Self-Supervised Learning for Data Scarcity in a Fatigue Damage Prognostic Problem

Anass Akrim, Christian Gogu, Rob A. Vingerhoeds, Michel Salaün

Self-Supervised Learning for data scarcity in a fatigue damage prognostic problem

Anass Akrim [a,b,*], Christian Gogu [a,b], Rob Vingerhoeds [b], Michel Salaün [a,b]

[a] Institut Clément Ader (UMR CNRS 5312) INSA/UPS/ISAE/Mines Albi, Université de Toulouse, 3 rue Caroline Aigle, 31400 Toulouse, France
[b] ISAE-SUPAERO, Université de Toulouse, 10 Avenue Edouard Belin, 31400 Toulouse, France

Dataset link: https://github.com/ansak95/DeepSSL

Keywords:
Prognostics and Health Management (PHM)
Remaining Useful Life (RUL)
Deep Learning (DL)
Data scarcity
Self-Supervised Learning (SSL)

A B S T R A C T

With the increasing availability of data for Prognostics and Health Management (PHM), Deep Learning (DL) techniques are now the subject of considerable attention for this application, often achieving more accurate Remaining Useful Life (RUL) predictions. However, one of the major challenges for DL techniques resides in the difficulty of obtaining large amounts of labelled data on industrial systems. To overcome this lack of labelled data, an emerging learning technique is considered in our work: Self-Supervised Learning, a sub-category of unsupervised learning approaches. This paper aims to investigate whether pre-training DL models in a self-supervised way on unlabelled sensors data can be useful for RUL estimation with only Few-Shots Learning, *i.e.* with scarce labelled data. In this research, a fatigue damage prognostics problem is addressed, through the estimation of the RUL of aluminium alloy panels (typical of aerospace structures) subject to fatigue cracks from strain gauge data. Synthetic datasets composed of strain data are used allowing to extensively investigate the influence of the dataset size on the predictive performance. Results show that the self-supervised pre-trained models are able to significantly outperform the non-pre-trained models in downstream RUL prediction task, and with less computational expense, showing promising results in prognostic tasks when only limited labelled data is available.

1. Introduction

Prognostics and Health Management (PHM) is a research domain addressing failure mechanisms of real systems in order to better manage the use of information on equipment operating conditions (Shao-feng et al., 2013). Its implementation can improve the efficiency of maintenance support (Mao et al., 2010), optimize the maintenance plan and therewith equipment availability (Atamuradov et al., 2017), help industry to balance safety and economic profit (Wen and Liu, 2011). For many mechanical structures and notably aerospace structures, fatigue damage is one of the major modes of failure. Therefore, fatigue monitoring and prediction of fatigue life in structures, *i.e.* Remaining Useful Life (RUL) estimation, represents one of the major challenges to be solved for paving the way towards predictive structural maintenance.

Among the approaches used for PHM, Data-Driven models have gained more and more attention in the PHM community, especially the latest Deep Learning (DL) techniques (Tsui et al., 2015), redefining state-of-the-art performances in a wide range of areas in recent years (LeCun et al., 2015). However, their effectiveness depends on the quantity and quality of available labelled data. Currently, data scarcity represents a scientific bottleneck in many engineering fields (*e.g.* in healthcare Jadon, 2021, in energy Berthou et al., 2019, water and environmental engineering Borzooei et al., 2019; Gutierrez-Torre et al., 2020, etc.), which makes it difficult to apply the latest Machine Learning (ML) methods. Many approaches have been proposed to address data scarcity in these various domains, as recently reviewed by Nandy et al. (2022), Gorgoglione et al. (2020) and Bansal et al. (2022). As faults are rare and structures can be replaced before reaching failure, data scarcity is becoming one of the most important challenges in PHM (Fink et al., 2020; Theissler et al., 2021). Nevertheless, while labelled data is lacking, the availability of raw sensors data is increasing due to the advancements in sensing technologies. This data is considered as "unlabelled" in the context of prognostics as, for sensor data at a given point in time, the true RUL is unknown and cannot be determined unless the sensor measurements are available all the way to failure. In most engineering applications, this is unattainable, since the parts will be replaced before failure, and this is particularly true for aerospace mechanical structures, thus the majority of sensor data is unlabelled, meaning that no associated RUL is available for it. Exploiting such unlabelled sensors data during training has become a major goal in ML in order to improve learning performance. Therefore, the research question addressed in this paper can be stated as follows: *is it possible to learn meaningful representations from unlabelled data and use it to enhance related supervised predictive tasks on a fatigue damage prognostics problem?*

* Corresponding author currently at: Square Research Center, Square Management Group, 173 Avenue Achille Peretti, 92200 Neuilly-sur-Seine, France
E-mail address: anass.akrim@gmail.com (A. Akrim).

https://doi.org/10.1016/j.engappai.2023.105837

Nomenclature

Abbreviations

AE	Autoencoder
AR	Autoregressive
DGN	Deep Gated Recurrent Unit Network
DL	Deep Learning
GRU	Gated Recurrent Unit
$LSTM$	Long Short-Term Memory
$MAPE$	Mean Absolute Percentage Error
ML	Machine Learning
MSE	Mean Squared Error
$MSPA$	Multi-Steps Prediction Autoregressive
PHM	Prognostics and Health Management
RNN	Recurrent Neural Networks
RUL	Remaining Useful Life
SSL	Self-Supervised Learning

Notations

D_L	Labelled Dataset
D_U	Unlabelled Dataset
T_f	Time of failure
X^L/y^L	Labelled input signal/Corresponding RUL label
X^U	Unlabelled input signal

Variables

d	Ratio of the total lifetime of a sequence
h	Length of the sliding window
n_g	Number of sensor time series
N_L	Number of labelled structures
n_L	Number of labelled samples
N_L^{Test}	Number of labelled structures for testing
N_L^{Train}	Number of labelled structures for training
N_U	Number of unlabelled structures
n_U	Number of unlabelled samples
N_U^{Train}	Number of unlabelled structures for training
n_U^{Train}	Number of unlabelled samples for training

In the Artificial Intelligence (AI) community, a recent learning technique to extract knowledge from unlabelled data was proposed to address the challenge of data scarcity: Self-Supervised Learning (SSL) (Jaiswal et al., 2021), a sub-category of unsupervised learning approaches. SSL has already shown tremendous performances in many AI fields such as in Natural Language Processing (*e.g.* GPT-3 Brown et al., 2020) or Image Processing (Chen et al., 2020a). Nevertheless, the applicability of this approach remains largely unexplored in the engineering fields, a domain in which data scarcity is an increasingly challenging issue (Borzooei et al., 2019; Zhu et al., 2021; Rocchetta et al., 2022). Currently, there is only a limited amount of existing research that focuses on the potential of Self-Supervised Learning for Prognostics (Krokotsch et al., 2022; Guo et al., 2022), and particularly for fatigue damage prognostics problems.

In order to address this limitation, this paper aims to investigate whether pre-training DL models in a self-supervised way on unlabelled sensors data on a fatigue damage prognostics problem can be useful for RUL estimation with only Few-Shots Learning, *i.e.* with scarce labelled data. The interest is in estimating the RUL of aluminium alloy panels (typical of aerospace structures) subject to fatigue cracks from strain gauge data. A synthetically generated dataset is used for this purpose, composed of a large unlabelled dataset (i.e. strain gauges data of structures before failure) for pre-training, and a smaller labelled dataset (i.e. strain gauges data of structures until failure) for fine-tuning on the RUL prediction task. The synthetic dataset is based on a framework previously developed by the authors (Akrim et al., 2022).

The remainder of the paper is structured as follows. Section 2 provides a background on the state-of-the-art DL techniques in prognostics for PHM. In Section 3, the proposed methodology is presented. Section 4 presents the experimental settings used in this study for pre-training and fine-tuning phases, and the results obtained with the deep learning-based approaches trained in a self-supervised manner are analysed. The impact of the size of the data available for pre-training as well as the choice of the pre-text task will be investigated, and different DL models are compared for the self supervised learning task. Section 5 summarizes the aspects of the approach considered in this paper and identifies potential future work. Finally, Section 6 concludes the current research paper and provides future outlooks. Fig. 1 illustrates the outline of this paper and summarizes the objective of each section.

2. Background

In this section, the application of DL techniques in the field of prognostics for PHM and related work on Self-Supervised Learning are presented.

2.1. Deep learning in prognostics for PHM

As more data becomes available in the engineering domain, there is a recent surge of interest in using Deep Learning in Prognostics and Health Management (Jimenez et al., 2020; Voulodimos et al., 2018). In prognostics applications, Time Series Forecasting models are most commonly used to predict the RUL of systems or structures, given the format of acquired data in PHM (*e.g.* data collected from sensors, vibration signals, etc.), and the most commonly used algorithms for these tasks are Recurrent Neural Networks (RNN) (Hewamalage et al., 2021). Given the sequential nature of the sensor data in the prognostics field (*e.g.* sensors data), good results have been obtained within the PHM community by using RNNs, such as Standard RNNs, Long Short-Term Memory (LSTM) networks, and Gated Recurrent Unit networks (GRU) (Fink et al., 2020; Rana, 2016; Baptista et al., 2020).

The lack of available labelled data is becoming a major challenge in the application of machine learning to PHM, which, as a field, suffers from a high data acquisition cost compared to other domains in which machine learning has proven useful (*e.g.* Natural Language Processing). In Prognostics tasks, a label can constitute the RUL at each time step of measurements, which is generally difficult to acquire and often can be a time-consuming and expensive investment for experts. However, due to advancements in sensing technologies in engineering fields, the availability of unlabelled data is increasing (*e.g.* raw sensors data of structures replaced before reaching failure). Since failure is not reached when the structures are replaced, the RUL at replacement and at each previous timestep is not known. Thus the sensor data is unlabelled according to the previously introduced definition of a label in the PHM context. Exploiting unlabelled data during training has therefore become a major goal in order to improve learning performance.

2.2. Self-Supervised Learning

Similarly to *self-taught learning* as presented in Raina et al. (2007), Self-Supervised Learning (SSL) consists in learning meaningful and general representations from unlabelled data (during a *pre-training phase*) by solving a so-called *pretext task* without requiring the data to be labelled. These representations are then applicable to a wide range of related supervised tasks (*i.e. downstream task*) with only few labelled data (*i.e.* "Few-Shots Learning"). SSL aims to improve predictive

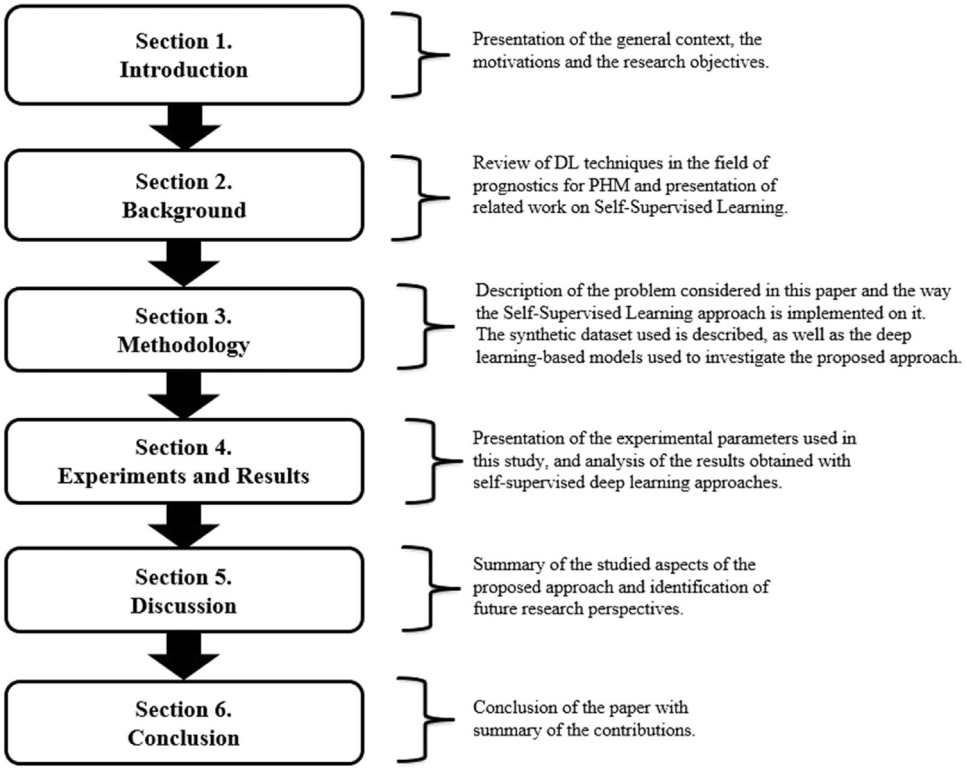

Fig. 1. Outline of this paper.

performance on the downstream task through the use of unlabelled data, thus avoiding the extensive cost of collecting and annotating large-scale datasets (Jing and Tian, 2020). This learning paradigm has already proven that it can significantly improve the performance of downstream tasks for many AI applications such as in Natural Language Processing (*e.g.* GPT-3 Brown et al., 2020) or Image Processing (Chen et al., 2020a,b). GPT-3 (Brown et al., 2020) was one of the largest self-supervised learning systems released by OpenAI in 2019.

There are few recent developments that have shown the potential of the SSL paradigm in engineering fields (Yengera et al., 2018; Endo et al., 2022; Yu et al., 2022; Shurrab and Duwairi, 2022), and several PHM researchers considered this approach to address the data scarcity in fault diagnostics problems, showing promising results (Hahn and Mechefske, 2021; Ding et al., 2022). However, to date, there is, to the best of the authors' knowledge, only a limited amount of existing research that focuses on the application of SSL to prognostic problems in PHM, for example for RUL estimation on the NASA C-MAPSS[1] dataset (Yoon et al., 2017; Ellefsen et al., 2019; Krokotsch et al., 2022; Guo et al., 2022). One of the first to explore this learning paradigm in prognostics in order to deal with the problem of lack of labelled data is Yoon et al. (2017), using a pre-trained variational autoencoder (VAE Kingma and Welling, 2013) that makes use of available unlabelled data to learn a latent space representation in an unsupervised manner; the pre-text task being the minimization of the reconstruction error. The extracted features by the VAE model are then fed as inputs to an RNN model for RUL estimation, trained in a supervised manner by varying the fraction of labelled engines data down to 1% in order to investigate if the SSL approach can enhance predictive tasks when only a small amount of labelled data is available. Results showed that their proposed method was able to outperform a non pre-trained supervised RNN-model when all the labelled dataset is available as well as in other scenarios when the available labelled data is highly limited with only a small labelled fraction of the training data. The authors in Ellefsen et al. (2019) used a Restricted Boltzmann Machine model (RBM) (Hinton, 2009) for pre-training on unlabelled dataset with a reconstruction pre-text task, and an LSTM model for RUL prediction. Results showed that this SSL approach could improve the RUL prediction accuracy compared to the purely supervised learning approach (*i.e.* predictive model without the initial pre-training stage), both when the training data is completely labelled and when the labelled training data is reduced. It is worth noting that the methods proposed in Yoon et al. (2017) and Ellefsen et al. (2019) were not presented as SSL approaches, but are considered as such in this paper since the proposed methods follow the same procedure as described earlier. However, Krokotsch et al. (2022) highlighted two shortcomings of these two previous studies:

1. the approaches were evaluated only on one subset of the C-MAPSS dataset out of four in each study, rendering these investigations limited;
2. pre-training was performed on unlabelled data of engines that contain the point of failure, which should not be the case in real scenarios, since the RUL labels for all the data could be deduced based on the knowledge of the failure time.

To overcome these limitations in Krokotsch et al. (2022), the investigation was performed over all subsets of the C-MAPSS data set and the unsupervised pre-training phase was performed over truncated time series, assuming that realistic unlabelled data does not contain features near the time of failure (corresponding to sensors data of structures replaced before reaching failure). Results showed that:

1. the proposed SSL approach can outperform the supervised baseline that used only the labelled data. Both approaches were trained on only few labelled time series for RUL estimation (*i.e.* Few-Shots learning);

[1] NASA C-MAPSS (Saxena et al., 2008) is a publicly available dataset of simulated turbofan engines commonly used to benchmark RUL estimation algorithms. The dataset is divided into four subsets (FD001–FD004) of different operating conditions and possible fault modes.

2. the proposed pre-training model outperformed two competing pre-training models, including AE and RBM using a reconstruction pre-text task (*i.e.* the output y corresponds to an estimation of the input x).

These results suggest that the choice of the pre-training model (or pre-text task) matters. Recently, Guo et al. (2022) proposed a pre-training method based on masked autoencoders (He et al., 2022) to perform SSL on the C-MAPSS datasets. Results showed that their pre-trained model outperformed the fully supervised model in RUL estimation. Unfortunately, there are no clear guidelines for selecting the right pre-text task that learns meaningful representations from unlabelled time series data (*e.g.* sensors data) during the pre-training phase. Furthermore, one of the main challenges for extensive investigations on the potential of SSL in PHM resides in the difficulty of having scalable open-source dataset, similar to those available in Natural Language Processing or Image Processing. Thus, despite demonstrating encouraging results, the domain of SSL is still largely unexplored in the prognostics field and is in contrast with the increasing amount of unlabelled data available in industry, having the potential to enable predictive maintenance. Table 1 summarizes the applications of self-supervised learning in PHM identified in this paper.

3. Methodology

The authors of this paper seek to advance the field of data scarcity in fatigue damage prognostic problems by investigating Deep Self-Supervised Learning on an associated RUL estimation problem. In this section, a description of the dataset involved is provided, followed by a description of the problem considered in this paper and the way the Self-Supervised Learning approach is implemented on it. The deep learning-based models used to investigate the SSL approach are also presented and detailed in this section. Note that data and code for the learning procedure are publicly available on https://github.com/ansak95/DeepSSL.

3.1. Data description

In the current research study, a synthetic dataset for a realistic fatigue damage prognostics problem is generated, based on a framework previously proposed by the authors (Akrim et al., 2022). It consists of synthetic multivariate run-to-failure time series data for structures subject to fatigue crack propagation (*e.g.* fuselage panels). Indeed, the proposed framework generates synthetic data sets of mechanical strain data (i.e. virtual strain gauges), by simulating the crack growths in structures based on the Paris–Erdogan model (Paris and Erdogan, 1963). Strain data was considered as sensor data since we consider a mechanical fatigue propagation problem and strain data is one of the main, easily measurable, quantities of interest allowing to determine crack propagation. Furthermore, strain gauge measurement is a mature technique that can be relatively easily implemented on various kinds of structures. The strain data, or measurement sequences, are obtained until the crack size a reaches the critical crack size a_{crit}, considered as the time of failure (necessary to compute the RUL at each time step for example). Finally, the generated strain data are used as sensors time series data available for prognostics problem such as RUL estimation. This setup can be seen representative of real experiments under fatigue loading where the strain state is monitored at multiple strain gauge positions (blue, orange and green crosses), illustrated in Fig. 2.

In the current research, the multivariate dataset used contains the variations of the strains at $n_g = 3$ positions in the panel as a function of the number of cycles, where n_g is the number of the time series. More details about the dataset are given in Akrim et al. (2022), and an illustration of a generated sequence (*i.e.* three placed gauges) for a single structure until failure is given in Fig. 3.

Given the sequential nature of the sensors data, the time series generated are processed sequentially on a *sliding window* approach of size h: at each time-step t, the input of the predictive models corresponds to the current and past measurements, such that $X_t := (x_{t-h+1}, \dots, x_t) \in \mathbb{R}^{n_g \times h}$ where $h = 30$ is the length of the sliding window (note that the value of parameter h was set after preliminary experiments). Fig. 4 illustrates the sliding window approach used in this work.

3.2. The proposed Self-Supervised Learning approach

The Self-Supervised learning paradigm aims to extract useful features from unlabelled data in a self-supervised manner that can subsequently benefit supervised training on few labelled samples. Hence it is typically composed of:

1. *a pre-training phase*: a predictive data-driven model is trained on a raw unlabelled dataset in an unsupervised (or *self-supervised*) manner in order to learn abstract features.
2. *a fine-tuning phase*: the pre-trained model is coupled to a non-pre-trained model (*e.g.* for neural networks a linear layer or data-driven model) and then trained on a set of labelled data in a supervised manner.

The pre-training in SSL is essentially performed with deep learning models. Indeed, the architecture of DL models is in the form of a stack of layers of neurons, and the last layer is used to obtain the final output. Knowledge transfer is typically performed by removing this last layer and replacing it with a new non-trained output linear layer (or a predictive model). The working of SSL can be illustrated in Fig. 5. This strategy allows to reuse the learned knowledge in terms of global architecture of the pre-trained network, which works as a *features extractor*, and to exploit it as a starting point for a downstream task (*i.e.* fine-tuning phase). It also provides faster learning time in downstream predictive tasks compared to non-pre-trained models, since it is not necessary to train the pre-trained layers but only the new output linear layer (or predictive model). This aspect will be discussed in Section 4.4.2. Note that some machine learning models are not suitable for pre-training in SSL paradigm, as their architecture is not composed of layers that can be easily extracted and reused for knowledge transfer (*e.g.* Support Vector Machines Cortes and Vapnik, 1995, Random Forests Ho, 1995, Gaussian Processes Rasmussen, 2003). Nevertheless, there are recent developments of these models that can be used in knowledge transfer (*e.g.* Deep Gaussian Processes Damianou and Lawrence, 2013; Kandemir, 2015).

Table 1
Summary of identified applications of SSL approaches in PHM.

Authors	Year	Downstream task	Pre-training tasks	Application
Yoon et al. (2017)	2017	RUL estimation	Reconstruction of the input signal (variational autoencoder).	Turbofan engines
Ellefsen et al. (2019)	2019	RUL estimation	Reconstruction of the input signal (restricted Boltzmann machine).	Turbofan engines
Krokotsch et al. (2022)	2022	RUL estimation	Reconstruction of the input signal (autoencoder and restricted Boltzmann machine); learn a distance or similarity metric between pairs of data (siamese network).	Turbofan engines
Guo et al. (2022)	2022	RUL estimation	Reconstruction of the input signal (masked autoencoder).	Turbofan engines
Hahn and Mechefske (2021)	2021	Fault diagnostics	Reconstruction of the input signal (variational autoencoder).	Milling tools
Ding et al. (2022)	2022	Fault diagnostics	Contrastive learning (deep convolutional network).	Bearings

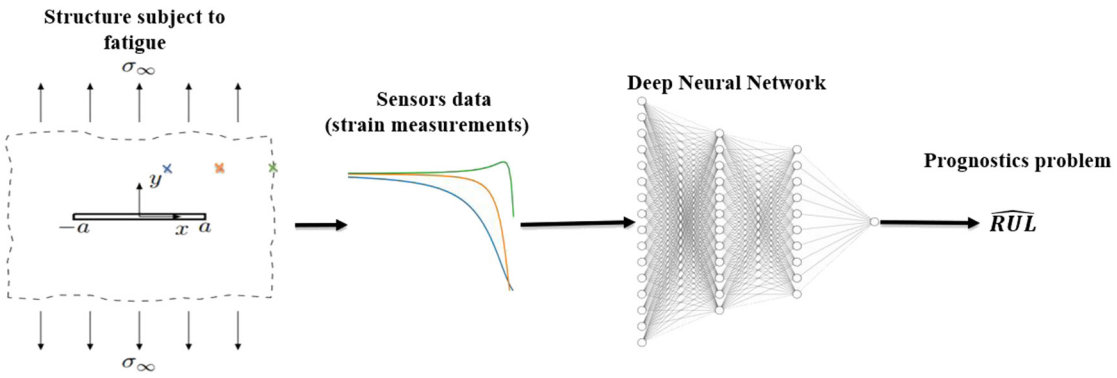

Fig. 2. Illustration of 3 run-to-failure time series generated (*i.e.* strain data) used as sensors data, and as input for prognostic problems (*e.g.* RUL estimation).

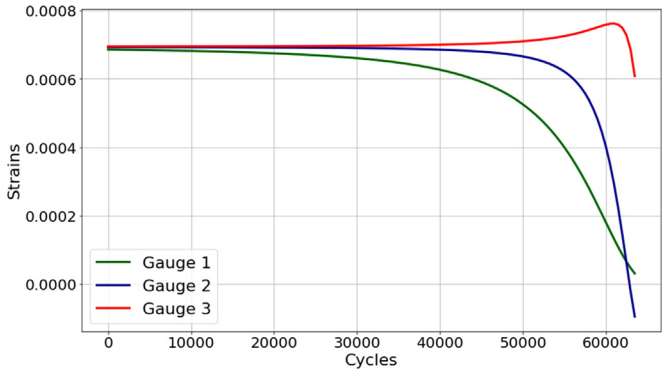

Fig. 3. Strain values time series corresponding to a random sensor sample generated reaching failure (*i.e.* a labelled sequence).

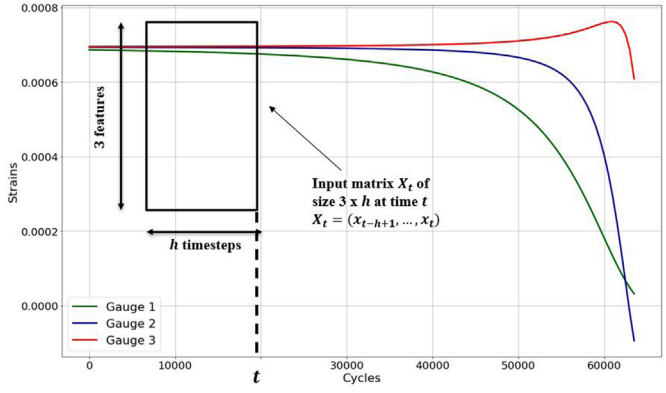

Fig. 4. Illustration of the processing of the input data using a "sliding window" of size h to predict the corresponding output at each timestep t.

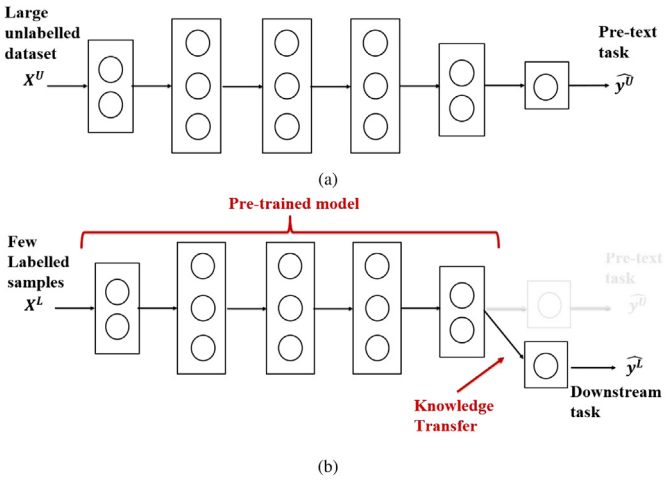

Fig. 5. A schematic view of the Self-Supervised Learning procedure (a): Pre-training phase in a self-supervised way, (b): Fine-tuning phase (supervised training on downstream tasks).

3.2.1. Problem statement

To clearly formulate the problem, the synthetic data used in this work is composed of:

1. A large set of unlabelled data $D_U = \{X_i^U\}_{i=1}^{n_U}$, where n_U is the number of unlabelled samples, $X_i^U \in \mathbb{R}^{n_g \times h}$ the input signal with n_g sensors and h time steps. The unlabelled set D_U refers to strain measurement sequences of structures before reaching failure.
2. A smaller set of labelled data $D_L = \{(X_i^L, y_i^L)\}_{i=1}^{n_L}$, where n_L is the number of labelled samples, $X_i^L \in \mathbb{R}^{n_g \times h}$ the input signal with n_g sensors and h time steps, $y_i^L \in \mathbb{R}$ the corresponding RUL label. The labelled set D_L refers to strain measurement sequences of structures until failure.

Note that the samples of both domains D_U and D_L are multivariate time series sampled from related distributions.

Therefore in this paper, the pre-training phase of the proposed SSL approach consists of pre-training a DL model on unlabelled sensors dataset D_U in a self supervised manner, called *pre-text task*. The pre-trained model is then fine-tuned on a specific downstream Prognostics task, *i.e.* RUL estimation, using only limited amounts of labelled data (*i.e.* strain data of structures until failure, on which the RUL is known at each timestep). The pre-trained model is then fine-tuned on a specific downstream Prognostics task, *i.e.* RUL estimation, using only limited amounts of labelled data D_L. In this work, Deep Gated Recurrent Unit (GRU Rana, 2016) networks, or DGN, are used as the basic deep prediction model, because of their sequential properties and good regressive performance found in previous work (Akrim et al., 2022) (see Appendix for more details about the GRU networks). Note that a DGN consists of a stack of GRU layers in this work. Fig. 6 summarizes the proposed SSL approach in this paper.

3.2.2. Pre-training phase

In order to vary the pre-text tasks and inspired by Liu et al. (2021) and Chen et al. (2020b), two types of models are used and compared in this work: (1) Autoencoders (AE) and (2) Autoregressive (AR) models.

3.2.2.1. Autoencoder architecture in pre-training phase. An Autoencoder (AE) is an artificial neural network that is often used in learning the discriminating features of a dataset in an unsupervised manner (Rumelhart et al., 1986). It is composed of two blocks: encoder and decoder (see

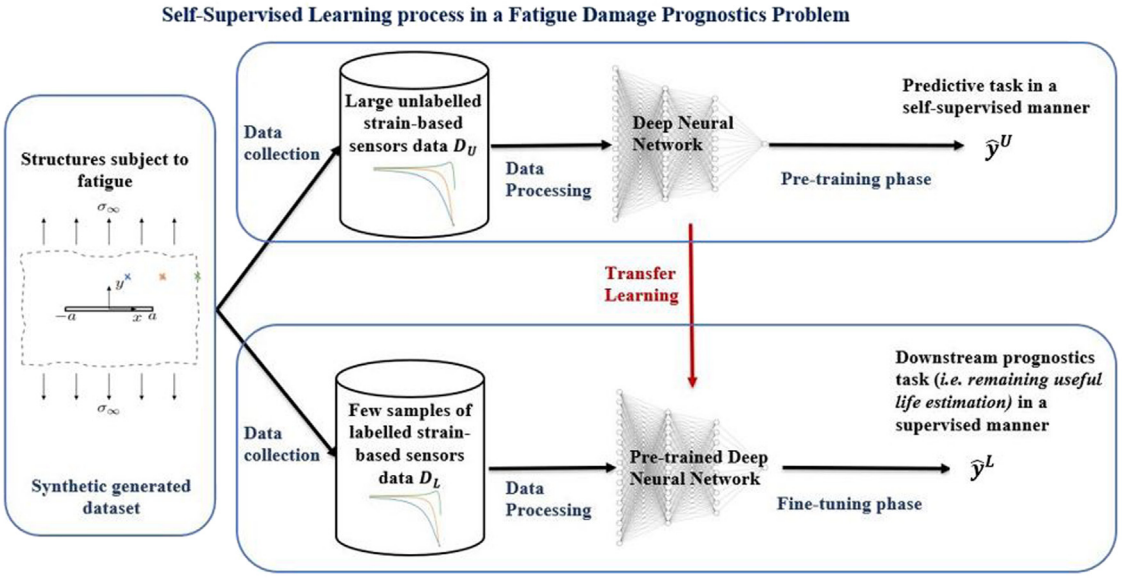

Fig. 6. Flow chart of the proposed Self-Supervised Learning framework.

Fig. 7 for a simplified architecture of the model). The encoder seeks to learn the underlying features of the input data X_t at time step t. These learned features z_t are generally of reduced dimension (number of neurons less than the number of input features). The goal of the decoder is thus to recreate the original data from these underlying learned features. In recent years, Autoencoders have been successful in prognostics applications in terms of feature extraction (Ren et al., 2018; Ma et al., 2018; Sun et al., 2018), which motivated the use of its architecture as a reference model for abstract representation learning in this work.

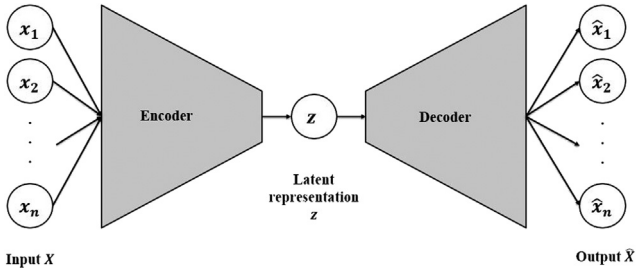

Fig. 7. The architecture of basic Autoencoders. Note that encoders and decoders can be composed of one or more hidden layers.

In pre-training, the output of the Autoencoder (AE) is an estimation of the unlabelled input signal $X_t^U = (x_{t-h+1}^U, \ldots, x_t^U)$ such that $y_t^U = X_t$. A schematic view of the investigated AE model in the proposed SSL framework is given in Fig. 8.

The architecture of the AE model is organized as follows:

1. The input data X_t^U is first embedded through a linear layer[2] f in order to expand the dimension of the data and learn abstract features;
2. The output of the following layer e_t corresponds to a normalized[3] transformation of the embedded input $f(X_t^U)$;
3. The resulting embedded and normalized transformation of the data e_t is then fed to an encoder θ and decoder ψ. Note that both encoder and decoder are Deep GRU networks (DGN), *i.e.* stack of GRU layers;
4. z_t is considered as the learned representation by the model and will be used for feature extraction in the following;
5. In this architecture, two skip connections[4] are used through deep GRU networks such that $z_t = \theta(e_t) + e_t$ and $o_t = \psi(z_t) + z_t$;
6. The output linear layer g is then used to generate an estimation $\hat{y}_t^U$ of the unlabelled input signal X_t^U, *i.e.* an estimation of the input signal such that $\hat{y}_t^U = \hat{X}_t^U$.

[2] The input linear layer is used as an alternative to the embedding layers used in Natural Language Processing (Hrinchuk et al., 2019) since the input data is continuous is this work, converting each time step data into a fixed length vector of defined size.

[3] The layer normalization (Ba et al., 2016) are used for regularized training and faster convergence.

[4] Skip connections in DL architectures, also called *residual connections* or *shortcut connections*, consist in skipping some layers in the neural network and feeding the output of one layer as the input to the next layers (Adaloglou, 2020), used to solve the *degradation problem* (e.g. ResNet He et al., 2016). In this paper, skip connections are proposed to establish a direct connection through deep GRU networks in order to avoid information loss and learn robust sequential representation, which has already proven to be effective for deep recurrent neural networks (Yue et al., 2018).

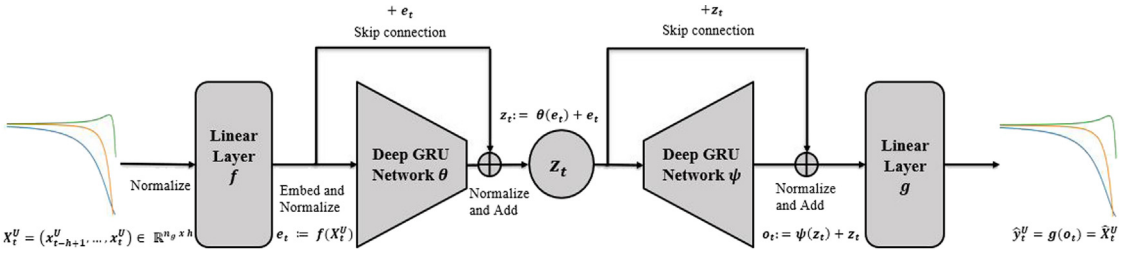

Fig. 8. Flow chart of the pre-training phase of the Autoencoder model (AE) in the proposed Self-Supervised Learning framework.

3.2.2.2. Autoregressive architecture in pre-training phase. An Autoregressive (AR) model g_h can be defined as a sequential model governed by an Autoregressive process of order h that models the future outcome of a sequence at time $t+1$, using its previous h realizations. Autoregressive modelling captures the temporal dependencies between sequential input data, which makes it useful in learning better features. Inspired by the autoregressive DL models used in Brown et al. (2020) and Chen et al. (2020b) for pre-training, the proposed AR model in this paper consists of a Deep GRU network in which the input is a sequence of h time steps at t such that $X_t^U = (x_{t-h+1}^U, \dots, x_t^U)$, and the output is an estimation of the data of the next timestep such that $y_t^U = x_{t+1}^U$, according to the following formula:

$$\hat{y}_t^U = \hat{x}_{t+1}^U = \mathcal{F}_h(x_t^U, x_{t-1}^U, \dots, x_{t-h+1}^U) \tag{1}$$

where $\mathcal{F}_h$ denotes the AR model governed by an autoregressive process of order h. A schematic view of the investigated AR models in the proposed SSL framework is given in Fig. 9.

The architecture of the AR model is organized as follows:

1. The embedding linear layer f is used in order to expand the dimension of the input data and learn abstract features;
2. The output of the following layer e_t corresponds to a normalized transformation of the embedded input $f(X_t^U)$;
3. The resulting embedded and normalized transformation of the data e_t is then fed to a Deep GRU Network ψ, composed of a stack of GRU layers;
4. z_t is considered as the learned representation by the model and will be used for feature extraction in the following;
5. In this architecture, a skip connection is used such that $z_t = \psi(e_t) + e_t$;
6. The output linear layer g is then used to generate an estimation $\hat{y}_t^U$ of the input signal X_t^U, *i.e.* the data of the next timestep such that $\hat{y}_t^U = \hat{x}_{t+1}^U$.

3.2.3. Fine-tuning phase

In the fine-tuning phase, an RUL estimation problem is considered, hence the output of the predictive models is a point-wise estimation of the RUL such that $y_t^L = RUL_t$. The embedding z_t of the input data is extracted (see Figs. 10 and 11), the weights of the hidden pre-trained layers are frozen, then for fine-tuning a simple GRU layer ϕ followed by an output linear layer $\tilde{g}$ are used such that:

$$\begin{aligned} \hat{y}_t^L &= \tilde{g} \circ \phi(z_t) \\ &= \hat{RUL}_t \end{aligned} \tag{2}$$

where the function ϕ refers to the fine-tuning GRU layer and the function $\tilde{g}$ to the output linear layer. Note that, in the fine-tuning phase it is common to use only a linear layer for training, but the authors found that adding a GRU layer significantly improves the performance of the approach on this RUL estimation problem.

Finally, in order to investigate the added value of the SSL approach in prognostics, the pre-trained models are compared with their non-pre-trained counterpart architecture, illustrated in Figs. 10 and 11. Note that the architectures of the pre-trained and non-pre-trained models are the same, the difference residing in the absence of pre-training on unlabelled data and the corresponding *knowledge transfer*. Also, only the non-pre-trained weights for the pre-trained models are trained (*i.e.* trainable weights of the fine-tuning model), while all the trainable parameters of the non-pre-trained models are trained. Note that the pre-trained model with an autoregressive pre-text task followed by a GRU model for fine-tuning will be referred to as the "autoregressive model" in the following for simplicity.

4. Experiments and results

4.1. Preparation of data

In this experiment, an Aluminium alloy 7075-T6 plate was considered, which is typical of aeronautic structures. Considering that the evolution of the changes from one cycle to another are small (see Fig. 3), it was decided to collect the data every $\Delta k = 500$ loading–unloading cycles, as in Akrim et al. (2022).

In the current paper, a training set and a testing set are generated. The training set is composed of:

1. N_U^{Train} unlabelled structures for the pre-training phase,
2. N_L^{Train} labelled structures for the fine-tuning phase.

Note that the number of structures N_U^{Train} and N_L^{Train} are varied; this will be described in the following subsections. The testing set is composed of N_L^{Test} labelled structures. It is used to evaluate the RUL estimation performance of the trained models (in fine-tuning) as a data set that was not used during training. The parameters used to generate the dataset according to the framework described in Akrim et al. (2022) are summarized in Table 2.

4.2. Experimental settings in pre-training phase

As the structures subjected to fatigue can be replaced before reaching failure at any time, the proposed approach has been investigated on four degradation scenarios: for pre-training, available sequences of unlabelled data are incomplete at $d = 60\%$, 70%, 80%, and 90% of their total lifetime, where d is the ratio of the total lifetime of a sequence. To illustrate the size of the strain data sequences available, these four degradation scenarios are illustrated in Fig. 12. Moreover, the number of pre-training structures, denoted N_U^{Train}, for which strain sequences were available was varied in order to investigate the effect of the amount of unlabelled data. The investigated models (autoencoder and autoregressive model) were therefore pre-trained on $N_U^{Train} = 100, 1000, 5000$, and 10 000 unlabelled structures. As mentioned before, strain data are collected every 500 cycles, and a sliding window approach of $h = 30$ is used (see Section 3). Thus, as an illustration, Table 3 summarizes the number of pre-training samples n_U for the autoencoder model in each degradation scenario. In each training procedure, 95% of the dataset was used for training (in terms of the number of structures), while 5% of it was used for validation. The validation set is used for monitoring and adjusting the training phase, using the mean

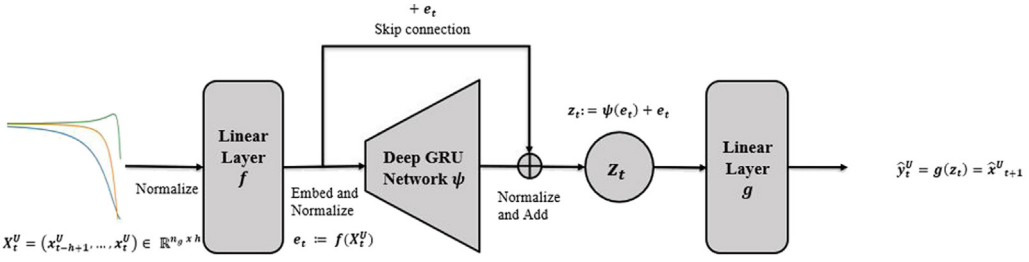

Fig. 9. Flow chart of the pre-training phase of the Autoregressive model (AR) in the proposed Self-Supervised Learning framework.

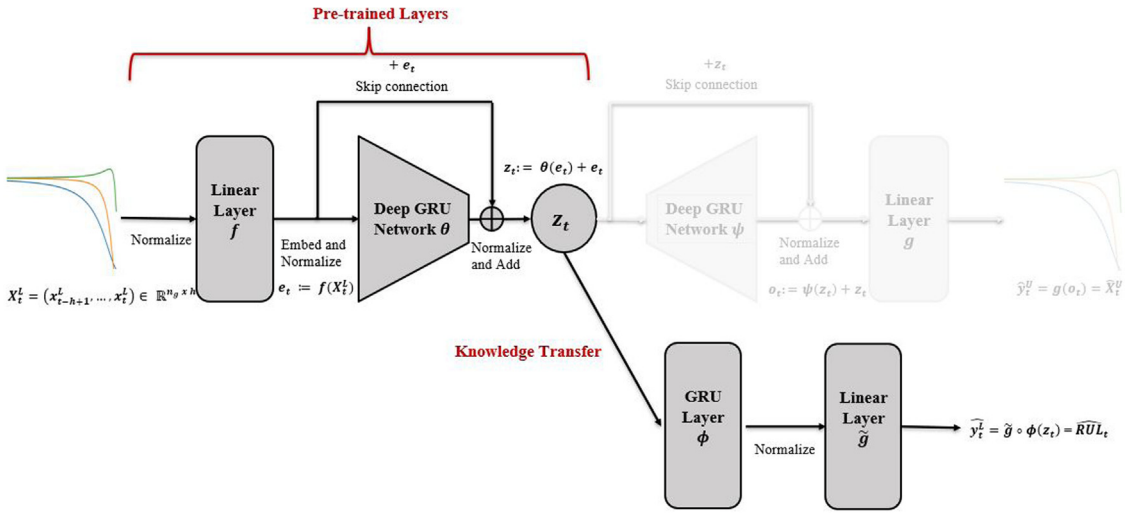

Fig. 10. Flow chart of the fine-tuning phase of the Autoencoder model (AE) in the proposed Self-Supervised Learning framework.

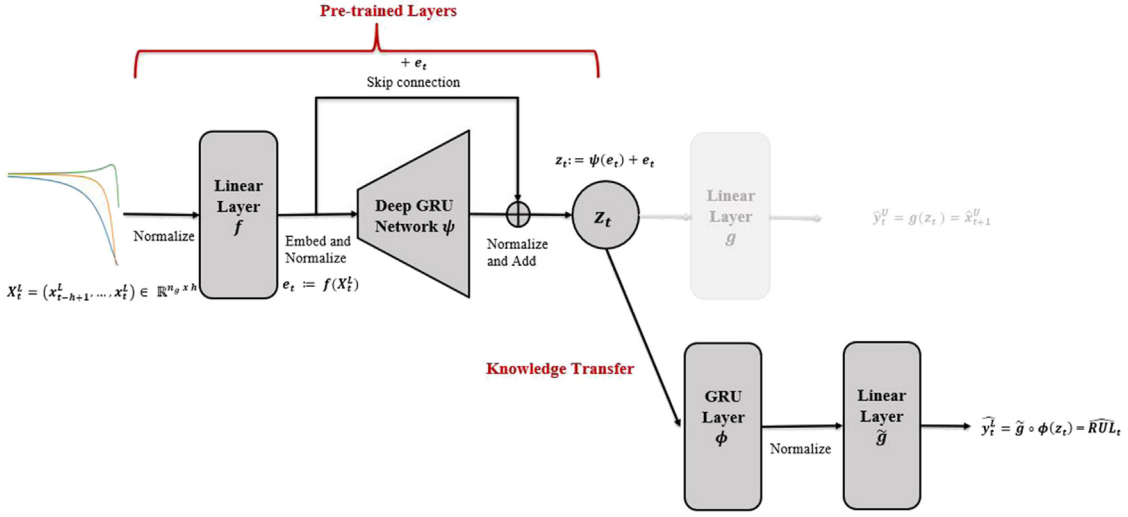

Fig. 11. Flow chart of the fine-tuning phase of the Autoregressive model (AR) in the proposed Self-Supervised Learning framework.

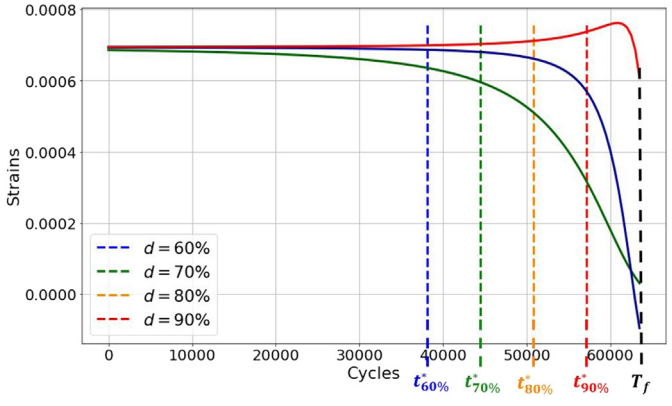

Fig. 12. Four degradation scenarios depicted on the sequence of a structure. For each scenario, the available strain data correspond to the measurements from time 0 to time $t_d^* := d \times T_f$, where d is the ratio of the total lifetime of a sequence, and T_f is the time of failure.

absolute percentage error (*MAPE*) metric. During training, the aim is to minimize the mean squared error (*MSE*) loss function L_{MSE} such that:

$$L_{MSE} = \frac{1}{n_U} \sum_{i=1}^{n_U} (y_i^U - \hat{y}_i^U)^2 \tag{3}$$

$$MAPE = \frac{1}{n_U} \sum_{i=1}^{n_U} \left| \frac{y_i^U - \hat{y}_i^U}{y_i^U} \right| * 100 \tag{4}$$

where n_U is the number of unlabelled samples with $\hat{y}^U$ being the prediction and y^U the target value. Note that $y_i^U = (x_{i-h+1}^U, \dots, x_i^U)$ for the AE model and $y_i^U = x_{i+1}^U$ for the AR model. The Adam optimizer (Kingma and Ba, 2014) was used with default parameters and the learning rate was decreased incrementally. The learning rates of $10^{-2}, 10^{-3}, 10^{-4}$ were sequentially used for a predefined number of epochs, saving the model weights each time the validation loss decreases; the weights of the best model were loaded each time the learning rate was lowered. At the end of the procedure, the model was trained on the whole dataset (training and validation sets) with a lower learning rate of 10^{-5} until convergence. Calculations were performed using PyTorch's core library in Python on NVIDIA V100 GPUs, hence the batch size was chosen to be as large as possible in order to speed up calculations, here $2^{12} = 4096$ depending on the available memory of the used GPUs, and not too large in order to avoid numerical instability. The model hyperparameters were optimized using a Grid Search algorithm, listed in Table 4:

Table 2
Parameters for numerical study. The full model description is available in Akrim et al. (2022).

Parameter	Denotation	Type	Value	Unit
Elastic parameters				
Young's modulus	E	Deterministic	71.7	GPa
Poisson's ratio	ν	Deterministic	0.33	–
Strain field parameters				
Maximum stress intensity	σ_{max}	Uniform distribution	$\mathcal{U}(75, 85).10^6$	Pa
Fracture toughness	K_I	Deterministic	$19,7.10^6$	$\text{Pa}\sqrt{m}$
Strain gauges				
Number of gauges placed	n_g	Deterministic	3	–
Position of the gauges placed	$(x_i, y_i)_{i=1,\dots,n_g}$	Deterministic	(3, 14), (14, 14), (25, 14)	mm
Angle of the gauges placed	θ	Deterministic	45	deg
Initialization parameters				
Initial crack size	a_0	Gaussian distribution	$\mathcal{N}(\mu_{a_0}, \sigma_{a_0})$	m
Mean of a_0	μ_{a_0}	Deterministic	5.10^{-4}	m
Standard deviation of a_0	σ_{a_0}	Deterministic	$2,5.10^{-4}$	m
Paris–Erdogan's law parameters	$(m, \log C)$	Multivariate Gaussian distribution	$\mathcal{N}(\mu_m, \sigma_m, \mu_{\log C}, \sigma_{\log C}, \rho)$	–
Mean of m	μ_m	Deterministic	3,4	-
Standard deviation of m	σ_m	Deterministic	0,25	–
Mean of C	μ_C	Deterministic	1.10^{-10}	–
Standard deviation of C	σ_C	Deterministic	5.10^{-11}	–
Correlation coefficient of m and $\log C$	ρ	Deterministic	−0.996	–
Generated data set				
Number of unlabelled structures for training	N_U^{Train}	Deterministic	(100, 1000, 5000, 10 000)	–
Number of labelled structures for training	N_L^{Train}	Deterministic	(5, 10, 20, 50, 100)	–
Number of labelled structures for testing	N_L^{Test}	Deterministic	100	–
Data collection interval	Δk	Deterministic	500	–

Table 3
Number of unlabelled pre-training samples n_U^{Train} used in this work for the autoencoder. Note that in this work, strain data are collected every $\Delta_k = 500$ cycles and a sliding window approach of $h = 30$ is used.

Number of pre-training structures N_U^{Train}	Ratio of the total lifetime d			
	$d = 60\%$	$d = 70\%$	$d = 80\%$	$d = 90\%$
$N_U^{Train} = 100$	$n_U^{Train} = 11\,880$	$n_U^{Train} = 14\,346$	$n_U^{Train} = 16\,819$	$n_U^{Train} = 19\,283$
$N_U^{Train} = 1000$	$n_U^{Train} = 114\,537$	$n_U^{Train} = 138\,451$	$n_U^{Train} = 162\,511$	$n_U^{Train} = 186\,443$
$N_U^{Train} = 5000$	$n_U^{Train} = 571\,515$	$n_U^{Train} = 690\,932$	$n_U^{Train} = 811\,015$	$n_U^{Train} = 930\,545$
$N_U^{Train} = 10\,000$	$n_U^{Train} = 1137\,959$	$n_U^{Train} = 1375\,948$	$n_U^{Train} = 1\,615\,261$	$n_U^{Train} = 1\,853\,470$

Table 4
Hyperparameters of the pre-training phase.

Hyperparameters	Search space	Autoencoder model	Autoregressive model
Linear layer f - neurons	{32, 64}	64	64
Deep GRU network θ - neurons	{32, 64, 128, 256}	64	–
Deep GRU network θ - layers	{1, 2, 4, 8}	2	–
Deep GRU network θ - dropout	{0, 0.1, 0.2, 0.3}	0.1	–
Deep GRU network ψ - neurons	{32, 64, 128, 256}	64	64
Deep GRU network ψ - layers	{1, 2, 4, 8}	2	4
Deep GRU network ψ - dropout	{0, 0.1, 0.2, 0.3}	0.1	0.1

- Autoencoder model illustrated in Fig. 8: the embedding linear layer f is composed of 64 neurons, the Deep GRU Networks θ and ψ were each composed of 2 layers of GRU, 64 neurons, and a dropout of 0.1, that is to say nearly 100.000 parameters.
- Autoregressive model illustrated in Fig. 9: the embedding linear layer f is composed of 64 neurons, the Deep GRU Network ψ was composed of 4 GRU layers, 64 neurons, and a dropout of 0.1, that is to say nearly 100.000 parameters.

Note that the authors found that, given the training data, the search space considered was sufficient to obtain good results, whereas deep neural networks with a larger number of layers/neurons performed poorer with longer training (probably due to more difficult convergence).

4.3. Experimental settings in fine-tuning phase

For fine-tuning, as illustrated in Figs. 10 and 11, the embedding z_t of the input data was extracted, the weights of the hidden layers were frozen, and a GRU model was used for the downstream task. The models are then trained on N_L available labelled structures, using a sliding window approach similar to that used in the previous pre-training phase. The fine-tuning model was composed of a single GRU layer, 32 neurons, 0.1 in dropout to regularize, and followed by an output linear layer (the hyperparameters were optimized using a Grid Search algorithm on the autoencoder pre-trained model, listed in Table 5).

Table 5
Hyperparameters of the fine-tuning phase.

Hyperparameters	Search space	Fine-tuning model
Deep GRU network ϕ - neurons	{32, 64}	32
Deep GRU network ϕ - layers	{1, 2}	1
Deep GRU network ϕ - dropout	{0, 0.1, 0.2, 0.3}	0.1
Batch size	{32, 64}	32

The pre-trained models were then compared with their non-pre-trained "counterpart" (*i.e.* same architecture but all model weights were reset) on few shots learning. The number of available labelled training structures N_L^{Train} were varied, such that: $N_L^{Train} = 5, 10, 20, 50$ and 100 labelled structures (*i.e.* strain data of structures reaching failure

at time T_f, thus for which the RUL is available for each timestep between times 0 and T_f). Calculations were performed using PyTorch's core library in Python on a machine with 62 GB of RAM and an NVIDIA GeForce GTX 1080 Ti 11 GB GPU.

After training during the fine-tuning phase, the models are evaluated on the testing set composed of $N_L^{Test} = 100$ different labelled structures. For each structure, a unique RUL estimation is performed at a time t_n^*. For each structure $n \in \{1, \dots, N_L^{Test}\}$, the parameter t_n^* is randomly drawn such that $t_n^* \sim T_f^n \times \mathcal{U}([0,33; 0,9])$, where T_f^n is the time of failure for the nth structure. This means that the test prediction for the RUL is done at a time t_n^* which is drawn uniformly between 33% and 90% of the sequence's length. Hence, the input data for the model is $X_{t_n^*}^L = (x_{t_n^*-h+1}^L, \dots, x_{t_n^*}^L) \in \mathbb{R}^{n_g \times h}$ and the output of the model is $\hat{y}_{t_n^*}^L = \hat{RUL}_{t_n^*} \in \mathbb{R}$. As the RUL estimation problem is considered as a regression problem in this paper, the aim is to minimize a mean squared error loss L_{MSE} during training, and the mean absolute percentage error (*MAPE*) metric is used to evaluate the performance of the investigated models such that:

$$L_{MSE} = \frac{1}{n_L} \sum_{i=1}^{n_L} (y_i^L - \hat{y}_i^L)^2 \tag{5}$$

$$MAPE = \frac{1}{n_L} \sum_{i=1}^{n_L} \left| \frac{y_i^L - \hat{y}_i^L}{y_i^L} \right| * 100 \tag{6}$$

where n_L is the number of labelled samples with $\hat{y}^L$ being the RUL prediction and y^L the target RUL value.

As a limited amount of labelled data leads to epistemic uncertainty, it is difficult to make a reliable comparison. Hence, a 5-fold cross validation was used by varying the split between the training and validation set, as illustrated in Fig. 13, which gives an average *MAPE* error and its standard deviation to quantify the uncertainty when evaluated on the test set.

4.4. Results

4.4.1. Pre-training analysis and comparison of pre-text tasks

First, the performance of the pre-trained Autoencoder (AE) and non-pre-trained counterpart were compared on the considered RUL estimation problem, by varying the number of unlabelled samples in pre-training. The results are presented in Table 6.

A first remark that can be drawn from the results in Table 6 is that pre-training the model is not always beneficial. For example, for an AE model pre-trained on 100 structures, the accuracy of the RUL estimation is not always better compared to the non-pre-trained model (AE). The term *negative transfer* can be used when the transfer method decreases predictive performance (Torrey and Shavlik, 2010). Moreover, considerable variability in results can be observed when models are pre-trained on very few unlabelled samples (for example when fine-tuned on 5 labelled samples), which could be due to overfitting during pre-training. However, it can also be observed that as the number of unlabelled samples increases, the pre-training becomes more efficient and allows to have better results than a non-pre-trained model when few labelled structures are available, especially for the model pre-trained on 10 000 structures. For example, the AE pre-trained models with $N_U^{Train} = 10\,000$ unlabelled structures and fine-tuned on $N_L^{Train} = 10$ structures has an $MAPE$ of about 11%–12% while the non-pre-trained model with the same $N_L^{Train} = 10$ structures has an $MAPE$ of about 23%. Overall, results in Table 6 show that for the Autoencoder model, the number of unlabelled samples in pre-training matters: the more unlabelled samples, the more efficient the self supervised learning is for each of the 4 scenarios. The autoregressive model shows similar performances, presented in Table 7.

In order to illustrate the differences in performance between the autoencoder and the autoregressive pre-text tasks, the $MAPE$ of these models for $N_U^{Train} = 10\,000$ structures as well as the $MAPE$ of their non-pre-trained counterparts are provided in Fig. 14. For very few labelled structures ($N_L^{Train} = 5$), results illustrated in Fig. 14 do not allow to clearly distinguish between the two models due to the limited number of labelled samples, leading all models to work relatively poorly.

Nevertheless, results show that both pre-trained models clearly outperform their non-pre-trained counterpart in Few-Shots learning (more than 5 but less than 50 structures). The AR pre-trained model significantly outperforms the AE pre-trained one when fine-tuned on 10 or 20 structures, and has almost three times less estimation error than the best non-pre-trained model. These results make sense since the autoregressive task and the RUL estimation task have in common the task of predicting future outcome, and may need to capture the temporal dependencies of the input signal. However, it can also be seen that as the number of labelled samples increases, the difference between the pre-trained and non-pre-trained models is reduced (*e.g.* trained on more than 50 structures).

Given the good performance of the autoregressive model, some further variations of this concept were investigated. Hence, an extended Autoregressive model was proposed, denoted multi-steps prediction autoregressive (MSPA) model, for which the pre-text task consists in estimating at each timestep t the data from the next timestep $t + 1$ until the timestep $t + q$, such that $y_t^U = (\hat{x}_{t+1}^U, \dots, \hat{x}_{t+q}^U)$, with $q \in \mathbf{N}^*$, as illustrated in Fig. 15.

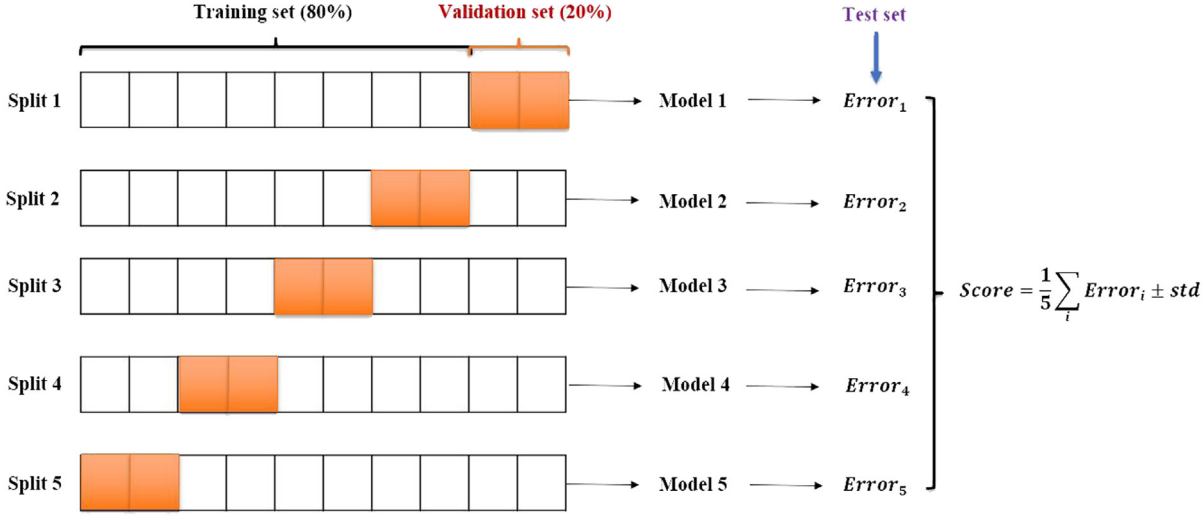

Fig. 13. Flow chart of the 5-fold cross validation used in this paper.

Table 6
MAPE mean values (in %) plus or minus its standard deviation as a function of the number of labelled structures used and of the training scenario for the autoencoder case. Best performance is represented in bold for each case.

Labelled structures N_L^{Train}		$MAPE$ (%)	Mean ± St.	Dev.	
	5	10	20	50	100
Pre-trained model (d = **60%**)					
Autoencoder N_U^{Train} = 100	29.36 ± 3.81	21.16 ± 2.86	11.84 ± 4.12	1.95 ± 0.25	1.87 ± 0.36
Autoencoder N_U^{Train} = 1.000	25.95 ± 0.74	14.78 ± 3.13	5.59 ± 1.47	1.42 ± 0.17	1.31 ± 0.06
Autoencoder N_U^{Train} = 5.000	28.55 ± 0.99	**12.01 ± 2.31**	4.80 ± 0.92	**1.27 ± 0.11**	**1.08 ± 0.03**
Autoencoder N_U^{Train} = 10.000	**24.60 ± 4.30**	12.70 ± 1.76	**3.48 ± 0.93**	1.41 ± 0.04	1.13 ± 0.12
Pre-trained model (d = **70%**)					
Autoencoder N_U^{Train} = 100	28.02 ± 1.96	20.99 ± 3.73	11.70 ± 3.78	1.61 ± 0.43	1.43 ± 0.25
Autoencoder N_U^{Train} = 1.000	30.76 ± 1.83	18.20 ± 3.76	6.42 ± 1.69	1.65 ± 0.36	1.29 ± 0.16
Autoencoder N_U^{Train} = 5.000	26.10 ± 0.88	18.73 ± 2.72	6.67 ± 1.30	1.58 ± 0.23	1.25 ± 0.08
Autoencoder N_U^{Train} = 10.000	**25.38 ± 5.99**	**11.85 ± 2.64**	**3.77 ± 0.61**	**1.29 ± 0.06**	**1.08 ± 0.12**
Pre-trained model (d = **80%**)					
Autoencoder N_U^{Train} = 100	34.96 ± 9.59	17.72 ± 4.43	8.92 ± 3.33	1.33 ± 0.13	1.41 ± 0.20
Autoencoder N_U^{Train} = 1.000	26.46 ± 2.51	15.78 ± 3.77	4.74 ± 0.79	1.31 ± 0.13	1.10 ± 0.05
Autoencoder N_U^{Train} = 5.000	30.19 ± 6.56	17.18 ± 3.03	6.19 ± 2.07	1.60 ± 0.22	1.17 ± 0.18
Autoencoder N_U^{Train} = 10.000	**24.77 ± 4.80**	**12.06 ± 3.82**	**4.27 ± 0.72**	**1.18 ± 0.13**	**1.01 ± 0.06**
Pre-trained model (d = **90%**)					
Autoencoder N_U^{Train} = 100	28.27 ± 2.47	21.56 ± 0.98	9.91 ± 3.29	1.57 ± 0.35	1.56 ± 0.16
Autoencoder N_U^{Train} = 1.000	30.27 ± 1.15	18.53 ± 5.68	6.51 ± 1.63	1.74 ± 0.43	1.13 ± 0.15
Autoencoder N_U^{Train} = 5.000	25.99 ± 1.73	17.06 ± 4.09	5.24 ± 1.60	1.43 ± 0.31	1.06 ± 0.07
Autoencoder N_U^{Train} = 10.000	**22.97 ± 5.69**	**11.04 ± 3.61**	**3.39 ± 0.67**	**1.22± 0.12**	**0.88 ± 0.09**
Non-pre-trained model					
Autoencoder architecture	27.72 ± 0.65	23.07 ± 5.94	8.12 ± 1.87	1.40 ± 0.33	**0.83 ± 0.18**

Table 7
MAPE mean values (in %) plus or minus its standard deviation as a function of the number of labelled structures used and of the training scenario for the autoregressive case. Best performance is represented in bold for each case.

Labelled structures N_L^{Train}		$MAPE$ (%)	Mean ± St.	Dev.	
	5	10	20	50	100
Pre-trained model (**d** = **60%**)					
Autoregressive N_U^{Train} = 100	36.15 ± 13.48	20.18 ± 5.19	12.17 ± 3.31	2.29 ± 0.22	1.70 ± 0.19
Autoregressive N_U^{Train} = 1.000	28.14 ± 3.20	16.65 ± 1.21	8.80 ± 3.13	1.48 ± 0.18	1.21 ± 0.13
Autoregressive N_U^{Train} = 5.000	26.53 ± 1.57	13.58 ± 2.14	7.25 ± 3.09	1.22 ± 0.02	1.00 ± 0.01
Autoregressive N_U^{Train} = 10.000	**22.48 ± 6.06**	**7.34 ± 0.95**	**2.63 ± 0.80**	**1.14 ± 0.04**	**1.03 ± 0.06**
Pre-trained model (**d** = **70%**)					
Autoregressive N_U^{Train} = 100	28.95 ± 1.74	16.98 ± 2.96	11.85 ± 2.73	2.01 ± 0.20	1.51 ± 0.34
Autoregressive N_U^{Train} = 1.000	26.76 ± 2.49	16.34 ± 1.38	8.54 ± 1.89	1.53 ± 0.20	1.18 ± 0.16
Autoregressive N_U^{Train} = 5.000	25.18 ± 2.70	10.96 ± 2.46	6.79 ± 2.51	1.33 ± 0.13	1.27 ± 0.21
Autoregressive N_U^{Train} = 10.000	**24.43 ± 4.08**	**8.46 ± 1.53**	**2.42 ± 0.53**	**1.20 ± 0.06**	**1.14 ± 0.17**
Pre-trained model (**d** = **80%**)					
Autoregressive N_U^{Train} = 100	30.47 ± 3.47	20.04 ± 2.59	10.68 ± 4.25	2.34 ± 0.85	1.48 ± 0.07
Autoregressive N_U^{Train} = 1.000	**26.10 ± 2.14**	13.66 ± 3.22	6.01 ± 1.45	1.44 ± 0.05	1.17 ± 0.03
Autoregressive N_U^{Train} = 5.000	26.46 ± 2.46	12.60 ± 1.33	6.91 ± 1.28	1.38 ± 0.11	1.09 ± 0.05
Autoregressive N_U^{Train} = 10.000	26.50 ± 2.58	**8.09 ± 3.28**	**2.39 ± 0.26**	**1.39 ± 0.20**	**1.07 ± 0.11**
Pre-trained model (**d** = **90%**)					
Autoregressive N_U^{Train} = 100	35.76 ± 7.41	17.64 ± 2.58	8.66 ± 2.26	1.60 ± 0.17	1.33 ± 0.19
Autoregressive N_U^{Train} = 1.000	25.00 ± 3.89	17.31 ± 1.72	8.59 ± 2.79	1.45 ± 0.15	1.29 ± 0.20
Autoregressive N_U^{Train} = 5.000	23.01 ± 3.23	12.97 ± 2.91	3.15 ± 0.45	1.21 ± 0.08	1.05 ± 0.03
Autoregressive N_U^{Train} = 10.000	**22.69 ± 2.36**	**8.83 ± 1.61**	**3.39 ± 0.40**	**1.25 ± 0.07**	**0.99 ± 0.06**
Non-pre-trained model					
Autoregressive architecture	28.09 ± 1.63	21.10 ± 1.92	7.52 ± 1.59	1.15 ± 0.09	**0.79 ± 0.09**

Results in Table 8 show that increasing the prediction time horizon in pre-training can improve the predictive performance of the fine-tuned models, when few labelled structures for training are available ($N_L^{Train} = 5$ or 10). For example, the MSPA model with a time horizon of $q = 30$ and fine-tuned on $N_L^{Train} = 5$ structures has an $MAPE$ of about 13% when $d = 90\%$, while the initial autoregressive model trained under the same conditions has an $MAPE$ of about 22%. Moreover, on very few labelled training structures ($N_L^{Train} = 5$), the results show that the parameter d has a significant influence on the pre-training of the MSPA models: the higher d is, the more the available sequence data is close to the given failure time T_f (see Section 4.1) and the better the RUL estimation performance of the MSPA pre-trained models. This improvement of the MSPA performance compared to the AR one makes sense since when at $d = 90\%$, predicting $q = 30$ timesteps means predicting the strain data until the time of failure. Being able to accurately predict until time of failure facilitates of course the downstream RUL prediction task. However, note that this performance does not hold when more labelled samples for training are available (N_L^{Train} greater than 20) by becoming worse than those of the initial autoregressive model and the non-pre-trained model, which does not

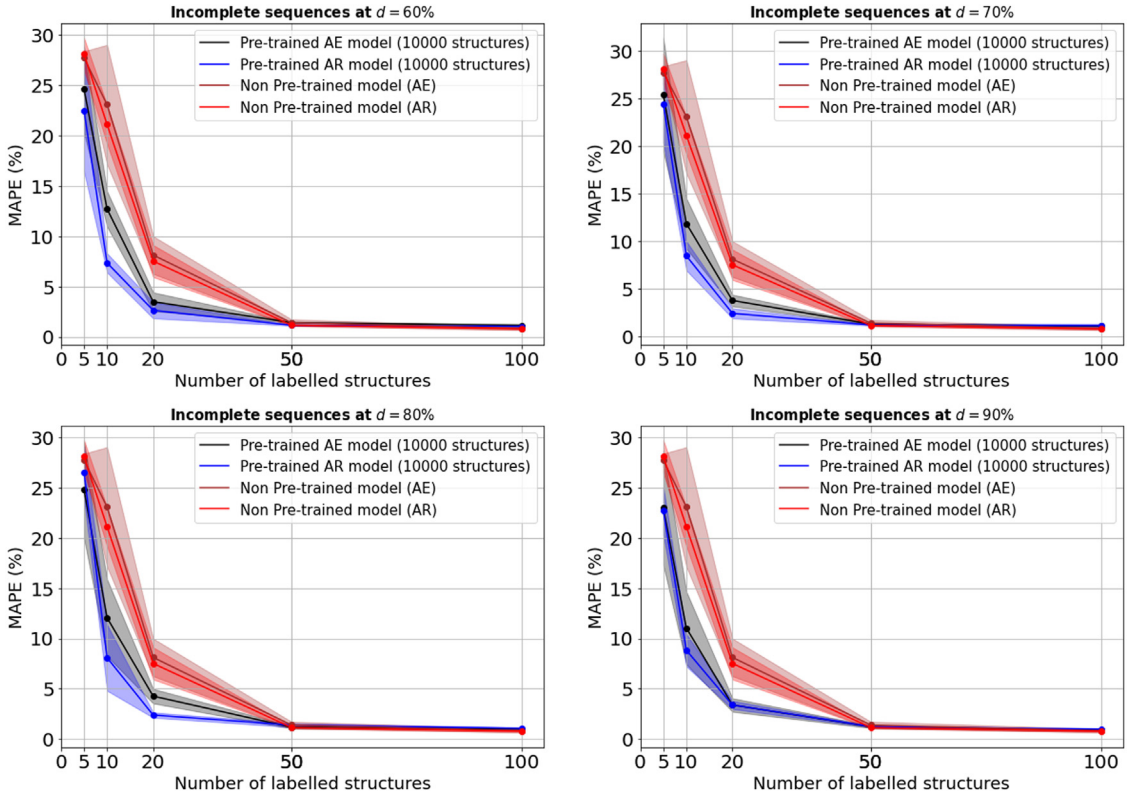

Fig. 14. Comparison of pre-trained and non-pre-trained models in RUL estimation for the testing set (100 samples). The *MAPE* metric (%) is used, and here the target value to estimate is the RUL.

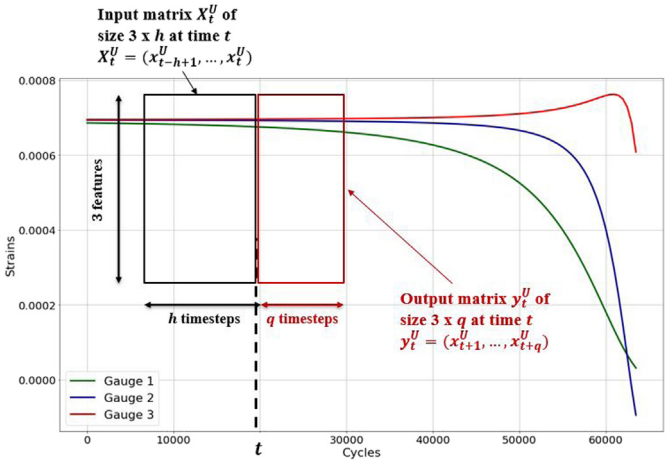

Fig. 15. Illustration of the extended Autoregressive model (MSPA) pre-text task.

allow general conclusions to be drawn. One possible explanation for this worsening is that training the DL models with values of q greater than 1 is significantly more challenging. Some of the variations seen may then be related to the pre-training phase being not yet fully converged.

Future work could seek to better control the training convergence of the pre-text task for the MSPA models. As a final remark, note that the authors also tried as outputs of the pre-text task predicting $y_t^U = \hat{x}_{t+q}^U$ only, instead of predicting the entire time-windows $(\hat{x}_{t+1}^U, \dots, \hat{x}_{t+q}^U)$. As the results obtained after fine-tuning were similar for the two approaches, in this study only the pre-text task considering the entire time-windows has been presented and described in this paper.

4.4.2. *Freezing pre-trained layers during learning*

In the fine-tuning phase of the previous subsection, the weights of the pre-trained layers were frozen, and only the weights of the fine-tuning model were trainable (*i.e.* GRU network for fine-tuning as illustrated in Fig. 11. Therefore, the authors also sought to investigate the effect of not freezing the pre-trained layers during the fine-tuning phase. Unfreezing the layers means that the pre-text task is basically used to find a good starting point for the training of the full network architecture. As the autoregressive model showed the best performance so far, the investigation was done on this model. In the fine-tuning phase, the model based on the autoregressive structure is composed of 125 121 trainable parameters when the pre-trained layers are not frozen, against 25 025 trainable parameters when they are frozen. Results in Table 9 show that the two approaches perform almost similarly, so it is difficult to determine whether it is better to freeze or not the layers in this RUL estimation problem. Note that both pre-trained models remain better in RUL estimation than their non-pre-trained counterpart, with or without frozen pre-trained layers, which confirms the benefits of the pre-training in all the cases.

Nevertheless, it should be noted that freezing the weights of the pre-trained layers considerably reduces the number of trainable parameters (25 025 trainable parameters when the pre-trained layers are frozen, against 125 121 trainable parameters when they are not), and therefore reduces the computational complexity during training. Indeed, Fig. 16 shows that freezing the pre-trained layers speeds up the calculations considerably (1.5 to 2 times less time than other models), while both investigated learning approaches in this subsection perform almost similarly as shown in Table 9.

5. Discussion

Based on the previous results we now summarize and discuss some of the effects observed:

Table 8
MAPE mean values (in %) plus or minus its standard deviation as a function of the number of labelled structures used and of the training scenario for the autoregressive case. Best performance is represented in bold for each case.

Labelled structures N_L^{Train}		$MAPE$ (%)	Mean ± St.	Dev.	
	5	10	20	50	100
Pre-trained models (d = 60%)					
Autoregressive $q = 1$	22.48 ± 6.06	8.08 ± 1.27	**2.63 ± 0.80**	**1.14 ± 0.04**	**1.03 ± 0.06**
MSPA $q = 10$	25.90 ± 4.02	8.81 ± 2.08	2.86 ± 0.80	1.22 ± 0.13	1.04 ± 0.07
MSPA $q = 20$	23.93 ± 6.54	**7.81 ± 0.56**	3.55 ± 0.67	1.34 ± 0.19	1.06 ± 0.08
MSPA $q = 30$	**18.88 ± 4.02**	8.24 ± 1.33	4.95 ± 0.80	1.57 ± 0.21	1.23 ± 0.15
Pre-trained models (d = 70%)					
Autoregressive $q = 1$	24.43 ± 4.08	8.46 ± 1.53	**2.42 ± 0.53**	**1.20 ± 0.06**	**1.14 ± 0.17**
MSPA $q = 10$	21.33 ± 3.85	8.25 ± 0.88	4.34 ± 0.90	1.43 ± 0.12	1.17 ± 0.04
MSPA $q = 20$	20.99 ± 5.43	7.95 ± 1.03	4.62 ± 0.56	1.70 ± 0.17	1.22 ± 0.24
MSPA $q = 30$	**16.56 ±3.16**	**7.52 ± 0.99**	4.79 ± 0.87	2.68 ± 0.42	1.45 ± 0.04
Pre-trained models (d = 80%)					
Autoregressive $q = 1$	26.50 ± 2.58	8.09 ± 3.28	**2.39 ± 0.26**	**1.39 ± 0.20**	**1.07 ± 0.11**
MSPA $q = 10$	17.75 ± 4.53	8.96 ± 0.53	3.53 ± 1.03	1.47 ± 0.11	1.18 ± 0.10
MSPA $q = 20$	14.84 ± 3.22	7.03 ± 1.30	4.68 ± 0.77	2.27 ± 0.41	1.55 ± 0.17
MSPA $q = 30$	**13.32 ± 2.78**	**5.83 ± 0.53**	4.55 ± 0.98	2.88 ± 0.47	2.16 ± 0.18
Pre-trained models (d = 90%)					
Autoregressive $q = 1$	22.69 ± 2.36	8.83 ± 1.61	**3.39 ± 0.40**	**1.25 ± 0.07**	**0.99 ± 0.06**
MSPA $q = 10$	14.68 ± 3.55	8.84 ± 1.73	6.55 ± 0.85	1.44 ± 0.05	1.19 ± 0.06
MSPA $q = 20$	13.92 ± 5.16	**7.15 ± 0.81**	5.22 ± 0.59	1.85 ± 0.25	1.39 ± 0.14
MSPA $q = 30$	**13.24 ± 2.34**	7.45 ± 0.73	4.42 ± 2.04	1.48 ± 0.21	1.18 ± 0.09
Non-pre-trained models					
Autoregressive architecture	28.09 ± 1.63	21.10 ± 1.92	7.52 ± 1.59	1.15 ± 0.09	**0.79 ± 0.09**

Table 9
MAPE mean values (in %) plus or minus its standard deviation as a function of the number of labelled structures used and of the training scenario for the autoregressive case, with or without freezing pre-trained layers. Best performance is represented in bold for each case.

Labelled structures N_L^{Train}		$MAPE$ (%)	Mean ± St.	Dev.	
	5	10	20	50	100
Pre-trained models (d = 60%)					
Autoregressive - Freeze layers	22.48 ± 6.06	8.08 ± 1.27	**2.63 ± 0.80**	**1.14 ± 0.04**	**1.03 ± 0.06**
Autoregressive - Unfreeze layers	**21.13 ± 4.87**	**6.52 ± 0.56**	3.36 ± 0.80	**1.13 ± 0.08**	**1.05 ± 0.06**
Pre-trained models (d = 70%)					
Autoregressive - Freeze layers	**24.43 ± 4.08**	**8.46 ± 1.53**	**2.42 ± 0.53**	1.20 ± 0.06	1.14 ± 0.17
Autoregressive - Unfreeze layers	24.75 ± 3.64	8.93 ± 1.93	3.20 ± 1.18	**1.14 ± 0.12**	**1.01 ± 0.09**
Pre-trained models (d = 80%)					
Autoregressive - Freeze layers	**26.50 ± 2.58**	8.09 ± 3.28	**2.39 ± 0.26**	1.39 ± 0.20	1.07 ± 0.11
Autoregressive - Unfreeze layers	27.04 ± 4.74	**7.81 ± 1.63**	2.99 ± 0.85	**1.16 ± 0.08**	**1.02 ± 0.08**
Pre-trained models (d = 90%)					
Autoregressive - Freeze Layers	22.69 ± 2.36	8.83 ± 1.61	**3.39 ± 0.40**	1.25 ± 0.07	0.99 ± 0.06
Autoregressive - Unfreeze Layers	**22.41 ± 3.05**	**8.74 ± 2.65**	4.37 ± 1.29	**1.16 ± 0.10**	**0.85 ± 0.12**
Non-pre-trained models					
Autoregressive architecture	28.09 ± 1.63	21.10 ± 1.92	7.52 ± 1.59	1.15 ± 0.09	**0.79 ± 0.09**

- *Number of unlabelled samples for pre-training:* Results obtained confirmed that the number of pre-training samples matters. They have shown that pre-training does not always improve predictive performance when the number of pre-training samples is not sufficient, and may even decrease predictive performance (*i.e. negative transfer*). Nevertheless, these investigations indicated that as the number of unlabelled samples increases, the pre-training becomes more efficient and allows to have better results than a non-pre-trained model when few labelled structures are available. A research direction to further improve the pre-training process would be to select the available unlabelled sample, with the aim of extracting the most useful features from the data and avoiding over-fitting, in the spirit of deep active learning (Ren et al., 2021). On the application we considered, for example, it could be interesting to implement an adaptive pre-training strategy to select the training samples and remove unnecessary samples (*e.g.* sensor data with very little variation).
- *Pre-text task:* In this work, several pre-training tasks were compared (*i.e.* input signal estimation and prediction of the future outcome of a sequence) in order to identify which one is most appropriate for the considered case study, and by extension for other engineering case studies using time series sensor data. Experiments have shown that autoregressive pre-training tasks outperform the AE model in pre-training, and capture useful representations from the sensor data (*i.e.* temporal dependencies of the input signal) for RUL estimation tasks. Moreover, results showed that increasing the prediction time horizon of autoregressive models in pre-training can improve the predictive performance, notably when few labelled structures are available. In next steps, it would be interesting to explore other pre-text tasks (*e.g.* Contrastive learning, which aims at learning similar or dissimilar representations from source data Jaiswal et al., 2021). Another interesting research direction would be to embed a Bayesian framework to the models in the pre-training phase (*e.g.* Variational Autoencoders Kingma and Welling, 2013), in order to address the random nature of the data such as noise or measurement errors (*i.e.* aleatoric uncertainty). Note that the VAE model, although it has shown promising results in learning meaningful representations from raw unlabelled data (Zhu et al., 2020; Dhariwal et al., 2020), has been studied and implemented in this work but the results were unsatisfactory. This could be due to the stochastic nature of the model during sampling: thus it requires further investigation in the future.

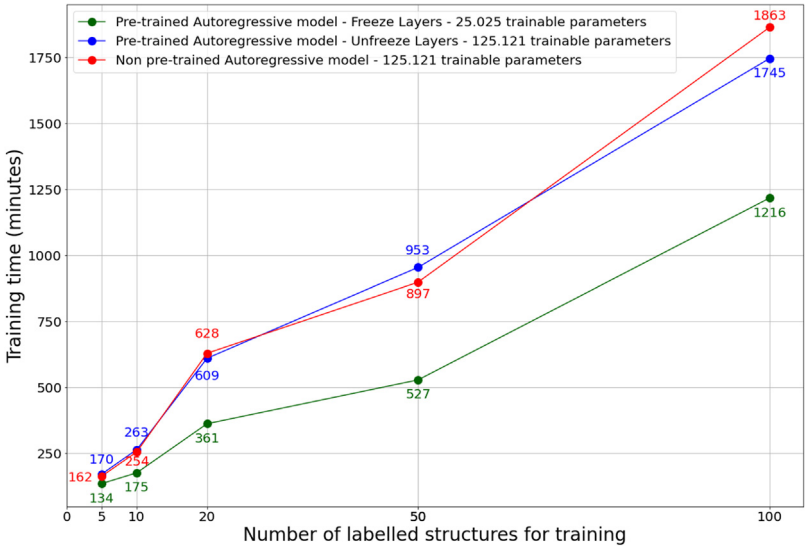

Fig. 16. Mean training time during fine-tuning phase (in minutes).

6. Conclusion

In this paper, a Self-Supervised Learning approach for fatigue damage prognostics problem was proposed and investigated. The approach is based on combination of a pretext and a downstream task. In the pretext task a model is trained using a large number of raw (unlabelled) sensor data with the aim of learning general representations linked to the degradation process. No RUL data is available during this pretext task, only raw sensor data (strains in our case), as the data is obtained only on structures that have not yet reached failure. Then, in a subsequent downstream task, a new model, aimed at predicting the RUL, is adjusted based on the previously pretrained model and based on a limited number of labelled RUL data obtained on structures that have reached failure. Multiple scenarios were investigated within this framework, including varying the pretext task, the models and the dataset properties.

The results obtained showed that self supervised learning is efficient in prognostics and can improve RUL estimation performances especially when only a limited amount of labelled data is available. Overall, these investigations indicate that pre-trained models are able to significantly outperform the respective non-pre-trained counterpart models in the RUL prediction task, while at the same time lowering training computational costs. Accordingly, the proposed approach can significantly reduce the need for labelled data for a given prediction accuracy, or alternatively significantly improve the prediction accuracy for the same amount of (limited) labelled data. Furthermore, the authors of this paper believe that the potential of this learning approach will benefit researchers in a variety of similar engineering fields using sensors or time series data (*e.g.* in energy Jain et al., 2006, or water and environmental engineering Borzooei et al., 2019) and that it can be reused to overcome the lack of available labelled data.

In next steps, it would be interesting to explore other pre-text tasks (*e.g.* contrastive learning, ensemble learning, etc.) or other models (*e.g.* masked autoencoders), adaptive activation functions Jagtap et al., 2020b,a, 2022; Jagtap and Karniadakis, 2022). Furthermore, as uncertainty quantification remains a challenging and ubiquitous task in real-world ML applications (*e.g.* in engineering domains such as transportation engineering Mazloumi et al., 2011 or water and environmental applications Ghiasi et al., 2022), it could be interesting to use Bayesian machine learning models in SSL (*e.g.* Deep Gaussian Processes) to quantify uncertainty in downstream prognostics tasks. Another future work perspective consists in combining strain data with other type of sensor data (*e.g.* ultrasound mappings) in a self-supervised framework in order to further improve prediction results. Future work is also aimed at investigating how the proposed self-supervised prognostics framework behaves on an actual engineering problem involving real-world data.

CRediT authorship contribution statement

Anass Akrim: Conceptualization, Formal analysis, Methodology, Investigation, Visualization, Writing – original draft. **Christian Gogu:** Conceptualization, Investigation, Validation, Supervision, Writing – review & editing. **Rob Vingerhoeds:** Investigation, Validation, Supervision, Writing – review & editing. **Michel Salaün:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data and code for the learning procedure are publicly available on https://github.com/ansak95/DeepSSL.

Acknowledgements

This work was partially funded by the French "Occitanie Region" under the Predict project. This funding is gratefully acknowledged. This work has been partly carried out on the supercomputers PANDO (ISAE-SUPAERO, Toulouse) and Olympe (CALMIP, Toulouse, project n° 21042). Authors are grateful to ISAE-SUPAERO and CALMIP for the hours allocated to this project.

Appendix. Deep gated recurred unit networks

Introduced by Cho et al. (2014) Gated Recurrent Unit, or GRU, are a variant of recurrent neural networks which solves the time-delay problem existing in traditional RNNs. This approach has gained in popularity in recent years due to its relative simplicity (*i.e.* lower complexity and faster computation Rana, 2016), while the same ability to capture the mapping relationships among time series data (Yamak et al., 2019). The structure of the GRU network is shown in Fig. A.17.

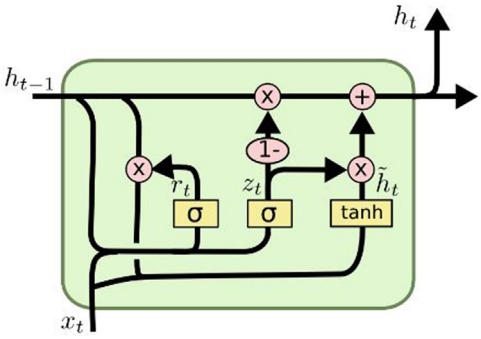

Fig. A.17. GRUs architecture.
Source: In Olah (2015).

The formulas that govern the computation happening in a GRU network are as follow (Cho et al., 2014):

$$\begin{aligned} z_t &= \sigma(W_z X_t + U_z h_{t-1}) \\ r_t &= \sigma(W_f X_t + U_f h_{t-1}) \\ \tilde{h}_t &= tanh(W_{\tilde{h}} X_t + U_{\tilde{h}}[h_{t-1} * r_t]) \\ h_t &= z_t * \tilde{h}_t + (1 - z_t) * h_{t-1} \end{aligned} \tag{A.1}$$

where x_t is the input sequence at time step t, h_t a hidden state, z_t the update gate, r_t the reset gate, $\tilde{h}_t$ a cell state, $\sigma(.)$ represents the sigmoid activation function and $tanh(.)$ the hyperbolic tangent non-linear function, W and U denote the weight matrices which are learned during training. The pink circles represent pointwise operations (*e.g.* addition, multiplication). The idea behind the GRU network is that in each unit, the update gate z_t must select whether the hidden state h_t is to be updated with a new hidden state $\tilde{h}_t$; the reset gate r_t must decide whether the previous hidden state h_{t-1} is ignored. More details can be found in Cho et al. (2014).

References

Adaloglou, N., 2020. Intuitive explanation of skip connections in deep learning. AI Summer.

Akrim, A., Gogu, C., Guillebot de Nerville, T., Strähle, P., Waffa Pagou, B., Salaün, M., Vingerhoeds, R., 2022. A framework for generating large data sets for fatigue damage prognostic problems. In: 2022 IEEE International Conference on Prognostics and Health Management. ICPHM, IEEE, pp. 25–33. http://dx.doi.org/10.1109/ICPHM53196.2022.9815692.

Atamuradov, V., Medjaher, K., Dersin, P., Lamoureux, B., Zerhouni, N., 2017. Prognostics and health management for maintenance practitioners-review, implementation and tools evaluation. Int. J. Progn. Health Manag. 8 (060), 1–31.

Ba, J.L., Kiros, J.R., Hinton, G.E., 2016. Layer normalization. arXiv preprint arXiv:1607.06450.

Bansal, M.A., Sharma, D.R., Kathuria, D.M., 2022. A systematic review on data scarcity problem in deep learning: solution and applications. ACM Comput. Surv. 54 (10s), 1–29.

Baptista, M., Prendinger, H., Henriques, E., 2020. Prognostics in aeronautics with deep recurrent neural networks. In: PHM Society European Conference, Vol. 5. p. 11.

Berthou, T., Duplessis, B., Stabat, P., Rivière, P., Marchio, D., 2019. Urban energy models validation in data scarcity context: Case of the electricity consumption in the French residential sector. In: Building Simulation 2019.

Borzooei, S., Amerlinck, Y., Abolfathi, S., Panepinto, D., Nopens, I., Lorenzi, E., Meucci, L., Zanetti, M.C., 2019. Data scarcity in modelling and simulation of a large-scale WWTP: stop sign or a challenge. J. Water Process Eng. 28, 10–20.

Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al., 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., Hinton, G.E., 2020a. Big self-supervised models are strong semi-supervised learners. Adv. Neural Inf. Process. Syst. 33, 22243–22255.

Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., Sutskever, I., 2020b. Generative pretraining from pixels. In: International Conference on Machine Learning. PMLR, pp. 1691–1703.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Cortes, C., Vapnik, V., 1995. Support-vector networks. Mach. Learn. 20 (3), 273–297.

Damianou, A., Lawrence, N., 2013. Deep gaussian processes. In: Artificial Intelligence and Statistics. pp. 207–215.

Dhariwal, P., Jun, H., Payne, C., Kim, J.W., Radford, A., Sutskever, I., 2020. Jukebox: A generative model for music. arXiv preprint arXiv:2005.00341.

Ding, Y., Zhuang, J., Ding, P., Jia, M., 2022. Self-supervised pretraining via contrast learning for intelligent incipient fault detection of bearings. Reliab. Eng. Syst. Saf. 218, 108126.

Ellefsen, A.L., Bjørlykhaug, E., Æsøy, V., Ushakov, S., Zhang, H., 2019. Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. Reliab. Eng. Syst. Saf. 183, 240–251.

Endo, M., Poston, K.L., Sullivan, E.V., Fei-Fei, L., Pohl, K.M., Adeli, E., 2022. GaitForeMer: Self-supervised pre-training of transformers via human motion forecasting for few-shot gait impairment severity estimation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, pp. 130–139.

Fink, O., Wang, Q., Svensén, M., Dersin, P., Lee, W.-J., Ducoffe, M., 2020. Potential, challenges and future directions for deep learning in prognostics and health management applications. Eng. Appl. Artif. Intell. 92, 103678.

Ghiasi, B., Noori, R., Sheikhian, H., Zeynolabedin, A., Sun, Y., Jun, C., Hamouda, M., Bateni, S.M., Abolfathi, S., 2022. Uncertainty quantification of granular computing-neural network model for prediction of pollutant longitudinal dispersion coefficient in aquatic streams. Sci. Rep. 12 (1), 1–15.

Gorgoglione, A., Castro, A., Chreties, C., Etcheverry, L., 2020. Overcoming data scarcity in earth science. Data 5 (1), 5.

Guo, H., Zhu, H., Wang, J., Prahlad, V., Ho, W.K., Lee, T.H., 2022. Masked self-supervision for remaining useful lifetime prediction in machine tools. arXiv preprint arXiv:2207.01219.

Gutierrez-Torre, A., Berral, J.L., Buchaca, D., Guevara, M., Soret, A., Carrera, D., 2020. Improving maritime traffic emission estimations on missing data with CRBMs. Eng. Appl. Artif. Intell. 94, 103793.

Hahn, T.V., Mechefske, C.K., 2021. Self-supervised learning for tool wear monitoring with a disentangled-variational-autoencoder. Int. J. Hydromechatron. 4 (1), 69–98.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R., 2022. Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16000–16009.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778.

Hewamalage, H., Bergmeir, C., Bandara, K., 2021. Recurrent neural networks for time series forecasting: Current status and future directions. Int. J. Forecast. 37 (1), 388–427.

Hinton, G.E., 2009. Deep belief networks. Scholarpedia 4 (5), 5947.

Ho, T.K., 1995. Random decision forests. In: Proceedings of 3rd International Conference on Document Analysis and Recognition, Vol. 1. IEEE, pp. 278–282.

Hrinchuk, O., Khrulkov, V., Mirvakhabova, L., Orlova, E., Oseledets, I., 2019. Tensorized embedding layers for efficient model compression. arXiv preprint arXiv:1901.10787.

Jadon, S., 2021. COVID-19 detection from scarce chest x-ray image data using few-shot deep learning approach. In: Medical Imaging 2021: Imaging Informatics for Healthcare, Research, and Applications, Vol. 11601. SPIE, pp. 161–170.

Jagtap, A.D., Karniadakis, G.E., 2022. How important are activation functions in regression and classification? A survey, performance comparison, and future directions. arXiv preprint arXiv:2209.02681.

Jagtap, A.D., Kawaguchi, K., Em Karniadakis, G., 2020a. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 476 (2239), 20200334.

Jagtap, A.D., Kawaguchi, K., Karniadakis, G.E., 2020b. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. J. Comput. Phys. 404, 109136.

Jagtap, A.D., Shin, Y., Kawaguchi, K., Karniadakis, G.E., 2022. Deep Kronecker neural networks: A general framework for neural networks with adaptive activation functions. Neurocomputing 468, 165–180.

Jain, S., Shah, R.C., Brunette, W., Borriello, G., Roy, S., 2006. Exploiting mobility for energy efficient data collection in wireless sensor networks. Mob. Netw. Appl. 11 (3), 327–339.

Jaiswal, A., Babu, A.R., Zadeh, M.Z., Banerjee, D., Makedon, F., 2021. A survey on contrastive self-supervised learning. Technologies 9 (1).

Jimenez, J.J.M., Schwartz, S., Vingerhoeds, R., Grabot, B., Salaün, M., 2020. Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics. J. Manuf. Syst. 56, 539–557.

Jing, L., Tian, Y., 2020. Self-supervised visual feature learning with deep neural networks: A survey. IEEE Trans. Pattern Anal. Mach. Intell..

Kandemir, M., 2015. Asymmetric transfer learning with deep gaussian processes. In: International Conference on Machine Learning. PMLR, pp. 730–738.

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

Kingma, D.P., Welling, M., 2013. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

Krokotsch, T., Knaak, M., Gühmann, C., 2022. Improving semi-supervised learning for remaining useful lifetime estimation through self-supervision. Int. J. Progn. Health Manag. 13 (1), http://dx.doi.org/10.36001/ijphm.2022.v13i1.3096.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.

Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., Tang, J., 2021. Self-supervised learning: Generative or contrastive. IEEE Trans. Knowl. Data Eng..

Ma, J., Su, H., Zhao, W.-l., Liu, B., 2018. Predicting the remaining useful life of an aircraft engine using a stacked sparse autoencoder with multilayer self-learning. Complexity 2018.

Mao, D., Lv, C., Shi, J., Zou, Y., Guo, Z., 2010. Research of the military aircraft maintenance support mode based on the prognostics and health management. In: 2010 Prognostics and System Health Management Conference. IEEE, pp. 1–6.

Mazloumi, E., Rose, G., Currie, G., Moridpour, S., 2011. Prediction intervals to account for uncertainties in neural network predictions: Methodology and application in bus travel time prediction. Eng. Appl. Artif. Intell. 24 (3), 534–542.

Nandy, A., Duan, C., Kulik, H.J., 2022. Audacity of huge: overcoming challenges of data scarcity and data quality for machine learning in computational materials discovery. Curr. Opin. Chem. Eng. 36, 100778.

Olah, C., 2015. Understanding LSTM networks. URL colah.github.io/posts/2015-08-Understanding-LSTMs/.

Paris, P., Erdogan, F., 1963. A critical analysis of crack propagation laws. J. Basic Eng. 85 (4), 528–533. http://dx.doi.org/10.1115/1.3656900.

Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y., 2007. Self-taught learning: transfer learning from unlabeled data. In: Proceedings of the 24th International Conference on Machine Learning. pp. 759–766.

Rana, R., 2016. Gated recurrent unit (GRU) for emotion classification from noisy speech. arXiv preprint arXiv:1612.07778.

Rasmussen, C.E., 2003. Gaussian processes in machine learning. In: Summer School on Machine Learning. Springer, pp. 63–71.

Ren, L., Sun, Y., Cui, J., Zhang, L., 2018. Bearing remaining useful life prediction based on deep autoencoder and deep neural networks. J. Manuf. Syst. 48, 71–77.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Gupta, B.B., Chen, X., Wang, X., 2021. A survey of deep active learning. ACM Comput. Surv. 54 (9), 1–40.

Rocchetta, R., Gao, Q., Mavroeidis, D., Petkovic, M., 2022. A robust model selection framework for fault detection and system health monitoring with limited failure examples: Heterogeneous data fusion and formal sensitivity bounds. Eng. Appl. Artif. Intell. 114, 105140.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. Nature 323 (6088), 533–536.

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

Shao-feng, X., Yun-fei, E., Xiao-ling, L., Yu-dong, L., Yi-qiang, C., 2013. Development and application of prognostics and health management technology. In: Proceedings of the 20th IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits. IPFA, IEEE, pp. 3–7.

Shurrab, S., Duwairi, R., 2022. Self-supervised learning methods and applications in medical imaging analysis: A survey. PeerJ Comput. Sci. 8, e1045.

Sun, C., Ma, M., Zhao, Z., Tian, S., Yan, R., Chen, X., 2018. Deep transfer learning based on sparse autoencoder for remaining useful life prediction of tool in manufacturing. IEEE Trans. Ind. Inform. 15 (4), 2416–2425.

Theissler, A., Pérez-Velázquez, J., Kettelgerdes, M., Elger, G., 2021. Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry. Reliab. Eng. Syst. Saf. 215, 107864.

Torrey, L., Shavlik, J., 2010. Transfer learning. In: Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques. IGI Global, pp. 242–264.

Tsui, K.L., Chen, N., Zhou, Q., Hai, Y., Wang, W., 2015. Prognostics and health management: A review on data driven approaches. Math. Probl. Eng. 2015.

Voulodimos, A., Doulamis, N., Bebis, G., Stathaki, T., 2018. Recent developments in deep learning for engineering applications. Comput. Intell. Neurosci. 2018.

Wen, Z., Liu, Y., 2011. Applications of Prognostics and Health Management in aviation industry. In: 2011 Prognostics and System Health Managment Confernece. IEEE, pp. 1–5.

Yamak, P.T., Yujian, L., Gadosey, P.K., 2019. A comparison between arima, lstm, and gru for time series forecasting. In: Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence. pp. 49–55.

Yengera, G., Mutter, D., Marescaux, J., Padoy, N., 2018. Less is more: Surgical phase recognition with less annotations through self-supervised pre-training of CNN-LSTM networks. arXiv preprint arXiv:1805.08569.

Yoon, A.S., Lee, T., Lim, Y., Jung, D., Kang, P., Kim, D., Park, K., Choi, Y., 2017. Semi-supervised learning with deep generative models for asset failure prediction. arXiv preprint arXiv:1709.00845.

Yu, J., Yin, H., Xia, X., Chen, T., Li, J., Huang, Z., 2022. Self-supervised learning for recommender systems: A survey. arXiv preprint arXiv:2203.15876.

Yue, B., Fu, J., Liang, J., 2018. Residual recurrent neural networks for learning sequential representations. Information 9 (3), 56.

Zhu, Q.-X., Hou, K.-R., Chen, Z.-S., Gao, Z.-S., Xu, Y., He, Y.-L., 2021. Novel virtual sample generation using conditional GAN for developing soft sensor with small data. Eng. Appl. Artif. Intell. 106, 104497.

Zhu, Y., Min, M.R., Kadav, A., Graf, H.P., 2020. S3vae: Self-supervised sequential vae for representation disentanglement and data generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6538–6547.