimating RUL more than underestimating it as the former has a bigger impact on maintenance actions. We report the sum over all samples in the test set following previous work in the field.

We will discuss the results using both metrics. Although, we will focus on RMSE as we view it as the more intuitive measure of performance.

4.4. Supervised Baseline

The baseline for our experiments is training the RUL estimation network in a supervised fashion as described in section 3.1. The training will only incorporate the available labeled data. We used the Adam optimizer (Kingma & Ba, 2014) with the default values for 0.9 and 0.99 for β_1 and β_2.

4.5. Competing Approaches

Aside from the baseline, we compare our pre-training approach to two other methods found in the literature. The first approach is unsupervised pre-training via a deep AE. We construct the AE so that our feature extractor network is used as the encoder. The decoder is an exact mirror image of the encoder with transposed convolutions replacing the convolutional layers. The AE is then trained to reconstruct its input via mini-batch gradient descent with a MSE loss. As for the baseline, we used the Adam optimizer.

The second approach is unsupervised pre-training via a RBM. The first layer of the feature extractor is interpreted as a RBM with Gaussian visible units and ReLU hidden units and trained until convergence. The other layers of the feature extractor remain randomly initialized. Again, Adam was used as the optimizer.

Even though (Listou Ellefsen et al., 2019; Yoon et al., 2017) reported results for RBMs and variational AEs, we choose to reproduce the competing approaches ourselves. This has several reasons. First, the mentioned papers evaluated their approaches only on FD001 or FD004 which paints a limited picture of the approaches' performance. Second, the mentioned papers used a slightly different experimental setup, i.e. a different RUL_{max}, making comparison difficult. Furthermore, the aforementioned RUL_{max} was optimized as a hyperparameter in one of the papers. As this value controls the possible range of the performance metrics, optimizing it results in improperly optimistic results.

4.6. Evaluation Procedure

The baseline, our approach and the competing SSL methods are evaluated on each of the subsets of the C-MAPSS dataset subject to each of our selected data scenarios. This results in 100 different experimental setups (4 subsets, 5 number of failed engines, 5 grades of degradation) for each SSL method. The baseline includes only 20 setups, as it does not use unlabeled data which makes varying the grade of degradation superfluous.

Each experimental setup is replicated ten times. For each replication, a new split of labeled and unlabeled data is randomly samples according to the data scenario. The splits are held constant across methods, i.e. the baseline receives the same labeled data as the SSL methods given the same data scenario. This makes comparison easier as much more replications than ten would be needed for the low labeled data scenarios to receive statistically stable performance estimates.

The pre-training stage for the SSL methods receives the labeled and unlabeled portions of the data and trains for at least 100 epochs. The RBM is trained for five epochs as it contains only one layer. Early stopping is used to select the model with the lowest validation loss. For our method we monitor the MSE loss for the relative RUL target and for the autoencoder the MSE reconstruction loss.

The supervised training stage receives only the labeled portion of the data and is trained for at least 200 epochs. The weights of the selected pre-trained model are used to initialize the network in this stage for the SSL methods. The baseline is initialized randomly. It should be noted that we divide the output of the feature extractor by its norm if the pre-training was self-supervised. Again, early stopping on the validation RMSE is used to select the best model for which the performance metrics over the test set are calculated. We report the performance for each subset/data scenario combination separately and averaged over the ten replications.

4.7. Hyperparameter Selection

We began hyperparameter selection using the fixed network architecture shown in figure 2 and conducted a random search in two steps. First, the hyperparameters of the supervised stage (i.e. learning rate, dropout, and batch size) were optimized for each subset of C-MAPSS. A network was trained in a supervised fashion with all available labeled data as described in section 4.6. After 100 trials, the hyperparameters of the network with the best validation RMSE were selected. The hyperparameters in question can be seen in table 2 under supervised stage.

In a second step, the hyperparameters of the pre-training stages (i.e. learning rate, dropout, batch size, and minimum pair distance) were optimized similarly. The networks are trained without labels at 80% grade of degradation as described in section 4.6. For our method we selected hyperparameters according to the validation MSE loss for the relative RUL target and for the autoencoder according to the validation MSE reconstruction loss. As stability was an issue, each hyperparameter configuration was trained five times and the mean of these replications is used for selection. For the RBM we adopted the hyperparameters from (Listou Ellefsen et al., 2019). We set the learning rate for this method to 10^{-4} by hand as it was not given in the paper.

It should be noted that all optimizations used no labels at all and can therefore be conducted independently from the amount of labeled data. The selected hyperparameters can be seen in table 2, too.

5. RESULTS

First, we will describe the results of our experiments for each C-MAPSS subset. Afterward, we will interpret the findings and set them into context.

5.1. Comparison of Approaches

Our experiments produced too many data points to present them all in detail. We will therefore show only slices of our results as plots. The results plotted against the percentage of labeled data are shown in figures 5 and 6 at 80% degradation. The results plotted against grade of degradation are shown in figures 7 and 8 at 2% of the labeled data. The complete results are shown in appendix A in tables 3, 4, 5 and 6.

Overall we can conclude that a significant drop in baseline performance was mostly observable for very low amounts of labeled data. In figure 5 we can see that the performance of the baseline (blue) has a relatively small standard deviation for 100%, 40%, and 20% of labeled data. For the subsets FD002 and FD004, the mean and standard deviation increase only at 2% labeled data. For FD001, the performance already drops at 40% labeled data and for FD003 at 10%. One has to keep in mind, though, that FD002 and FD004 are more than twice as large as FD001 and FD003, which means that they have a higher amount of labeled data available at the same percentage.

Performance on **FD001** shows only marginal improvement through SSL. In figure 5 we can see that the median RMSE performances is well inside of each others IQRs with the exception of 40% labeled data where AE pre-training achieves much better performance. In one case, i.e. for 2% labeled data, the performance of RBM pre-training was even worse than the baseline. The RUL-Score metric in figure 6 paints a similar picture, even though pre-trained models seem slightly better for 20% and 40% labeled data. For 2% labeled data, the SSL approaches had a worse mean RUL-Score performance than the baseline. These findings are stable for different grades of degradation, as well. Figures 7 and 8 show no discernible trend with respect to the grade of degradation for any method. Overall, the AE seems to perform slightly better than the other approaches.

Performance on **FD002** clearly benefits from RBM and self-supervised pre-training. The self-supervised pre-training, in particular, beats the baseline and the other approaches most of the time, especially in low-labeled data scenarios. On the other hand, there is an extreme drop in performance for our method at 40% degradation. This is true for all percentages of

Hyperparameter	Search Space	Configuration for FD001	FD002	FD003	FD004
Supervised Stage					
Learning Rate	$\mathrm{qlogu}(1^{-4}, 1^{-1}, 5^{-5})$	0.0056	0.0903	0.095	0.06635
Dropout	$\mathrm{qu}(0.0, 0.5, 0.1)$	0.4	0.3	0.2	0.0
Batch Size	[64, 128, 256, 512]	128	512	64	64
Self-Supervised Pre-Training					
Learning Rate	$\mathrm{qlogu}(1^{-4}, 1^{-1}, 5^{-5})$	0.00015	0.01155	0.00615	0.07455
Dropout	$\mathrm{qu}(0.0, 0.5, 0.1)$	0.2	0.4	0.1	0.1
Batch Size	[64, 128, 256, 512]	64	64	64	64
Minimum Distance	[1, 10, 15, 30]	10	15	15	10
Autoencoder Pre-Training					
Learning Rate	$\mathrm{qlogu}(1^{-4}, 1^{-1}, 5^{-5})$	0.0001	0.0248	0.015	0.0006
Dropout	$\mathrm{qu}(0.0, 0.5, 0.1)$	0.1	0.4	0.0	0.0
Batch Size	[64, 128, 256, 512]	64	256	64	64
Minimum Distance	[1, 10, 15, 30]	1	15	1	10

Table 2. **Hyperparameters:** The search space $\mathrm{qlogu}(a, b, c)$ draws samples uniformly on a logarithmic scale from the interval $[a, b]$ quantized to c. The search space $\mathrm{qu}(a, b, c)$ draws samples uniformly from the interval $[a, b]$ quantized to c. The minimum distance was optimized for the autoencoder, as well, due to implementation reasons. It should not influence the results significantly, though.

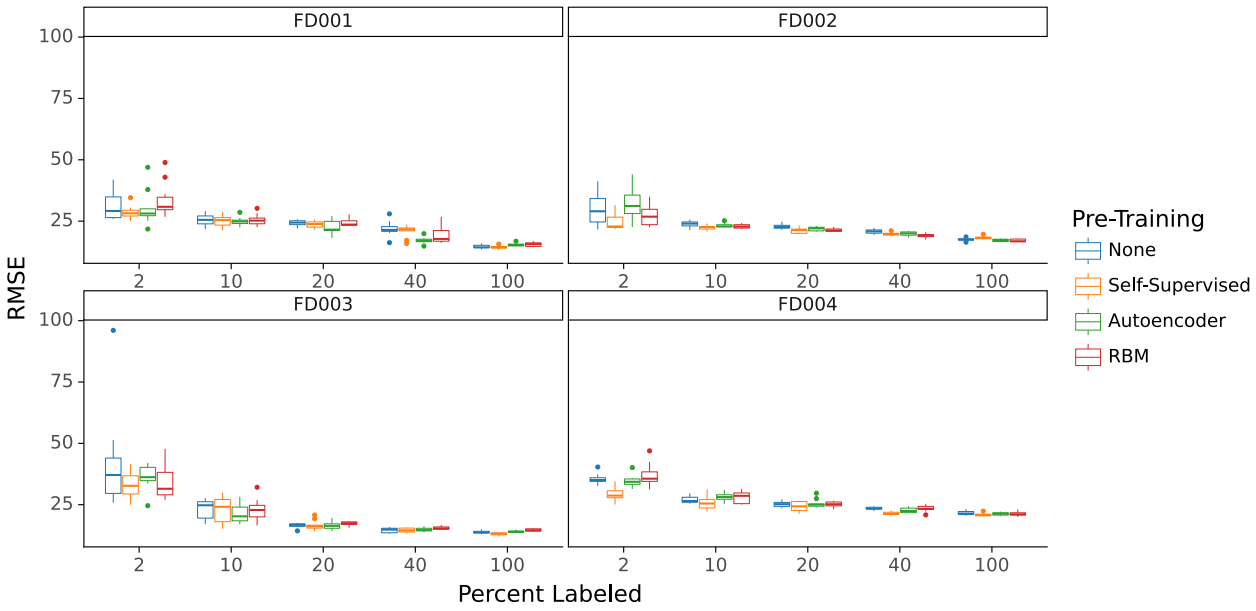


Figure 5. **RMSE at 80% degradation:** The plots show the results on the four subsets of the C-MAPSS dataset. We plot the RMSE test performance against the percentage of labeled data. The remaining data were used as unlabeled data with a grade of degradation of 80%. The box represents the interquartile range (IQR) and the middle line the median performance. The ends of the whiskers depict the minimum and maximum performance that was not deemed an outlier. Outliers are defined as more than 1.5 times IQR away from the lower or upper end of the box. We can see that the performance degrades only slowly with less labeled data. Significant drops in performance can be seen for 10% labeled data for FD003, 40% for FD001, and 2% for the other subsets.

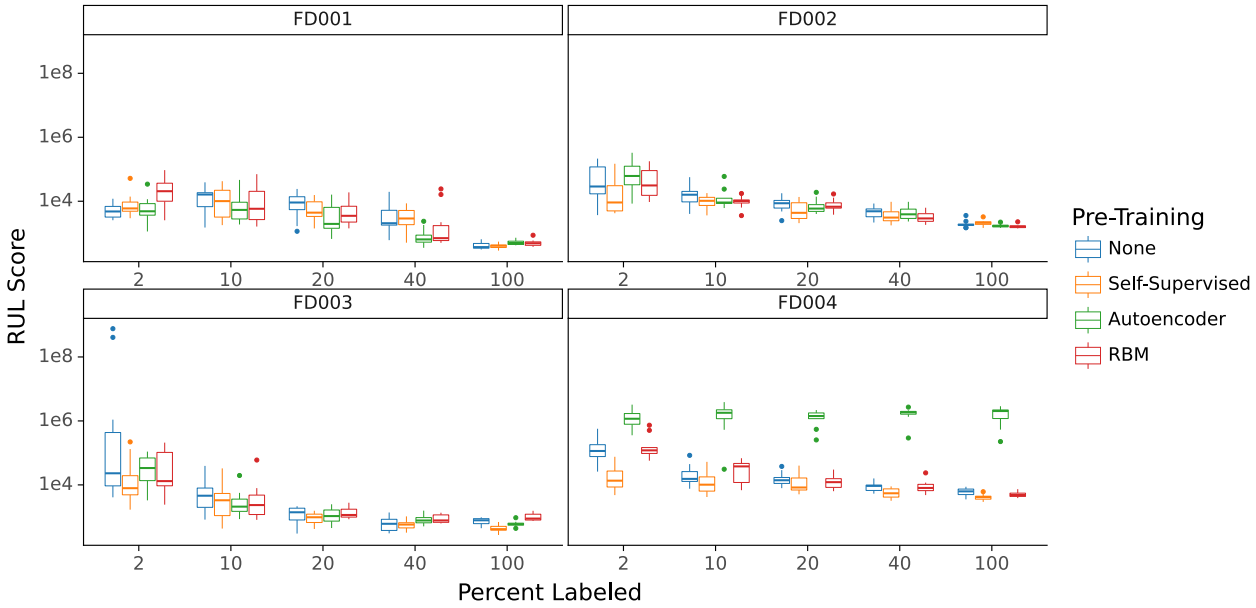


Figure 6. **RUL-Score at 80% degradation:** The plots show the results on the four subsets of the C-MAPSS dataset. We plot the RUL Score test performance on a logarithmic scale against the percentage of labeled data. The remaining data was used as unlabeled data with a grade of degradation of 80%. See figure 5 for an explanation of the box plots. The results are similar to the RMSE in figure 5 but show significantly worse scores for the AE pre-training on FD004.

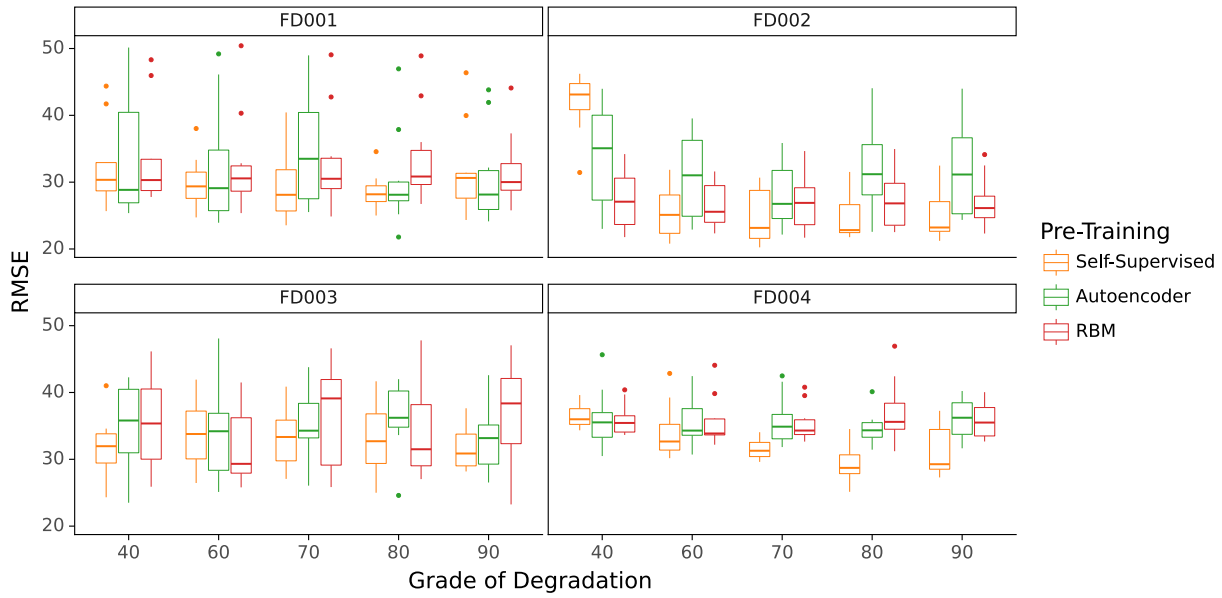


Figure 7. **RMSE at 2% of labeled data:** The plots show the results on the four subsets of the C-MAPSS dataset. We plot the RMSE test performance against the grade of degradation when only 2% of the labeled data was used. The median performance of the baseline is depicted in blue. See figure 5 for an explanation of the box plots. We can observe that the performance of the AE and the RBM is relatively constant on all grades of degradation. The self-supervised method's performance degrades for lower degradation, which can be seen best in FD004. Nevertheless, the self-supervised method shows much better performance on higher grades of degradation in FD002 and FD004 than the competing methods. On the remaining subsets, the performance of the self-supervised method is comparable to the other approaches.

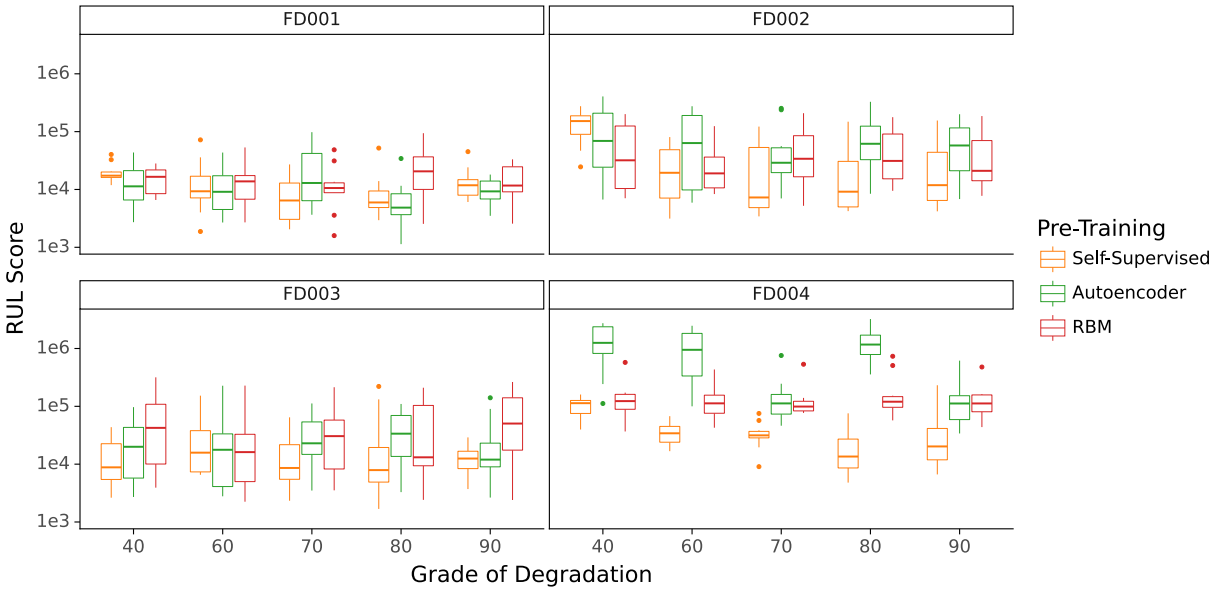


Figure 8. **RUL-Score at 2% of labeled data:** The plots show the results on the four subsets of the C-MAPSS dataset. We plot the RUL Score test performance on a logarithmic scale against the grade of degradation when only 2% of the labeled data was used. The median performance of the baseline is depicted in blue. See figure 5 for an explanation of the box plots. The results are similar to the RMSE in figure 5 but show significantly worse scores for the AE pre-training on FD004. We can also observe that the median score of the self-supervised approach is the lowest for all grades of degradation on FD002. FD003 and FD004.

labeled data but most apparent for 2%. When looking at figure 8, only self-supervised pre-training reliably beats the median performance of the baseline in terms of RUL-Score for grades of degradation above 40%.

Performance gains on **FD003** are better than on FD001 but still minor. For 2% labeled data, the self-supervised pre-training beats the baseline and the other methods for next to all grades of degradation in terms of mean RMSE and RUL-score. Nevertheless, the IQRs of our method and the RBM are often highly overlapping. As on FD001, we can see no trend concerning the grade of degradation.

On **FD004**, we can see the benefits of our approach most clearly. While AE and RBM pre-training bring next to no performance gains in terms of median RMSE, self-supervised pre-training outperforms the baseline reliably, as seen in figure 5 and 7. Nevertheless, we can observe a downward trend in performance with respect to the grade of degradation for our method. The competing approaches do not suffer from this. Figures 6 and 8 reveal devastating performance losses for AE pre-training in terms of RUL-Score.

We can conclude that SSL can be beneficial for RUL estimation even under realistic conditions with varying grades of degradation. However, no approach was able to reliably beat the baseline on all subsets under all data scenarios. A representative validation set and careful hyperparameter tuning are needed to assure improved performance and detect negative outcomes. Self-supervised pre-training seems to be a step in the right direction, as it often outperforms the competing approaches. Nevertheless, its performance trends downward on FD002 and FD004 with a falling grade of degradation.

5.2. Discussion of Findings

Our results revealed several points worthy of further discussion. First of all, there is the matter of minimal baseline performance drops when using as few as 40% of the labeled data. A possible explanation for this is the fact that the C-MAPSS dataset is the product of a simulation which may lead to little variation between the individual time series. If the variation between time series is small, the network needs less data to learn useful patterns from it. The differences between the subsets may be explained by the varying number of available time series per subset. Additional experiments where the absolute number of labeled data is varied instead of a percentage can be used to confirm this hypothesis.

The next point would be the discrepancies between RMSE and RUL-Score performance best seen on FD004 for the AE pre-training. Even though the RMSE performance is similar to the other approaches, the RUL-Score is much higher. This phenomenon can be explained through the asymmetric nature of the RUL-score, i.e. that late predictions incur a higher penalty than early ones. The AE pre-trained models seem to make predominantly late predictions, even though the absolute difference from the real RUL is similar to the other approaches.

Another interesting finding is, that RBM pre-training is com-

petitive with AE pre-training, even though it pre-trains only the first layer. Both methods use a reconstruction-based pre-training task, which makes the comparison even more interesting. An explaining factor could be that the bottleneck size of the autoencoder was not optimized as a hyperparameter, although it has a significant influence on the reconstruction capabilities of the AE. We did not optimize it because changing the bottleneck size would change the number of model parameters making the comparison with other methods difficult. A bigger bottleneck may increase performance because of the increased parameter count independently from pre-training. Additional experiments where the bottleneck size is optimized for the autoencoder and then compared to the other approaches on the same model architecture could prove this hypothesis.

The last point is the downward trend in performance for self-supervised pre-training with respect to the grade of degradation. At least on FD002 and FD004, we can observe that performance drops at lower grades of degradation and even makes our approach worse than the baseline in some cases. This problem may lie in the nature of the pre-text task we choose. It is based on the assumption that differences in the features of two time frames are correlated with the difference in RUL. If we take the piece-wise linear degradation model of C-MAPSS at face value, there are no differences in RUL above RUL_{max}. Consequently, there should not be a difference in the features either. Our pre-text task on the other hand is only accurate for a linear degradation model as we cannot take RUL_{max} into account on unlabeled data. Looking at the percentage of time frames above and below RUL_{max} for different grades of degradation, we can see that in FD004 only 11% of the training data has a label below a RUL_{max} of 125 at 40% grade of degradation. Coincidentally, the trend is most obvious on this subset. The training data of FD001, FD002, and FD003 has 33%, 23%, and 16% labels below 125 at 40% grade of degradation. Less data with labels below RUL_{max} makes the approximation of our pre-text task less accurate.

6. CONCLUSION & FUTURE WORK

We presented a study of three SSL approaches on the NASA C-MAPSS dataset under improved, more realistic evaluation conditions. These conditions take into account that realistic, unlabeled data does not contain features near the point of failure. Concerning our first research question, our results show that, contrary to previous studies, SSL does not always improve the accuracy of RUL estimation. This underlines the importance of a representative validation set to identify negative outcomes which may not always be available in settings with few labeled time series.

In answering our second research question, we have shown that our SSL approach, based on self-supervised pre-training, has superior performance compared to the competing approaches under certain conditions. More work is necessary to replicate these findings on other datasets. Nevertheless, our approach was not able to beat the baseline performance reliably under all data scenarios. Most notably, the performance dropped when the grade of degradation was low.

Future work includes conducting the experiments outlined in the discussion section to test the proposed explanations to the observed phenomena. Additionally, advanced techniques from the field of metric learning, e.g. hard sample mining or triplet loss, could be used to improve the self-supervised pre-training. Theoretically, our approach could be used for Unsupervised Domain Adaption (DA), too, as it shares many characteristics with SSL. Unsupervised DA is of high interest for RUL estimation as labeled and unlabeled data often do not share the same domain. Investigating the effectiveness of our approach for general, non-linear degradation processes, e.g. tool wear estimation, is another direction for future work.

ACKNOWLEDGEMENT

This work was funded by the Technische Universität Berlin with support of IAV GmbH.

REFERENCES

Bachman, P., Alsharif, O., & Precup, D. (2014). Learning with pseudo-ensembles. In *Proceedings of the 27th international conference on neural information processing systems-volume 2* (pp. 3365–3373).

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv*.

Baldi, P., & Chauvin, Y. (1993). Neural networks for fingerprint recognition. *neural computation*, *5*(3), 402–418.

Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., ... Shah, R. (1993). Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, *7*(04), 669–688.

Cheng, C., Zhou, B., Ma, G., Wu, D., & Yuan, Y. (2019). Wasserstein Distance based Deep Adversarial Transfer Learning for Intelligent Fault Diagnosis. *arXiv*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the north american chapter of the association for computational linguistics: Human language technologies* (Vol. 1, pp. 4171–4186).

Doersch, C., Gupta, A., & Efros, A. A. (2015, December). Unsupervised visual representation learning by context prediction. In *Proceedings of the ieee international conference on computer vision (iccv)*.

Franceschi, J.-Y., Dieuleveut, A., & Jaggi, M. (2019). Unsupervised scalable representation learning for multivariate time series. In *Advances in neural information*

processing systems (Vol. 32, pp. 4650–4661). Curran Associates, Inc.

Gebraeel, N., Lawley, M., Liu, R., & Parmeshwaran, V. (2004). Residual life predictions from vibration-based degradation signals: A neural network approach. *IEEE Transactions on Industrial Electronics*, *51*, 694–700. doi: 10.1109/TIE.2004.824875

Gidaris, S., Singh, P., & Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. In *International conference on learning representation.*

He, R., Dai, Y., Lu, J., & Mou, C. (2018). Developing ladder network for intelligent evaluation system: Case of remaining useful life prediction for centrifugal pumps. *Reliability Engineering & System Safety*, *180*, 385-393. doi: https://doi.org/10.1016/j.ress.2018.08.010

Heimes, F. O. (2008). Recurrent neural networks for remaining useful life estimation. In *International conference on prognostics and health management.*

Ioffe, S., & Szegedy, C. (2015, 07–09 Jul). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd international conference on machine learning* (Vol. 37, pp. 448–456). Lille, France: PMLR.

Jiang, J.-R., Lee, J.-E., & Zeng, Y.-M. (2020). Time Series Multiple Channel Convolutional Neural Network with Attention-Based Long Short-Term Memory for Predicting Bearing Remaining Useful Life. *Sensors*, *20*(166).

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

Kingma, D. P., Rezende, D. J., Mohamed, S., & Welling, M. (2014). Semi-supervised learning with deep generative models. In *Proceedings of the 27th international conference on neural information processing systems-volume 2* (pp. 3581–3589).

Krokotsch, T., Knaak, M., & Gühmann, C. (2020). A novel evaluation framework for unsupervised domain adaption on remaining useful lifetime estimation. In *2020 ieee international conference on prognostics and health management (icphm).* doi: 10.1109/ICPHM49022.2020.9187058

Li, X., Ding, Q., & Sun, J. Q. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering and System Safety*, *172*. doi: 10.1016/j.ress.2017.11.021

Listou Ellefsen, A., Bjørlykhaug, E., Æsøy, V., Ushakov, S., & Zhang, H. (2019). Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering and System Safety*, *183*, 240–251. doi: 10.1016/j.ress.2018.11.027

McInnes, L., Healy, J., Saul, N., & Grossberger, L. (2018). Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, *3*(29), 861.

Musgrave, K., Belongie, S., & Lim, S.-N. (2020). A metric learning reality check. In A. "Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), *"computer vision – eccv 2020"* (pp. "681–699"). "Springer International Publishing".

Oliver, A., Odena, A., Raffel, C. A., Cubuk, E. D., & Goodfellow, I. (2018). Realistic evaluation of deep semi-supervised learning algorithms. *Advances in Neural Information Processing Systems*, *31*, 3235–3246.

Rasmus, A., Valpola, H., Honkala, M., Berglund, M., & Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Proceedings of the 28th international conference on neural information processing systems-volume 2* (pp. 3546–3554).

Sateesh Babu, G., Zhao, P., & Li, X.-L. (2016). Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life. *Database Systems for Advanced Applications*, *9642*, 214–228. doi: 10.1007/978-3-319-32025-0

Saxena, A., & Goebel, K. (2008). Turbofan engine degradation simulation data set. *NASA Ames Prognostics Data Repository*.

Si, X. S., Wang, W., Hu, C. H., & Zhou, D. H. (2011). Remaining useful life estimation - A review on the statistical data driven approaches. *European Journal of Operational Research*, *213*. doi: 10.1016/j.ejor.2010.11.018

van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning*, *109*, 373–440. doi: 10.1007/s10994-019-05855-6

Wu, Y., Yuan, M., Dong, S., Lin, L., & Liu, Y. (2018). Remaining useful life estimation of engineered systems using vanilla LSTM neural networks. *Neurocomputing*, *275*, 167–179. doi: 10.1016/j.neucom.2017.05.063

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics* (pp. 189–196).

Yoon, A. S., Lee, T., Lim, Y., Jung, D., Kang, P., Kim, D., ... Choi, Y. (2017). Semi-supervised learning with deep generative models for asset failure prediction. *arXiv*.

Zheng, S., Ristovski, K., Farahat, A., & Gupta, C. (2017). Long Short-Term Memory Network for Remaining Useful Life estimation. In *Ieee international conference on prognostics and health management* (pp. 88–95). IEEE. doi: 10.1109/ICPHM.2017.7998311

Zhu, J., Chen, N., & Peng, W. (2019). Estimation of Bearing Remaining Useful Life Based on Multiscale Convolutional Neural Network. *IEEE Transactions on Industrial Electronics*, *66*. doi: 10.1109/TIE.2018.2844856

A. COMPLETE RESULT TABLES

		FD001				
		2%	10%	20%	40%	100%
	None	31.30 ± 6.16	25.45 ± 2.31	24.21 ± 1.31	21.94 ± 3.08	14.63 ± 0.81
40%	AE	33.37 ± 9.53	24.71 ± 1.28	$\mathbf{23.56 \pm 3.12}$	$\mathbf{17.27 \pm 1.47}$	15.03 ± 0.61
	RBM	33.49 ± 7.49	25.90 ± 1.93	25.09 ± 1.61	21.02 ± 3.17	15.67 ± 0.57
	Ours	32.35 ± 6.08	$\mathbf{24.42 \pm 1.58}$	24.91 ± 1.87	21.38 ± 2.32	$\mathbf{14.34 \pm 0.81}$
60%	AE	32.17 ± 8.94	$\mathbf{24.62 \pm 1.75}$	$\mathbf{22.65 \pm 2.46}$	$\mathbf{17.95 \pm 1.92}$	15.09 ± 0.52
	RBM	32.64 ± 7.45	27.06 ± 1.79	24.78 ± 2.73	19.50 ± 2.93	15.68 ± 1.01
	Ours	$\mathbf{29.88 \pm 3.83}$	25.81 ± 2.36	23.93 ± 1.72	22.41 ± 1.82	$\mathbf{14.48 \pm 0.87}$
70%	AE	34.93 ± 8.64	25.10 ± 2.23	$\mathbf{21.70 \pm 2.09}$	$\mathbf{17.71 \pm 1.79}$	15.02 ± 0.58
	RBM	33.00 ± 7.40	26.00 ± 1.77	23.95 ± 1.55	19.21 ± 1.92	15.45 ± 0.71
	Ours	$\mathbf{29.52 \pm 5.26}$	$\mathbf{24.90 \pm 1.47}$	22.96 ± 2.38	18.51 ± 2.76	15.35 ± 0.78
80%	AE	30.24 ± 7.16	25.32 ± 2.23	$\mathbf{22.41 \pm 3.07}$	$\mathbf{17.13 \pm 1.31}$	15.40 ± 0.62
	RBM	33.58 ± 7.05	25.53 ± 2.36	24.43 ± 1.66	19.56 ± 4.02	15.55 ± 0.88
	Ours	$\mathbf{28.50 \pm 2.68}$	$\mathbf{25.13 \pm 2.28}$	23.64 ± 1.52	20.98 ± 2.52	$\mathbf{14.33 \pm 0.74}$
90%	AE	$\mathbf{30.57 \pm 7.00}$	26.04 ± 1.92	$\mathbf{21.84 \pm 1.82}$	$\mathbf{17.26 \pm 1.34}$	14.89 ± 0.65
	RBM	31.86 ± 5.30	$\mathbf{25.05 \pm 2.17}$	25.01 ± 2.53	19.44 ± 2.36	15.55 ± 0.47
	Ours	31.69 ± 6.62	25.20 ± 1.92	23.52 ± 2.24	18.74 ± 3.53	15.39 ± 0.78

		FD002				
		2%	10%	20%	40%	100%
	None	29.80 ± 6.38	23.87 ± 1.30	22.83 ± 1.03	20.73 ± 0.94	17.55 ± 0.62
40%	AE	33.86 ± 7.88	23.12 ± 1.11	21.91 ± 0.96	20.48 ± 0.90	$\mathbf{17.19 \pm 0.30}$
	RBM	$\mathbf{27.43 \pm 4.30}$	$\mathbf{22.79 \pm 0.70}$	$\mathbf{20.98 \pm 0.65}$	$\mathbf{19.41 \pm 0.92}$	17.19 ± 0.51
	Ours	42.04 ± 4.47	24.04 ± 0.84	22.09 ± 1.31	20.44 ± 0.80	18.34 ± 0.84
60%	AE	30.95 ± 6.38	23.25 ± 0.92	22.26 ± 1.05	20.69 ± 0.91	17.78 ± 1.22
	RBM	26.50 ± 3.22	22.87 ± 0.90	$\mathbf{21.36 \pm 1.01}$	$\mathbf{19.42 \pm 0.68}$	$\mathbf{17.54 \pm 0.74}$
	Ours	$\mathbf{25.46 \pm 3.79}$	$\mathbf{22.86 \pm 1.17}$	22.14 ± 1.35	20.45 ± 0.84	18.45 ± 0.78
70%	AE	28.03 ± 4.85	23.84 ± 1.06	22.43 ± 1.83	20.49 ± 0.82	17.56 ± 1.22
	RBM	26.83 ± 3.91	22.85 ± 1.20	21.41 ± 1.19	19.33 ± 0.58	17.17 ± 0.46
	Ours	$\mathbf{24.82 \pm 4.00}$	$\mathbf{21.79 \pm 1.42}$	$\mathbf{20.25 \pm 0.77}$	$\mathbf{18.66 \pm 0.69}$	$\mathbf{16.83 \pm 0.44}$
80%	AE	32.36 ± 7.24	23.29 ± 0.90	21.87 ± 0.90	19.92 ± 0.90	17.21 ± 0.58
	RBM	27.13 ± 4.05	22.84 ± 0.86	21.35 ± 0.66	$\mathbf{19.13 \pm 0.87}$	$\mathbf{17.10 \pm 0.57}$
	Ours	$\mathbf{24.82 \pm 3.43}$	$\mathbf{22.43 \pm 1.11}$	$\mathbf{21.21 \pm 1.26}$	19.79 ± 0.65	18.23 ± 0.63
90%	AE	31.55 ± 6.75	23.70 ± 1.07	22.30 ± 1.00	20.71 ± 1.36	17.55 ± 0.53
	RBM	26.98 ± 3.73	22.55 ± 1.12	20.99 ± 1.07	19.46 ± 0.70	17.28 ± 0.39
	Ours	$\mathbf{24.99 \pm 3.80}$	$\mathbf{21.48 \pm 1.39}$	$\mathbf{20.26 \pm 1.23}$	$\mathbf{18.82 \pm 0.72}$	$\mathbf{16.80 \pm 0.44}$

Table 3. **RMSE results for FD001 and FD002**: We report the mean and standard deviation. The rows represent the *grade of degradation* and the columns the *percent of labeled data* of the data scenario. The second column contains the pre-training method, where *None* is the baseline without any pre-training, and *Ours* is the self-supervised pre-training. The bold results mark the best mean performance for each data scenario. If no result is bold, no approach was able to beat the baseline.

		FD003				
		2%	10%	20%	40%	100%
	None	42.33 ± 20.56	22.99 ± 4.02	16.55 ± 0.94	14.58 ± 1.06	13.88 ± 0.58
40%	AE	35.08 ± 6.38	$\mathbf{22.12 \pm 5.31}$	17.05 ± 1.63	14.75 ± 1.12	13.96 ± 0.59
	RBM	35.83 ± 7.00	22.98 ± 4.85	16.82 ± 1.13	15.19 ± 0.82	14.50 ± 0.50
	Ours	$\mathbf{31.82 \pm 4.51}$	22.98 ± 4.30	$\mathbf{16.00 \pm 0.86}$	$\mathbf{14.39 \pm 0.85}$	$\mathbf{13.47 \pm 0.18}$
60%	AE	34.00 ± 6.89	$\mathbf{21.26 \pm 4.86}$	16.77 ± 1.62	15.09 ± 1.14	14.16 ± 0.49
	RBM	$\mathbf{31.63 \pm 6.02}$	22.63 ± 3.93	17.04 ± 1.89	15.33 ± 0.84	14.45 ± 0.42
	Ours	33.79 ± 4.74	24.80 ± 4.77	17.10 ± 1.19	$\mathbf{14.52 \pm 1.10}$	$\mathbf{13.09 \pm 0.74}$
70%	AE	35.01 ± 5.99	$\mathbf{22.00 \pm 4.80}$	$\mathbf{16.36 \pm 1.15}$	15.39 ± 0.93	13.85 ± 0.42
	RBM	36.30 ± 7.52	22.36 ± 4.23	16.68 ± 1.35	15.51 ± 0.88	14.61 ± 0.45
	Ours	$\mathbf{33.10 \pm 4.34}$	22.35 ± 4.82	16.73 ± 1.45	15.14 ± 1.18	$\mathbf{13.64 \pm 0.62}$
80%	AE	36.35 ± 5.06	$\mathbf{21.24 \pm 3.75}$	$\mathbf{16.49 \pm 1.60}$	14.82 ± 0.76	14.03 ± 0.50
	RBM	34.29 ± 6.83	22.87 ± 4.63	17.31 ± 0.86	15.52 ± 0.66	14.68 ± 0.56
	Ours	$\mathbf{33.47 \pm 5.57}$	22.98 ± 5.18	16.54 ± 2.04	14.60 ± 0.91	$\mathbf{13.05 \pm 0.53}$
90%	AE	33.37 ± 5.36	$\mathbf{20.14 \pm 4.09}$	17.10 ± 2.03	14.84 ± 0.88	14.17 ± 0.35
	RBM	36.48 ± 7.74	23.47 ± 4.27	17.26 ± 1.35	15.06 ± 0.92	14.34 ± 0.57
	Ours	$\mathbf{31.79 \pm 3.53}$	21.19 ± 4.13	$\mathbf{16.34 \pm 1.19}$	14.78 ± 0.75	$\mathbf{13.83 \pm 0.49}$
		FD004				
		2%	10%	20%	40%	100%
	None	35.44 ± 2.22	27.00 ± 1.37	25.18 ± 1.13	23.51 ± 0.62	21.66 ± 0.90
40%	AE	36.07 ± 4.39	28.53 ± 2.26	25.39 ± 1.36	$\mathbf{23.20 \pm 1.06}$	21.62 ± 0.71
	RBM	35.99 ± 2.38	28.46 ± 2.20	25.74 ± 0.90	23.55 ± 0.62	21.48 ± 0.54
	Ours	36.41 ± 1.71	31.15 ± 3.66	25.44 ± 1.67	23.24 ± 1.84	$\mathbf{21.11 \pm 0.52}$
60%	AE	35.60 ± 3.68	28.34 ± 2.36	$\mathbf{24.88 \pm 0.97}$	23.42 ± 0.97	21.30 ± 0.37
	RBM	35.59 ± 3.69	27.61 ± 2.44	25.48 ± 1.22	23.22 ± 0.48	21.18 ± 0.62
	Ours	$\mathbf{34.20 \pm 4.08}$	29.40 ± 2.77	25.31 ± 2.28	$\mathbf{22.24 \pm 1.21}$	$\mathbf{20.96 \pm 0.67}$
70%	AE	35.72 ± 3.72	28.28 ± 1.82	25.34 ± 1.30	23.19 ± 0.71	21.26 ± 0.56
	RBM	35.41 ± 2.69	27.32 ± 2.30	25.40 ± 1.39	23.02 ± 0.73	21.08 ± 0.52
	Ours	$\mathbf{31.58 \pm 1.57}$	27.02 ± 2.07	$\mathbf{23.88 \pm 2.14}$	$\mathbf{22.48 \pm 1.51}$	$\mathbf{20.95 \pm 0.67}$
80%	AE	34.68 ± 2.34	28.16 ± 1.73	25.47 ± 1.79	22.74 ± 1.08	21.23 ± 0.62
	RBM	37.15 ± 4.56	27.95 ± 2.36	25.20 ± 1.08	23.43 ± 1.25	21.26 ± 0.89
	Ours	$\mathbf{29.23 \pm 2.74}$	$\mathbf{25.84 \pm 2.84}$	$\mathbf{24.26 \pm 2.11}$	$\mathbf{21.46 \pm 0.68}$	$\mathbf{20.95 \pm 0.67}$
90%	AE	35.95 ± 2.96	27.57 ± 1.28	25.66 ± 1.14	23.43 ± 1.20	21.73 ± 0.59
	RBM	35.88 ± 2.70	27.84 ± 1.82	25.69 ± 1.31	23.73 ± 0.78	21.25 ± 0.44
	Ours	$\mathbf{31.07 \pm 3.81}$	$\mathbf{24.20 \pm 2.14}$	$\mathbf{24.17 \pm 2.88}$	$\mathbf{21.82 \pm 1.22}$	$\mathbf{20.90 \pm 0.62}$

Table 4. **RMSE results for FD003 and FD004**: Please consult table 3 for further information.

		FD001				
		2%	10%	20%	40%	100%
	None	$5.64e3 \pm 3.25e3$	$1.49e4 \pm 1.11e4$	$1.04e4 \pm 7.48e3$	$5.28e3 \pm 6.35e3$	$4.24e2 \pm 1.15e2$
	AE	$1.51e4 \pm 1.28e4$	$\mathbf{4.57e3 \pm 2.37e3}$	$\mathbf{7.39e3 \pm 7.22e3}$	$\mathbf{1.03e3 \pm 7.22e2}$	$4.79e2 \pm 1.03e2$
40%	RBM	$1.62e4 \pm 8.24e3$	$1.73e4 \pm 1.83e4$	$9.79e3 \pm 9.05e3$	$3.69e3 \pm 4.48e3$	$5.32e2 \pm 6.25e1$
	Ours	$2.04e4 \pm 9.05e3$	$1.14e4 \pm 8.40e3$	$1.57e4 \pm 1.32e4$	$3.90e3 \pm 2.89e3$	$\mathbf{3.97e2 \pm 8.31e1}$
	AE	$1.38e4 \pm 1.28e4$	$\mathbf{9.80e3 \pm 1.20e4}$	$\mathbf{4.76e3 \pm 4.94e3}$	$\mathbf{9.85e2 \pm 5.46e2}$	$4.79e2 \pm 6.05e1$
60%	RBM	$1.76e4 \pm 1.56e4$	$1.89e4 \pm 2.34e4$	$1.55e4 \pm 1.58e4$	$2.95e3 \pm 4.13e3$	$5.14e2 \pm 9.48e1$
	Ours	$1.80e4 \pm 2.13e4$	$1.08e4 \pm 5.36e3$	$8.08e3 \pm 5.60e3$	$6.04e3 \pm 5.29e3$	$\mathbf{4.04e2 \pm 8.41e1}$
	AE	$3.16e4 \pm 3.61e4$	$1.42e4 \pm 1.67e4$	$\mathbf{2.71e3 \pm 2.45e3}$	$\mathbf{8.92e2 \pm 4.70e2}$	$4.76e2 \pm 7.56e1$
70%	RBM	$1.49e4 \pm 1.42e4$	$1.37e4 \pm 1.51e4$	$5.71e3 \pm 5.43e3$	$1.06e3 \pm 5.36e2$	$4.91e2 \pm 1.05e2$
	Ours	$9.03e3 \pm 8.16e3$	$\mathbf{9.96e3 \pm 8.28e3}$	$7.18e3 \pm 8.90e3$	$1.45e3 \pm 1.53e3$	$4.86e2 \pm 9.98e1$
	AE	$8.31e3 \pm 9.62e3$	$\mathbf{1.32e4 \pm 1.20e4}$	$\mathbf{4.74e3 \pm 5.51e3}$	$\mathbf{9.11e2 \pm 6.53e2}$	$5.22e2 \pm 9.65e1$
80%	RBM	$3.00e4 \pm 2.83e4$	$1.55e4 \pm 2.14e4$	$6.26e3 \pm 6.15e3$	$4.72e3 \pm 8.42e3$	$5.13e2 \pm 1.41e2$
	Ours	$1.13e4 \pm 1.47e4$	$1.54e4 \pm 1.46e4$	$6.48e3 \pm 4.50e3$	$3.62e3 \pm 2.74e3$	$\mathbf{3.98e2 \pm 7.89e1}$
	AE	$1.01e4 \pm 5.01e3$	$1.25e4 \pm 1.48e4$	$\mathbf{2.99e3 \pm 2.53e3}$	$\mathbf{9.04e2 \pm 3.92e2}$	$5.06e2 \pm 1.74e2$
90%	RBM	$1.62e4 \pm 1.09e4$	$1.37e4 \pm 1.05e4$	$1.22e4 \pm 1.37e4$	$1.27e3 \pm 7.67e2$	$4.95e2 \pm 6.30e1$
	Ours	$1.50e4 \pm 1.18e4$	$\mathbf{1.06e4 \pm 1.03e4}$	$7.26e3 \pm 8.44e3$	$3.50e3 \pm 6.38e3$	$4.89e2 \pm 9.89e1$
		FD002				
		2%	10%	20%	40%	100%
	None	$6.95e4 \pm 7.74e4$	$2.02e4 \pm 1.70e4$	$9.21e3 \pm 4.80e3$	$4.79e3 \pm 1.95e3$	$1.99e3 \pm 6.11e2$
	AE	$1.25e5 \pm 1.32e5$	$\mathbf{1.21e4 \pm 5.90e3}$	$8.23e3 \pm 4.71e3$	$5.80e3 \pm 2.51e3$	$1.86e3 \pm 5.80e2$
40%	RBM	$7.16e4 \pm 7.74e4$	$1.35e4 \pm 8.68e3$	$\mathbf{6.08e3 \pm 2.18e3}$	$4.33e3 \pm 2.30e3$	$\mathbf{1.69e3 \pm 2.60e2}$
	Ours	$1.43e5 \pm 8.00e4$	$1.45e4 \pm 1.00e4$	$8.20e3 \pm 3.92e3$	$\mathbf{4.12e3 \pm 2.29e3}$	$2.71e3 \pm 1.74e3$
	AE	$1.01e5 \pm 1.03e5$	$\mathbf{1.20e4 \pm 5.64e3}$	$\mathbf{6.71e3 \pm 2.70e3}$	$5.46e3 \pm 2.16e3$	$2.08e3 \pm 1.13e3$
60%	RBM	$3.92e4 \pm 4.56e4$	$1.29e4 \pm 7.54e3$	$8.26e3 \pm 4.54e3$	$\mathbf{3.10e3 \pm 8.06e2}$	$\mathbf{1.82e3 \pm 3.69e2}$
	Ours	$\mathbf{2.99e4 \pm 2.72e4}$	$1.25e4 \pm 1.04e4$	$7.69e3 \pm 6.09e3$	$3.71e3 \pm 9.67e2$	$2.78e3 \pm 1.72e3$
	AE	$7.06e4 \pm 9.35e4$	$1.92e4 \pm 1.71e4$	$1.02e4 \pm 9.86e3$	$5.08e3 \pm 1.72e3$	$2.59e3 \pm 1.93e3$
70%	RBM	$6.14e4 \pm 6.51e4$	$\mathbf{1.30e4 \pm 4.50e3}$	$7.90e3 \pm 5.90e3$	$3.57e3 \pm 1.23e3$	$1.76e3 \pm 2.73e2$
	Ours	$\mathbf{3.33e4 \pm 4.39e4}$	$1.78e4 \pm 2.29e4$	$\mathbf{4.74e3 \pm 4.02e3}$	$\mathbf{2.76e3 \pm 1.38e3}$	$\mathbf{1.67e3 \pm 1.97e2}$
	AE	$9.93e4 \pm 1.01e5$	$1.56e4 \pm 1.63e4$	$7.76e3 \pm 4.89e3$	$4.58e3 \pm 2.32e3$	$1.74e3 \pm 2.47e2$
80%	RBM	$6.16e4 \pm 6.11e4$	$\mathbf{1.02e4 \pm 3.96e3}$	$8.27e3 \pm 3.91e3$	$\mathbf{3.32e3 \pm 1.34e3}$	$\mathbf{1.65e3 \pm 2.47e2}$
	Ours	$\mathbf{3.55e4 \pm 5.25e4}$	$1.04e4 \pm 4.79e3$	$\mathbf{6.25e3 \pm 4.44e3}$	$3.88e3 \pm 2.41e3$	$2.13e3 \pm 4.91e2$
	AE	$7.85e4 \pm 7.03e4$	$1.48e4 \pm 6.75e3$	$8.13e3 \pm 3.31e3$	$5.43e3 \pm 3.63e3$	$1.88e3 \pm 3.15e2$
90%	RBM	$4.95e4 \pm 5.63e4$	$1.03e4 \pm 6.69e3$	$7.04e3 \pm 5.64e3$	$3.44e3 \pm 1.30e3$	$1.68e3 \pm 2.25e2$
	Ours	$\mathbf{3.75e4 \pm 5.15e4}$	$\mathbf{8.11e3 \pm 4.40e3}$	$\mathbf{3.91e3 \pm 2.10e3}$	$\mathbf{2.41e3 \pm 6.83e2}$	$\mathbf{1.66e3 \pm 2.01e2}$

Table 5. **RUL-Score results for FD001 and FD002**: We report the mean and standard deviation. The rows represent the *grade of degradation* and the columns the *percent of labeled data* of the data scenario. The second column contains the pre-training method, where *None* is the baseline without any pre-training, and *Ours* is the self-supervised pre-training. The bold results mark the best mean performance for each data scenario. If no result is bold, no approach was able to beat the baseline. Please note that the standard deviation of these results can be misleading as RUL-Score is an exponential measure.

		FD003				
		2%	10%	20%	40%	100%
	None	$1.18e8 \pm 2.62e8$	$8.96e3 \pm 1.20e4$	$1.31e3 \pm 6.83e2$	$7.03e2 \pm 3.82e2$	$7.48e2 \pm 1.71e2$
	AE	$2.93e4 \pm 3.11e4$	$2.02e4 \pm 5.66e4$	$1.40e3 \pm 9.24e2$	$6.94e2 \pm 3.24e2$	$7.13e2 \pm 2.08e2$
40%	RBM	$9.06e4 \pm 1.13e5$	$\mathbf{4.18e3 \pm 3.15e3}$	$3.43e3 \pm 7.44e3$	$9.74e2 \pm 5.84e2$	$9.88e2 \pm 3.10e2$
	Ours	$\mathbf{1.51e4 \pm 1.31e4}$	$7.16e3 \pm 9.81e3$	$\mathbf{1.02e3 \pm 5.83e2}$	$\mathbf{6.20e2 \pm 2.76e2}$	$\mathbf{4.95e2 \pm 1.12e2}$
	AE	$4.03e4 \pm 6.81e4$	$\mathbf{7.16e3 \pm 1.35e4}$	$1.39e3 \pm 7.65e2$	$9.76e2 \pm 3.91e2$	$7.10e2 \pm 1.44e2$
60%	RBM	$4.07e4 \pm 6.87e4$	$9.77e3 \pm 2.35e4$	$2.64e3 \pm 4.86e3$	$1.03e3 \pm 6.20e2$	$9.02e2 \pm 3.54e2$
	Ours	$\mathbf{3.35e4 \pm 4.47e4}$	$7.24e3 \pm 7.05e3$	$1.71e3 \pm 1.55e3$	$\mathbf{6.13e2 \pm 2.42e2}$	$\mathbf{4.74e2 \pm 1.71e2}$
	AE	$3.91e4 \pm 3.60e4$	$7.12e3 \pm 1.50e4$	$\mathbf{1.05e3 \pm 4.12e2}$	$1.04e3 \pm 6.46e2$	$6.78e2 \pm 1.75e2$
70%	RBM	$6.09e4 \pm 7.69e4$	$\mathbf{3.49e3 \pm 3.45e3}$	$1.42e3 \pm 6.43e2$	$1.01e3 \pm 3.75e2$	$9.28e2 \pm 2.79e2$
	Ours	$\mathbf{1.65e4 \pm 1.88e4}$	$1.20e4 \pm 3.14e4$	$1.16e3 \pm 5.47e2$	$7.17e2 \pm 2.83e2$	$\mathbf{5.25e2 \pm 1.27e2}$
	AE	$4.36e4 \pm 3.58e4$	$\mathbf{4.11e3 \pm 5.66e3}$	$1.21e3 \pm 6.58e2$	$8.46e2 \pm 3.09e2$	$6.14e2 \pm 1.39e2$
80%	RBM	$5.96e4 \pm 7.59e4$	$8.60e3 \pm 1.82e4$	$1.47e3 \pm 7.13e2$	$9.10e2 \pm 2.92e2$	$1.01e3 \pm 2.76e2$
	Ours	$\mathbf{4.20e4 \pm 7.41e4}$	$6.77e3 \pm 9.98e3$	$\mathbf{9.61e2 \pm 3.88e2}$	$\mathbf{5.96e2 \pm 2.08e2}$	$\mathbf{4.59e2 \pm 1.39e2}$
	AE	$3.22e4 \pm 4.59e4$	$\mathbf{3.40e3 \pm 5.39e3}$	$1.73e3 \pm 1.23e3$	$8.26e2 \pm 4.42e2$	$6.66e2 \pm 8.99e1$
90%	RBM	$8.56e4 \pm 8.92e4$	$5.24e3 \pm 6.90e3$	$1.97e3 \pm 1.67e3$	$9.11e2 \pm 5.46e2$	$9.92e2 \pm 4.93e2$
	Ours	$\mathbf{1.31e4 \pm 7.50e3}$	$3.44e3 \pm 4.80e3$	$\mathbf{1.14e3 \pm 5.58e2}$	$\mathbf{6.69e2 \pm 2.35e2}$	$\mathbf{5.55e2 \pm 9.86e1}$
		FD004				
		2%	10%	20%	40%	100%
	None	$1.63e5 \pm 1.58e5$	$2.52e4 \pm 2.33e4$	$1.69e4 \pm 9.37e3$	$9.15e3 \pm 3.29e3$	$6.21e3 \pm 1.62e3$
	AE	$1.44e6 \pm 9.82e5$	$1.56e6 \pm 8.75e5$	$2.08e6 \pm 6.41e5$	$2.27e6 \pm 1.34e6$	$1.77e6 \pm 9.15e5$
40%	RBM	$1.60e5 \pm 1.52e5$	$3.90e4 \pm 3.43e4$	$1.69e4 \pm 8.72e3$	$\mathbf{7.79e3 \pm 2.62e3}$	$6.03e3 \pm 2.25e3$
	Ours	$\mathbf{1.02e5 \pm 4.04e4}$	$5.25e4 \pm 7.18e4$	$\mathbf{1.65e4 \pm 1.28e4}$	$1.12e4 \pm 7.05e3$	$\mathbf{5.26e3 \pm 1.71e3}$
	AE	$1.10e6 \pm 8.70e5$	$1.82e6 \pm 8.91e5$	$1.71e6 \pm 1.29e6$	$1.94e6 \pm 1.00e6$	$1.42e6 \pm 6.38e5$
60%	RBM	$1.41e5 \pm 1.12e5$	$3.66e4 \pm 3.26e4$	$\mathbf{1.29e4 \pm 5.23e3}$	$8.53e3 \pm 5.18e3$	$5.76e3 \pm 1.30e3$
	Ours	$\mathbf{3.69e4 \pm 1.60e4}$	$3.04e4 \pm 1.76e4$	$1.66e4 \pm 1.35e4$	$\mathbf{6.67e3 \pm 3.77e3}$	$\mathbf{4.29e3 \pm 9.92e2}$
	AE	$1.80e5 \pm 2.11e5$	$3.86e4 \pm 3.61e4$	$1.64e4 \pm 7.75e3$	$1.08e4 \pm 8.82e3$	$5.73e3 \pm 1.43e3$
70%	RBM	$1.43e5 \pm 1.39e5$	$3.68e4 \pm 5.13e4$	$1.53e4 \pm 9.11e3$	$9.53e3 \pm 4.82e3$	$5.38e3 \pm 1.94e3$
	Ours	$\mathbf{3.52e4 \pm 1.87e4}$	$\mathbf{1.54e4 \pm 6.76e3}$	$\mathbf{8.43e3 \pm 4.92e3}$	$\mathbf{8.52e3 \pm 5.06e3}$	$\mathbf{4.25e3 \pm 1.01e3}$
	AE	$1.46e6 \pm 1.01e6$	$1.76e6 \pm 1.09e6$	$1.37e6 \pm 6.03e5$	$1.75e6 \pm 6.24e5$	$1.71e6 \pm 8.83e5$
80%	RBM	$2.11e5 \pm 2.24e5$	$3.41e4 \pm 2.14e4$	$1.42e4 \pm 8.29e3$	$9.48e3 \pm 5.50e3$	$5.13e3 \pm 1.09e3$
	Ours	$\mathbf{2.18e4 \pm 2.11e4}$	$\mathbf{1.61e4 \pm 1.51e4}$	$\mathbf{1.36e4 \pm 1.10e4}$	$\mathbf{5.87e3 \pm 2.13e3}$	$\mathbf{4.25e3 \pm 1.01e3}$
	AE	$1.58e5 \pm 1.73e5$	$3.28e4 \pm 2.23e4$	$1.59e4 \pm 1.02e4$	$1.04e4 \pm 5.96e3$	$5.74e3 \pm 4.26e2$
90%	RBM	$1.46e5 \pm 1.24e5$	$2.51e4 \pm 2.21e4$	$1.61e4 \pm 1.16e4$	$1.09e4 \pm 5.88e3$	$5.60e3 \pm 1.27e3$
	Ours	$\mathbf{5.05e4 \pm 7.02e4}$	$\mathbf{1.14e4 \pm 7.05e3}$	$\mathbf{1.17e4 \pm 1.27e4}$	$\mathbf{5.52e3 \pm 2.95e3}$	$\mathbf{4.04e3 \pm 8.03e2}$

Table 6. **RUL-Score results for FD003 and FD004**: Please consult table 5 for further information.