A Coevolutionary Optimization Approach with Deep Sparse Autoencoder for the Extraction of Equipment Degradation Indicators

Ali Eftekhari Milani
Energy Department, Politecnico di Milano, Via Lambruschini 4, 20156, Milan, Italy
E-mail: ali.eftekhari@polimi.it

Federico Antonello
Energy Department, Politecnico di Milano, Via Lambruschini 4, 20156, Milan, Italy
E-mail: federico.antonello@polimi.it

Piero Baraldi
Energy Department, Politecnico di Milano, Via Lambruschini 4, 20156, Milan, Italy
E-mail: piero.baraldi@polimi.it

Enrico Zio
Energy Department, Politecnico di Milano, Via Lambruschini 4, 20156, Milan, Italy
MINES ParisTech, PSL Research University, CRC, Sophia Antipolis, France
Eminent Scholar, Dep. of Nuclear Engineering, College of Engineering, Kyung Hee University, Republic of Korea
E-mail: enrico.zio@polimi.it

We present a coevolutionary optimization approach for the automatic and unsupervised extraction of industrial component degradation indicators from a set of signals collected during operation. It embeds a deep sparse autoencoder (SAE) for the extraction of the degradation indicators, into a multi-objective coevolutionary optimization algorithm, which maximizes the SAE's performance by optimizing its architecture and hyperparameters. The effectiveness of the proposed approach is shown by its application to a synthetic dataset, which mimics the operation of a degrading component in an environment affected by seasonal changes.

Keywords: Prognostics and health management (PHM), degradation indicator, coevolutionary optimization algorithm, sparse autoencoder.

1. Introduction

The identification of the degradation state of industrial components is at the basis of condition-based maintenance (Kim, et al., 2012); (Marton, et al., 2013); (Termite, et al., 2019). Typically, degradation indicators are hand-crafted features, designed by experts and estimated by applying signal processing techniques and supervised machine learning algorithms in the time, frequency, and time-frequency domains (Das, 2001); (Feng, et al., 2013); (Liao, 2014); (Yan, et al., 2014); (Zhao, et al., 2017). Wavelet-based wrapper feature selection approaches have been used to build degradation indicators in (Yan, et al., 2014), binary differential evolution algorithm with k-nearest neighbor classifiers in (Baraldi, et al., 2016), greedy search methods in (Rauber, et al., 2015), and LSSVM[a] in (Lu, et al., 2019). These methods typically require manual parameter setting, are based on trial and error approaches, which can be time-consuming and computation-intensive, and are heavily dependent on expert knowledge (Miikkulainen, et al., 2018).

In (Yang, et al., 2018), a low-dimensional latent representation of the input data is extracted using deep sparse autoencoder and the Mann-Kendall (M-K) monotonicity test (Pohlert, 2016) is applied to select the most monotonic feature as degradation indicator. The method is shown able to extract satisfactory degradation indicators from different types of signals, in an unsupervised manner, and without requiring the use of expert knowledge. However, it requires tuning the deep SAE's architecture and its hyperparameters, which is done by the following iterative trial and error procedure: firstly, an architecture is tried, followed by a grid search to find the optimal hyperparameters; then, the architecture is modified and another grid search is performed on the new architecture, and the process is repeated until a satisfactory solution is found.

[a] Least-Squares Support Vector Machine

Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference
Edited by Piero Baraldi, Francesco Di Maio and Enrico Zio
Published by Research Publishing, Singapore.
ISBN: 978-981-14-8593-0; doi:10.3850/978-981-14-8593-0

This approach can be time-consuming, computation-intensive, and does not guarantee finding globally optimal solutions (Gruau, et al., 1996).

Numerous frameworks have been suggested and implemented to optimally set the SAE architectures and hyperparameters (O. Stanley, et al., 2002); (Garcia-Pedrajas, et al., 2003); (Lu, et al., 2019). Evolution-based optimization algorithms (also referred to as neuroevolutionary algorithms) evolve a population of network architectures and/or parameters which are encoded into strings of numbers called chromosomes (Likothanassis, et al., 1997). Recently, coevolutionary algorithms, which comprise multiple genetically isolated populations and species that coevolve together, have emerged among neuroevolutionary algorithms, since they are compatible with the modular nature of neural networks and are effective in preserving diversity during the search for the optimal solutions (Garcia-Pedrajas, et al., 2003).

In this work, we present an approach which embeds a deep sparse autoencoder for extracting degradation indicators from vibrational sensor data of industrial components, into a cooperative coevolutionary algorithm (Popovici, et al., 2012), which optimizes the SAE's architecture and hyperparameters setting.

The main contribution is that the proposed algorithm fully automatizes the procedure of extracting degradation indicators using deep sparse autoencoders.

The effectiveness of the proposed method is demonstrated by its application to a case study, which comprises the synthetic dataset used in (Yang, et al., 2018).

The remainder of the paper is organized as follows: Section 2 introduces sparse autoencoders. In section 3, the proposed method is presented. The case study, the dataset, and the obtained results are discussed in section 4. Finally, section 5 draws some conclusions.

2. Sparse Autoencoders

An autoencoder (AE) is a neural network composed of an encoder and a decoder, trained to replicate its input signals. It allows reducing the input data dimensionality by providing a low-dimensional representation of them (Salakhutdinov, 2006); (Yang, et al., 2018). Let $\boldsymbol{X} = \{\boldsymbol{x}_i, \mathrm{i} = 1, \cdots, \mathrm{N}\}, \boldsymbol{x}_i \in \mathbb{R}^K$ be an unlabeled dataset containing a run-to-failure degradation trajectory of a component for which the measurements of K signals at N time instances are available. The encoder maps the input vector $\boldsymbol{x}_i$ to a latent representation $\boldsymbol{z}_i \in \mathbb{R}^{K_F}$, whereas the decoder reconstructs the input data from the extracted features $\boldsymbol{z}_i$. A sparse autoencoder (SAE) is a variant of an AE with a regularization term added to the cost function to encourage the neurons in the hidden layers to be inactive most of the time, making the AE sparse (Le, 2013). This is because it has been demonstrated that the extraction of discriminative features $z_j^{(1)}, j = 1, 2, \dots, K_1$ is favored when the hidden neurons are constrained to be inactive most of the time, i.e. by requiring the sparsity of the AE (NG., 2011).

SAEs are typically organized in multi-layered deep architectures called Stacked-SAEs (S-SAEs) (Fig. 1). Their training process is composed of two steps: layer-wise pretraining and fine-tuning. Pretraining consists of training a stack of basic single-layered AEs. For example, the first basic AE (bottom in Fig. 1a) is trained to transform the input $\boldsymbol{x}_i$ to the feature $\boldsymbol{z}_i^{(1)} = [z_1^{(1)}(i), z_2^{(1)}(i), \dots, z_{K_1}^{(1)}(i)]$:

$$\boldsymbol{z}_1^{(1)} = f(\boldsymbol{W}_1 \boldsymbol{x}_1 + \boldsymbol{b}_1) \tag{1}$$

where $f, \boldsymbol{W}_1, \boldsymbol{b}_1$ are respectively the encoder activation function, weight matrix, and bias vector. After obtaining the features $\boldsymbol{z}_i^{(1)}$, they are fed into the next basic SAE as input data to transform $\boldsymbol{z}_i^{(1)}$ to the feature vector $\boldsymbol{z}_i^{(2)}$. This procedure is repeated until the innermost basic SAE is pretrained. Subsequently, the basic SAEs are stacked together to create a deep autoencoder, which is, then, fine-tuned (Rumelhart, et al., 1986).

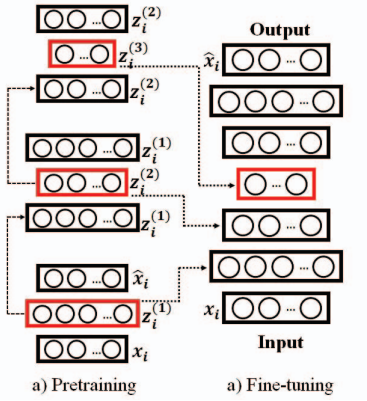

Fig. 1. SAE training

$$\boldsymbol{z}_1^{(1)} = f(\boldsymbol{W}_1\boldsymbol{x}_1 + \boldsymbol{b}_1) \quad (1)$$

The training of a generic basic SAE aims at minimizing the cost function:

$$C = R_{error} + \beta R_{sparse} + \lambda R_{L2} \quad (2)$$

where R_{error}, R_{sparse}, and R_{L2} are the reconstruction error, the sparsity regularization, and the $L2$ regularization terms respectively. β and λ are coefficients indicating the relative importance of the terms in the cost function. The reconstruction error, R_{error}, which is the mean squared error (MSE), quantifies the difference between the reconstructed signal $\hat{\boldsymbol{x}}_i$ and the input vector $\boldsymbol{x}_i$:

$$R_{error} = \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2 \quad (3)$$

The sparsity regularization term R_{sparse} is calculated using the Kullback-Leibler (KL) divergence function to measure the degree of sparsity of the SAE:

$$R_{sparse} = \sum_{i=1}^{K_1} KL\big(p\|\hat{p}_j\big) = \sum_{i=1}^{K_1}\left[p\,log\frac{p}{\hat{p}_j} + (1-p)\,log\frac{1-p}{1-\hat{p}_j}\right] \quad (4)$$

where $\hat{p}_j$ is the average activation of the j^{th} neuron of the SAE hidden layer, evaluated over all the training patterns $\boldsymbol{x}_i, i = 1, \dots, N$:

$$\hat{p}_j = \frac{1}{N}\sum_{i=1}^{N} z_j(i)\ , j = 1, \dots, K \quad (5)$$

and $z_j(i),\ j = 1, 2, \dots, K$ is the j^{th} element of the extracted feature vector $\boldsymbol{z}(i)$ with K the number of neurons in the hidden layer. The KL function is zero when all $\hat{p}_j$ are equal to p and increases when they diverge.

The weight regularization term R_{L2} is introduced in Eq. (2) to prevent the increase of the weight values, which can lead to R_{sparse} becoming small, without actual sparsity of the network (Olshausen, et al., 1997). It is defined as:

$$R_{L2} = \frac{1}{2}\|\boldsymbol{W}\|^2 \quad (6)$$

where $\boldsymbol{W}$ is the SAE weight matrix.

2.1. Degradation indicator identification through Deep SAEs

According to (Yang, et al., 2018), deep SAEs can be used to extract degradation indicators from run-to-failure trajectories. The training of these networks consists of layer-wise pretraining (i.e. training each hidden layer with the data of the previous layer as the input and the output) and fine-tuning.

Subsequently, the decoder is discarded, and the encoder is used for obtaining a set of indicators, which are the activation values of the innermost layer neurons. Finally, the obtained indicators are analysed by Mann-Kendall (MK) monotonicity test (Pohlert, 2016), and a monotonicity coefficient ρ_{MK} is calculated for each of them. For a generic signal $F(t)$, ρ_{MK} is:

$$\rho_{MK} = \begin{cases} \dfrac{V-1}{\sigma} & if \quad V > 0 \\ 0 & if \quad V = 0 \\ \dfrac{V+1}{\sigma} & if \quad V < 0 \end{cases} \quad (7)$$

where $V = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} sign(F(j) - F(i))$ and $\sigma = \sqrt{N(N-1)(2N+5)/18}$ with N indicating the number of training patterns. The larger the value of ρ_{MK}, the more monotonic is the extracted feature and the more likely is the feature to be a good degradation indicator.

3. Proposed Approach

Deep SAEs, and neural networks in general, are composed of modules (i.e., layers, nodes, weights, etc.) cooperating to improve the capability of the overall network (Popovici, et al., 2012). The network is, then, evaluated by assessing its capability of learning to solve a given problem at hand. In this light, cooperative coevolutionary frameworks are the natural solution for Deep SAEs architecture and hyperparameters optimization (Garcia-Pedrajas, et al., 2003); (Popovici, et al., 2012). Specifically, a Deep SAE can be broken down into its constituents (layers, computation modules, weights, etc.) and each constituent can be evolved inside a dedicated genetically isolated sub-population (Garcia-Pedrajas, et al., 2003).

In this context, we propose a cooperative coevolutionary optimization approach, which automatically optimizes the architecture and hyperparameters of a deep SAE, and, at the same time, trains the Deep SAE to identify degradation indicators from run-to-failure signal trajectories. The proposed framework comprises a four-hidden-layer deep SAE, embedded in a two-population cooperative coevolutionary algorithm composed of a population of individuals, which describe deep SAEs architectures and the hyperparameters related to the whole network

rather than to the layers (e.g. optimizer, learning rate, etc.), and a population of layers, which is divided into four subpopulations, each one dedicated to a single hidden layer of the SAE. Each individual of the first population encodes a deep SAE, built by stacking the layers together (one from each of the layer species) to build the encoder, and a symmetric decoder (based on (Garcia-Pedrajas, et al., 2003); (Miikkulainen, et al., 2018). Thus, the proposed algorithm evolves the individuals of the two populations by evaluating the performances of deep SAEs built by decoding the chromosomes. Each deep SAE is trained according to (Yang, et al., 2018), the degradation indicator is extracted and evaluated by MK test, and the network and layer individuals are evolved. Finally, when the maximum number of generations is reached, the first front of the final population is ordered according to the monotonicity of the extracted features and the individual with the largest monotonicity is selected as the best deep SAE. The detailed specifications of the framework and the two populations are presented in the following sections.

3.1. Population of networks

The network population represents a population of architectures and hyperparameters of four-layered deep SAEs, whose layers are evolved in the layer population. Each individual of the population is a chromosome, i.e. a string of binary numbers divided into several genes. The first part of the chromosome contains genes dedicated to the network-specific hyperparameters.

The second part is composed of four genes, each one dedicated to an SAE layer which points to one specific layer chromosome in the corresponding species (Fig. 2 bottom). Hence, it functions as a blueprint for assembling the SAEs (Miikkulainen, et al., 2018); (Lu, et al., 2019). The network-specific hyperparameters (which make up the first part of the network chromosomes) and their ranges are reported in Table 1.

Table 1. Network-specific hyperparameters

Parameter	Range
Batch Size	$1, 2, 4, 8, 16, 32, 64, 128$
Optimizer	*Adam, Nadam, RMSProp, Adadelta*
Learning Rate	$\log(10^{-8}, 10)$
Learning Rate Decay	$\log(10^{-8}, 1)$
Momentum	$\log(0.9, 0.9999)$
Init. Random Seed	$0, 1, \dots, 63$

The final four genes contain the indexes of the selected layer chromosomes in the four layer species. For a more effective participation of the layers in the networks, the network population is required to have more individuals compared to the layer population.

The optimization of the network individuals has two objectives: ρ_{MK} and the validation loss of the training. Notice that including the validation loss as one of the objectives is required, since it has been observed in the experiments that it is possible to have a large value of ρ_{MK} without a proper training of the SAE, thus, obtaining degradation indicators that are not robust. Thus, a multi-objective steady-state scheme is adopted for their evolution towards optimality (Garcia-Pedrajas, et al., 2003); (Miikkulainen, et al., 2018). For this, we adopt the NSGA-II algorithm (Deb, et al., 2002), which ranks the individuals in the network population based on non-dominated fronts of solutions and has been shown superior to other optimization schemes in the literature (Lu, et al., 2019). In each generation of the proposed algorithm, two individuals are selected randomly from the top 70% of the population and their hyperparameters are mutated (Garcia-Pedrajas, et al., 2003). The severity of mutation (i.e. the number of hyperparameters mutated) is an exponential function of the individual's rank (Garcia-Pedrajas, et al., 2003); (García-Pedrajas, et al., 2009). Then, two individuals are selected from the bottom 30% and go through a uniform layer-level crossover, followed by a structural and parametric mutation whose probability for the i^{th} network individual is:

$$p_i^{mut} = \sqrt{\frac{r_i - r_{min}}{r_{max} - r_{min}}} \tag{8}$$

where r_i is the corresponding rank (O. Stanley, et al., 2002); (Garcia-Pedrajas, et al., 2003). Finally, the four worst networks are deleted and replaced by the offspring. Structural changes (mutation and crossover) are limited to the worst 30% of the network population, because they are more aggressive, and applying them to already good solutions tends to destabilize the algorithm (Garcia-Pedrajas, et al., 2003).

3.2. Population of layers

The population of layers is composed of four species, each representing one of the layers of the SAE. A layer chromosome is a string of binary numbers which is divided into several genes, each representing one layer-specific hyperparameter (Fig. 2 top). The layer-specific hyperparameters and their ranges are reported in Table 2:

Table 2. Layer-specific hyperparameters

Parameter	Range
L2 coefficient	$(10^8, 1)$
Sparcity proportion	$(0.001, 0.01)$
Sparcity coefficient	$(10^{-3}, 10)$
Activation function	$sigmoid, tanh, softsign, selu$
Initialization scheme	*Glorot Normal, He Normal, Glorot Uniform, He Uniform*
Number of neurons in layer 1	(3,10)
Number of neurons in layer 2	(14,28)
Number of neurons in layer 3	(20,90)
Number of neurons in layer 4	(50,120)

An index is assigned for each individual in each of the four layer species. For a layer individual, the index is fixed as long as it does not go through mutation.

The populations are initialized in a way that all the layers participate in the networks in the initial population. The layers are, then, evaluated according to a credit assignment procedure (Garcia-Pedrajas, et al., 2003); (Miikkulainen, et al., 2018), which assigns for each layer the rank of the fittest network containing that layer. If a layer does not participate in a network at a specific generation, it holds the last fitness assigned to it in previous generations. A flip-bit mutation scheme is used, where the severity of the mutation (i.e. the number of genes mutated) is an exponential function of the fitness of the layer (Garcia-Pedrajas, et al., 2003); (García-Pedrajas, et al., 2009). Crossover is not used due to its disadvantages (Angeline, et al., 1994). A steady-state evolution scheme is adopted with 70% elitism (Garcia-Pedrajas, et al., 2003).

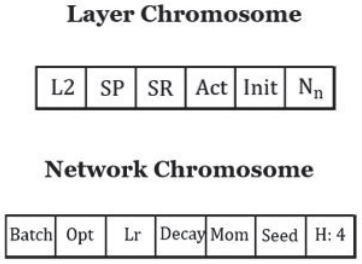

Fig. 2. Chromosome structure

At each generation, the bottom 30% of the layers are deleted and replaced by mutated individuals, randomly selected from the top 70%. The pseudo-code of the evolution process is reported in Fig. 3.

4. Case Study

In order to assess the effectiveness of degradation indicators extracted from signal data, (Yang, et al., 2018) proposed a synthetic dataset, which mimics the behavior of an industrial component characterized by a linear degradation process:

$$D(t) = \alpha t\,, t = 1, 2, \dots, 200 \tag{9}$$

where $\alpha = 0.025$ is the intrinsic degradation rate. They assume that $D(t)$ is not directly measurable, and that 10 signals, whose values are measured by sensors, are influenced by the component degradation $D(t)$, the operational and environmental conditions $C(t)$, and a process noise $w(t)$:

$$\begin{cases} C(t) = sin\left(\frac{1}{25}\pi t\right) + w_c(t) \\ w_c(t) \sim N(\mu = 0, \sigma = 0.2) \end{cases} \tag{10}$$

$$w(t) \sim N(0, 0.1) \tag{11}$$

The *sin* function describes the periodic trend of the seasonal effects and $w_c(t)$ represents the stochasticity of the environmental changes. Fig. 4 depicts the trajectories of these three factors. Subsequently, the three above-mentioned factors are non-linearly combined to make up 10 signals $s_i(t), i = 1, 2, \dots, 10$, which represent the data collected by 10 sensors during the operation of the industrial component:

$$\begin{cases} s_1(t) = sin\big(1/2\,D(t) + 2C(t)\big) \\ s_2(t) = sin\big(D(t) + \tan\big(C(t)\big)\big) \\ s_3(t) = tan\big(1/2\,C(t) - 2/3\,w(t)\big) \\ s_4(t) = cos^3\big(w(t)\big) sin\big(C(t)\big) \\ s_5(t) = cos\big(D(t) - 3\tan\big(w(t)\big)\big) \\ s_6(t) = sin\big(D(t) + w(t) - tan^2(C(t) + 1)\big) \\ s_7(t) = tan\big(1/6\,D(t)C(t)\big) + 1/2\,w(t) \\ s_8(t) = cos^2\big(7/10\,D(t) + 2C(t)\big) \\ s_9(t) = cos\big(D(t) + 2\tan\big(C(t)\big) - \tan\big(1/2\,w(t)\big)\big) \\ s_{10}(t) = tan\big(1/4\,w(t)\big) - 1/4\,cos\big(1/3\,D(t)\sin\big(C(t)\big)\big) \end{cases} \tag{12}$$

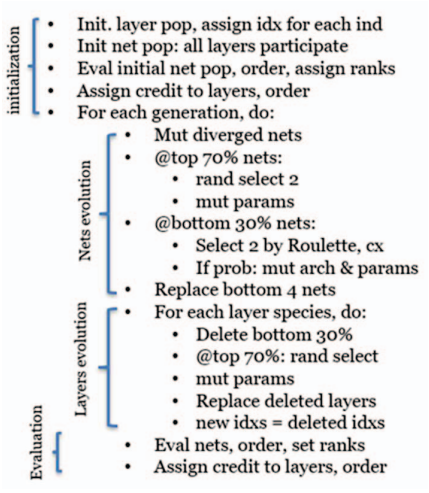

Fig. 3. Algorithm pseudo-code

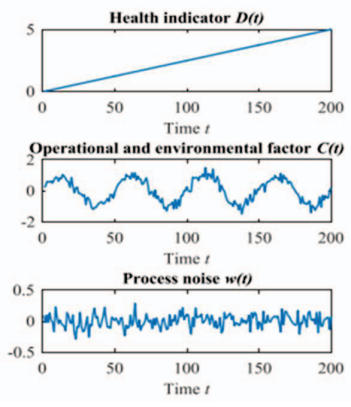

Fig. 4. Time evolution of the three factors used for generating the 10 signals of the synthetic dataset

Fig. 5 shows an example of the evolution of the 10 signals during a run-to-failure trajectory. The proposed coevolutionary optimization approach is applied to the dataset using a layer population size of $2^6 = 64$ and a network population size of 80 individuals. The algorithm is run for 50 generations and the first front of the final network population is ordered according to the monotonicity. The fittest network in the coevolutionary approach produces an indicator with monotonicity of $\rho_{MK} = 15.5647$, which is comparable with the largest value obtained in (Yang, et al., 2018). This indicates that the obtained indicator has a large correlation with $D(t)$, which is the linear degradation factor of the input signals. The obtained degradation indicator, the network architecture and hyperparameters are reported in Fig. 6, Table 3, and Table 4, respectively. Furthermore, it is interesting to observe that the obtained architecture and parameter setting is difficult to find by trial and error or grid search, because using these methods, the parameter search grid needs to be necessarily coarse due to the computational burden.

Table 3. Result: Network-specific hyperparameters

Parameter	Value
Batch size	64
Optimizer	Adam
Learning rate	0.0044
Learning rate decay	2.27×10^{-5}
Momentum	0.9147
Initialization random seed	13

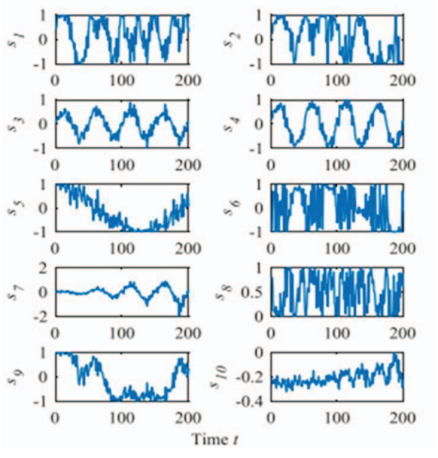

Fig. 5. Time evolution of the 10 simulated signals during the considered run-to-failure trajectory

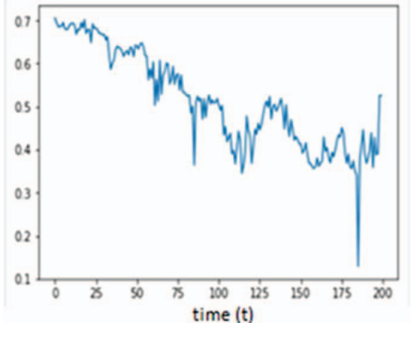

Fig. 6. Indicator obtained by the presented approach

Table 4. Result: Layer-specific hyperparameters

Param.	Layer 1	Layer 2	Layer 3	Layer 4
L2	0.0031	0.0074	4.32 $\times 10^{-6}$	9.64 $\times 10^{-5}$
SP	0.0957	0.0317	0.0569	0.0111
SR	0.7979	0.0309	0.0117	0.0309
Act.	Tanh	Tanh	Softsign	SeLU
Init.	Glorot_N	He_U	Glorot_N	Glorot_N
n_neuron	10	26	40	120

5. Conclusion and Future Work

A coevolutionary optimization approach has been developed for the automatic and unsupervised extraction of a degradation indicator from sensor data measured on a degrading industrial component. The proposed approach automatically sets the architecture and hyperparameters of an SAE used for extracting the degradation indicator from the raw data. It has been shown able to effectively search the space of possible SAE architectures and hyperparameters, and automatically find an SAE that is on par, in terms of performance, with the SAEs obtained by applying computation-intensive trial and error and grid search approaches. Future studies will be devoted to applying the method to real case studies and to extending the method to consider the optimization of other parameters of the AE architecture, such as the number of hidden layers.

Acknowledgement

The participation of Piero Baraldi, Ali Eftekhari Milani, and Enrico Zio has been funded by "Smart Maintenance of Industrial Plants and Civil Structures by 4.0 Monitoring Technologies and Prognostic Approaches – MAC4PRO", sponsored by the call BRIC-2018 of the National Institute for Insurance against Accidents at Work – INAIL. We would also like to acknowledge Dr. Zhe Yang for providing us the dataset used in this study and for the relative insights about the data.

References

Angeline P. J., Saunders G. M. and Pollack J. B. An evolutionary algorithm that constructs recurrent neural networks [Journal] // IEEE Transactions on Neural Networks. - 1994. - Vol. 5. - pp. 54-65.

Baraldi P. [et al.] Hierarchical k-nearest neighbours classification and binary differential evolution for fault diagnostics of automotive bearings operating under variable conditions [Journal] // Engineering Applications of Artificial Intelligence. - 2016. - 56. - pp. 1-13.

Chawla Nitesh V. [et al.] SMOTE: Synthetic Minority Over-sampling Technique [Journal] // Journal of Artificial Intelligence Research. - 2002. - Vol. 16. - pp. 321–357.

Das Sanmay Filters, Wrappers and a Boosting-Based Hybrid for Feature Selection [Conference] // ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning. - 2001.

Deb Kalyanmoy [et al.] A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II [Journal] // IEEE Transactions on Evolutionary Computation. - 2002. - 2 : Vol. 6. - pp. 182-197.

Dozat Timothy Incorporating Nesterov Momentum into Adam [Conference] // International Conference on Learning Representations (ICLR). - 2016.

Feng Z., Liang M. and Chu F. Recent advances in time–frequency analysis methods for machinery fault diagnosis: A review with application examples [Journal] // Mechanical Systems and Signal Processing. - 2013. - 33 : Vol. 1. - pp. 165-205.

Garcia-Pedrajas Nicolas, Hervas-Martinez Cesar and Munoz-Perez Jose COVNET: A Cooperative Coevolutionary Model for Evolving Artificial Neural Networks [Journal] // IEEE TRANSACTIONS ON NEURAL NETWORKS. - 2003. - 3 : Vol. 14. - pp. 575-596.

García-Pedrajas Nicolás, Hervás-Martínez César and Ortiz-Boyer Domingo Cooperative Coevolution of Artificial Neural Network Ensembles for Pattern Classification [Journal] // IEEE Transactions on Evolutionary Computation. - 2009. - 3 : Vol. 9. - pp. 271-302.

Gruau Frederic, Whitley Darrell and Pyeatt Larry A Comparison between Cellular Encoding and Direct Encoding for Genetic Neural Networks [Journal] // Genetic Programming 1996: Proceedings of the First Annual Conference. - 1996. - pp. 81-89.

Kim Hack-Eun [et al.] Bearing fault prognosis based on health state probability estimation [Journal]. - [s.l.] : Elsevier, 2012. - 5 : Vol. 39.

Le Quoc V. Building high-level features using large scale unsupervised learning [Conference] // IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). - 2013.

Liao Linxia Discovering Prognostic Features Using Genetic Programming in Remaining Useful Life Prediction [Journal] // IEEE Transactions on Industrial Electronics. - 2014. - 5 : Vol. 61. - pp. 2464-2472.

Likothanassis S. D., Georgopoulos E. and Fotakis D. Optimizing the structure of neural networks using evolution techniques [Journal] // Transactions on Information and Communications Technologies. - 1997. - Vol. 15. - pp. 157-168.

Lu C. [et al.] Degradation trend estimation of slewing bearing based on LSSVM mode [Journal] // Mechanical Systems and Signal Processing. - 2019. - Vol. 76. - pp. 353,366.

Lu Zhichao [et al.] NSGA-Net: Neural Architecture Search using Multi-Objective Genetic Algorithm [Conference] // GECCO '19 Proceedings of the Genetic and Evolutionary Computation Conference. - 2019.

Marton Isabel [et al.] Application of Data Driven Methods for Condition Monitoring Maintenance [Journal]. - [s.l.] : Chemical Engineering Transactions, 2013. - Vol. 33.

Miikkulainen Risto [et al.] Evolving Deep Neural Networks [Journal] // Elsevier. - 2018.

NG. Andrew Sparse autoencoder [Book Section] // CS294A Lecture notes. - 2011.

O. Stanley Kenneth and Miikkulainen Risto Evolving Neural Networks through Augmenting Topologies [Journal] // Evolutionary Computation. - 2002. - 2 : Vol. 10. - pp. 99-127.

Olshausen Bruno A. and Field David J. Sparse coding with an overcomplete basis set: A strategy employed by V1? [Journal] // Vision Research. - 1997. - 23 : Vol. 37. - pp. 3311-3325.

Pohlert T. Non-parametric trend tests and change-point detection [Journal] // CC BY-ND. - 2016.

Popovici Elena [et al.] Coevolutionary Principles [Journal] // Handbook of Natural Computing. Springer, Berlin, Heidelberg. - 2012. - pp. 987-1033.

Rauber T. W., de Assis Boldt F. and Varejão F. M. Heterogeneous feature models and feature selection applied to bearing fault diagnosis [Journal] // IEEE Transactions on Industrial Electronics. - 2015. - 1 : Vol. 62. - pp. 637-646.

Rumelhart David E., Hinton Geoffrey E. and Williams Ronald J. Learning representations by back-propagating errors [Journal] // Nature. - 1986. - 6088 : Vol. 323. - pp. 533-536.

Salakhutdinov G. E. Hinton and R. R. Reducing the dimensionality [Journal] // Science. - 2006. - 5786 : Vol. 313. - pp. 504-507.

Saxena A. et. al. Metrics for evaluating performance of prognostic techniques [Conference] // International Conference on Prognostics and Health Management, IEEE. - 2008.

Termite Maria Rosaria [et al.] A Never-Ending Learning Method for Fault Diagnostics in Energy Systems Operating in Evolving Environments [Journal]. - [s.l.] : Energies, 2019. - 4802 : Vol. 12.

Yan R., Gao R. X. and Chen X. Wavelets for fault diagnosis of rotary machines: A review with applications [Journal] // Signal Processing. - 2014. - 96. - pp. 1-15.

Yang Zhe, Baraldi Piero and Zio Enrico Automatic Extraction of a Health Indicator from Vibrational Data by Sparse Autoencoders [Conference] // 3rd International Conference on System Reliability and Safety (ICSRS). - 2018.

Zhao J. [et al.] Health indicator selection and health assessment of rolling element bearing