Periodica Polytechnica Mechanical Engineering, 66(3), pp. 244–252, 2022

Machine Learning Approach for Degradation Path Prediction Using Different Models and Architectures of Artificial Neural Networks

Basheer Shaheen[1*], István Németh[1]

[1] Department of Manufacturing Science and Engineering, Faculty of Mechanical Engineering, Budapest University of Technology and Economics, Műegyetem rkp. 3, 1111 Budapest, Hungary
[*] Corresponding author, e-mail: shaheen.basheer@gpk.bme.hu

Received: 13 March 2022, Accepted: 09 May 2022, Published online: 25 May 2022

Abstract

Degradation and failure prediction has become more and more crucial for maintenance planning and scheduling, the decision-making process, and many other areas of manufacturing systems. This paper presents an approach where different artificial neural network models were developed to predict the degradation path of a machine component using different architectures, including fully connected networks (FCN) and arbitrarily connected networks (ACN). These models were trained using the Neuron-by-Neuron (NBN) training algorithm with forward-backward computations, where NBN is an improved form of the Levenberg-Marquardt (LM) algorithm, combined with FCN and ACN architectures, which can be trained efficiently, it can give more accurate predictions with a fewer number of neurons used. The developed models were evaluated using the statistical performance measure of the sum of squared error (SSE). The results show that the used networks are successfully able to predict the degradation path; the 8-neurons model of FCN architecture and the 3-neurons model of ACN architecture with tanh (mbib) hidden layers activation function and linear function (mlin) of the outputs have the lowest prediction error (SSE) among all the developed models. The use of such architectures combined with NBN training algorithm can easily model manufacturing systems with complex component structures that provide a vast amount of data.

Keywords

machine learning, artificial intelligence (AI), degradation prediction, maintenance, artificial neural network (ANN), neuron-by-neuron (NBN), fully connected networks (FCN), arbitrarily connected networks (ACN)

1 Introduction

Nowadays, industries are directed toward the Fourth Industrial Revolution or the so-called "Industry 4.0", which aims at integrating the physical and cyber parts of the production systems, including maintenance activities through different technologies such as Cyber-Physical Systems (CPS), Artificial Intelligence (AI), Internet of Things (IoT), Big Data, Cloud Computing, and others, leading to smart and autonomous production systems [1]. Hence, new trends of maintenance planning and scheduling, including predictive maintenance (PdM), prognostic and health management (PHM), condition-based maintenance (CBM), and smart decision-making, have been introduced in many related research areas, utilizing different approaches of AI [1–3]. Machine learning (ML) is one of the most researched areas of AI, which comprises prediction and optimization methods to discover knowledge and make smart decisions [4]. ML, or the "data-driven approach", uses historical data to learn the system's behavior [3]. ML approaches are commonly used in maintenance planning and scheduling where the required historical data is highly available [3, 5].

Many different algorithms have been used in ML, such as support vector machines (SVM), random forests, logistic regression, artificial neural networks (ANN), Naive Bayes, genetic algorithms, and much more. Paturi and Cheruku [6] showed that through the past two decades, ANN algorithms had gained significant importance in the field of maintenance planning; ANN has been proven to work with high accuracy and low level of error, proving its applicability in manufacturing operations. Consequently, neural networks are currently widely used in manufacturing systems applications, especially in maintenance

Cite this article as: Shaheen, B., Németh, I. "Machine Learning Approach for Degradation Path Prediction Using Different Models and Architectures of Artificial Neural Networks", Periodica Polytechnica Mechanical Engineering, 66(3), pp. 244–252, 2022. https://doi.org/10.3311/PPme.20145

planning and scheduling, condition-based maintenance, predictive maintenance, and degradation and failure prediction [7].

This paper presents the research work where a newly used training algorithm "Neuron-by-Neuron (NBN)" with backward-forward computation has been applied to predict the degradation path of a machine component leading to failure and replacement of the component then. Different models with various architectures of artificial neural networks, mainly fully connected networks (FCN) and arbitrarily connected networks (ACN), have been developed to predict the degradation path within the available range of the given data set. It is the first step toward predicting the degradation and failure data and developing a comprehensive predictive maintenance framework. Then the prediction results can be used to conduct simulations for better maintenance planning and scheduling. The NBN algorithm has been used here due to its well-known ability to train these ACN and FCN neural network architectures with lower prediction errors, less training time, less memory consumption, and fewer neurons.

2 Literature review

Training neural networks is a complicated process. Selecting the proper training algorithm can determine the success of an application. Different training algorithms are used to train the developed neural networks. The most popular one is the error-back propagation algorithm (EBP) which is an inefficient algorithm [7]. Li et al. [8] and Hamid et al. [9] indicated that EBP has many general shortcomings in terms of training speed, the complexity of the required activation functions, getting stuck at local minima, and the slow convergence speed. Among the usually used algorithms, EBP works with many errors; it is slower and inefficient compared to the other newly developed second-order training algorithms such as Levenberg-Marquardt (LM) and Neuron-by-Neuron (NBN) algorithms [10, 11]. The LM algorithm is efficient and very fast, but it can only train multi-layer perceptron (MLP) networks architecture, and a limited number of patterns can be handled [7]. These issues were solved by improving the LM algorithm with the recently developed NBN training algorithm. In addition, the NBN algorithm shows improvements in computation time and memory consumption [7, 12, 13].

The forward and backward computations can be performed based on the NBN routings; this makes the algorithm suitable for arbitrarily connected networks (ACN) and offers a proper method for modeling the complicated connections in manufacturing systems resulting from the tremendous amount of data provided by their complex machines.

Previous studies used EBP algorithms for prediction purposes; Rajmohan and Palanikumar [14] modeled and predicted the surface roughness in the drilling process of hybrid metal matrix composites using an artificial neural network based on an EBP training algorithm where MLP architecture was used to design the network. The results showed an efficient surface roughness prediction with a 9-4-1 MLP architecture. Similarly, Leh et al. [15] developed a model with EBP to detect failures in a power transmission line. The results show a tolerable accuracy in failure detection.

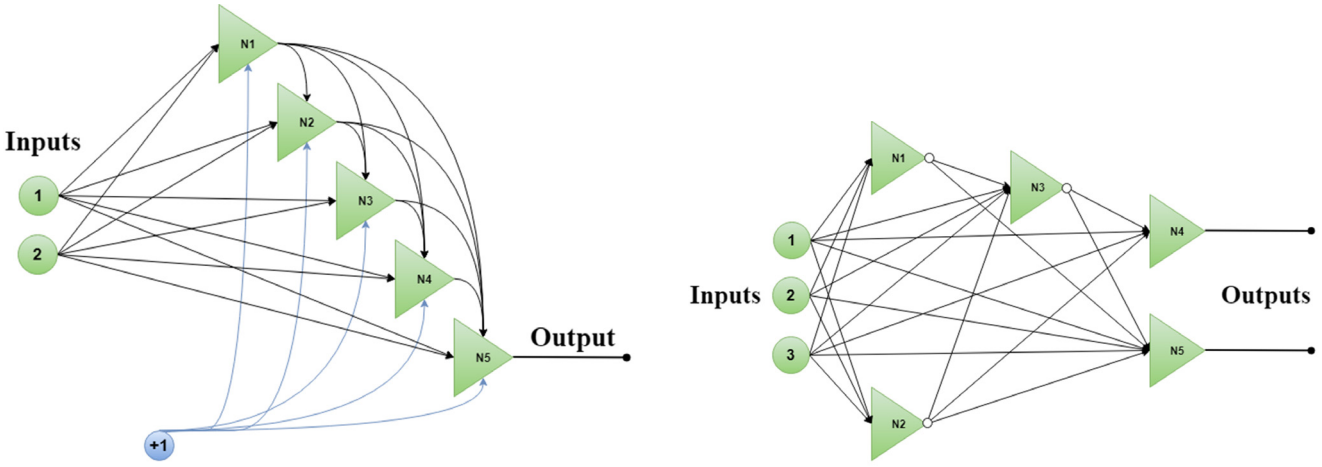
On the other hand, Hunter et al. [7] conducted a comparative study to select the proper network architecture and the size using different training algorithms, including the LM, EBP, and NBN algorithms. The study concluded that, as the NBN and LM are second-order algorithms, they are 100 to 1000 times faster than the EBP algorithm. Besides this, the NBN algorithm has some advantages against the LM algorithm:

1. The LM algorithm can handle only MLP networks, while the NBN algorithm can handle any feed-forward and arbitrarily connected neural networks.
2. The NBN method can be used for problems with an unlimited number of patterns.
3. The NBN method is more accurate, with a better success rate than LM.
4. Finally, the NBN algorithm can use both forward-backward or forward-only computational scheme for faster training and consume less memory.

The NBN training algorithm combined with FCN architecture was used by Hussain et al. [16] to develop estimators from an aircraft's sensor measurements. The developed estimator based on the NBN method was able to predict with high accuracy and few neurons.

3 Neural network architectures

Multi-layer perceptron (MLP) is considered as the most popular neural network (NN) architecture. However, MLP is inefficient and not powerful due to the high number of the required neurons to solve problems with a limited number of patterns [7, 12, 17, 18]. Thus, ACN and FCN are better and more efficient than the MLP [12, 16, 18]. Fig. 1 shows examples of FCN and ACN architectures.



(a) (b)

Fig. 1 Neural network architectures – (a) FCN / 2 inputs, 1 output and (b) ACN / 3 inputs, 2 outputs

3.1 NBN training algorithm

The NBN training algorithm is an improved version of the second-order LM algorithm [11]. The updated rule of the LM algorithm is the following [11, 12, 19]:

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \left(J^T J + \mu I\right)^{-1} J^T \boldsymbol{e}\,, \tag{1}$$

where $\boldsymbol{w}_{n+1}$ is the new vector of weights; $\boldsymbol{w}_n$ is the old vector of weights; J is the Jacobian matrix; I is the matrix of identity; $\boldsymbol{e}$ is the vector of error, μ is the coefficient of combination. The size of the Jacobian matrix is $(P \times M) \times N$ and the vector of error is $(P \times M) \times 1$, where P is the training patterns number, M is the network outputs number, and N is the weights number [17].

$$J = \begin{bmatrix} \text{neuron } 1 & & & \text{neuron } j & & \\ \frac{\partial e_{1,1}}{\partial w_{1,1}} & \frac{\partial e_{1,1}}{\partial w_{1,2}} & \cdots & \frac{\partial e_{1,1}}{\partial w_{j,1}} & \cdots & m=1 \\ \frac{\partial e_{1,2}}{\partial w_{1,1}} & \frac{\partial e_{1,2}}{\partial w_{1,2}} & \cdots & \frac{\partial e_{1,2}}{\partial w_{j,1}} & \cdots & m=2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{1,M}}{\partial w_{1,1}} & \frac{\partial e_{1,M}}{\partial w_{1,2}} & \cdots & \frac{\partial e_{1,M}}{\partial w_{j,1}} & \cdots & m=M \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{P,1}}{\partial w_{1,1}} & \frac{\partial e_{P,1}}{\partial w_{1,2}} & \cdots & \frac{\partial e_{P,1}}{\partial w_{j,1}} & \cdots & m=1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{P,M}}{\partial w_{1,1}} & \frac{\partial e_{P,M}}{\partial w_{1,2}} & \cdots & \frac{\partial e_{P,M}}{\partial w_{j,1}} & \cdots & m=M \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \\ \\ \\ p=1 \\ \\ \\ \\ \\ p=P \end{matrix} \tag{2}$$

Equation (2) shows how the Jacobian matrix is built, with p denoting the training pattern ($p = 1 \ldots P$), j denoting the neuron index, $\boldsymbol{w}_{j,x}$ denoting the x^{th} connection weight to neuron j, and m denoting the network output neuron index ($m = 1 \ldots M$). For training pattern p at network output neuron m, the error $e_{p,m}$ is calculated as follows:

$$e_{p,m} = d_{p,m} - o_{p,m}\,, \tag{3}$$

where $d_{p,m}$ is the targeted output and $o_{p,m}$ is the actual output for the training pattern p at the network output neuron m.

Usually, the Jacobian matrix is generated and then stored in order to update the weights using Eq. (1). This is useful for problems that just require a few training patterns. Due to the massive size of the Jacobian matrix [20], memory restriction may become a major challenge to be addressed for larger numbers of training patterns. The weights in the NBN algorithm are updated using the following update rule:

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \left(Q + \mu I\right)^{-1} \boldsymbol{g}\,. \tag{4}$$

The gradient vector is $\boldsymbol{g}$, and the quasi-Hessian matrix is Q. This represents an updated form of the LM rule [12], where

$$Q = J^T J \tag{5}$$

$$\boldsymbol{g} = J^T \boldsymbol{e}\,. \tag{6}$$

The matrix Q is created in the NBN algorithm by adding the quasi-Hessian sub-matrix $\Theta_{p,m}$ for pattern p and network output neuron m:

$$Q = \sum_{p=1}^{P} \sum_{m=1}^{M} \Theta_{p,m}\,. \tag{7}$$

Summing the gradient sub-vectors $\eta_{p,m}$ for pattern p and network output neuron m yields the gradient vector $\boldsymbol{g}$:

$$\boldsymbol{g} = \sum_{p=1}^{P} \sum_{m=1}^{M} \eta_{p,m}\,. \tag{8}$$

The number of patterns and outputs has no bearing on the size of the matrix Q, which is $N \times N$.

The NBN algorithm, unlike the LM algorithm, directly calculates the matrix Q and vector $\boldsymbol{g}$ when the patterns are applied. As a result, the Jacobian matrix ($\boldsymbol{J}$) does not need to be computed or stored [11]. The vector $\boldsymbol{j}_{p,m}$ is calculated as the patterns are applied to achieve this. The Jacobian row for pattern p and network output neuron m is represented by this vector. The matrix Q and vector $\boldsymbol{g}$ can be updated using this vector as each pattern is applied using Eq. (9) and Eq. (10):

$$\Theta_{p,m} = \boldsymbol{j}_{p,m}^{T} \boldsymbol{j}_{p,m} \tag{9}$$

$$\boldsymbol{\eta}_{p,m} = \boldsymbol{j}_{p,m} e_{p,m} . \tag{10}$$

NBN combined with FCN and ACN architectures are chosen here due to their highly accurate capabilities in prediction, a fewer number of neurons in each network, less training time, and less memory consumption [7, 12, 17, 18, 21]. Hence, the NBN training algorithm with forward-backward computations has been used to predict the degradation levels of a machine component of a manufacturing system. Its analysis and results are discussed in the below sections.

4 Models design and implementation

4.1 Tools and software

Models were developed using a laptop with the following specifications:

- Processor: Intel(R) Core (TM) i7-8750H CPU @ 2.20 GHz (12 CUs), 2.2 GHz
- Memory: 12288 MB RAM.

The used training software is NBN trainer 2.08 developed based on the C++ programming language by Yu and Wilamowski [22], in addition to other statistical software for data pre-processing and data visualization.

4.2 Data

Failure datasets with high reliability are scarce [23]. Here in this study, a simulated dataset has been used to investigate the applicability of the ANN algorithms for degradation modeling and prediction. The simulated dataset represents the degradation of a mechanical component, as depicted in Fig. 2.

The assumptions for the data simulation process are based on Bagdonavičius and Nikulin's methods [23] for simulating (lifetime data) degradation models and degradation characteristics from data with covariates. The data

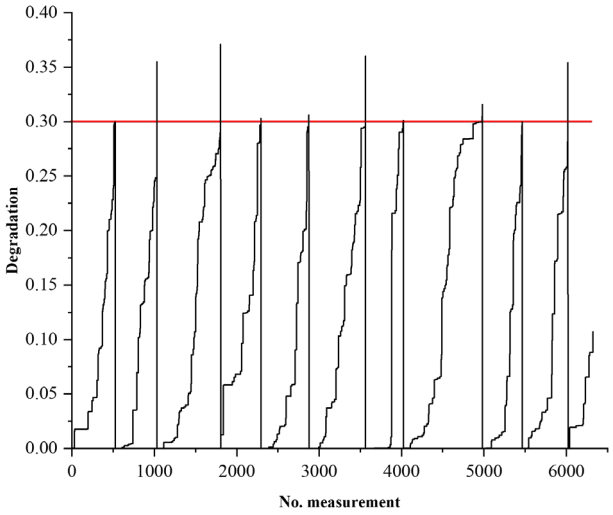


Fig. 2 The overall degradation dataset

set includes the degradation of a single component, including 6323 points (patterns). The failure occurs once the degradation measure exceeds the pre-set threshold of 0.3. Then the component is replaced with a new one, and the degradation starts from zero. The parameters in the degradation dataset should be monotonic and trendy [23, 24]. This means that identification of the increasing or decreasing trajectories of the parameters is needed. The dataset was pre-processed; suspicious data and outliers such as the extremely deviated or non-monotonous values, if existing, were cleaned.

In the training process, it is essential to split the primary dataset into training and testing datasets to avoid overfitting and to check the performance of the trained models. According to Gholmany et al. [25], many recent studies indicated that the best training results are achieved when using 70–80% of the data for training and the remaining 20–30% for testing the models. The most commonly used split percentage is 80% for training and 20% for testing [25, 26].

Table 1 shows the main characteristics of the simulated degradation dataset.

4.3 Models evaluation

The developed models were evaluated using the sum squared error (SSE) statistical performance measure, as follows:

$$\mathrm{SSE} = \sum_{i=1}^{n} \left(y_i' - y_i \right)^2 , \tag{11}$$

Table 1 Descriptive statistics of the simulated dataset

Total count	Mean	Standard deviation	Min	Max	Range
6323	0.09445	0.09967	0.00000	0.37083	0.37083

where n is the number of patterns, y_i' is the predicted value, and y_i is the target value.

5 ANN design and implementation

The selected network architecture strongly affects the training success [27]; the use of the NBN algorithm could solve the problem of the inability to handle other types of architectures rather than the most common one (MLP) such as BMLP (Bridged multi-layer perceptron), FCN, ACN and multi-layer perceptron with linear neurons (MLPL) [28].

Thus, different numbers of neurons and two types of network architectures (FCN and ACN) were used to model and predict the degradation path.

Table 2 shows the used topologies and architectures for the developed models.

Random weights in the range of −1 to 1 are used to initialize the neural networks. The bipolar hyperbolic tangent (tanh) activation function symbolized as "mbib" is used for hidden layers (used by neurons), knowing that bipolar neurons have positive or negative outputs.

Likewise, the linear activation function "mlin" is used to be the output neurons, where linear activation functions are used if the desired output is larger than 1 [22]. Table 3 shows the numbers of input patterns for the used training and testing datasets according to the 80% training and 20% testing schema. Refer to Table 3 for more details.

The output of neuron j is defined as

$$\text{out}_j = \tanh\left(\text{gain} \times net_j\right) + \text{der} \times net_j . \tag{12}$$

As seen in Fig. 3, net_j is the summation of the weighted inputs to neuron j and out_j is the output of neuron j.

The "gain" and "der" are parameters of the activation functions. The parameter "der" is introduced to adjust the slope of the activation function, where the slope is approaching zero [22]. The activation functions' parameters are "gain" and "der". When the slope of the activation function approaches zero, the parameter "der" is introduced to adjust it [22].

These parameters are set to be gain = 0.50 and der = 0.01 for the "mbib" neurons, while gain = 1.00 and der = 0.01 for the "mlin" neurons.

The degradation measurements, which should be the output of the training, are normalized using the min-max normalization process using Eq. (13):

$$x_n = (b-a) \times \frac{x_o - x_{\min}}{x_{\max} - x_{\min}} + a , \tag{13}$$

where the normalized value denoted by x_n and the value to be normalized denoted by x_o; the minimum and maximum values of the range to be normalized to are a and b; $x_{\max}$ and $x_{\min}$ are the max and min range values from which x_o has been normalized, knowing that the normalization process speeds up learning and leads to quicker convergence.

6 Results and discussion

Table 3 shows the summary of the conducted experiments with the six ANN models presented above to predict the degradation path of the component.

The sum squared errors (SSE) of the training and testing degradation data sets and using a different number of neurons for the six ANN models are depicted in Fig. 4.

The adequacy of the ANN models is validated using the SSE performance measure. The selected (close to optimal) best-trained models are those with the least SSE. According to Table 3 and Fig. 4, model no. 3 (8 neurons - FCN) and model no. 6 (3 neurons - ACN) are the best-trained models.

The NBN algorithm was able to train the network to a small error $\text{SSE}_{\text{Training}}$ = 0.000299 and 0.000204 for FCN and ACN, respectively. The results are acceptable with SSE_{Test} = 0.000101 and 0.000055 for FCN and ACN, respectively.

Fig. 5 shows the training curve of the best two models.

Table 4 shows the training results: the success rate (success rate for multiple times training from 0–100%), the average number of iterations, and average time. From Table 3 and Table 4, it can be seen that the used NBN training algorithm is able to successfully train both FCN and ACN architectures and achieve significant results (low prediction error) with a fewer number of neurons and iterations. It is worth mentioning that the average training time of ACN is greater than FCN due to the complexity of such architecture.

Figs. 6 and 7 show the degradation path prediction results of models 3 and 6, respectively. It can be noticed that there is a high correlation between the target and precited values in both models (red and blue values). The "target-test dataset" is 20% of the whole simulated data set, which was used to test the accuracy of the trained models, as mentioned in Table 3. The "predicted-test dataset" contains the predicted values by the trained model compared to the "target-test dataset", which are highly correlated due to the very low prediction error.

Table 2 Topologies and architectures of the developed models

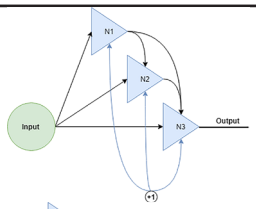
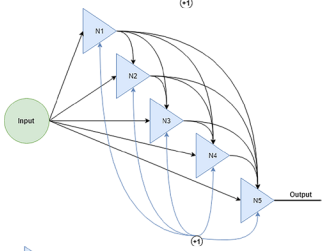
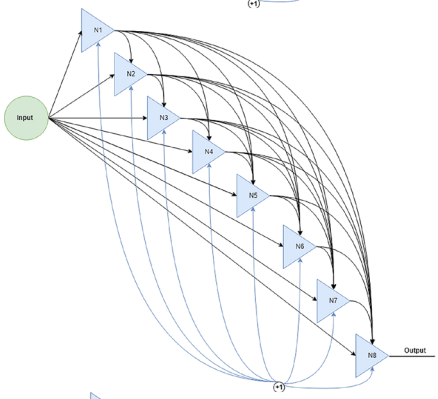
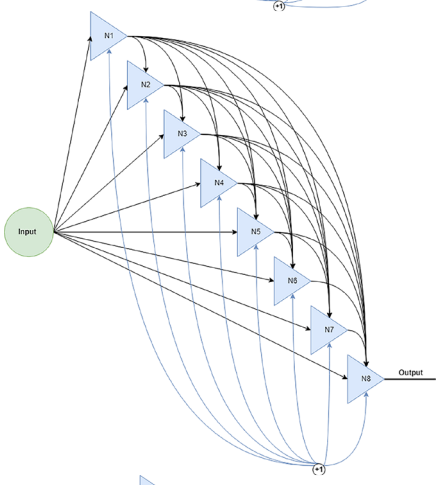
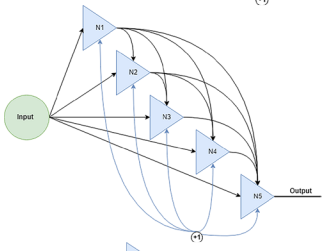
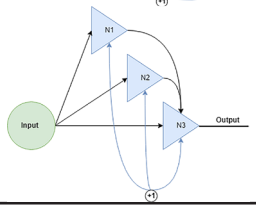
Model	Type	Topology	ANN architecture
1	FCN	n 2 mbip 1 n 3 mbip 1 2 n 4 mlin 1 2 3	
2	FCN	n 2 mbip 1 n 3 mbip 1 2 n 4 mbip 1 2 3 n 5 mbip 1 2 3 4 n 6 mlin 1 2 3 4 5	
3	FCN	n 2 mbip 1 n 3 mbip 1 2 n 4 mbip 1 2 3 n 5 mbip 1 2 3 4 n 6 mbip 1 2 3 4 5 n 7 mbip 1 2 3 4 5 6 n 8 mbip 1 2 3 4 5 6 7 n 9 mlin 1 2 3 4 5 6 7 8	
4	ACN	n 2 mbip 1 n 3 mbip 1 2 n 4 mbip 1 2 3 n 5 mbip 1 2 3 n 6 mbip 1 2 3 4 5 n 7 mbip 1 2 3 4 5 6 n 8 mbip 1 2 3 4 5 6 n 9 mlin 1 2 3 4 5 6 7 8	
5	ACN	n 2 mbip 1 n 3 mbip 1 2 n 4 mbip 1 2 3 n 5 mbip 1 2 3 n 6 mlin 1 2 3 4 5	
6	ACN	n 2 mbip 1 n 3 mbip 1 n 4 mlin 1 2 3	

Table 3 Summary - "training and testing" processes of the ANN models

Parameter	ANN models					
Architecture	FCN	FCN	FCN	ACN	ACN	ACN
Variable's normalization	Min-max normalization					
Hidden layers activation function	mbib (tanh)					
Output activation function	mlin					
No. neurons	3	5	8	8	5	3
No. nodes	4	6	9	9	6	4
Training algorithm	NBN-forward backward					
No. patterns / training (80%)	5058					
No. patterns / test (20%)	1265					
Error function	Sum squared error (SSE)					
Maximum error	0.001					
Maximum no. iterations	50					
SSE / training	0.000345	0.000919	0.000299	0.000280	0.000467	0.000204
SSE / test	0.000093	0.000228	0.000101	0.000083	0.000109	0.000055

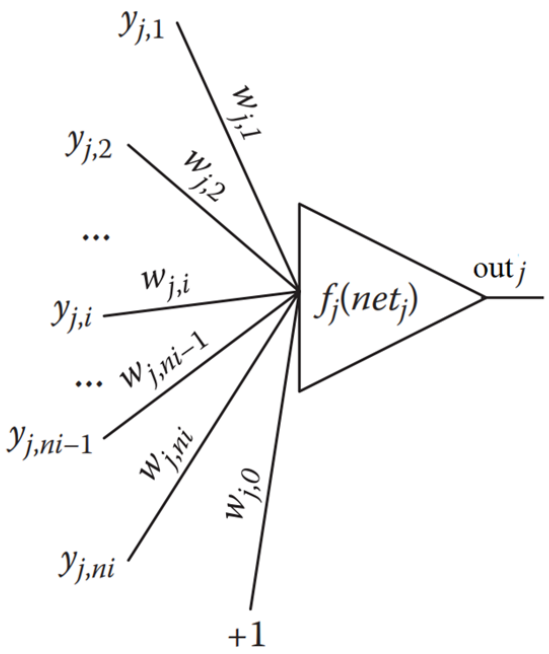


Fig. 3 Neuron *j* connections

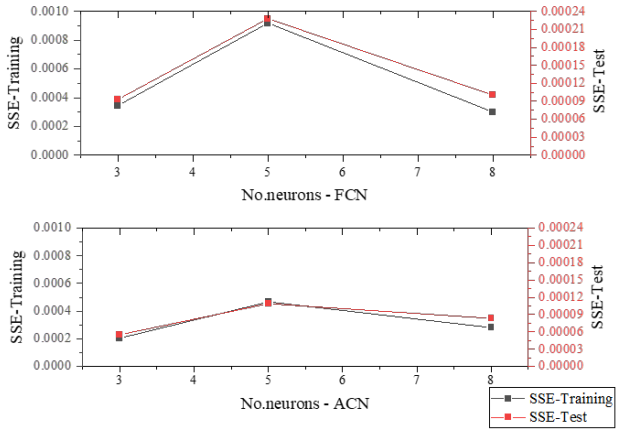
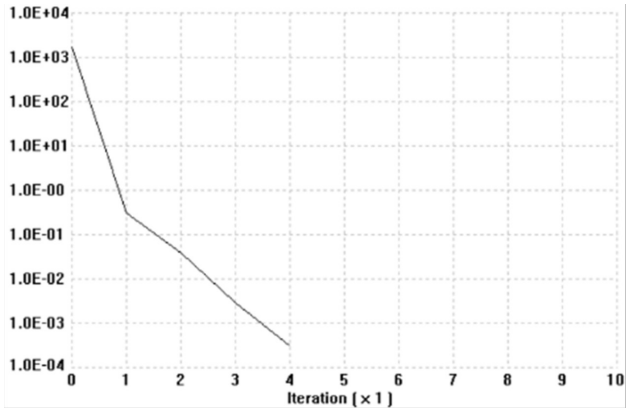
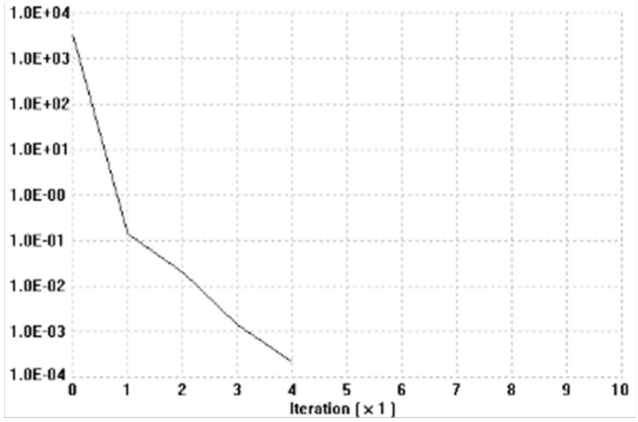


Fig. 4 $SSE_{Training}$ vs SSE_{Test} - FCN and ACN for all models



Model no.3



Model no.6

Fig. 5 Training curves of the best-trained models

Table 4 Training results

	Success rate (%)		Avg no. iterations		Avg time (ms)	
No. neurons	FCN	ACN	FCN	ACN	FCN	ACN
3	100	100	2.55	2.07	0.62	9.22
5	100	100	2.14	2.09	6.45	13.11
8	100	100	2.22	2.28	34.86	35.49

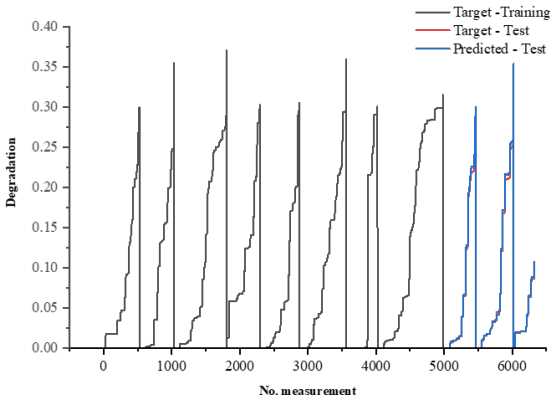


Fig. 6 Target vs. predicted degradation path / Model no. 3

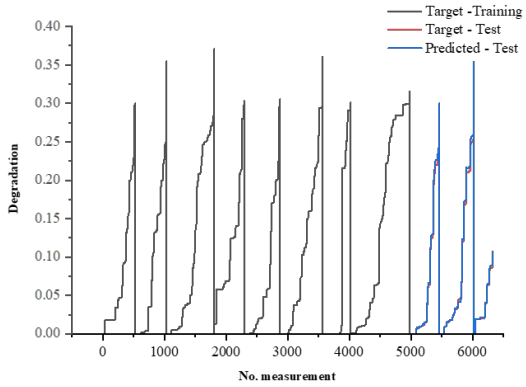


Fig. 7 Target vs. predicted degradation path / Model no. 6

7 Conclusion and future work

In this research work, different artificial neural networks were developed to model the degradation path of a mechanical component based on combinations of FCN and ACN architectures and a second-order Neuron-by-Neuron (NBN) training algorithm with forward-backward computations. The ANN models were analyzed, compared, and discussed. The results show that the used NBN training algorithm and the FCN and ACN architectures can precisely predict the degradation path of the machine component with a high correlation for the values of the target degradation path. Furthermore, the developed models can produce reasonable predictions with 3 and 8 neurons.

Results also confirmed the effect of choosing a proper architecture on the success of the training process. In addition, the use of ACN architecture combined with NBN connects the network neurons arbitrarily; thus, a manufacturing system (consisting of several independent machines) can be modeled easily despite its complexity by using this type of neural network.

Future research work aims to develop advanced models (combination of ACN architecture and NBN training algorithm) and estimate the remaining useful life (RUL) of the components based on the historical degradation data. This type of prediction is the first step to developing a predictive maintenance framework (PdM), including a health monitoring and diagnosis system, failure detection, failure prediction, and failure identification algorithms to boost the maintenance planning and scheduling system and reduce downtimes and maintenance associated costs. More research opportunities could be addressed by utilizing this approach to develop probabilistic models which are able to predict the failure rate and the failure probability to be exploited in a discrete event simulation system to improve the maintenance planning and scheduling as required to be integrated into the industry 4.0 era. Moreover, extended comparative studies of the performance of different training algorithms for such applications can be an opportunity for future research.

Acknowledgment

The research reported in this paper and carried out at the Budapest University of Technology and Economics has been supported by the National Laboratory of Artificial Intelligence funded by the National Research Development and Innovation Office under the auspices of the Ministry for Innovation and Technology, and by the European Commission through the H2020 project EPIC under grant No. 739592.

References

[1] Çinar, Z. M., Nuhu, A. A., Zeeshan, Q., Korhan, O., Asmael, M., Safaei, B. "Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0", Sustainability, 12(19), 8211, 2020. https://doi.org/10.3390/su12198211

[2] Sirvio, K. M. "Intelligent Systems in Maintenance Planning and Management", In: Kahraman, C., Çevik Onar, S. (eds.) Intelligent Techniques in Engineering Management, Springer International Publishing, , 2015, pp. 221–245. ISBN: 978-3-319-17905-6 https://doi.org/10.1007/978-3-319-17906-3_10

[3] Paolanti, M., Romeo, L., Felicetti, A., Mancini, A., Frontoni, E., Loncarski, J. "Machine Learning approach for Predictive Maintenance in Industry 4.0", In: 2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), IEEE, 2018, pp. 1–6. ISBN: 978-1-5386-4644-1 https://doi.org/10.1109/MESA.2018.8449150

[4] Wang, L., Alexander, C. A. "Machine Learning in Big Data", International Journal of Mathematical, Engineering and Management Sciences, 1(2), pp. 52–61, 2016. https://doi.org/10.33889/ijmems.2016.1.2-006

[5] Dalzochio, J., Kunst, R., Pignaton, E., Binotto, A., Sanyal, S., Favilla, J., Barbosa, J. "Machine learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges", Computers in Industry, 123, 103298, 2020. https://doi.org/10.1016/j.compind.2020.103298

[6] Paturi, U. M. R., Cheruku, S. "Application and performance of machine learning techniques in manufacturing sector from the past two decades: A review", Materials Today: Proceedings, 38(5), pp. 2392–2401, 2021. https://doi.org/10.1016/j.matpr.2020.07.209

[7] Hunter, D., Yu, H., Member, S., Pukish, M. S., Member, S., Kolbusz, J., Wilamowski, B. M. "Selection of Proper Neural Network Sizes and Architectures — A Comparative Study", IEEE Transaction on Industrial Informatics, 8(2), pp. 228–240, 2012. https://doi.org/10.1109/TII.2012.2187914

[8] Li, J., Cheng, J.-H., Shi, J.-Y., Huang, F. "Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement", In: Jin, D., Lin, S. (eds.) Advances in Computer Science and Information Engineering, Springer, 2012, pp. 553–558. ISBN: 978-3-642-30222-0 https://doi.org/10.1007/978-3-642-30223-7_87

[9] Hamid, N. A., Mohd, N., Ghazali, R. "A review on improvement of back propagation algorithm", AWERProcedia Information Technology & Computer Science, 1, pp. 882–888, 2012.

[10] Nawi, N. M., Khan, A., Rehman, M. Z. "A New Levenberg Marquardt Based Back Propagation Algorithm Trained with Cuckoo Search", Procedia Technology, 11, pp. 18–23, 2013. https://doi.org/10.1016/j.protcy.2013.12.157

[11] Wilamowski, B. M., Yu, H., Cotton, N. "NBN Algorithm", In: Wilamowski, B. M., Irwin, J. D. (eds.) Intelligent Systems, CRC Press, 2011, pp. 13-1–13-23. ISBN: 9781315218427

[12] Wilamowski, B. M. "Neural network architectures and learning algorithms", IEEE Industrial Electronics Magazine, 3(4), pp. 56–63, 2009. https://doi.org/10.1109/MIE.2009.934790

[13] Wilamowski, B. M. "Challenges in applications of computational intelligence in industrial electronics", In: IEEE International Symposium on Industrial Electronics, IEEE, 2010, pp. 15–22. ISBN: 978-1-4244-6390-9 https://doi.org/10.1109/ISIE.2010.5637934

[14] Rajmohan, T., Palanikumar, K. "Ann Model To Predict Surface Roughness in Drilling Hybrid Composites", Advances in Production Engineering and Management, 6(4), pp. 281–290, 2011.

[15] Leh, N. A. M., Zain, F. M., Muhammad, Z., Hamid, S. A., Rosli, A. D. "Fault Detection Method Using ANN for Power Transmission Line", In: 2020 10th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), IEEE, 2020, pp. 79–84. ISBN: 978-1-7281-7244-6 https://doi.org/10.1109/ICCSCE50387.2020.9204921

[16] Hussain, S., Mokhtar, M., Howe, J. M. "Aircraft sensor estimation for fault tolerant flight control system using fully connected cascade neural network", In: The 2013 International Joint Conference on Neural Networks (IJCNN), Dallas, TX, USA, 2013, pp. 1–8. https://doi.org/10.1109/IJCNN.2013.6706763

[17] Wilamowski, B. M. "How to not get frustrated with neural networks", In: 2011 IEEE International Conference on Industrial Technology, IEEE, 2011, pp. 5–11. ISBN: 978-1-4244-9064-6 https://doi.org/10.1109/ICIT.2011.5754336

[18] Yu, H., Auburn, W. "Fast and Efficient and Training of Neural Networks", In: 3rd International Conference on Human System Interaction, IEEE, 2010, pp. 175–181. ISBN: 978-1-4244-7560-5 https://doi.org/10.1109/HSI.2010.5514571

[19] Wilamowski, B. M. "Advanced learning algorithms", In: 2009 International Conference on Intelligent Engineering Systems, IEEE, 2009, pp. 9–17. ISBN: 978-1-4244-4111-2 https://doi.org/10.1109/INES.2009.4924730

[20] Wilamowski, B. M., Yu, H. "Improved Computation for Levenberg – Marquardt Training", IEEE Transaction on Neural Networks, 21(6), pp. 930–937, 2010. https://doi.org/10.1109/TNN.2010.2045657

[21] Wilamowski, B., Hunter, D., Maljnowski, A. "Solving Parity-N Problems with Feedforward Neural Networks", In: Proceedings of the International Joint Conference on Neural Networks, 2003, IEEE, Vol. 4, 2003, pp. 2546–2551. ISBN: 0-7803-7898-9 https://doi.org/10.1109/IJCNN.2003.1223966

[22] Yu, H., Wilamowski, B. M. "C++ implementation of neural networks trainer", In: 2009 International Conference on Intelligent Engineering Systems, Barbados, 2009, pp. 257–262. https://doi.org/10.1109/INES.2009.4924772

[23] Bagdonavičius, V., Nikulin, M. S. "Estimation in Degradation Models with Explanatory Variables", Lifetime Data Analysis, 7(1), pp. 85–103, 2001. https://doi.org/10.1023/A:1009629311100

[24] Singh, J., Darpe, A. K., Singh, S. P. "Bearing remaining useful life estimation using an adaptive data-driven model based on health state change point identification and *K*-means clustering", Measurement Science and Technology, 31(8), 085601, 2020. https://doi.org/10.1088/1361-6501/ab6671

[25] Gholmany, A., Kreinovich, V., Kosheleva, O. "Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation", University of Texas, El Paso, TX, USA, Rep. UTEP-CS-18-09, 2018. [online] Available at: https://scholarworks.utep.edu/cs_techrep/1209/ [Accessed: 10 March 2022]

[26] Sheskin, D. J. "Handbook of Parametric and Nonparametric Statistical Procedures", [e-book] Chapman and Hall/CRC, 2011. ISBN: 9780429186196 https://doi.org/10.1201/9780429186196

[27] Rozycki, P., Kolbusz, J., Malinowski, A., Wilamowski, B. "The Impact of Architecture on the Deep Neural Networks Training", In: 2019 12th International Conference on Human System Interaction (HIS), IEEE, 2019, pp. 41–46. ISBN: 978-1-7281-3981-4 https://doi.org/10.1109/HSI47298.2019.8942622

[28] Wilamowski, B. M., Yu, H. "Neural Network Learning without Backpropagation", IEEE Transaction on Neutral Networks, 21(11), pp. 1793–1803, 2010. https://doi.org/10.1109/TNN.2010.2073482