Çukurova Üniversitesi Mühendislik Fakültesi Dergisi, 37(4), ss. 885-898, Aralık 2022
Cukurova University Journal of the Faculty of Engineering, 37(4), pp. 885-898, December 2022

Torque Estimation with Artificial Intelligence Methods in a Brushed Geared Dc Motor

Serkan BELLER[*1] **ORCID** *0000-0002-8475-6619*

[1]*Çukurova University, Vocational School of İmamoğlu, Natural Gas, and Fitting Technologies Department, Adana*

Geliş tarihi: 25.05.2022 Kabul tarihi: 27.12.2022

Atıf şekli/ How to cite: BELLER, S., (2022). Torque Estimation with Artificial Intelligence Methods in a Brushed Geared Dc Motor. Çukurova Üniversitesi, Mühendislik Fakültesi Dergisi, 37(4), 885-898.

Abstract

Today, with the advances in electricity-electronics, the usage areas of DC motors have increased considerably. DC motors have high starting torques and speed can be adjusted over a wide range. In the present experimental study, different weights connected to the motor shaft were rotated at different speeds, at variable distances, in the angle range of 0º-345º degrees. Thus, different torque values produced by the DC motor were observed. In some cases, the amount of torque produced at low rotational speeds may have non-linear values. This allows the use of artificial intelligence methods for accurate torque estimation. In the present study, different uses of Elman Backpropagation Neural Network (EBNN) and General Regression Neural Network (GRNN) are given for the estimation of the best torque values. Performance comparisons were made according to mean square error (MSE), regression coefficient (R^2), root square error (RSE), and mean absolute error (MAE) values.

Keywords: DC motor, Torque estimation, Artificial intelligence, EBNN, GRNN

Bir Fırçalı Redüktörlü Dc Motorda Yapay Zeka Yöntemleriyle Tork Tahmini

Öz

Günümüzde elektrik-elektronikteki ilerlemelerle birlikte DC motorların kullanım alanları oldukça artmıştır. DC motorlar yüksek başlangıç torklarına sahiptir ve hızları geniş bir aralıkta ayarlanabilir. Mevcut deneysel çalışmada motor miline bağlı olan farklı ağırlıklar, farklı hızlarda, değişken uzaklıklarda, 0º-345º derece açı aralığında döndürülmüştür. Böylece DC motorun ürettiği farklı tork değerleri gözlemlenmiştir. Bazı durumlarda düşük dönme hızlarında üretilen tork miktarı doğrusal olmayan değerlere sahip olabilmektedir. Bu durum doğru tork tahmini için yapay zeka metotlarının kullanılmasına imkan sağlamaktadır. Mevcut çalışmada en iyi tork değerlerinin tahmini için Elman Backpropagation Neural Network (EBNN) ve General Regression Neural Network (GRNN) ağlarının farklı kullanımlarına yer verilmiştir. Performans kıyaslamaları ortalama karesel hata (MSE), regresyon katsayısı (R^2), kök karesel hata (RSE), ve ortalama mutlak hata (MAE) değerlerine göre yapılmıştır.

Anahtar Kelimeler: DC motor, Tork tahmini, Yapay zeka, EBNN, GRNN

[*]Corresponding author (Sorumlu yazar): Serkan BELLER, *sbeller@cu.edu.tr*

1. INTRODUCTION

With the development of technology, electronic and motorized systems have become a part of life. DC motor is one of the most used motor types. It provides the necessary energy conversion with the windings and permanent magnets in the DC motor, which converts the straight electric current into mechanical energy. When electric current is applied to the windings in the motor, motion is obtained with the effect of the magnetic force, which is formed in the opposite direction to the permanent magnets inside the motor. The direction of this current must be reversed to create a permanent magnetic field opposite the permanent magnet. This change is made by the brushes in brushed motors, and by electronic speed control circuits in brushless motors [1].

Basically, DC motors, which can be diversified as brushed, brushless, stepper, and servo DC motors, each have different characteristics. Brushed DC motors are the most basic type of DC motor. The use of these motors is easy. But they have worn parts called brushes or coals that must be replaced periodically [1]. Also, DC motors are used in many projects with or without a reducer. A geared brushed DC motor (Pololu, 12V25 mm, 2250 RPM, 48 CPR Encoder [2]) was used in this study. The use of this DC motor is seen in Figure 1. The purpose of use of the reducer is to obtain higher torque by reducing the rotation speed of the motor. There are also gear systems that work in the opposite way, that is, reducing the torque and increasing the speed.

When previous studies are examined, Nouri et. al. (2008) proposed adaptive control for a nonlinear dc motor drive using Recurrent Neural Networks (RNN). A model-following adaptive control structure is suggested for the speed control of a nonlinear motor drive system [3]. Yang et. al. (2009) improved a mechatronic positioning system. It has a manipulator arm, a DC-motor-driven propeller assembly and a positioning control interface. Three different control methods are tried to regulate the displacement of the arm. These are, a fixed gain PID controller, a function-based variable gain PID controller and a fuzzy gain mixing PID controller.

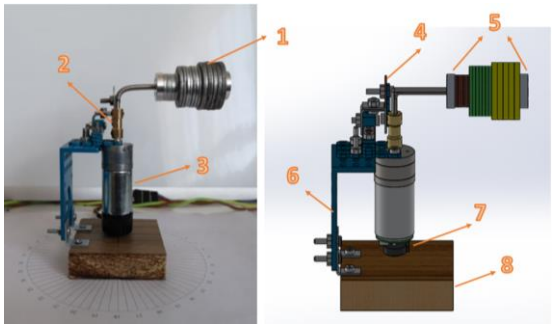


Figure 1. Experiment Set: 1. Test weights, 2. Coupling, 3. DC motor, 4. Stopper, 5. Limiters, 6. Motor holder, 7. Encoder, 8. Bottom table

Third method was more succesfull in suppressing the overshoot of the arm [4]. Reyes-Reyes et. al. (2010) presented a simple neuro-control law for controlling a geared DC motor. They formalized mathematically the stability of a geared DC motor. The formulation process was done using an artificial neural network and a neuro control law generated by Lyapunov-like analysis. The proposed approach was devoted to angular position regulation [5]. Premkumar et. al. (2014) suggested a novel controller for Brushless DC motor (BLDC). The proposed controller is based on ANFIS. The performance of the ANFIS is compared with PI controller, Fuzzy Tuned PID controller, and Fuzzy Variable Structure controller. Simulation was analyzed for varying load and varying set speed conditions. A more successful control strategy was developed for speed control of the BLDC motor with the ANFIS [6]. Ramadan et. al. (2014) presented an adaptive fuzzy logic speed controller for a DC motor, based on Field Programmable Gate Array (FPGA) hardware. Presented speed control was validated with good tracking results under distinct conditions [7]. Sabir et. al. (2016) considered designing of an optimum PID controller for DC motors of dual axis solar tracker system by using swarm intelligence techniques of Particle Swarm Optimization (PSO), Firefly Algorithm (FFA), and Cuckoo Search Algorithm (CSA) [8]. Rodr´ıguez-Molina et. al.

(2017) suggested an adaptive control for the speed regulation of the DC motor using meta-heuristic algorithms. Several adaptive controllers based on the optimizers of Differential Evolution (DE), PSO, Bat Algorithm (BAT), Firefly Algorithm (FFA), and Wolf Search Algorithm (WSA) are suggested. According to the results, PSO based controller is one of the best options [9]. El-samahy et. al. (2018) developed brushless DC motor tracking control using self-tuning fuzzy PID control and Model Reference Adaptive Control (MRAC). The aim of the algorithm was to force the rotor speed to follow the desired reference speed with good accuracy at all times. According to simulation results MRAC had better performance than self-tuning fuzzy PID controller [10]. Gamazo-Real et. al. (2022) proposed ANN-based position and speed sensorless estimation for brushless DC motors. According to the results overall position estimation improved many methods, and the speed estimation improved the traditional methods a little, but it was not very successful in advanced methods [11].

According to the previous studies, it was mostly focused on speed control in DC motors. Afterwards, position control is one of the subjects studied. On the other hand, torque control is not a much studied area in DC motors. Therefore, in this article, torque control is examined with artificial intelligence methods that have not been used before.

2. MATERIAL AND METHOD

When voltage is applied to a DC motor, current starts to flow through the rotor windings. Since these windings are in a magnetic field, each winding is subjected to a force. As the winding conductors are wrapped around the rotor, the rotor starts to rotate with the force it is exposed to. This angular force that causes rotation is called torque. In some cases, unexpected torque values can be seen at low speeds, depending on the DC motor characteristics. This variability can also occur from time to time at high speeds. The use of artificial intelligence methods is extremely beneficial in estimating unstable torque values at low speeds in DC motors working with high tooth backlash reducer. They also provide an estimate of the deformations and life cycles of the DC motor too. In the present study, machine learning methods Elman Back-propagation Neural Network (EBNN) and feed forward General Regression Neural Network (GRNN) were used. In both methods, they were trained with the supervised learning technique. Serial and parallel+serial uses of these networks are also included.

2.1. Calculation of Torque

The mass rotating around a center is shown in Figure 2. As seen in Equation 1 and Equation 2, multiplying the tangential acceleration (a_t) of the body P with the amount of mass (m) creates the torque force (F_t) in the same direction as the tangential velocity (V). Multiplying this torque force by the perpendicular distance (r) to the center (O) gives the numerical value of the torque (τ) (Equation 1, Equation 2).

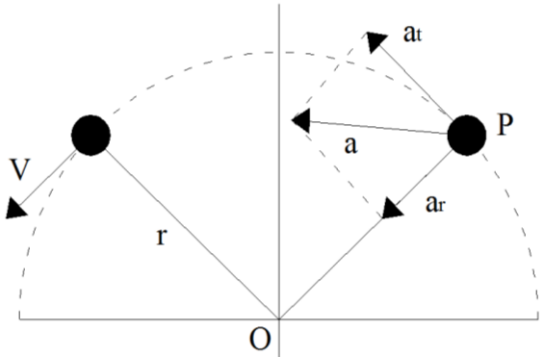


Figure 2. Rotating mass

$$\tau = F_t r \tag{1}$$

$$\tau = (ma_t)r \tag{2}$$

Torque is directly related to the mass of the rotating object (m), its tangential acceleration (a_t) and its distance from the center (r). Increasing one or all of these values also increases the amount of torque to be generated. Firstly in order to calculate the torque, the tangential velocity (V) of the rotating object must be calculated as in Equation 3.

$$v = \frac{x}{t} = \frac{2\pi r(\frac{345^0}{360^0})}{t} \quad (3)$$

Calculating the tangential velocity allows the angular velocity to be calculated as in Equation 4.

$$v = \omega r \ , \ \omega = \frac{v}{r} \quad (4)$$

Angular acceleration (α) is calculated as in Equation 5 with the angular velocity equation, which is one of the rotational motion equations with constant angular acceleration.

$$\omega_{last} = \omega_{first} + \alpha t \quad (5)$$

Knowing the angular acceleration (α) enables the tangential acceleration (a_t) to be found as in Equation 6.

$$\alpha = \frac{a_t}{r} \ , \ a_t = \alpha r \quad (6)$$

Thus, by finding the tangential acceleration (a_t), the torque calculation is done by Equation 2.

2.2. Elman Backpropagation Neural Network (EBNN)

Elman (1990) neural network has a multi-layer artificial neural network structure. The only difference is that it contains the hidden layer outputs as a parallel input layer. This artificial neural network starts with the input layer that receives the input data. Then, it continues the neural network operation by returning the initial output values from the hidden layer and adding them to the additional input layer. This structure is seen in Figure 3. Since the return is delayed, the additional layer is also called the delayed input layer [12]. Since the weights (*W*) of the recycles are constant in this network structure, the Elman network can also be called a partially reversible network. The learning of the Elman Network, is generalized delta learning rule as in multilayer perceptrons [13]. According to the delta rule, the weight values of neuron connections should be constantly changed in order to reduce the difference between the expected result and the result obtained from the network. This rule was developed according to this logic. The generalized delta rule has two stages. In the first stage, forward calculation is made in the network. In the second step, backward calculation is done [13]. While training these network structures, "nntool" neural network toolbox was used in Matlab/Simulink program. In all trials, the epoch number was 1000, goal was 0, and the max_fail value was 6. In this network structure, the toolbox determines the data split rates for training, testing and validation by itself. It cannot be interfered with. While specifying the data, it makes its selections scattered from the data pool. The graphical results of the training are given by the toolbox.

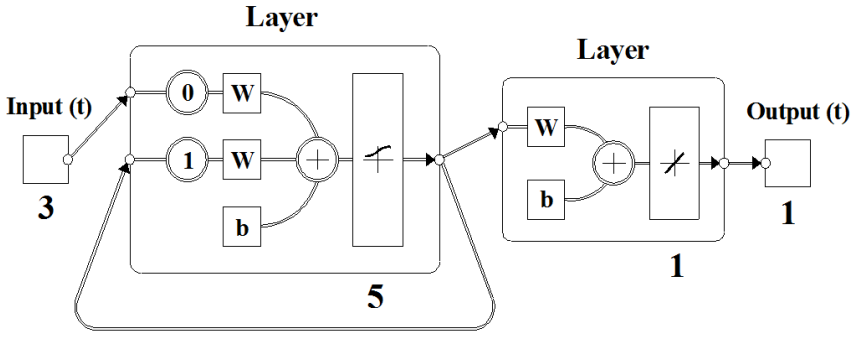


Figure 3. View of EBNN in MATLAB

2.3. General Regression Neural Network (GRNN)

GRNN was presented by Donald F. Specht in 1991. In this network, the learning phase is fast and can be effective with a small number of data. As seen in Figure 4, GRNN structure is a feedforward network structure. There are four layers in this network. The number of input neurons depends on the number of problem inputs. The number of neurons in the pattern layer is equal to the number of samples. The number of neurons in the summation layer is one more than the number of outputs. The number of neurons in the output layer is equal to the number of data types requested in the trainings [14]. During training, the user chooses a *spread* value. For the most suitable performance, the *spread* value must be selected between 0-1 [15]. In this study, maximum nine different spread values (100 between 0.000001) were tried. Since the numbers 10 and 100 are out of the range of 0-1, they were chosen for experimental purposes. Values of 1 and less were chosen by decreasing them by ten digits. Thus, it is easier to understand the result has evolved in which direction. In addition, while determining the smallest *spread* value, it was checked whether the Training R^2 result was 1, the Training RSE result was 0, and the Training MAE result was 0. Because, after reaching these values, the same results are obtained in every next 10-digit reduction of the *spread* value for the training and test results at the four digits after zero. So there is no need to lower the *spread* value further. For this reason, sometimes 7 different *spread* values and sometimes 9 different *spread* values were used in this study.

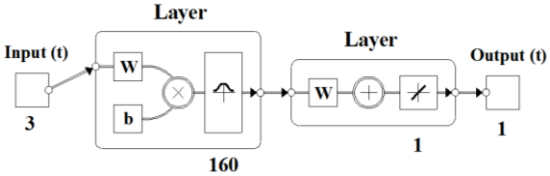


Figure 4. View of GRNN in MATLAB

2.4. System Design

In this study, Pololu 12V brushed geared DC motor (2250 RPM, 48 CPR quadrature encoder) [2], Pololu motor controller card 24V23A [16], Humusoft MF634 DAQ card [17], power source (12V, 16.7A), and a computer (Windows 7, 64 bit, Intel Core i5-7500, 3.40 GHz, 16384 MB ram) were used. DC motor speed capacity was selected also 56.25% percent.

According to the control block in Figure 5, three different data were used as input data. The first is the analog value for the DC motor velocity, the second is the test weight (g) value, and the third is the distance (mm) of the center of gravity of the test weight to the DC motor shaft. The output data is sweep duration (s) the 345^0 degrees angle. As seen in Figure 6, studies were performed on four different neural network (NN) usage methods in the Matlab/Simulink environment.

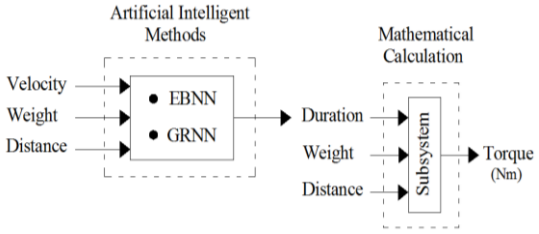


Figure 5. Control block

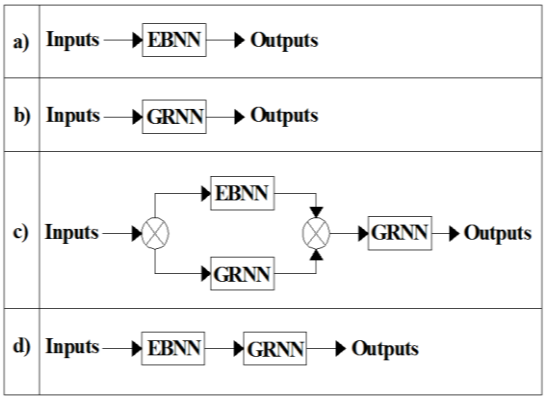


Figure 6. NN usage methods: a) EBNN, b) GRNN, c) (EBNN+GRNN)+GRNN, d) EBNN+ GRNN

The neural network (NN) methods in Figure 6a and Figure 6b, namely EBNN and GRNN, consist of using neural networks alone. In the Figure 6c, EBNN and GRNN blocks were first used in parallel, then the output data of both sides were retrained in a second serial connected GRNN block. Also, in Figure 6d, EBNN and GRNN blocks were used serially. Success rates of control blocks are shown in the results section.

2.5. Calculation of Error Values

Performance measurements of training and test results were made with Mean Squared Error (MSE), Regression Coefficient (R^2), Root Square Error (RSE), and Mean Absolute Error (MAE) values. They were calculated in Excel with the help of Equations 8, 9, 10, and 11, respectively.

Mean Squared Error (MSE) [18],

A_j = Actual values, P_j = Predicted values, n = Size of the data set, e_j =Error (Equation 7)

$$e_j = A_j - P_j \tag{7}$$

$$MSE = \frac{1}{n}\sum_{t=1}^{n} e_j^2 \tag{8}$$

Regression Coefficient (R^2) [19],

TSS = Total Sum of Squares, *RSS* = Residuals Sum of Squares

$$R^2 = \frac{TSS - RSS}{TSS} \tag{9}$$

Root Square Error (RSE) [20],

$$RSE = \sqrt{\sum_{t=1}^{n} \frac{(P_j - A_j)^2}{A_j}} \tag{10}$$

Mean Absolute Error (MAE) [21],

$$MAE = \frac{\sum_{t=1}^{n} \left| \frac{(P_j - A_j)}{A_j} \right|}{n} \tag{11}$$

MSE represents the average of the squared difference between the original and predicted values. It measures the variance of the residuals. R^2 represents the proportion of the variance in the dependent variable which is explained by the linear regression model. It summarizes the explanatory power of the regression model and is computed from the sums of squares terms. RSE is the square root of Squared Error. It measures the standard deviation of residuals. MAE expresses the average of absolute errors between forecast and actual value. It measures the average of the residuals in the dataset.

3. RESULTS

In this study, machine learning methods EBNN and GRNN were used. During the training, 160 lines of input data were used. While the DC motor speed value, the amount of test weight and the distance of the weight to the motor shaft were used as input data, the time to finish the 345^0 degrees tour was requested as output data in the trainings. The relevant experimental set measurement parameters are shown in Figure 7. For the testing of artificial neural networks, 49 lines of data were used. These data were the values found between the first and last row data of the training data, which had never been introduced to the networks before. Thus, an attempt was made to measure the capabilities of neural networks. In addition, performance evaluations of parallel and serial usage of these networks were made. The success rates of the networks were evaluated with the results of MSE, R^2, RSE, and MAE. With this study, it was seen how close the predictions produced by the networks to the true values. The fact that the results for R^2 were close to 1 and the results for MSE, RSE and MAE were close to 0 determined the success rates of the networks.

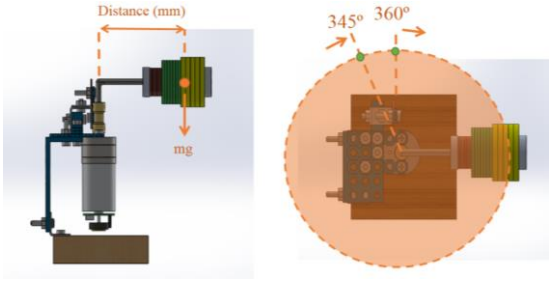


Figure 7. Experiment set measurement parameters

During study, different analog signal values were prefered to drive the DC motor at different speeds. Analog signal values were used during training in the range of 1.6 to 3.0 with an increment of 0.2. Also, analog signal values were used during the test in the range of 1.7 to 2.9 with an increment of 0.2. The values of these signals as tangential velocity on the DC motor shaft are shown in Figure 8a and Figure 9a. The values of tangential velocity are constantly changing according to the analog signal values, test weight amounts, and the distance of the center of gravity of these weights to the DC motor shaft. These changes are seen in Figure 8c, Figure 8d, Figure 9c, and Figure 9d, respectively. In Figure 8b and Figure 9b, completion times of 345^0 degrees tours for the training and test weights are shown. The fluctuations in the finishing times at low speeds can also be seen at high speeds from time to time. The best estimation method for these nonlinear cases can be achieved by using artificial intelligence methods.

During the training, 20 different weights were connected to the experimental set at 20 different distances. Trainings were held at 8 different speeds. The weights used are, 17, 24, 34, 42, 53, 58, 65, 78, 84, 88, 97, 104, 116, 124, 129, 139, 145, 152, 166, and 169 grammes, respectively. The distances used are, 42.5, 53.79, 59.75, 61.55, 61.85, 61.59, 60.78, 58.92, 57.85, 57.02, 62.77, 61.67, 59.29, 66.01, 65.53, 64.73, 64.19, 63.53, 61.86, and 61.47 millimeters, respectively. Thus, 160 lines of training data emerged. During the test, 7 different weights were connected to the experimental set at 7 different distances. Tests were carried out at 7 different speeds. The weights used are, 20.5, 48, 70, 92, 119, 143, and 162 grammes, respectively. The distances used are, 49.33, 64.24, 67, 64.58, 62.86, 62.06, and 58.4 millimeters, respectively. Thus, 49 lines of test data were obtained.

3.1. EBNN Experiment Results

As seen in Table 1, four different functions were used as training function during the training of 32 different networks. These are *Trainlm*, *Trainbfg*, *Trainoss*, *Trainscg*. Training was performed with 5 different neuron numbers (5, 10, 15, 20) and two different transfer functions (*Log*, *Tan*) in the hidden layer. In the output layer, the *Purelin* transfer function was preferred. According to Table 2, the best training results were obtained with the first-line neural network structure. The training function is *trainlm*, the hidden layer transfer function is *logsig*, and the number of neuron is 5. The output layer transfer function is *purelin* and the number of neuron is 1. In Figure 10, the training performance graphs of the training results are shown.

3.2. GRNN Experiment Results

In this study, seven different *spread* values (100, 10, 1, 0.1, 0.01, 0.001, 0.0001) were used tried to find the best performance. According to Table 3, it is seen that the training and test results do not change in values where the *spread* value is 0.01 or less for the four digits after zero. The 'Train R^{2}' result is 1, the 'Train RSE' result is 0, and the 'Train MAE' result is 0 at the fifth, sixth, and seventh *spread* values. So, the best performance was accepted by *spread* value of 0.01.

3.3. (EBNN+GRNN)+GRNN Experiment Results

In this control technique, EBNN and GRNN blocks were firstly used in parallel, then the output data of both sides were retrained in a second serial connected GRNN block. According to Table 4, seven different *spread* values (100, 10, 1, 0.1, 0.01, 0.001, 0.0001) were used tried to find the best performance.

Because; the 'Train R^{2}' result is 1, the 'Train RSE' result is 0, and the 'Train MAE' result is 0 at the seventh *spread* value. At lower *spread* values, training and test results will not change for the four digits after zero. The best performance was provided by *spread* value of 0.01.

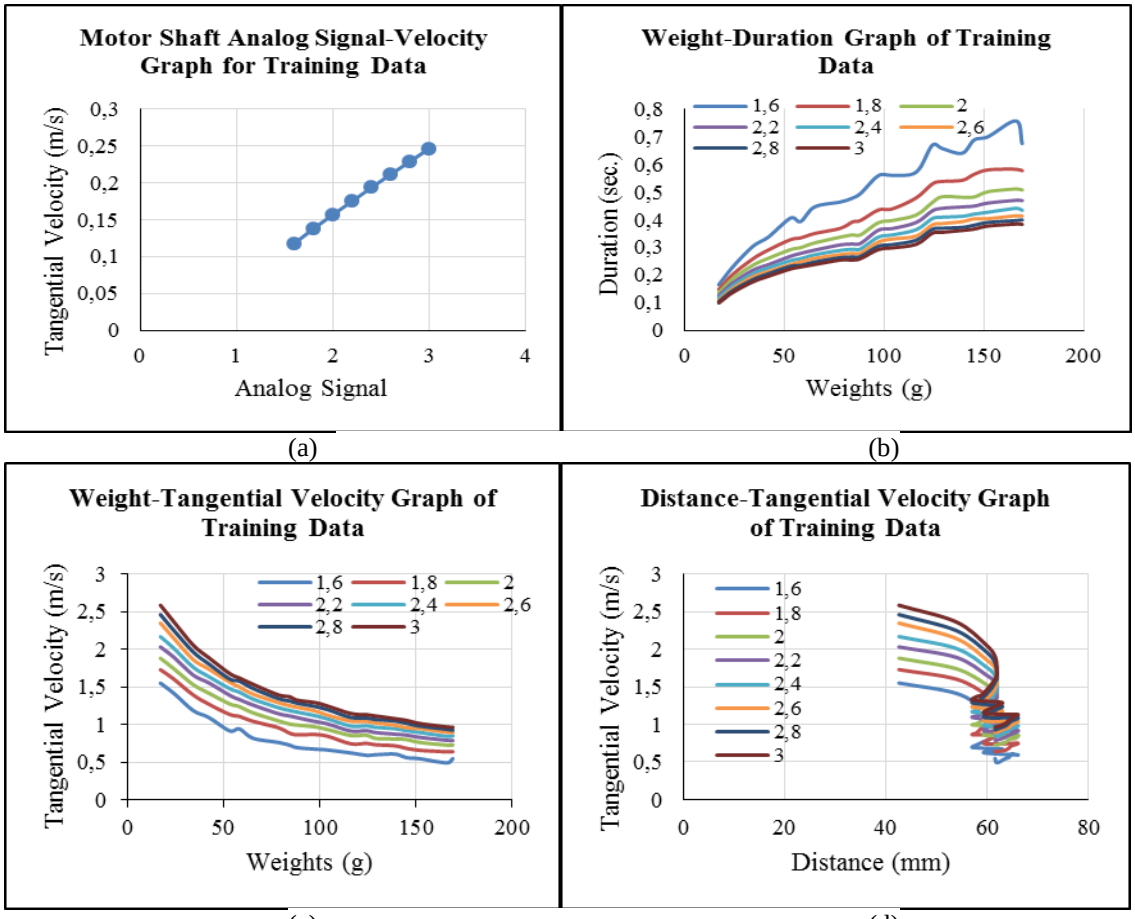
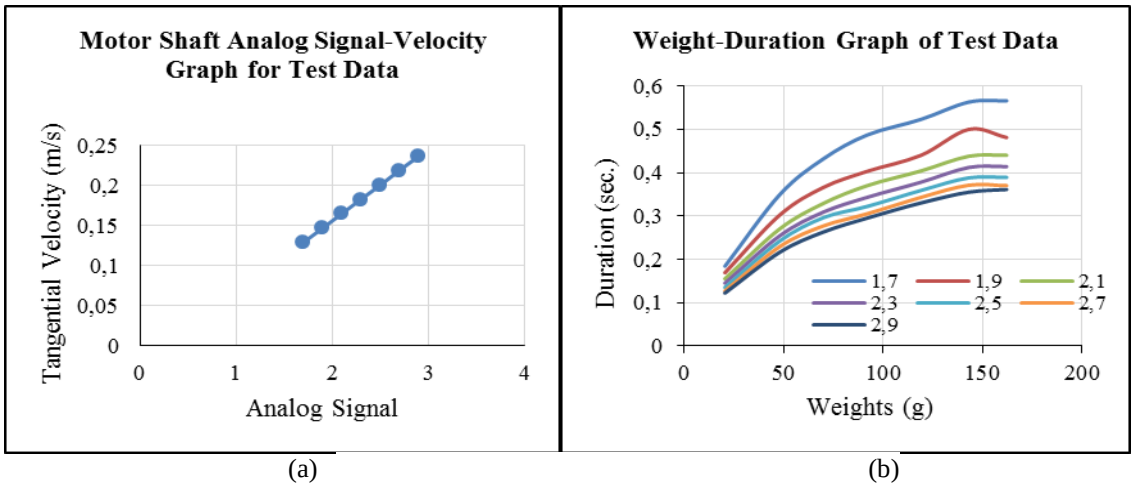
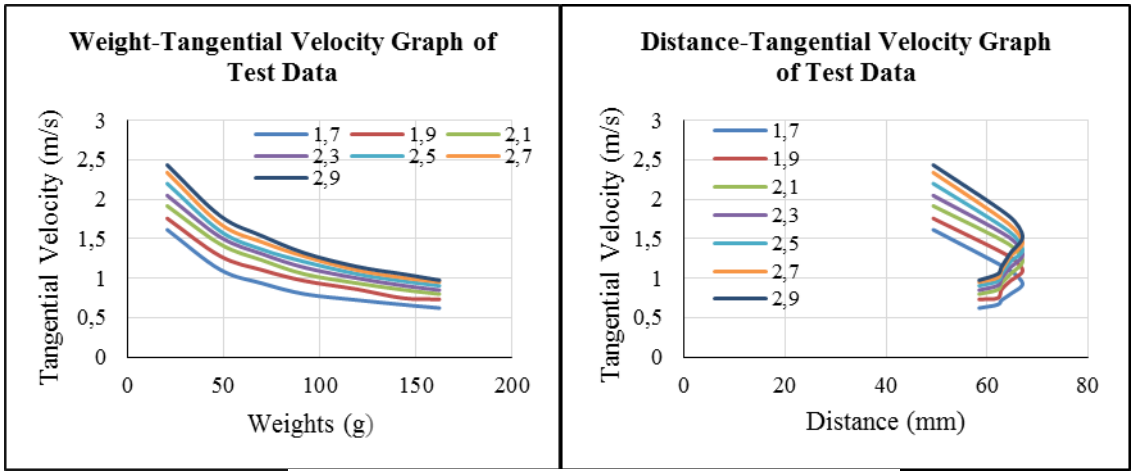


Figure 8. Training data; a) Analog signal-velocity graph b) Weight-duration graph c)Weight-velocity graph d) Distance-velocity graph





(c) (d)

Figure 9. Test data; a) Analog signal-velocity graph b) Weight-duration graph c)Weight-velocity graph d) Distance-velocity graph

3.4. EBNN+GRNN Experiment Results

In this method, EBNN and GRNN blocks were used serial connected. According to Table 5, nine different *spread* values (100, 10, 1, 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001) were used tried to find the best performance. Because; the 'Train R^2' result is 1, the 'Train RSE' result is 0, and the 'Train MAE' result is 0 at the nineth *spread* value. At lower *spread* values, training and test results will not change for the four digits after zero. The best performance was provided by *spread* value of 0.01.

Table 1. EBNN training features

	Network Type	Training Function	Layer 1 Tran. F.	Layer 1 Neuron	Layer 2 Tran. F.	Layer 2 Neuron
1	**Elman Back.**	**TRAINLM**	**LOGSIG**	**5**	**PURELIN**	**1**
2	Elman Back.	TRAINLM	LOGSIG	10	PURELIN	1
3	Elman Back.	TRAINLM	LOGSIG	15	PURELIN	1
4	Elman Back.	TRAINLM	LOGSIG	20	PURELIN	1
5	Elman Back.	TRAINLM	TANSIG	5	PURELIN	1
6	Elman Back.	TRAINLM	TANSIG	10	PURELIN	1
7	Elman Back.	TRAINLM	TANSIG	15	PURELIN	1
8	Elman Back.	TRAINLM	TANSIG	20	PURELIN	1
9	Elman Back.	TRAINBFG	LOGSIG	5	PURELIN	1
10	Elman Back.	TRAINBFG	LOGSIG	10	PURELIN	1
11	Elman Back.	TRAINBFG	LOGSIG	15	PURELIN	1
12	Elman Back.	TRAINBFG	LOGSIG	20	PURELIN	1
13	Elman Back.	TRAINBFG	TANSIG	5	PURELIN	1
14	Elman Back.	TRAINBFG	TANSIG	10	PURELIN	1
15	Elman Back.	TRAINBFG	TANSIG	15	PURELIN	1
16	Elman Back.	TRAINBFG	TANSIG	20	PURELIN	1
17	Elman Back.	TRAINOSS	LOGSIG	5	PURELIN	1
18	Elman Back.	TRAINOSS	LOGSIG	10	PURELIN	1

19	Elman Back.	TRAINOSS	LOGSIG	15	PURELIN	1
20	Elman Back.	TRAINOSS	LOGSIG	20	PURELIN	1
21	Elman Back.	TRAINOSS	TANSIG	5	PURELIN	1
22	Elman Back.	TRAINOSS	TANSIG	10	PURELIN	1
23	Elman Back.	TRAINOSS	TANSIG	15	PURELIN	1
24	Elman Back.	TRAINOSS	TANSIG	20	PURELIN	1
25	Elman Back.	TRAINSCG	LOGSIG	5	PURELIN	1
26	Elman Back.	TRAINSCG	LOGSIG	10	PURELIN	1
27	Elman Back.	TRAINSCG	LOGSIG	15	PURELIN	1
28	Elman Back.	TRAINSCG	LOGSIG	20	PURELIN	1
29	Elman Back.	TRAINSCG	TANSIG	5	PURELIN	1
30	Elman Back.	TRAINSCG	TANSIG	10	PURELIN	1
31	Elman Back.	TRAINSCG	TANSIG	15	PURELIN	1
32	Elman Back.	TRAINSCG	TANSIG	20	PURELIN	1

3.5. Comparison of Experiment Results

In the current study, two different artificial intelligence methods were used in four different ways to achieve the best results. When looked at Table 6, the serial-connected EBNN+GRNN method obtained the best results.

Table 2. EBNN training performances

	MSE	**Train R^2**	**Test R^2**	**Train RSE**	**Test RSE**	**Train MAE**	**Test MAE**
1	**5.0160E-06**	**0.9967**	**0.9975**	**0.1319**	**0.1249**	**0.0115**	**0.0241**
2	2.6103E-05	0.9970	0.9949	0.1240	0.1839	0.0103	0.0371
3	2.6134E-05	0.9960	0.9938	0.1423	0.1839	0.0111	0.0354
4	6.7697E-06	0.9970	0.9913	0.1238	0.2156	0.0099	0.0387
5	7.2691E-05	0.9965	0.9972	0.1391	0.1273	0.0120	0.0259
6	3.6900E-05	0.9962	0.9961	0.1612	0.1341	0.0160	0.0251
7	2.0247E-05	0.9965	0.9958	0.1349	0.1403	0.0114	0.0239
8	1.7891E-05	0.9955	0.9950	0.1596	0.1635	0.0121	0.0309
9	9.0484E-05	0.9928	0.9958	0.2178	0.1581	0.0226	0.0294
10	8.8255E-05	0.9924	0.9928	0.2286	0.1681	0.0258	0.0301
11	3.2254E-05	0.9958	0.9967	0.1659	0.1527	0.0163	0.0302
12	4.5193E-05	0.9955	0.9968	0.1659	0.1623	0.0159	0.0322
13	4.8156E-05	0.9944	0.9952	0.2180	0.1640	0.0214	0.0319
14	6.7363E-05	0.9955	0.9966	0.1784	0.1401	0.0180	0.0287
15	4.5285E-05	0.9951	0.9948	0.1734	0.1587	0.0172	0.0304
16	6.8692E-05	0.9940	0.9931	0.1974	0.1632	0.0198	0.0324
17	8.0814E-05	0.9919	0.9936	0.2313	0.1737	0.0246	0.0296
18	7.8762E-05	0.9931	0.9940	0.2200	0.2199	0.0228	0.0441
19	5.0604E-05	0.9919	0.9939	0.2281	0.1966	0.0233	0.0364
20	5.8189E-05	0.9945	0.9958	0.1806	0.1588	0.0178	0.0286
21	5.9081E-05	0.9940	0.9941	0.1987	0.2072	0.0211	0.0403
22	5.0640E-05	0.9939	0.9953	0.2014	0.1964	0.0211	0.0427
23	2.6075E-05	0.9944	0.9946	0.1840	0.1887	0.0178	0.0354
24	5.4275E-05	0.9936	0.9944	0.1988	0.1816	0.0203	0.0339

25	5.1542E-05	0.9955	0.9958	0.1749	0.1759	0.0178	0.0365
26	4.9899E-05	0.9951	0.9938	0.1720	0.1696	0.0171	0.0319
27	3.1127E-05	0.9958	0.9956	0.1606	0.1366	0.0158	0.0266
28	4.7405E-05	0.9956	0.9974	0.1605	0.1438	0.0146	0.0278
29	1.4031E-05	0.9962	0.9966	0.1531	0.1473	0.0142	0.0302
30	1.4414E-05	0.9960	0.9963	0.1521	0.1508	0.0143	0.0299
31	2.1622E-05	0.9957	0.9952	0.1596	0.1608	0.0148	0.0294
32	3.8453E-05	0.9951	0.9963	0.1813	0.1272	0.0171	0.0230

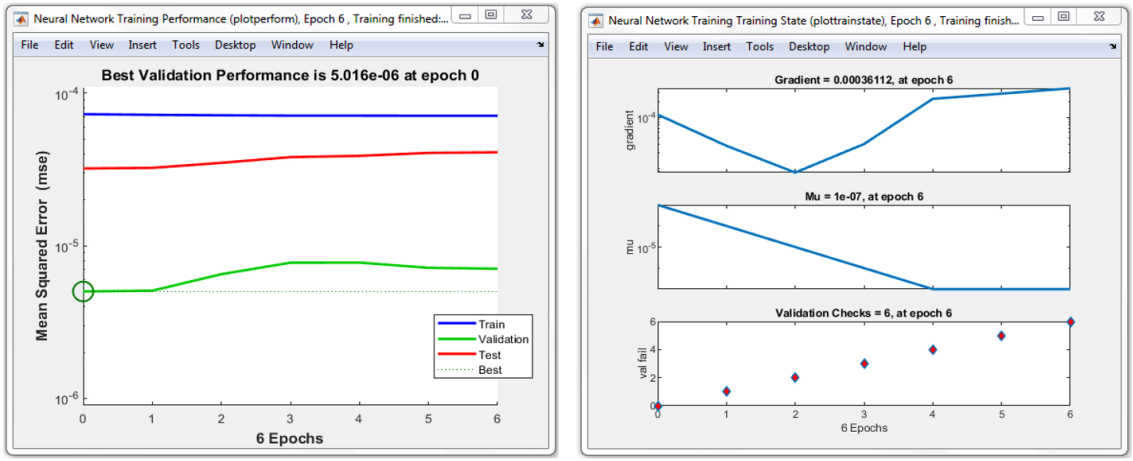


Figure 10. EBNN training performance graphs

Table 3. Performances of *spread* values for GRNN

Spread	**100**	**10**	**1**	**0.1**	**0.01**	**0.001**	**0.0001**
Train. R^2	0.6351	0.6533	0.7917	0.9997	**1**	1	1
Test. R^2	0.6793	0.7138	0.8157	0.9577	**0.9577**	0.9577	0.9577
Train. RSE	2.6291	1.4714	1.1236	0.0412	**0**	0	0
Test. RSE	1.3718	0.7173	0.5906	0.3285	**0.3285**	0.3285	0.3285
Train. MAE	0.3085	0.1642	0.1196	0.0030	**0**	0	0
Test. MAE	0.2939	0.1493	0.1286	0.0693	**0.0693**	0.0693	0.0693

Table 4. Performances of *spread* values for (EBNN+GRNN)+GRNN

Spread	**100**	**10**	**1**	**0.1**	**0.01**	**0.001**	**0.0001**
Train. R^2	0.9991	0.9991	0.9989	0.9948	**0.9997**	0.9999	1
Test. R^2	0.9860	0.9860	0.9837	0.9763	**0.9783**	0.9758	0.9760
Train. RSE	3.1884	3.1867	3.0299	0.5811	**0.0408**	0.0019	0
Test. RSE	1.7016	1.7007	1.6163	0.3688	**0.2325**	0.2448	0.2430
Train. MAE	0.3807	0.3805	0.3616	0.0681	**0.0045**	0.0001	0
Test. MAE	0.3630	0.3628	0.3446	0.0747	**0.0445**	0.0483	0.0474

Table 5. Performances of *spread* values for EBNN+GRNN

Spread	**100**	**10**	**1**	**$1*10^{-1}$**	**$1*10^{-2}$**	**$1*10^{-3}$**	**$1*10^{-4}$**	**$1*10^{-5}$**	**$1*10^{-6}$**
Trai. R^2	0.9967	0.9967	0.9966	0.9872	**0.9975**	0.9996	0.9999	0.9999	1
Test. R^2	0.9975	0.9975	0.9972	0.9910	**0.9977**	0.9963	0.9956	0.9955	0.9955
Trai. RSE	3.1884	3.1875	3.1070	0.9738	**0.1168**	0.0528	0.0231	0.0011	0
Test. RSE	1.7016	1.7012	1.6568	0.5233	**0.1148**	0.1289	0.1339	0.1338	0.1338
Trai. MAE	0.3807	0.3806	0.3709	0.1163	**0.0105**	0.0042	0.0008	2.49E-05	0
Test. MAE	0.3630	0.3629	0.3535	0.1084	**0.0238**	0.0255	0.0265	0.0265	0.0265

In Figure 11, torque values of 160 different positions for training and 49 different positions for testing are shown. Since eight different speeds for training and seven different speeds for testing are increased gradually, the torque values in the graphs are seen in increasing steps. Along with tangential velocity, the amounts of weight and the increases in the distance played a role in these changes too.

Table 6. Performance comparisons of the networks

	Network type	**Cor. predic. rate of train data for %5 error**	**Cor. predic. rate of test data for %5 error**	**Train R^2**	**Test R^2**	**Train RSE**	**Test RSE**	**Train MAE**	**Test MAE**
1	**EBNN+GRNN**	**% 98.75**	**% 93.87**	**0.9975**	**0.9977**	**0.1168**	**0.1148**	**0.0105**	**0.0238**
2	EBNN	% 98.12	% 93.87	0.9967	0.9975	0.1319	0.1249	0.0115	0.0241
3	(EBNN+GRNN)+GRNN	% 100	% 69.38	0.9997	0.9783	0.0408	0.2325	0.0045	0.0445
4	GRNN	% 100	% 40.81	1.0000	0.9577	0.0000	0.3285	0.0000	0.0693

According to Table 6, in the EBNN+GRNN method, the correct prediction rates for the 5% error value are 98.75% in the training data (561-1175 RPM) and 93.87% in the test data (616-1130 RPM). Most of the errors occurred at low motor speeds. During the training phase, 2 faults occurred at 1.6 motor speed (561 RPM), in the testing phase, 2 of the 3 faults occurred at 1.9 motor speed (701 RPM), and 1 at 2.7 motor speed (1046 RPM). The unstable torque values at low speeds in DC motors working with a gearbox with high tooth spacing force artificial intelligence methods. A similar results are seen in the study of Gamazo-Real et. al. [11]. The relative speed error is 5.5% when the motor speed is 325 RPM. At 725 RPM, it is 4.5%. Brushless DC motor relative speed error is 5% over the full motor speed range (125–1500 RPM) in [11].

4. DISCUSSION AND CONCLUSION

In DC motors, torque is very important in terms of speed control, position control, amount of load to be lifted, torque, and traction power. It has a direct relationship with the work to be done. In this study, artificial intelligence methods that will predict the most accurate torque values are presented. The success of these models was determined according to MSE, R^2, RSE, and MAE results. For the training of the models, 160 lines of data consisting of different speeds, weights and distances were used. For the test, a 49-row dataset, which was not used in the trainings and consisted of different speeds, weights and distances, was preferred. From machine learning methods Backpropagation EBNN network and feed-forward GRNN network were used for the prediction with different arrangements. In the EBNN+GRNN serial connected model, train R^2, test R^2, train RSE, test RSE, train MAE, and test MAE values were calculated as 0.9975, 0.9977, 0.1168, 0.1148, 0.0105, and 0.0238, respectively. This model is more successful than other models. The use of the serial-connected model had previously given successful results in the use of feed-forward NN+GRNN [22] too. In this study, serially

connected EBNN+GRNN usage provided similar success.

In future studies, it will be tried to increase the prediction performance of different models with different artificial neural network methods by increasing the data set. Thus, in robotic studies, more stable movements would be achieved despite unforeseen conditions with accurate torque estimations.

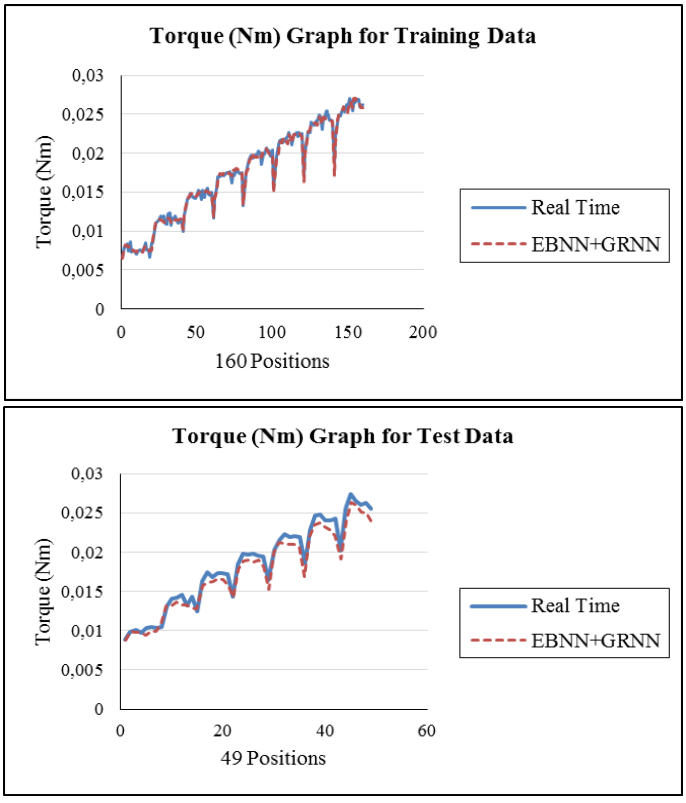


Figure 11. a) Torque graph of real-time and EBNN+GRNN training data b) Torque graph of real-time and EBNN+GRNN test data

5. REFERENCES

1. Direct Current Motor, https://www.sciencedirect.com/topics/engineering/direct-current-motor, Access date: 19.05.2022.
2. Pololu Brushed DC Motor, https://www.pololu.com/product/3213, Access date: 03.05.2022.
3. Nouri, K., Dhaouadi, R., Braiek, N.B., 2008. Adaptive Control of a Nonlinear Dc Motor Drive Using Recurrent Neural Networks. Applied Soft Computing, 8, 371–382.
4. Yang, S.F., Chou, J.H., 2009. A Mechatronic Positioning System Actuated Using a Micro DC-Motor-Driven Propeller–Thruster. Mechatronics, 19, 912–926.
5. Reyes-Reyes, J., Astorga-Zaragoza, C.M., Adam-Medina, M., Guerrero-Ramı´rez, G.V., 2010. Bounded Neuro-Control Position Regulation for a Geared DC Motor. Engineering Applications of Artificial Intelligence, 23, 1398–1407.
6. Premkumar, K., Manikandan, B.V., 2014. Adaptive Neuro-Fuzzy Inference System based Speed Controller for Brushless DC Motor. Neurocomputing, 138, 260–270.

7. Ramadan, E.A., El-bardini, M., Fkirin, M.A., 2014. Design and FPGA-Implementation of an Improved Adaptive Fuzzy Logic Controller for DC Motor Speed Control. Ain Shams Engineering Journal, 5, 803–816.

8. Sabir, M.M., Ali, T., 2016. Optimal PID Controller Design Through Swarm Intelligence Algorithms for Sun Tracking System. Applied Mathematics and Computation, 274, 690–699.

9. Rodr´ıguez-Molina, A., Villarreal-Cervantes, M.G., Aldape-P´erez, M., 2017. An Adaptive Control Study for a DC Motor Using Meta-Heuristic Algorithms. IFAC Papers On Line, 50-1, 13114–13120.

10. El-samahy, A.A., Shamseldin, M.A., 2018. Brushless DC Motor Tracking Control Using Self-Tuning Fuzzy PID Control and Model Reference Adaptive Control. Ain Shams Engineering Journal, 9, 341–352.

11. Gamazo-Real, J.C., Martínez-Martínez, V., Gomez-Gil, J., 2022. ANN-Based Position and Speed Sensorless Estimation for BLDC Motors. Measurement, 188, 110602.

12. Şen, Z., 2004. Yapay Sinir Ağları İlkeleri. Su Vakfı Yayınları, İstanbul, 183.

13. Öztemel, E., 2012. Yapay Sinir Ağları, Third Edition. Papatya Yayıncılık, İstanbul, 232.

14. Sağıroğlu, Ş., Beşdok, E., Erler, M., 2003. Mühendislikte Yapay Zeka Uygulamaları I, Yapay Sinir Ağları. UFUK Yayıncılık, Kayseri, 426.

15. Sahroni, A., 2013. Design of Intelligent Control System Based on General Regression Neural Network Algorithm. GSTF Journal on Computing (JoC), 2(4), 103–110.

16. Pololu Motor Controller Card, https://www .pololu.com/product/1383, Access date: 03.05.2022.

17. Humusoft Data Acquisition Card, https://www .humusoft.cz/datacq/mf634/, Access date: 03.05.2022.

18. Karunasingha, D.S.K., 2022. Root Mean Square Error or Mean Absolute Error? Use Their Ratio as Well, Information Sciences, 585, 609–629.

19. Chicco, D., Warrens, M.J., Jurman, G., 2021. The Coefficient of Determination R-Squared is More Informative than SMAPE, MAE, MAPE, MSE and RMSE in Regression Analysis Evaluation. PeerJ Computer Science, 7:e623, https://doi.org/10.7717/peerj-cs.623.

20. Schubert, A.L., Hagemann, D., Voss, A., Bergmann, K., 2017. Evaluating the Model Fit of Diffusion Models with the Root Mean Square Error of Approximation. Journal of Mathematical Psychology, 77, 29–45.

21. Myttenaere, M., Golden, B., Grand, B.L., Rossi, F., 2016. Mean Absolute Percentage Error for Regression Models. Neurocomputing, 192, 38–48.

22. Yavuz, H., Beller, S., 2021. An Intelligent Serial Connected Hybrid Control Method for Gantry Cranes, Mech. Syst. Sig. Process., 146, 107011.