Use of artificial neural networks for solving the problem of residual resource estimation of hoisting cranes

Roman Khvan[1*]

[1]Don State Technical University, 1, Gagarin Square, Rostov-on-Don, 344002, Russian Federation

Abstract. The work is devoted to the problems of practical application of artificial neural networks (ANN) for estimation of residual resource of hoisting machines on the example of estimation of residual resource of bridge-type cranes. The work describes in detail the programming process of ANN which allows to determine the residual resource of metal structures of a hoisting crane depending on degradation of hardness of the metal structures material. The purpose of the work was to create new software for solving the problem of estimating the residual resource of the hoisting crane which appointed service life has ended. The basis for training of ANN was the existing knowledge of hoisting cranes operation, namely the statistical database of typical crane damages, expert assessment of the technical condition of hoisting cranes, and recommendations of the existing standards to identify defect indicators of hoisting cranes. The software is written in Python programming language. The work lists different ways of programming a neural network for an estimation of residual resource of hoisting cranes. The first way is more complex and painstaking and consists of writing the programme code of the neural network manually, the second way involves using a free interactive library Skicit-learn. A new software has been created to estimate the residual resource of hoisting cranes depending on degradation of hardness of metal structures material. The programme can be used by specialists and experts as an intellectual decision-making support system for determining the residual resource of hoisting cranes. The detailed algorithm of neural network programming is presented for possible use of this information in other areas of technical diagnostics of various mechanical engineering objects.

1 Introduction

Nowadays, methods of estimating the technical condition of mechanical engineering objects using artificial neural networks are becoming more and more widespread. Above all it is due to the relative simplicity of revealing dependences of output data (in this case, the residual resource of a metal structure of a hoisting crane) on input data (degradation of hardness of a material of a metal structure). For example, establishing the correlation

* Corresponding author: khvanroman@yandex.ru

dependence of the residual resource of hoisting cranes on the degradation of hardness of the material of the metal structure, expressed in the parameters of the Weibull distribution, requires a large number of analytical operations to be performed and the multifactor regression problem to be solved. The software, the core of which is a neural network, will facilitate the work and help experts and novice specialists make decisions on the possibility of further operation of the hoisting crane depending on hardness degradation of the metal structures material.

In our country scientists of various branches of knowledge have been dealing with the problems of reliability and safety of technical systems with the use of modern artificial intelligence tools. Some of the examples are the works of V.A. Vorontsov, E.A. Fedorov, A.A. Korotkiy, A.V. Panfilov, N.N. Nikolaev, A.R. Yusupov, S.V. Zhernakov, T.I. Goreva, N.N. Portyagin, G.A. Pyukke, B.Ch. Meskhi, A.N. Beskopylynyi, S.A. Stelmakh, I.F. Razveeva, and others [1–6].

Speaking about the relevance of the work, it is worth noting the insufficient elaboration of the issue related to the practical implementation of building and training of ANN for solving problems of machine-building industry. This is largely due to the high cost of software for training and construction of neural networks (Alyuda NeuroIntelligence, Statistica, Amygdala, MatLab Simulink, etc.). Obviously, the most affordable way to build and train a neural network, from a financial point of view, is to write your own programming code, provided you have at least basic programming skills. This article presents the experience of programming a neural network for estimating the residual resource of hoisting cranes. The basis for training of ANN was the existing knowledge of hoisting cranes operation, namely the statistical database of typical damages of hoisting cranes, expert assessment of residual resource of hoisting cranes, recommendations of existing standards to identify defect indicators of hoisting cranes [7-9].

The aim of the work is to create a new software for estimating the residual life of hoisting cranes based on hardness degradation of the metal structures material. The programme can be used by specialists and experts as an intelligent decision support system for determining the residual resource of hoisting cranes. The detailed algorithm of neural network programming is presented for possible use in other areas of technical diagnostics of various mechanical engineering facilities.

2 Materials and methods

The Python programming language was used to write the ANN programme code. It should be noted that the integrated environment for writing a programme in the Python programming language is publicly available and free of charge. There are several ways of programming neural networks in Python. The first way is more complex and painstaking, consisting of writing the neural network programming code "manually". The complexity lies in the large number of different mathematical operations needed to build and train the neural network, which requires appropriate conversion into programming code. Fig. 1 shows the scheme of the neural network for estimating the residual resource of hoisting cranes [7].

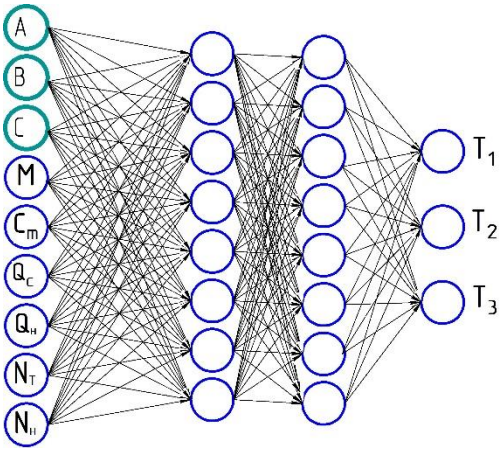

Fig. 1. Scheme of ANN for estimation of residual resource of hoisting cranes.

Following is the algorithm of basic operations for construction and training of neural network of residual resource estimation of hoisting cranes:

1. The so-called hyperparameters are set, which are the parameters that do not change during the training of the network: M – number of years of crane operating; $C_т$ – number of working cycles performed from the beginning of operating; Q_c – average load capacity; $Q_н$ – nominal load capacity of the crane; $N_н$ – normative value of the characteristic number; $N_т$ – current value of the characteristic number, as well as distribution parameters of the three-parameter Weibull law of steel hardness of metal structures: A – scale parameter; B – shape parameter; C – shear parameter. The categories of residual resources of the machines under study are fed to the output layer: T_1 – residual resource from 1 to 5 years; T_2 – residual resource from 6 to 10 years; T_3 – residual resource from 11 and more years; Thus, the architecture of the neural network is set.

2. Training parameters are set, which are the parameters that are changed (optimised) in the process of network training: values of synaptic weights *w* (strength of connection between neurons) and displacements *b*. At first, these values are set randomly, and then in the process of training the neural network they will be optimally adjusted.

3. The Forward propagation algorithm is implemented, i.e. the neural network is computed on random parameters *w, b*. At this stage, we need training data, i.e. samples of input parameters with known output parameters, as shown in Table 1.

Table 1. Data for training the neural network.

	Category	T, years	M, years	Q_c, tonnes	$Q_н$, tonnes	$N_т$, cycles	$N_н$, cycles	$C_т$, cycles	A	B	C
Crane No. 1	T_3	12	22	7.5	15	10312	16000	82500	47.3	7.3	120.7
Crane No.2	T_1	3	31	8	16	14531	16000	116250	47.0	4.8	130
Crane No.3	T_1	2.5	37	9.6	16	29970	32000	138750	53.1	3.8	135.8
Crane No.4	T_2	10.5	37	8	16	24975	32000	199800	44.1	4.8	120.4
Crane No.5	T_3	12.5	30	8	16	11250	16000	90000	32.3	5.5	130.6
Crane No.6	T_3	11.5	24	5	10	10800	16000	86400	37.3	3.4	122.3
Crane No.7	T_3	13.5	22	5	10	9900	16000	79200	39.2	7.1	124.3
Crane No.8	T_2	10	24	7.5	15	11250	16000	90000	37.2	3.3	134.3
Crane No.9	T_1	2.5	47	8.4	14	30456	32000	141000	45.3	2.7	120.5
Crane No.10	T_2	6.5	22	8	16	12375	16000	99000	49.0	5.9	121
…	T_2	8	49	8	16	27562	32000	220500	39.0	4.2	117.2
Crane No.100	T_1	1.5	47	8	16	30843	32000	246750	47.0	3.9	125.3

3.1 Activation values of neurons of each layer of the network are calculated (the process of formation of activation level of one neuron is described).

3.1.1. Input parameter values x are multiplied (at the first stage, and further the activation values of the neurons of the previous layer) by weighting coefficients w.

3.1.2. The weighted values are summed and the offset b is added.

3.1.3. The activation function F is calculated. The activation function is needed to add nonlinearity and an activation threshold to the output of each neuron. As a result, we obtain a formula for determining the activation level of a neuron:

$$y = F(x_1w_1 + x_2w_2 + x_3w_3 + b)$$

Fig. 2 shows a scheme of the formation of activation level of a neuron y.

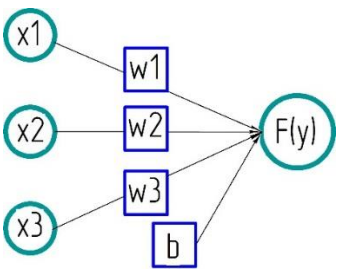

Fig. 2. Scheme of formation of activation level of a neuron.

3.2. Transformation of activation values of neurons of the last resulting layer of the network is normalised using Softmax function, so that their values are from 0 to 1 and the sum of these values is equal to 1. In other words, we implement the interpretation of neuron activation levels in a probabilistic sense. As a result, we obtain the prediction of the neural network: the category of the residual resource of the crane with some confidence level for the current values of synaptic weights (connections between neurons) w and from the displacements b.

3.3. The error E is calculated between the calculated activation levels of the neurons of the output layer $y_{выч}$ and the target activation values of the neurons of the output layer $y_{цел}$ using the *MSE* function (Euclidean distance) or the Cross-entropy function (used to

determine the distance between probability distributions). Remember that a neural network is trained "with a teacher", i.e. on examples with known inputs and outputs in advance (training sample).

4. The Backpropagation algorithm is implemented, which aims to solve the problem of minimising the error function E depending on the synaptic weights (connections between neurons) w and on the displacements b. For this purpose, the method of Gradient Descent is usually used.

4.1. The error gradient vector is calculated, which is a vector of partial derivatives of functions by all its arguments. Its dimension is equal to the number of all required weights w_i, and it is directed towards the local increase of the error E.

4.2. The gradient with the minus sign is directed towards error reduction. Thus, there is a movement along the anti-gradient in the direction of error reduction by updating the weights w. The size of the step with which the movement along the anti-gradient occurs is called the learning rate of the neural network. As a result, we obtain updated weight coefficients (connections between neurons).

5. The entire learning algorithm is repeated on the next training sample (or group of samples) which is designed to minimise the error of the neural network by updating the connections (weights) between neurons. Each such repetition is called a training epoch. After performing a given number of training epochs, the neural network with all settings is saved and can be used to predict the output parameters (residual life of the crane) from the input parameters (steel degradation) not previously used in the network training process.

Hardness measurements were made in the sections of crane booms, corresponding to the place of attachment of the hydraulic cylinder of boom lifting for truck cranes and in the section corresponding to the middle of the bridge crane span beam. This is due to the greatest bending moment being observed in this section when crane is lifting a load. This results in localised plastic deformations in these sections of the crane steel structures and the risk of fatigue cracks. A scheme of a truck crane boom showing a cross-section of the hardness measurement locations is shown in Fig. 3.

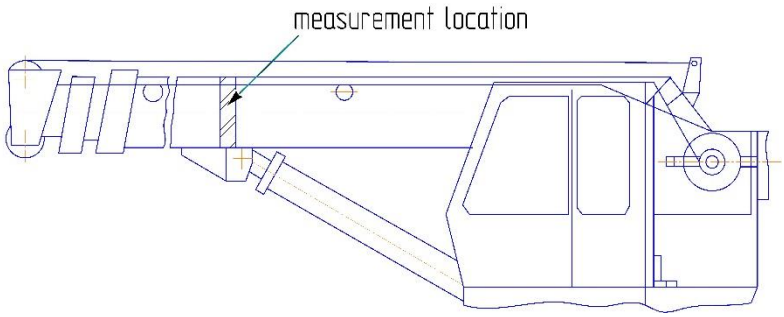

Fig. 3. Scheme of a truck crane boom with hardness measurement locations indicated.

Measurements of steel hardness of metal structures of overhead travelling cranes were made in the middle of the span girder of the crane, and also on the upper belt. The scheme of steel structures of overhead travelling crane with indication of the measurement location is presented in Fig. 4.

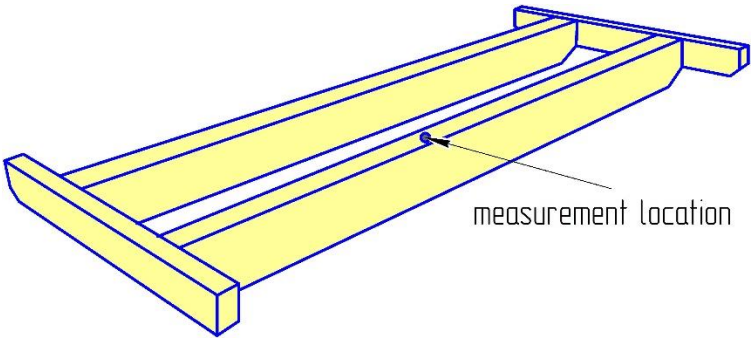

Fig. 4. Scheme of steel structures of overhead travelling crane with indication of hardness measurement location.

To take into account the hardness degradation of the steel structures material and to add the parameters of the Weibull law function of hardness distribution to the input layer of the neural network, an experiment of hardness measurements of steel hoisting cranes was carried out. Measurements were carried out using a portable ultrasonic hardness tester CONSTANTA K5U (КОНСТАНТА К5У). The sample size of measurements was taken as n=30 per the Student's criterion.

Following is a detailed algorithm for writing a neural network programming code based on the interactive library Skicit-learn. This library is designed for machine learning in the Python programming language [4-7].

1. In the first block, additional interactive libraries are installed (a library in this case means a set of ready-made subroutines that facilitate the programming process) through the input of the following code (the content or the code command to be executed is explained in brackets):

!python -m pip install pandas (data processing and analysis library).

!python -m pip install sklearn (library for machine learning).

!python -m pip install openpyxl (library for working with Excel files).

2. The second block imports modules by entering the following code:

import pandas as pd (data processing and analysis module).

from sklearn.neural_network import MLPClassifier (module for working with neural networks).

from sklearn.metrics import confusion_matrix, classification_report (additional module for working with neural networks).

import pickle (module used to save the trained neural network).

import joblib (module used to save the trained neural network).

3. The database, i.e. the training data set, is loaded. It is necessary to prepare in advance an Excel file with training data arranged in the format shown in Table 1 (a more simplified format is recommended in the form of only the columns of input and output parameters with their headings). To do this, enter the following programme code:

ds = pd.read_excel('book1.xlsx') , where book1.xlsx is an Excel file that should be located in the same folder with the file of the programme being created.

ds.head(10) (code for obvious visualisation of the first 10 rows of the table).

4. Next, assign values to variables *X* and *Y* by writing the following programming code:

X = ds.drop('Category',axis=1) (for variable *X*, all columns except the “Category” column are assigned (the literal name of the column is specified)).

y = ds[Category'] (for variable *Y* all columns except the column “Category” are assigned).

5. The neural network is built and trained by writing the following code (this two-line programming code replaces writing the neural network programming code "manually" from the previous algorithm):

nn=MLPClassifier(hidden_layer_sizes=(8,8), max_iter=2000) (the neural network architecture and its training parameters are set; in this case there are 3 hidden layers, with 8 neurons in each layer; the number of training epochs is 2000; the default settings are: activation function Relu (code: activation='relu'), error function Adam (code: solver='adam'), learning rate 0.0001(code: alpha=0.0001)). It is possible to select different training parameters for the neural network by writing appropriate programme codes separated with commas. Different kinds of activation functions, error functions, and other network parameters with their codes can be found on the official website of the interactive library Scikit-learn.

nn.fit(X, y) (previously set parameters X - input variables (steel degradation parameters) and Y - output target variable Residual resource of hoisting crane) are introduced into the neural network).

After starting this block, the neural network is built according to the given architecture and trained according to the given training parameters on the training data (samples) downloaded through the Excel file table. The condition for the training to be stopped is reaching the network control performance of 95%. The training process in this work was about 1.5 minutes.

3 Study findings

After training the network, we analyse the work of the constructed neural network. In order to identify the influence of each of the 9 input parameters on the output parameter of the network (residual resource category), a sensitivity analysis is performed (Table 2).

Table 2. Sensitivity analysis of the neural network.

$N_т$	$N_н$	B	A	Q_c	M	$C_т$	$Q_н$	C
86.61	74.38	46.08	44.26	33.64	11.80	4.92	3.04	1.04

Table 2 shows that the two most significant (sensitive) parameters are parameters $N_н$, $N_т$. However, it is significant within the framework of this work that the parameters of the Weibull law of hardness distribution *B* and *A* take sufficient significance with respect to other parameters of the network. Thus, the shape parameter *B* is 3rd, the scale parameter *A* is 4th; the shear parameter *C* has the lowest significance.

To test the ANN operation, 10 test samples were prepared, which did not participate in the training process. The results of ANN built in the STATISTICA software package are presented in Table 3.

Table 3. Neural network results.

Sampling (test)	Target	Network output	Category-1 (confidence level)	Category-2 (confidence level)	Category-3 (confidence level)
1	3	3	0.04	0.18	0.78
2	3	3	0.05	0.11	0.84
3	1	1	0.66	0.12	0.22
4	2	2	0.16	0.69	0.15
5	1	1	0.68	0.21	0.11
6	2	2	0.26	0.61	0.13
7	2	2	0.04	0.73	0.23
8	3	3	0.03	0.01	0.96
9	2	2	0.15	0.73	0.12
10	1	1	0.65	0.21	0.14

The column "Target" lists a previously known category of residual resource; the column "Network output" lists the result of the neural network; the subsequent columns indicate the confidence probabilities of determining a particular category of residual resource of the hoisting crane, which correspond to the activation levels of the output neurons of the network.

Table 2 shows that the neural network correctly determined the residual life category of the crane depending on hardness degradation of its steel structures in 10 out of 10 cases, i.e. the test performance was 100%. At the same time, the average value of confidence levels of determining the category of residual resource of the hoisting crane was 0.73.

4 Discussion and conclusions

A new software has been created to estimate the residual life of hoisting cranes depending on hardness degradation of metal structures material. The programme can be used by specialists and experts as an intellectual decision-making support system for determining the residual resource of hoisting cranes. The detailed algorithm of neural network programming is given for possible use in other areas of technical diagnostics of various mechanical engineering objects.

By editing the code of the presented programme, users can supplement it with new training data, change the network architecture, introduce new network parameters, output additional resulting estimates, such as the confidence level of the network.

The author expresses gratitude to the team of "Musl" Innovation Cultural Centre LLC SSTU of Novocherkassk for the opportunity to use the data of hoisting cranes operation, namely the statistical database of typical damages of crane metal structures.

References

1. A. N. Beskopylny, E. M. Shcherban, S. A. Stelmakh, et al., Appl. Sci. **13**, 5413 (2023)
2. S. A. Stelmakh, E. M. Shcherban, A. N. Beskopylny, et al., Materials **15**, 6740 (2022)
3. A. N. Beskopylny, S. A. Stelmakh, E. M. Shcherban, et al., Appl. Sci. **12**, 10864 (2022)
4. A. N. Beskopylny, E. M. Shcherban, S. A. Stelmakh, et al., Appl. Sci. **13**, 1904 (2023)
5. S. R. Ghoreishi, T. Messager, P. Cartraud, P. Davies, Int. J. Mech. Sci. **49 (11)** (2007) 1251–1261 (2007) DOI: https://doi.org/10.1016/j.ijmecsci.2007.03.014.
6. A. Frikha, P. Cartraud, F. Treyssède, Int. J. Solids Struct. **50 (9)**, 1373–1382 (2013) DOI: https://doi.org/10.1016/j.ijsolstr.2013.01.010.
7. A. A. Korotkiy, A. V. Panfilov, R. V. Khvan, A. R. Yusupov, Transport, Mining and Mechanical engineering: Science and Production **18**, 73-79 (2023) DOI: https://doi.org/10.26160/2658-3305-2023-18-73-79
8. F. Foti, A. de Luca di Roseto, Int. J. Mech. Sci. **115-116**, 202–214 (206) DOI: https://doi.org/10.1016/j. ijmecsci.2016.06.016.
9. V. Spak, G. Agnes, D. Inman, Appl. Mech. Rev. **65**, 1–18 (2013) DOI: https://doi.org/10.1115/1.4023489.