TẠP CHÍ KHOA HỌC & CÔNG NGHỆ ĐẠI HỌC DUY TÂN
DTU Journal of Science and Technology 3(58) (2023) 23-30
TRƯỜNG ĐẠI HỌC DUY TÂN
DUYTAN UNIVERSITY

Training deep neural network for regression analysis with the generalized delta rule: A case study in modeling the shear strength of soil

Huấn luyện mạng nơ-ron thần kinh nhân tạo sâu dùng cho phân tích hồi quy dựa trên quy tắc delta khái quát: ứng dụng cho mô phỏng sức chịu cắt của đất

Hoang Nhat Duc[a,b*], Tran Xuan Linh[a,b], Nguyen Quoc Lam[a,b]
Hoàng Nhật Đức[a,b*], Trần Xuân Linh[a,b], Nguyễn Quốc Lâm[a,b]

[a]*Institute of Research and Development, Duy Tan University, Da Nang, 550000, Vietnam*
[a]*Viện Nghiên cứu và Phát triển Công nghệ cao Trường Đại học Duy Tân, Đà Nẵng, Việt Nam*
[b]*Faculty of Civil Engineering, Duy Tan University, Da Nang, 550000, Vietnam*
[b]*Khoa Xây dựng, Trường Đại học Duy Tân, Đà Nẵng, Việt Nam*

(Ngày nhận bài: 03/01/2023, ngày phản biện xong: 06/3/2023, ngày chấp nhận đăng: 06/5/2023)

Abstract

This article presents the method for training a deep artificial neural network (DANN) for regression analysis; this method is based on the generalized delta rule. To illustrate the rule, a DANN with two hidden layers is used. The model's construction is described in the form of mathematical equations. Subsequently, a DANN program is written in Visual C# .NET. This program is tested with the task of estimating the shear strength of soil samples.

Keywords: Deep artificial neural network; Regression analysis; Soil shear strength; Generalized delta rule.

Tóm tắt

Bài báo này trình bày phương pháp huấn luyện một mạng nơ-ron thần kinh nhân tạo sâu dùng cho phân tích hồi quy. Phương pháp này được xây dựng dựa trên quy tắc delta khái quát. Để minh họa cách sử dụng quy luật, một mạng nơ ron thần kinh với hai lớp ẩn được sử dụng. Cách thức huấn luyện mô hình được mô tả dưới dạng các công thức toán học. Sau đó, một chương trình phần mềm đã được phát triển với ngôn ngữ Visual C#. NET. Chương trình này được sử dụng để dự báo sức kháng cắt của mẫu đất.

Từ khóa: Mạng nơ-ron thần kinh nhân tạo sâu; Phân tích hồi quy; Sức kháng cắt của đất; Quy tắc delta khái quát.

1. Introduction

Artificial neural network (ANN) is an important tool for researchers in the field of construction engineering. These machine learning approaches were used successfully in various sub-fields, including construction materials [1-5], structural analyses [6-8], geotechnical engineering [9-15]. Particularly for regression analysis, the task of interest is to construct an ANN model that can well predict the value of a dependent variable. This model should be able to learn and generalize the

[*]*Corresponding Author:* Hoang Nhat Duc, Institute of Research and Development, Duy Tan University, Da Nang, 550000, Vietnam; Faculty of Civil Engineering, Duy Tan University, Da Nang, 550000, Vietnam.
Email: hoangnhatduc@duytan.edu.vn

mapping relationship between this dependent variable and a set of influencing factors (also called predictor variables).

Recently, deep learning has increasingly attracted the attention of the research community in the task of regression analysis [11, 16]. Successful implementations of various deep artificial neural network (DANN) models have been reported [16-18]. The success of DANN models can be explained by their outstanding capability of feature engineering and generalization. The deep structure of these models can be understood as a complex feature engineering operator, in which increasingly informative features are constructed from raw datasets [19]. In addition, the stacking of multiple layers can be used as a form of model regularization because a single layer must operate in accordance with its preceding and succeeding layers. Therefore, providing that there are sufficient training samples, DANN provides many advantages over shallow ANN.

The goal of the training phase of a DANN model is to minimize the output errors on a dataset via the adjustment of the network weights [20-22]. The squared error loss (SEL) function is often used for regression analysis [21]. Using SEL, an algorithm based on a stochastic gradient-descent (SGD) can be used to train the DANN. The SGD performs gradient-descent updates with respect to a set of randomly created network weights. Since the number of weights in a DANN is large, the process of updating those weights is complicated. The generalized delta rule [23] can be used to neatly summarize the way of updating the network's weights as follows:

$$w = w + \alpha \times \Delta \times z \quad (1)$$

where $\Delta = \varphi'(v_j^{(i)}) \times e_j^{(i)}$; α is the learning rate; $e_j^{(i)}$ denotes the error of the neuron j at the layer i. $\varphi'(v_j^{(i)})$ represents the derivative of the activation function with respect to the weighted sum of the inputs $v_j^{(i)}$.

Most importantly, the generalized delta rule significantly eases the formulation of the error back-propagation via the following rule: The error of the neuron j at the layer i ($e_j^{(i)}$) is computed by the errors of the previous neurons multiplied by their connecting weights. With a DANN, there are multiple hidden layers. Therefore, the quantity of Δ and e must be calculated for each neuron in a layer. **Fig. 1.1** demonstrates the generalized delta rule used for computing the error (e) of a neuron with respect to the errors of the neurons in the succeeding layer. Herein, $e_1^{(2)}$ and $e_2^{(2)}$ are known. $e_1^{(1)}$ is computed as follows:

$$e_1^{(1)} = e_1^{(2)} \times w_{11}^{(2)} + e_2^{(2)} \times w_{21}^{(2)} \quad (2)$$

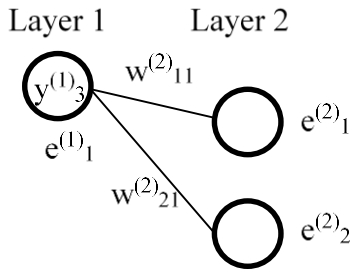

Fig. 1.1 Computing the error (e) of a neuron with respect to the errors of the neurons in the succeeding layer

2. Training a deep neural network

The previous section has described the method used to compute the error terms of a neuron with respect to the error terms of the neuron in the succeeding layer. This section presents the generalized delta rule used for training a DANN. The model of an ANN used for regression analysis is shown in **Fig. 2.1**. Herein, the network consists of three layers: input, hidden, and output layer. For the ease of presentation, a DANN with two inputs (x_1 and x_2) is used. Herein, the network consists of two hidden layers. The 1st hidden layer has three neurons; the 2nd hidden layer includes two neurons. There are three weight matrices in this

network: $W^{(1)}$ (which is the weights connecting the input to the hidden layer), $W^{(2)}$ (which connects the two hidden layers), and $W^{(3)}$ (which connects the hidden to the output layer). The training phase aims at adapting these two weight matrices to fit the collected dataset at hand. Herein, the sigmoid function is used as the activation function $\varphi()$ in the hidden layers.

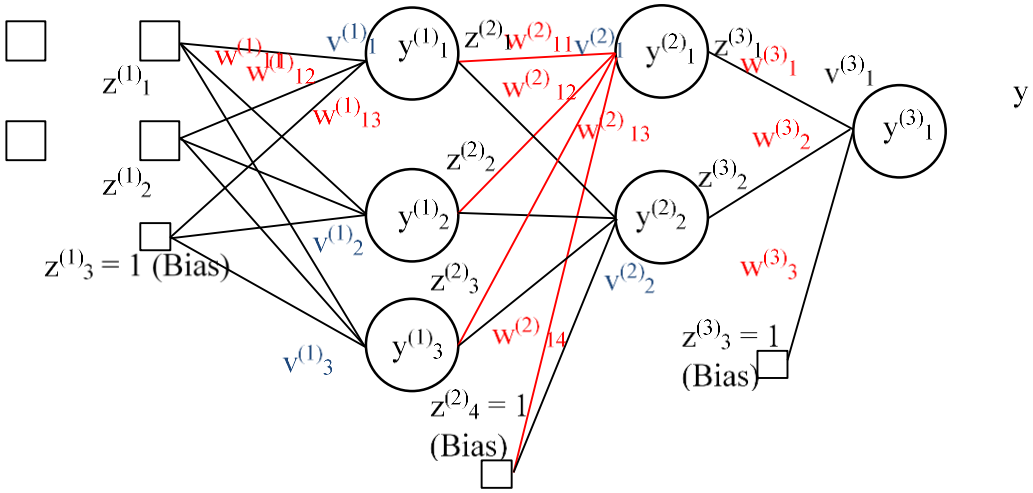

Fig. 2.1 The model of a DNN used for regression analysis

The three weight matrices in the network are given by:

$$W^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} & w_{13}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & w_{23}^{(1)} \\ w_{31}^{(1)} & w_{32}^{(1)} & w_{33}^{(1)} \end{bmatrix},\ W^{(2)} = \begin{bmatrix} w_{11}^{(2)} & w_{12}^{(2)} & w_{13}^{(2)} & w_{14}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} & w_{23}^{(2)} & w_{24}^{(2)} \end{bmatrix}, \text{and} W^{(3)} = \begin{bmatrix} w_1^{(3)} & w_2^{(3)} & w_3^{(3)} \end{bmatrix} \quad (3)$$

Let denote M_1 and M_2 as the number of neurons in the two hidden layer. D_X is the number of input variables. The size of the weight matrices can be specified as follows:

(i) The size of $W^{(1)}$ is M_1x(D_X +1).

(ii) The size of $W^{(2)}$ is M_2x(M_1 +1).

(iii) The size of $W^{(3)}$ is 1x(M_2 +1).

Given an input signal X, the input $Z^{(1)}$, weighted sum of the input $V^{(1)}$, and output $Y^{(1)}$ of the 1st layer are presented as follows:

$$Z^{(1)} = [z_1^{(1)}, z_2^{(1)}, z_3^{(1)}] = [x_1, x_2, 1]^T \quad (4)$$

$$V^{(1)} = [v_1^{(1)}, v_2^{(1)}, v_3^{(1)}] \quad (5)$$

where $v_1^{(1)} = z_1^{(1)} \times w_{11}^{(1)} + z_2^{(1)} \times w_{12}^{(1)} + z_3^{(1)} \times w_{13}^{(1)} = \sum_{d=1}^{D} z_d^{(1)} \times w_{1d}^{(1)}$

$$v_2^{(1)} = z_1^{(1)} \times w_{21}^{(1)} + z_2^{(1)} \times w_{22}^{(1)} + z_3^{(1)} \times w_{23}^{(1)} = \sum_{d=1}^{D} z_d^{(1)} \times w_{2d}^{(1)}$$

$$v_3^{(1)} = z_1^{(1)} \times w_{31}^{(1)} + z_2^{(1)} \times w_{32}^{(1)} + z_3^{(1)} \times w_{33}^{(1)} = \sum_{d=1}^{D} z_d^{(1)} \times w_{3d}^{(1)}$$

$$Y^{(1)} = [y_1^{(1)}, y_2^{(1)}, y_2^{(1)}] \quad (6)$$

where $y_1^{(1)} = \varphi(v_1^{(1)}) \quad y_2^{(1)} = \varphi(v_2^{(1)}) \quad y_3^{(1)} = \varphi(v_3^{(1)})$

Herein, $\varphi()$ denotes the sigmoid activation function, which is given by:

$$\varphi(x) = \frac{1}{1+e^{-x}} \quad (7)$$

The derivative of $\varphi(x)$ with respect to *x* is as follows:

$$\frac{\partial \varphi}{\partial x} = \varphi(x) \times (1 - \varphi(x)) \quad (8)$$

The input $Z^{(2)}$, weighted input $V^{(2)}$, and output $Y^{(2)}$ of the 2nd layer are given by:

$$Z^{(2)} = [z_1^{(2)}, z_2^{(2)}, z_3^{(2)}, z_4^{(2)}] = [z_1^{(2)}, z_2^{(2)}, z_3^{(2)}, 1] \quad (9)$$

where $z_1^{(2)} = y_1^{(1)} \qquad z_2^{(2)} = y_2^{(1)} \qquad z_3^{(2)} = y_3^{(1)} \qquad z_4^{(2)} = 1$

$$V^{(2)} = [v_1^{(2)}, v_2^{(2)}] \quad (10)$$

where $v_1^{(2)} = z_1^{(2)} w_{11}^{(2)} + z_2^{(2)} w_{12}^{(2)} + z_3^{(2)} w_{13}^{(2)} + z_4^{(2)} w_{14}^{(2)} = \sum_{u=1}^{4} z_u^{(2)} w_{1u}^{(2)}$

$$v_1^{(2)} = z_1^{(2)} w_{21}^{(2)} + z_2^{(2)} w_{22}^{(2)} + z_3^{(2)} w_{23}^{(2)} + z_4^{(2)} w_{24}^{(2)} = \sum_{u=1}^{4} z_u^{(2)} w_{2u}^{(2)}$$

$$Y^{(2)} = [y_1^{(2)}, y_2^{(2)}] = [\varphi(v_1^{(2)}), \varphi(v_2^{(2)})] \quad (11)$$

The input of the 3rd layer is given by:

$$Z^{(3)} = [z_1^{(3)}, z_2^{(3)}, z_3^{(3)}] = [z_1^{(3)}, z_2^{(3)}, 1] \quad (12)$$

The weighted sum of the input of the 3rd layer is as follows:

$$v_1^{(3)} = z_1^{(3)} \times w_1^{(3)} + z_2^{(3)} \times w_2^{(3)} + z_3^{(3)} \times w_3^{(3)} = z_1^{(3)} \times w_1^{(3)} + z_2^{(3)} \times w_2^{(3)} + z_3^{(3)} \times w_3^{(3)} = \sum_{u=1}^{3} z_u^{(3)} w_u^{(3)} \quad (13)$$

Since this is the last layer of the network, we simple have the following equations:

$$y_1^{(3)} = v_1^{(3)} \text{ and } y = y_1^{(3)} \quad (14)$$

The above equations are used to compute the response of a DANN model with respect to an input *x*. The next step is to employ the backpropagation framework to update the network's weight matrices. To do so, we need to calculate Δ for each node in each layer. The Δ in the last layer is given by:

$$\Delta^{(3)} = \varphi'(v_1^{(3)}) \times e_1^{(3)} = 1 \times e_1^{(3)} = e = t - y \quad (15)$$

where *t* and *y* denote the observed and the predicted output, respectively.

Hence, according to the generalized delta rule, the weight matrix in this layer is revised as follows:

$$w_u^{(3)} = w_u^{(3)} + \alpha \times e \times z_u^{(3)} \quad (16)$$

Since the 2nd layer includes two neurons and the sigmoid activation function is used, the Δ in the second layer is given by:

$$\Delta_1^{(2)} = \varphi'(v_1^{(2)}) \times e_1^{(2)} = y_1^{(2)}(1-y_1^{(2)}) \times (w_1^{(3)} e^{(3)}) = y_1^{(2)}(1-y_1^{(2)}) \times e \times (w_1^{(3)}) \quad (17)$$

$$\Delta_2^{(2)} = \varphi'(v_2^{(2)}) \times e_2^{(2)} = y_2^{(2)}(1-y_2^{(2)}) \times (w_2^{(3)} e^{(3)}) = y_2^{(2)}(1-y_2^{(2)}) \times e \times (w_2^{(3)}) \quad (18)$$

Accordingly, the 1st row of the weight matrix in the 2nd layer is updated as follows:

$$\begin{aligned} w_{1v}^{(2)} &= w_{1v}^{(2)} + \alpha \times \Delta_1^{(2)} \times z_v^{(2)} \quad = w_{1v}^{(2)} + \alpha \times \Delta_1^{(2)} \times z_v^{(2)} \\ &= w_{1v}^{(2)} + \alpha \times y_1^{(2)}(1-y_1^{(2)}) \times e \times (w_1^{(3)}) \times z_v^{(2)} \end{aligned} \quad (19)$$

where $\varphi'(v_1^{(2)}) = \varphi(v_1^{(2)}) \times (1-\varphi(v_1^{(2)}) = y_1^{(2)}(1-y_1^{(2)})$ because $\varphi()$ is the sigmoid function.

Similarly, the 2nd row of the weight matrix in the 2nd layer is revised as follows:

$$\begin{aligned} w_{2v}^{(2)} &= w_{2v}^{(2)} + \alpha \times \Delta_2^{(2)} \times z_v^{(2)} \quad = w_{2v}^{(2)} + \alpha \times \Delta_2^{(2)} \times z_v^{(2)} \\ &= w_{2v}^{(2)} + \alpha \times y_2^{(2)}(1-y_2^{(2)}) \times e \times (w_2^{(3)}) \times z_v^{(2)} \end{aligned} \quad (20)$$

Because the 1st layer consists of three neurons, the Δ in this layer is given by:

$$\Delta_1^{(1)} = \varphi'(v_1^{(1)}) \times e_1^{(1)} = y_1^{(1)}(1-y_1^{(1)}) \times e_1^{(1)} \quad (21)$$

$$\Delta_2^{(1)} = \varphi'(v_2^{(1)}) \times e_2^{(1)} = y_2^{(1)}(1-y_2^{(1)}) \times e_2^{(1)} \quad (22)$$

$$\Delta_3^{(1)} = \varphi'(v_3^{(1)}) \times e_3^{(1)} = y_3^{(1)}(1-y_3^{(1)}) \times e_3^{(1)} \quad (23)$$

According to the generalized delta rule, we have the following set of equations:

$$e_1^{(1)} = w_{11}^{(2)} e_1^{(2)} + w_{21}^{(2)} e_2^{(2)},\ e_2^{(1)} = w_{12}^{(2)} e_1^{(2)} + w_{22}^{(2)} e_2^{(2)}, \text{ and } \quad e_3^{(1)} = w_{13}^{(2)} e_1^{(2)} + w_{23}^{(2)} e_2^{(2)} \quad (24)$$

In general, e_v of the k^{th} layer is revised as follows: $e_v^{(k)} = \sum_{u=1}^{M_{k+1}} w_{uv}^{(k+1)} e_u^{(k+1)}$ (25)

where M_{k+1} is the number of neuron in the next layer.

Accordingly, the each row of the weight matrix in the 1st layer is revised as follows:

$$w_{1v}^{(1)} = w_{1v}^{(1)} + \alpha \times \Delta_2^{(1)} \times z_v^{(1)} = w_{1v}^{(1)} + \alpha \times y_1^{(1)}(1-y_1^{(1)}) \times e_1^{(1)} \times z_v^{(1)} \quad (26)$$

$$w_{2v}^{(1)} = w_{2v}^{(1)} + \alpha \times \Delta_2^{(1)} \times z_v^{(1)} = w_{2v}^{(1)} + \alpha \times y_2^{(1)}(1-y_2^{(1)}) \times e_2^{(1)} \times z_v^{(1)} \quad (27)$$

$$w_{3v}^{(1)} = w_{3v}^{(1)} + \alpha \times \Delta_3^{(1)} \times z_v^{(1)} = w_{3v}^{(1)} + \alpha \times y_3^{(1)}(1-y_3^{(1)}) \times e_3^{(1)} \times z_v^{(1)} \quad (28)$$

where v in these equations runs from 1 to D_X + 1; D_X is the number of input features. The last element of the vector $Z^{(1)}$ is always equal to 1 which corresponds to the bias in the 1st layer.

In general, the element of the weight matrix in the 1st layer is revised as follows:

$$w_{uv}^{(1)} = w_{uv}^{(1)} + \alpha \times y_u^{(1)}(1-y_u^{(1)}) \times e_u^{(1)} \times z_v^{(1)} \quad (29)$$

In summary, the training phase of the used DANN model is presented in the following pseudo code:

1: Normalize the dataset using Z-score equation

2: Prepare the training input (X) and output (T)

3: Randomly initialize the weight matrices W_0, W_1, W_2

4: Set the learning rate (α) and the maximum number of training epochs (MaxEp)

5: **For** k = 1 to MaxEp

6: **For** each data sample (x_i , t_i)in the training set:

7: Compute the predicted output y_i, refer to equations (4-14)

8: Revise W_0, W_1, W_2, refer to equations (15-29)

9: **End for**

10: **End for**

In the above pseudo code, the three weight matrices of the DANN are denoted as W_0, W_1, W_2. Herein, the index starts from 0 since the program is coded in Visual C#. The code used to revise these matrices is demonstrated in **Fig. 2.2**, **Fig. 2.3**, and **Fig. 2.4**.

```
double e2 = t - y;
double delta2 = e2;
for (int u = 0; u < M[1] + 1; u++)
{
    W2[0, u] = W2[0, u] + alpha * delta2 * Z2[u, 0];
}
```

Fig. 2.2 The C# code used for updating the weight matrix W_2

```
var e1 = new double[M[1], 1];
for (int k = 0; k < M[1]; k++)
{
    e1[k, 0] = e2 * W2[0, k];
}
var delta1 = new double[M[1], 1];
for (int k = 0; k < M[1]; k++)
{
    delta1[k, 0] = Y1[k, 0] * (1 - Y1[k, 0]) * e1[k, 0];
}
// mM.PrintMatrix("delta1", delta1);

for (int v = 0; v < M[1]; v++)
{
    for (int u = 0; u < M[0] + 1; u++)
    {
        W1[v, u] = W1[v, u] + alpha * delta1[v, 0] * Z1[v, 0];
    }
}
```

Fig. 2.3 The C# code used for updating the weight matrix W_1

```
var e0 = new double[M[0], 1];

for (int v = 0; v < M[0]; v++)
{
    // at row v
    for (int k = 0; k < M[1]; k++)
    {
        e0[v, 0] = e0[v, 0] + e1[k, 0] * W1[k, v];
    }
}
// mM.PrintMatrix("e0", e0);

var delta0 = new double[M[0], 1];
for (int v = 0; v < M[0]; v++)
{
    delta0[v, 0] = e0[v, 0] * Y0[v, 0] * (1 - Y0[v, 0]);
}

for (int u = 0; u < M[0]; u++)
{
    for (int v = 0; v < (Dx + 1); v++)
    {
        W0[u, v] = W0[u, v] + alpha * delta0[u, 0] * Z0[v, 0];
    }
}
```

Fig. 2.4 The C# code used for updating the weight matrix W_0

3. Model application

In this section, the DANN-based regression model is used to predict the shear strength of soil, which is a crucial property related to its capacity to resist failure. To train and test the DANN model, a dataset [24], including 249 samples, were used. The depth of sample, sand proportion, loam proportion, clay proportion, moisture content, wet density, dry density, void ratio, liquid limit, plastic limit, plastic index, and liquidity index are the 12 factors used for predicting the shear strength of soil (kg/cm^2).

The DANN are trained over 1000 epochs and the learning rate is 0.01. Six neurons are used in each hidden layer. Herein, 10% of the samples are used as the testing data; 90% of the samples are employed to train the model. **Table 1** summarizes the model's prediction results. The root mean square error (RMSE), mean absolute percentage error (MAPE), and coefficient of determination (R^2) [25] are used for assessing the results. **Table 1** also includes the prediction outcome of a shallow ANN with 12 neurons. This ANN model is also trained over 1000 epochs and with the learning rate of 0.01. Observably, with the same number of neurons, the DANN is able to achieve the performance better than that of the ANN. The DANN can explain 70% of the variance in the soil's shear strength and its MAPE is very close to 10%.

Table 1. The prediction results of the machine learning models

Indices	ANN	DANN
RMSE	0.049	0.048
MAPE (%)	11.39	10.28
R^2	0.67	0.70

4. Conclusion

The DANN is a highly potential method for fitting complex functional mapping tasks in civil engineering. This study presented the generalized delta rule for training a DANN model. This rule significantly simplified the formulation of the training process of a DANN featuring multiple hidden layers. Therefore, it can help facilitate the construction of computer programs based on the DANN. In this study, a DANN model was developed in Visual C# .NET and applied to estimating the shear strength of soil samples. Experimental results show good performance of the DANN model in comparison with the shallow ANN model.

References

[1] Ahmed H.U., Mohammed A.S., Mohammed A.A. (2022). "Proposing several model techniques including ANN and M5P-tree to predict the compressive strength of geopolymer concretes incorporated with nano-silica". *Environmental Science and Pollution Research*, DOI: 10.1007/s11356-022-20863-1.

[2] Cook R., Lapeyre J., Ma H., Kumar A. (2019). "Prediction of Compressive Strength of Concrete: Critical Comparison of Performance of a Hybrid Machine Learning Model with Standalone Models". *Journal of Materials in Civil Engineering* (31), 04019255. DOI: 10.1061/(ASCE)MT.1943-5533.0002902.

[3] Moradi M.J., Khaleghi M., Salimi J., Farhangi V., Ramezanianpour A.M. (2021). "Predicting the compressive strength of concrete containing metakaolin with different properties using ANN". *Measurement* (183), 109790. DOI: https://doi.org/10.1016/j.measurement.2021.109790.

[4] Tran T.-H., Hoang N.-D. (2016). "Predicting Colonization Growth of Algae on Mortar Surface with Artificial Neural Network". *Journal of Computing in Civil Engineering* (30), 04016030. DOI: 10.1061/(ASCE)CP.1943-5487.0000599.

[5] Hoang N.-D. (2022). "Machine Learning-Based Estimation of the Compressive Strength of Self-Compacting Concrete: A Multi-Dataset Study". *Mathematics* (10), 3771. DOI: 10.3390/math10203771.

[6] Barkhordari M.S., Massone L.M. (2022). "Failure Mode Detection of Reinforced Concrete Shear Walls Using Ensemble Deep Neural Networks". *International Journal of Concrete Structures and Materials* (16), 33. DOI: 10.1186/s40069-022-00522-y.

[7] Hoang N.-D. (2019). "Estimating Punching Shear Capacity of Steel Fibre Reinforced Concrete Slabs Using Sequential Piecewise Multiple Linear Regression and Artificial Neural Network". *Measurement* (137), 58-70. DOI: 10.1016/j.measurement.2019.01.035.

[8] Liu W., Moayedi H., Nguyen H., Lyu Z., Bui D.T. (2021). "Proposing two new metaheuristic algorithms of ALO-MLP and SHO-MLP in predicting bearing capacity of circular footing located on horizontal multilayer soil". *Engineering with Computers* (37), 1537-1547. DOI: 10.1007/s00366-019-00897-9.

[9] Kurnaz T.F., Kaya Y. (2019). "A novel ensemble model based on GMDH-type neural network for the prediction of CPT-based soil liquefaction". *Environmental Earth Sciences* (78), 339. DOI: 10.1007/s12665-019-8344-7.

[10] Pham T.A., Tran V.Q., Vu H.-L.T. (2021). "Evolution of Deep Neural Network Architecture Using Particle Swarm Optimization to Improve the Performance in Determining the Friction Angle of Soil". *Mathematical Problems in Engineering* (2021), 5570945. DOI: 10.1155/2021/5570945.

[11] Zhang Y., Xie Y., Zhang Y., Qiu J., Wu S. (2021). "The adoption of deep neural network (DNN) to the prediction of soil liquefaction based on shear wave velocity". *Bulletin of Engineering Geology and the Environment* (80), 5053-5060. DOI: 10.1007/s10064-021-02250-1.

[12] Achieng K.O. (2019). "Modelling of soil moisture retention curve using machine learning techniques: Artificial and deep neural networks vs support vector regression models". *Comput. Geosci.* (133), 104320. DOI: 10.1016/j.cageo.2019.104320.

[13] Gordan B., Koopialipoor M., Clementking A., Tootoonchi H., Tonnizam Mohamad E. (2018). "Estimating and optimizing safety factors of retaining wall through neural network and bee colony techniques". *Engineering with Computers*), 1-10. DOI: 10.1007/s00366-018-0642-2.

[14] Kurnaz T.F., Dagdeviren U., Yildiz M., Ozkan O. (2016). "Prediction of compressibility parameters of the soils using artificial neural network". *SpringerPlus* (5), 1801. DOI: 10.1186/s40064-016-3494-5.

[15] Kiran S., Lal B., Tripathy S.S. (2016). "Shear Strength Prediction of Soil based on Probabilistic Neural Network". *Indian Journal of Science and Technology* (9). DOI: 10.17485/ijst/2016/v9i41/99188.

[16] Asghari V., Leung Y.F., Hsu S.-C. (2020). "Deep neural network based framework for complex correlations in engineering metrics". *Adv Eng Inform* (44), 101058. DOI: 10.1016/j.aei.2020.101058.

[17] Zeng Z., Zhu Z., Yao W., Wang Z., Wang C., Wei Y., Wei Z., Guan X. (2022). "Accurate prediction of concrete compressive strength based on explainable features using deep learning". *Construction and Building Materials* (329), 127082. DOI: 10.1016/j.conbuildmat.2022.127082.

[18]Hoang N.-D. (2022). "Compressive Strength Estimation of Rice Husk Ash-Blended Concrete Using Deep Neural Network Regression with an Asymmetric Loss Function". *Iran. J. Sci. Technol. - Trans. Civ. Eng.* DOI: 10.1007/s40996-022-01015-4.

[19] Goodfellow I., Bengio Y., Courville A. (2016). *Deep Learning (Adaptive Computation and Machine Learning series).* United States: The MIT Press.

[20] Bullinaria J.A. (2004). "Introduction to Neural Networks". *Lecture Note, School of Computer Science - University of Birmingham <https://www.cs.bham.ac.uk/~jxb/NN/>*).

[21] Aggarwal C.C. (2018). *Neural Networks and Deep Learning.* United States: Springer.

[22] Silva I.N.d., Spatti D.H., Flauzino R.A., Liboni L.H.B., Alves S.F.d.R. (2017). *Artificial Neural Networks A Practical Course.* Switzerland : Springer.

[23] Kim P. (2017). *MatLab Deep Learning with Machine Learning, Neural Networks and Artificial Intelligence.* UK: Apress.

[24] Cao M.-T., Hoang N.-D., Nhu V.H., Bui D.T. (2020). "An advanced meta-learner based on artificial electric field algorithm optimized stacking ensemble techniques for enhancing prediction accuracy of soil shear strength". *Engineering with Computers (*38), 2185–2207. DOI: 10.1007/s00366-020-01116-6.

[25] Nguyen H., Cao M.-T., Tran X.-L., Tran T.-H., Hoang N.-D. (2022). "A novel whale optimization algorithm optimized XGBoost regression for estimating bearing capacity of concrete piles". *Neural Computing and Applications* (35), 3825–3852. DOI: 10.1007/s00521-022-07896-w.