Review of Artificial Neural Networks and A New Feed-Forward Network for Anchorage Analysis in Cracked Concrete

Article · November 2021

CITATIONS
0

READS
126

2 authors:

Salvio Aragao Almeida Junior
University of Toledo
3 PUBLICATIONS **4** CITATIONS
SEE PROFILE

Serhan Guner
University of Toledo
60 PUBLICATIONS **338** CITATIONS
SEE PROFILE

Review of Artificial Neural Networks and A New Feed-Forward Network for Anchorage Analysis in Cracked Concrete

Salvio A. Almeida Jr. and Serhan Guner

Synopsis: Soft computing applications through artificial intelligence (AI) are becoming increasingly popular in civil engineering. From concrete technology to structural engineering, AIs have provided successful solutions to various problems and greatly reduced the computational costs while achieving excellent prediction accuracy. In this study, a review of the main artificial neural network (ANN) types used in civil engineering is presented. Each ANN type is described, and example applications are provided. As a new research contribution, a deep feed-forward neural network (FFNN) is developed to predict the load capacities of post-installed adhesive anchors installed in cracked concrete, which is challenging and computationally expensive to achieve with conventional methods. The development of this FFNN is discussed, the influence of several parameters on its performance is demonstrated, and optimum parameter values are selected. In addition, a hybrid methodology that combines 2D nonlinear finite element (NLFE) techniques with the developed FFNN is briefly presented to account for real-life adverse effects in anchor analysis, including concrete cracking, wind-induced beam bending, and elevated temperatures. The results show that the developed network and methodology can rapidly and efficiently predict the load capacities of adhesive anchors installed into cracked concrete, accounting for the damage caused by the cracks, with high accuracies.

Keywords: adhesive anchors; anchorage to concrete; artificial intelligence; artificial neural network; cracked concrete; nonlinear finite element analysis.

Sálvio Aragão Almeida Júnior is a Master of Science Graduate from the University of Toledo, OH. His research expertise involves the numerical analysis of structures, including anchorage-to-concrete under hurricane loads and severe environmental exposure and soft computing applications. His research interests include applications of the finite element method for structural analysis and the use of artificial intelligence techniques, such as artificial neural networks and genetic algorithm, to find efficient solutions for computationally expensive problems.

ACI member **Serhan Guner** is an Associate Professor in the Department of Civil and Environmental Engineering at the University of Toledo, OH, USA. He received his Ph.D. from the University of Toronto, Toronto, ON, Canada. He is a member of the Joint ACI-ASCE Committee 447-Finite Element Analysis of Reinforced Concrete Structures, 370-Blast and Impact Load Effects, and 123-Research and Current Developments. His research interests include nonlinear modeling of structural response to extreme loads, shear effects, deep beams, disturbed regions, and machine learning.

INTRODUCTION

In recent years, there has been a significant increase in the application of soft computing techniques in civil engineering. These applications are seen in many civil engineering fields from earthquake engineering to water resources, as exemplified in Chandwani et al. (2013). These techniques differ from conventional analytical methods (also referred to as 'hard computing') in the sense that they seek approximate solutions (instead of exact ones) in a short time and with high accuracy. They are also significantly less computationally expensive than most conventional techniques, which adds to their popularity. Soft computing techniques are usually applied in the form of artificial intelligence (AI).

Artificial intelligence is the ability of an artificial entity (such as a computer program) to achieve specific goals under a variety of conditions, as defined by Luckey et al. (2020). It encompasses the fields of computer vision, robotics, machine learning, automated reasoning, knowledge representation, and natural language processing (Fig. 1). Among these, machine learning is especially beneficial to the engineering disciplines.

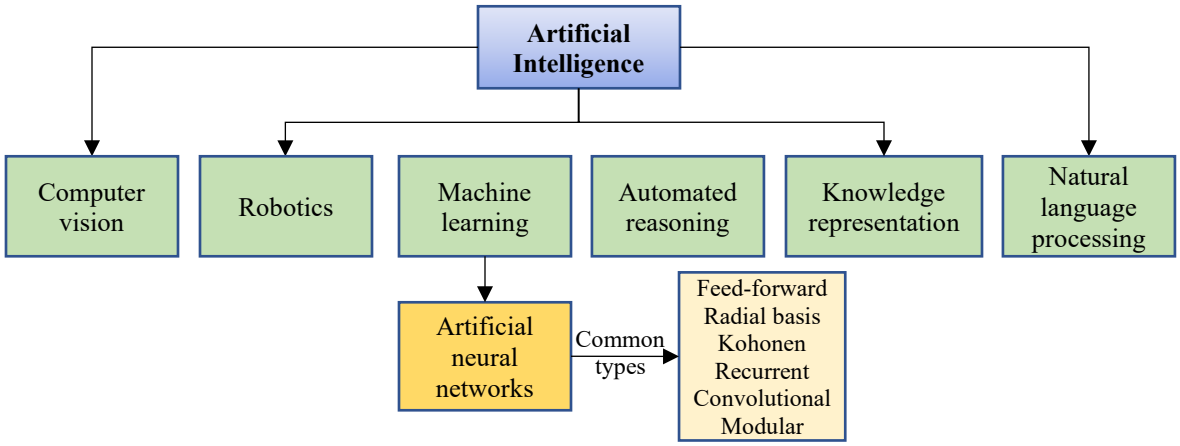


Fig. 1–Subcategories of artificial intelligence.

Machine learning is the category of AI that is responsible for learning patterns and generalizing the obtained knowledge to never-seen input. It is used for three main purposes: classification, regression, and clustering. One widely applied machine learning technique is the artificial neural network (ANN). ANNs find applications in virtually all fields of civil engineering for the classification and regression of data. They have been used to improve several aspects of construction management, as shown by Kulkarni et al. (2017), structural condition assessment and monitoring, and finite element mesh generation (e.g., Adeli, 2001), among others.

One relevant application of ANNs in structural engineering is to predict the load capacity of anchors installed into the concrete. These anchors are widely used to fasten non-structural components to the rooftop of buildings for wind load resistance. With the possible intensification of hurricanes impacting the United States due to climate change (Guo, 2017; Dinan, 2017; Cui and Caracoglia, 2016; Liu, 2014; Michalski, 2014; Mudd et al., 2014; Emanuel, 2013; Knutson et al., 2013), these anchors will play an ever-increasing role in mitigating hurricane damage. Several studies have applied ANNs to predict the load capacities of anchors installed into uncracked concrete (e.g., Gesoglu and Guneyisi, 2007; Ashour and Alquedra, 2005; Sakla and Ashour, 2005).

This paper describes the main ANN types and reviews their recent applications in civil engineering and other disciplines. To demonstrate how an ANN can be developed and applied to solve concrete-related problems, a deep FFNN is developed, trained, and tested for the purpose of predicting the load capacities of post-installed adhesive anchors installed in cracked concrete. Finally, a hybrid finite element-ANN methodology is developed to enable researchers and engineers to more accurately predict the anchor capacity while saving significant computational effort as compared to stand-alone nonlinear finite element (NLFE) analyses.

GENERAL STRUCTURE OF ARTIFICIAL NEURAL NETWORKS

Artificial neural networks consist of interconnected cells called neurons. Neurons modify themselves based on the information flowing through them, mimicking the learning process of biological neurons in the human brain (Fig. 2). Each neuron receives an input signal and plugs it into a mathematical function. This function is called an activation function. The role of this function is to convert the net input into the output of the neuron, also called 'output signal'. The output signals are modified by scalar values called 'weights' and 'biases' and then forwarded to the neurons in the subsequent layer. The neurons at the final layer output the ANN predictions.

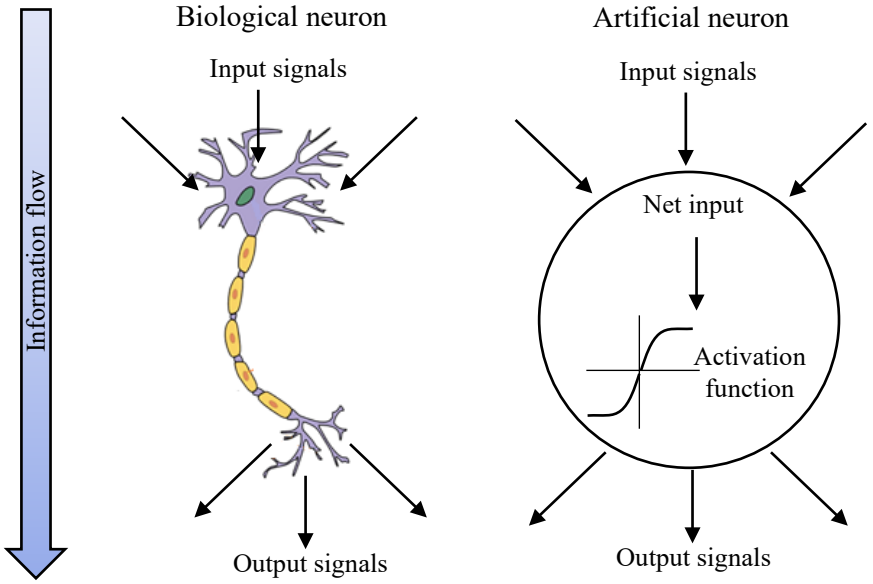


Fig. 2–Biological and artificial neurons.

A typical ANN will initially have random weights and biases with small values (e.g., between 0 and 1), which will lead to inaccurate results. To increase its accuracy, the ANN needs to learn either the patterns in the input data or the relationships between inputs and outputs. This will depend on the application. In both cases, the ANN needs be trained so that the weights and biases will be adjusted to give better results. Training of the ANN allows it to find approximate solutions to complex problems that would be challenging to solve with conventional techniques.

The training of the ANN may be supervised or unsupervised. The training is unsupervised when the dataset used has no detected patterns and is not human labeled. In this case, there is no correct answer prior to the training. A common application of unsupervised training is the clustering of data, in which the goal is to identify patterns in the input data and group this data in clusters that are not previously known. Alternatively, the training can be supervised when the target results are known. This type of training is typically used to find a relationship between the inputs and outputs for the classification of the data into categories, such as image recognition, or to predict a mathematical output such as regression analysis.

ANNs that aim to perform a regression analysis are commonly trained with the supervised backpropagation technique. In this method, several iterations are performed in which the error between the ANN outputs (i.e. predictions) and the targets is calculated and propagated back to the neurons in the previous layers to modify their weights and biases. The objective is to calculate how much the weights and biases influence the output and use the result to adjust their values, making the output closer to the targets. By iteratively repeating this procedure, the accuracy of the ANN predictions increases gradually and asymptotically, approaching a maximum value. The maximum accuracy depends on the ANN

layout (i.e., the number of hidden layers and neurons in each layer, also called ANN architecture) and the activation function used. In the end, if after a certain number of iterations (e.g., 10,000) the ANN has satisfactory accuracy, it is considered trained and can be used for testing and/or predictions; otherwise, a different layout or activation function must be used.

After the training stage, the ANN needs to be tested with new, never-seen data to ensure that it is able to make accurate predictions in new scenarios. The testing stage proceeds in a similar way to the training: the ANN is given the input data, forward propagation is applied, and the results are compared to the desired outputs. The main difference is that a backpropagation is not performed during the testing, thus the weights and biases remain unchanged. In other words, the weights and biases computed at the end of the training are final and used in all future predictions.

When the training and testing stages are successfully completed, the ANN is considered capable of making accurate predictions in both known and new scenarios and is ready to be used. It is important to note that ANNs are not appropriate for extrapolation of data. Therefore, they should not be used with the inputs outside the range of the parameters used in the training.

TYPES OF ARTIFICIAL NEURAL NETWORKS

The main types of ANNs are feedforward, radial basis, Kohonen self-organizing, recurrent, convolutional, and modular. Feedforward neural networks (FFNNs) are one of the simplest types of ANNs (Fig. 3). Conventional FFNNs consist of an input layer, one (and only one) hidden layer, and one output layer while deep FFNNs have multiple hidden layers (Fig. 3). Their name comes from the information flow which always goes in one direction, from inputs to outputs (i.e., forward). They are employed in several studies to predict the load capacities of structural members, such as concrete anchors (e.g., Gesoglu and Guneyisi, 2007; Ashour and Alquedra, 2005; Sakla and Ashour, 2005). In all of these applications, the authors included the anchor embedment depth, anchor diameter, and the concrete compressive strength as typical input variables. The FFNNs developed in each of these studies were successfully employed to accurately predict the pullout load capacities of cast-in and post-installed concrete anchors. They were, however, tested and trained with specimens installed in uncracked concrete only, and thus are limited to applications with this concrete condition.

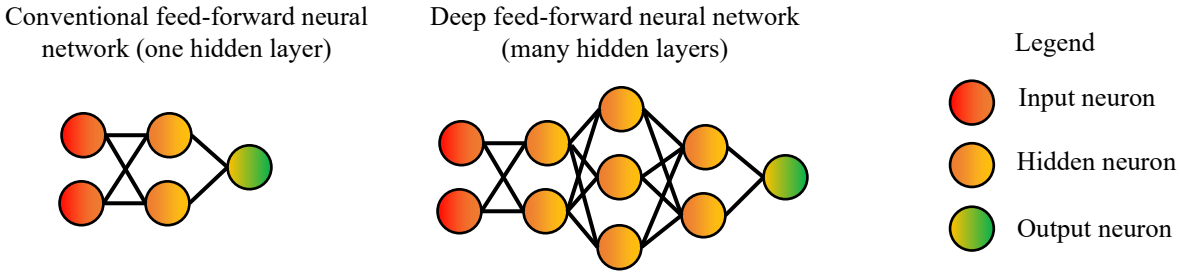


Fig. 3–Structure of conventional and deep feed-forward neural networks.

Radial basis neural networks (RBNNs) have a similar structure to FFNNs by having one input, one hidden, and one output layer. They differ in the type of activation function used (Fig. 4). RBNNs are applied in the civil engineering and medical fields. Chen et al. (2020) recently used such a network to evaluate the risk of natural flood disasters. Sun et al. (2019) successfully developed an RBNN to predict the 7-day and 28-day compressive strength of concrete based on the common physical parameters, such as the water-to-binder ratio, sand ratio, and weight of gravel. In the medical sciences, Cheruku et al. (2017) used an RBNN to diagnose patients with diabetes.

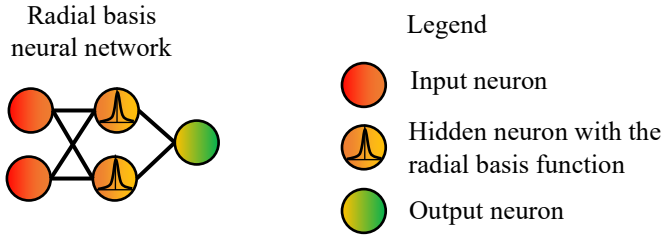


Fig. 4–Structure of a radial basis neural network.

In RBNNs, each neuron has a center vector which is used in the calculation of the activation function. The neuron centers are typically calculated by applying a technique called K-means clustering. In this technique, the input data is grouped into clusters and the centroids of the clusters become the centers of the neurons. After the neuron centers are obtained, the training of the RBNN begins.

During training, the weights of the neurons are modified based on the distance between their center and the inputs. The hidden layer applies a weighted summation on the inputs and passes this summation through a nonlinear radial basis function. The most common choice for a radial basis function is a Gaussian function shown in Eq. 1. Subsequently, the output layer performs a linear regression on the outputs of the hidden layer. Due to their formulation, RBNNs have fast convergence and are guaranteed to converge to the global optimum solution (Rosada et. al., 2019). RBNNs are trained similarly to FFNNs by adjusting the weights to improve the network's accuracy.

$$y = \exp(\beta \sum \|x_i - c_i\|)^2 \tag{1}$$

where x_i is the input of neuron i, c_i is the center of neuron i, β is a scalar coefficient and y is the neuron output.

Kohonen self-organizing neural networks (KSONNs) are typically a two-layer network, where the neurons in the input and output layers have one and two dimensions, respectively, as shown in Fig. 5. They are substantially different from the conventional neural networks in both motivation and operation. In terms of motivation, KSONNs are mainly used to classify data into regions. They can represent multi-dimensional data in two dimensions, creating a map of the input. Due to their ability to map multi-dimensional input into a two-dimensional region, KSONNs are typically used to classify data. Applications include the classification of individuals based on iris image recognition (e.g., Winston et al., 2019), and classification of fault in induction motors (e.g., Skowron et al., 2019).

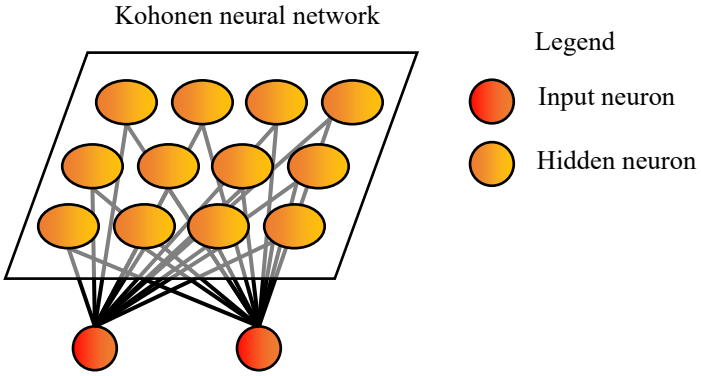


Fig. 5–Structure of a Kohonen neural network.

KSONNs also operate differently from conventional FFNNs, as seen in their typical algorithm, explained by Akinduko et al. (2016). For example, each neuron is connected to all input nodes. The neurons have specific coordinates and initially compete for the ownership of the input pattern, which helps find clusters in the data. The closest neuron (i.e., with the minimum Euclidean distance) is selected as the winner and excites its neighbor neurons. The weights of the winning neuron and its neighbors are adjusted according to a given rule so that they get closer to the input data. At the end, the input data is mapped according to the final positions of the neurons and classified according to their place in the map.

Recurrent neural networks (RNNs) use previous outputs as inputs in subsequent iterations during the training stage (Fig. 6). According to Scarpiniti et al. (2020), this recurrent use of outputs allows the network to discover correlations between subsequent events in time. Examples of applications of RNNs include audio classification (e.g., Scarpiniti et al., 2020), rain precipitation forecast (e.g., Asanjan et al., 2018), and others such as learning word representation, speech recognition, and music generation, as detailed by Amidi and Amidi (2019).

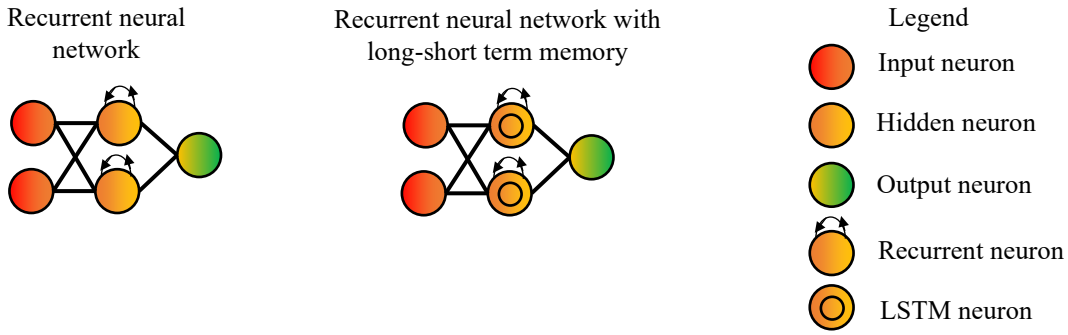


Fig. 6–Structure of a recurrent neural network.

RNNs commonly encounter the problems of exploding or vanishing error gradients during the backpropagation operations in the training. This occurs due to the large number of multiplications involved in the process. To help prevent these problems, modern RNNs include long-short term memory (LSTM) units. These units introduce a memory cell—a particular type of neuron (or cell) that preserves information about the error propagated in previous backpropagation steps. Memory cells have four gates: an input gate, a neuron with a self-recurrent connection, a forget gate, and an output gate, as described by Scarpiniti et al. (2020). These structures allow the cells to forget the irrelevant (e.g., noisy) data and give higher importance to more relevant data, which, according to Asanjan et al. (2018), gives the network a higher capacity for learning longer-range behaviors.

Convolutional neural networks (CNNs) are deep neural networks that use convolution in lieu of general matrix multiplication in at least one of its layers, as explained by Goodfellow et al. (2016). According to Amidi and Amidi (2018), they are generally composed of four layers: input, convolutions, pooling, and fully connected layers, as shown in Fig. 7. They have been successfully applied in image processing (e.g., Chen et al., 2017), voice recognition (e.g., Albawi et al., 2017), and enzyme classification (e.g., Amidi et al., 2018).

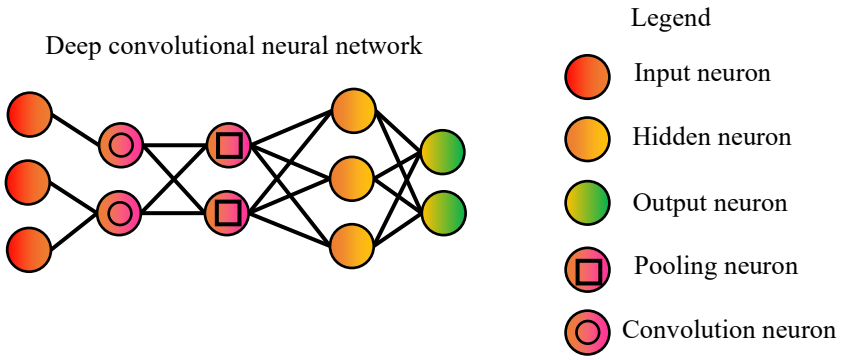


Fig. 7–Structure of a deep convolutional neural network.

A major advantage of the convolution is that it drastically reduces the number of parameters needed by the ANN. According to Albawi et al. (2017), in the case of image recognition the number of parameters needed drops from millions to a few tens by connecting each input neuron to only one neuron in the convolution layer (instead of a full connection to all neurons) and connecting the neurons in the following layer only to a fixed local region of the image while keeping the same weights for each connection between the local regions and the hidden neurons. Subsequently, in the pooling layer, the sampling size is reduced, which reduces the complexity of the network (such as reducing the resolution of an image). One of the most common pooling methods is to partition the sample into sub-regions and extract only the maximum value of those regions. These sub-regions can have some overlap for improved efficiency. The fully connected layer has all neurons connected to the ones in the adjacent layers and works similarly to an FFNN.

Modular neural networks (MNNs) are a combination of other types of ANNs working together (Fig. 8). Due to their configuration, each submodule network has a specialized role and prepares the input for the following network. They are used to predict the sorption efficiency of lead and manganese ions (e.g., Ilaboya and Izinyon, 2019) and the response aggregation for a COVID-19 time series in Mexico (e.g., Melin et al., 2020).

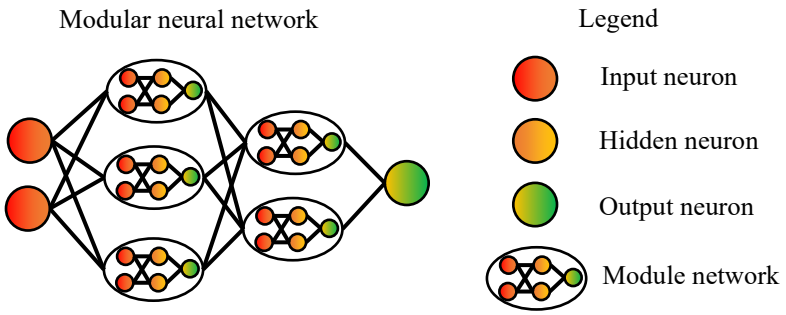


Fig. 8–Structure of a modular neural network.

DEVELOPMENT OF A DEEP FFNN FOR ANCHORAGE TO CRACKED CONCRETE

To demonstrate in detail how an ANN can be developed and applied to solve a concrete-related problem, this section will present the development, training, and testing of a deep FFNN to predict the experimental (i.e., nominal) load capacities of post-installed adhesive anchors installed in cracked concrete under short-term static loading. Ultimate load capacities can be obtained by applying appropriate material resistance factors depending on the design code used. The trained and tested FFNN can be downloaded in EXCEL format for ease of use at Almeida Jr and Guner (2019a) and an educational video of its use is available at Almeida Jr and Guner (2019b). The developed FFNN is part of a hybrid finite element-ANN methodology which is fully discussed at Almeida Jr and Guner (2020).

The feedforward network type is selected due to its simplicity and successful previous applications in anchorage to concrete. A computer code is developed to generate, train, and test FFNNs with any number of layers and neurons in each layer. The C++ programming language is used since it has the necessary mathematical tools to perform the calculations efficiently. Using this code, several network layouts (i.e., architectures) are trained and tested and the best configuration (i.e., with the optimum balance between accuracy and simplicity) is selected to consider cracking of concrete when predicting the load capacities of adhesive anchors.

FFNN formulation

The five input neurons considered in this study include: the concrete compressive strength (f'_c), anchor diameter (d_a), anchor embedment depth (h_{ef}), annular gap (A_g), and concrete crack width (w_{cr}), as shown in Fig. 9. The output layer contains only one neuron that returns the anchor load capacity (P).

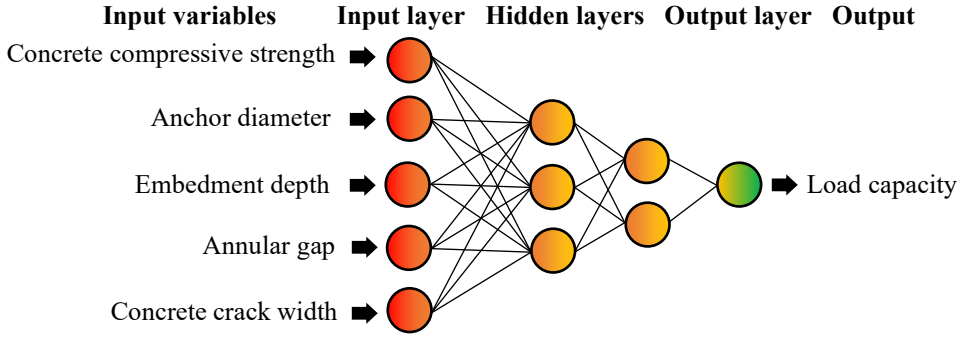


Fig. 9–Layout, input variables, and output variable of the chosen ANN.

During the forward propagation, each neuron receives inputs from the neurons in the previous layer and calculates a weighted sum of them and the corresponding bias, according to Eq. 2. The result of this weighted sum, the net input (x), is passed through the activation function in Eq. 3, to obtain the neuron output (y). Each output is passed to all the neurons in the following layer, becoming their input to be used again in Eq. 2. This happens sequentially for each layer until the final outputs are obtained in the last layer.

$$ {}^{L}x_i = \sum_{k=1}^{{}^{L-1}n} \left({}^{L-1}w_{ki} \; {}^{L-1}y_k \right) + {}^{L}b_i \tag{2} $$

$$ {}^{L}y_i = f({}^{L}x_i = x) = \frac{1}{1+e^{-x}} \tag{3} $$

where ${}^{L}x_i$ is the input of neuron i in layer L; ${}^{L-1}w_{ki}$ is the weight between neuron k in layer $L-1$ and i in layer L; ${}^{L-1}y_k$ is the output of neuron k in layer $L-1$; and ${}^{L}b_i$ is the bias of neuron i in layer L.

The activation function chosen for all the neurons is the sigmoid because it is smooth and differentiable over its entire length and has been used in several successful applications (e.g., Gesoglu and Guneyisi, 2007; Ashour and Alquedra, 2005; Sakla and Ashour, 2005). Since the sigmoid activation function only outputs values between 0 and 1, the inputs and target outputs are linearly normalized to a range of 0.1 and 0.8 through Eq. 4 before being used in the developed FFNN. To obtain the load capacity, the outputs are converted back to the physical range (i.e., unnormalized) using Eq. 5. Although not mandatory, the normalization of the inputs and target outputs speeds up the learning of the FFNN during the training. Other normalization techniques (such as Z-score and median) may also be used; they are not tested in this study since the linear normalization method used led to a satisfactory learning rate.

$$ In_{normalized} = 0.1 + \frac{0.8-0.1}{In_{\max} - In_{\min}} \times (In_{physical} - In_{\min}) \tag{4} $$

$$ Out_{physical} = Out_{\min} + \frac{Out_{\max} - Out_{\min}}{0.8-0.1} \times (Out_{normalized} - 0.1) \tag{5} $$

where In_{min} and In_{max} are the minimum and maximum values of the input in the physical range (i.e., not normalized), respectively; and Out_{min} and Out_{max} are the minimum and maximum values of the normalized outputs obtained, respectively.

To perform the backpropagation, the total error is computed according to Eq. 6. This error is then differentiated with respect to the weights and biases as per Eqs. 7 and 8. The result is multiplied by a learning rate and the weights and biases are updated according to Eqs. 9 and 10. The learning rate is a scalar between 0 and 1, which is applied to prevent a phenomenon known as 'overfitting'. Overfitting takes place when the update rate of the weights and biases is too large and the FFNN adjusts excessively to the specific data used in the training instead of finding the best global solution. Fig. 10 illustrates this phenomenon. An overfitted curve will adjust better to the training points but perform poorly in the testing and subsequently in practical applications. In this study, a learning rate of 0.5, which provides a good compromise between speed and accuracy, is applied.

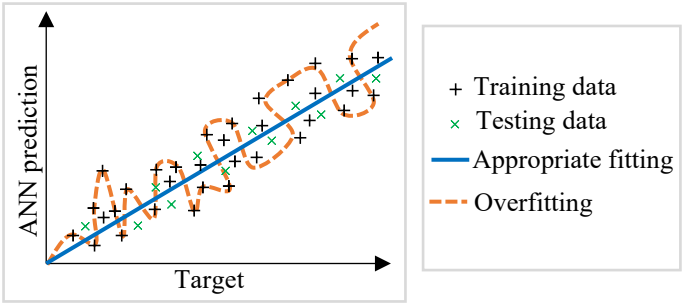


Fig. 10–Example of appropriate fitting and overfitting.

$$ E_{ANN} = \sum_{i=1}^{n_L} \frac{1}{2}\left[T_i - {}^{OL}y_i\right]^2 \tag{6} $$

$$ \delta\, {}^{L}w_{ij} = \frac{\partial E_{ANN}}{\partial\, {}^{L}w_{ij}} \tag{7} $$

$$ \delta\, {}^{L}b_i = \frac{\partial E_{ANN}}{\partial\, {}^{L}b_i} \tag{8} $$

$$^{L}w_{ij} = {}^{L}w_{ij} - \eta.\delta\,{}^{L}w_{ij} \tag{9}$$

$$^{L}b_{i} = {}^{L}b_{i} - \eta.\delta\,{}^{L}b_{i} \tag{10}$$

where E_{ANN} is the total error; T_i is the target output for neuron i; $^{OL}y_i$ is the output of neuron i in the output layer; and η is the learning rate.

Description of the procedure used for training and testing

Training of the FFNN is carried out by performing multiple iterations in which the forward and backpropagations are applied for each input data point. The iterations are completed once all data points have been used. At the end of each iteration, the calculation either continues to another one or stops if the maximum number of iterations is reached or the variation in the error is too small. In the latter case, the training is considered complete.

Subsequent to the training stage, testing of the FFNN is carried out by performing the forward propagation and computing the error for each input data point. Unlike during training, the backpropagation is not performed. Because of this, the weights and biases are not updated, and multiple iterations are not performed. If the calculated error is satisfactory (i.e., small enough), it indicates that the FFNN has correctly learned the relationship between inputs and outputs and is ready to be used. Otherwise, the developer should look for the reason for the significant error, correct it, and re-train and re-test the FFNN.

Database and datasets selection

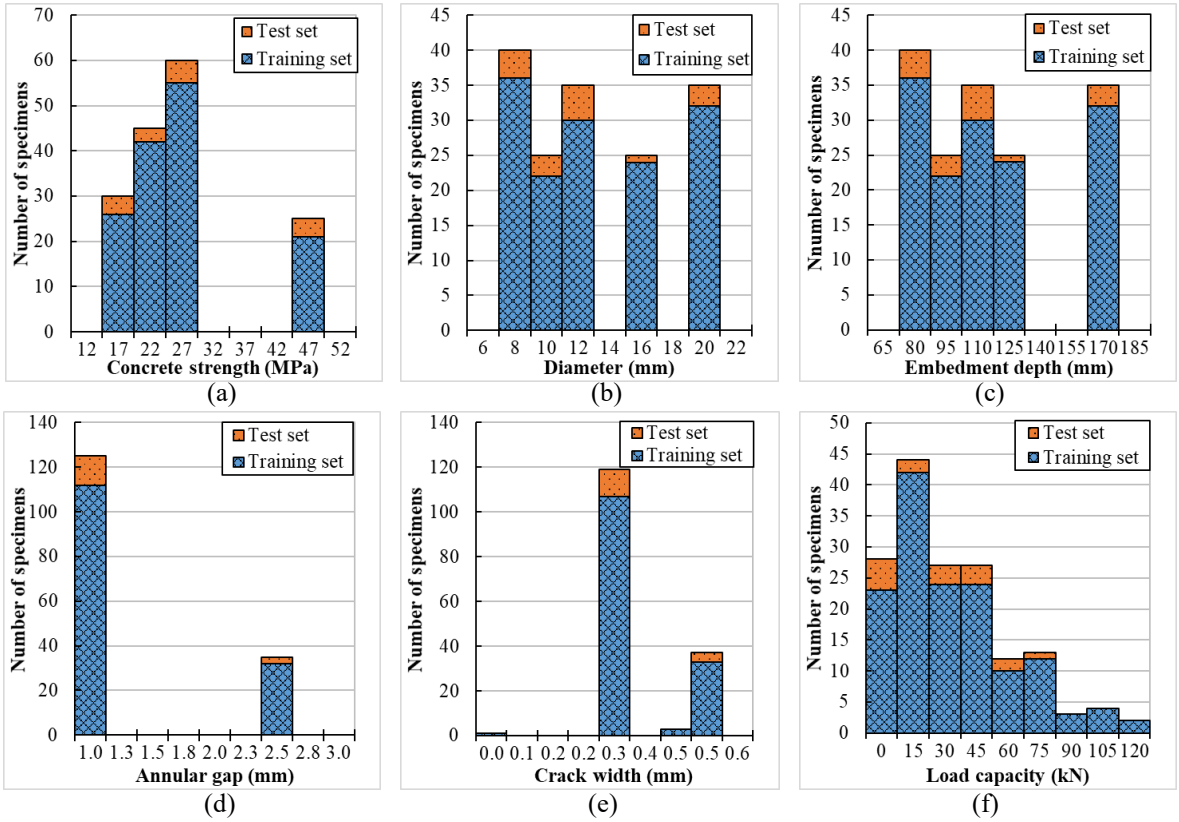
A large amount of data is required to successfully train an FFNN. In this study, the comprehensive database of ACI Committee 355 is used with proper permission. This database contains worldwide data of 2,929 adhesive anchors, including single and group anchors, tested in cracked and uncracked concrete. In this study, only the specimens consisting of individual anchors (i.e., not grouped) far from concrete edges and installed in cracked concrete are considered, resulting in a total number of 160 specimens. These specimens are divided into training and testing sets. The training set consisted of 90% of the specimens (144 specimens) while the remaining 10% (16 specimens) were used for testing.

The obtained database provides a large number of parameters from the experiments. To make the most efficient use of the FFNN, only the parameters considered most relevant for the calculation of the load capacity are used. Parameters that had the same value for all specimens (i.e., min value = max value, such as for test method, loading type, and cleaning) were not included. These non-numerical parameters can also be added to the FFNN in future studies (as more experimental test results become available) by adding input neurons that receive discrete values (such as 0 or 1) to indicate 'on' and 'off' status. The range of the parameters selected is presented in Table 1.

Table 1–Range of parameters in the selected dataset.

Parameter	Min	Max	Unit
Anchor diameter	8 [0.31]	20 [0.78]	mm [in]
Embedment Depth	80 [3.15]	170 [6.69]	mm [in]
Annular gap	1.0 [0.04]	2.5 [0.10]	mm [in]
Concrete strength	17.85 [2.59]	46.75 [6.78]	MPa [ksi]
Crack width	0.00 [0.00]	0.55 [0.02]	mm [in]
Load capacity	2.7 [0.61]	124.0 [27.88]	kN [kip]
Test method	Unconfined		
Loading type	Static		
Chemical system	Grout		
Adhesive type	Vynilester		
Bolt type	Threaded rod		
Cleaning	Clean, brushed, and dry		
Failure mode	Bolt, Concrete, B/M/C		

To ensure that the network will be able to make accurate predictions in the entire range of parameters, the training and testing sets are carefully selected so that both contain extreme values of all parameters, as presented in Fig. 11. Note that the bond stresses measured in the tests range from 1.1 to 10 MPa [0.16 to 1.45 ksi]; therefore, the resulting FFNN should only be used for adhesives with bond strength up to 10 MPa [1.45 ksi]. In addition, only two values of annular gaps and mostly two values of crack widths are contained in the database (as seen in Fig. 11); this implies that the network's accuracy will be higher around these values, as compared to the intermediate values of these parameters.



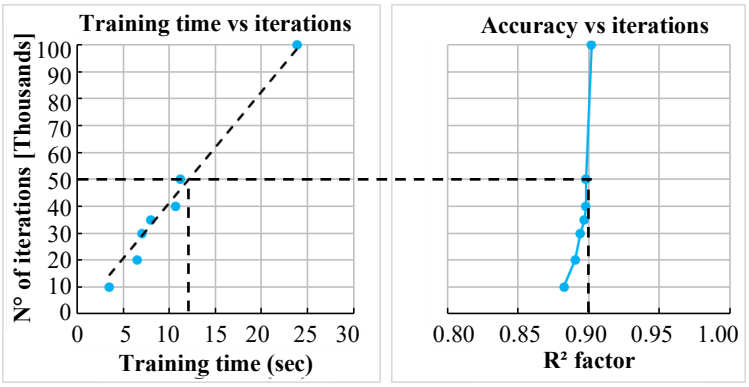
NOTE: 1 MPa = 0.145 ksi, 1 mm = 0.04 in, 1 kN = 0.225 kip

Fig. 11–Range of inputs and outputs used in the training and testing sets: (a) concrete strength; (b) anchor diameter; (c) anchor embedment depth; (d) annular gap; (e) crack width; and (f) anchor load capacity.

Selection of the optimum parameters and network layout

To determine the best network parameters, several training cycles were performed varying the number of iterations, initial values of weights and biases, and network layout. The findings of this investigation are discussed below.

First, as the number of iterations is increased, the training time increases almost linearly (Fig. 12a), while the accuracy (measured by R^2, where $R^2 = 1$ represents a perfect fit) increases asymptotically, converging to a maximum value (Fig. 12b). This indicates that excessively increasing the number of iterations is computationally expensive and does not significantly improve the accuracy of the results. From these results, 50,000 iterations were found to be an efficient value to provide satisfactory accuracy in this study.



(a) (b)

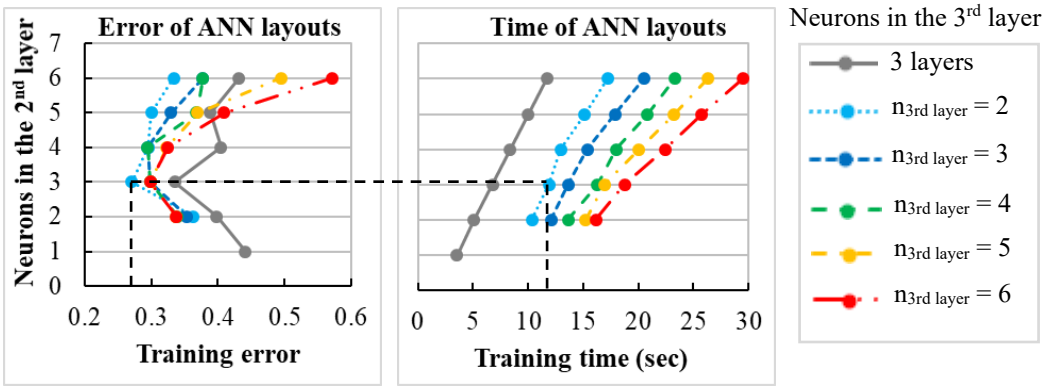
Fig. 12–(a) Increase in the ANN training time; and (b) accuracy with the number of iterations.

Second, the selection of initial values for the weights and biases changes the rate of error reduction, i.e. certain initial values help the FFNN reach the lowest error faster. After several trial-and-error attempts varying the initial weights and biases from 0 to 1, optimum values were found (shown in Table 2) and adopted, which converges to the smallest error in a lower number of iterations.

Table 2–Optimum initial values for the weights and biases.

Weights	Initial values	Weights	Initial values	Weights	Initial values	Biases	Initial values	Biases	Initial values	Biases	Initial values
$^{1}w_{11}$	0.0	$^{1}w_{12}$	0.1	$^{1}w_{13}$	0.2	$^{1}b_{1}$	0.0	$^{1}b_{2}$	0.1	$^{1}b_{3}$	0.2
$^{1}w_{21}$	0.0	$^{1}w_{22}$	0.1	$^{1}w_{23}$	0.2	$^{2}b_{1}$	0.0	$^{2}b_{2}$	0.1		
$^{1}w_{31}$	0.0	$^{1}w_{32}$	0.1	$^{1}w_{33}$	0.2	$^{3}b_{1}$	0.0				
$^{1}w_{41}$	0.0	$^{1}w_{42}$	0.1	$^{1}w_{43}$	0.2						
$^{1}w_{51}$	0.0	$^{1}w_{52}$	0.1	$^{1}w_{53}$	0.2						
$^{2}w_{11}$	0.0	$^{2}w_{12}$	0.1								
$^{2}w_{21}$	0.0	$^{2}w_{22}$	0.1								
$^{2}w_{31}$	0.0	$^{2}w_{32}$	0.1								
$^{3}w_{11}$	0.0										
$^{3}w_{21}$	0.0										

Lastly, the FFNN layout has a significant impact on the final error, convergence rate, and training time. After training a total of 31 FFNNs with three and four layers, the layout that led to the smallest error had two hidden layers containing three and two neurons, respectively (Fig. 13a). It was found that deeper networks (i.e., with more layers) with up to six neurons in each hidden layer tend to lead to a slightly lower final error, at the expense of higher number of iterations. Furthermore, the training time increased almost linearly with the number of neurons (Fig. 13b), indicating that it is not advantageous to choose an excessively deep FFNN. As a result, a simpler layout is chosen as already shown in Fig. 9.



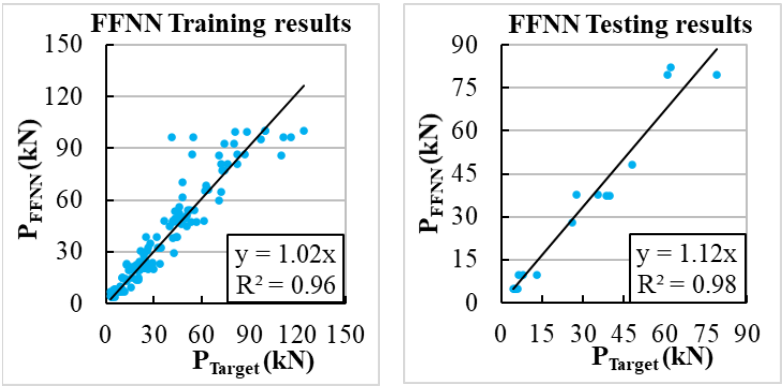
(a) (b)

Fig. 13–(a) Variation of the training error; and (b) training time with the number of layers and neurons.

Results of the training and testing of the chosen FFNN

The training of the chosen FFNN led to the results shown in Fig. 14a. In this figure, an ideal result (i.e., network predictions exactly equal to the target outputs) would be a 45° line ($y = x$) with $R^2 = 1$, which is close to the graph obtained. The discrepancy at some points is explained by the noisy nature of the experimental results. In some cases, similar anchors tested in the database had load capacities ranging from approximately 3 to 10 kN [0.67 to 2.25 kips]. In fact, on average, the load capacity given by similar anchors in the database had a coefficient variation of 18%. This high variation in the experimental data will limit the accuracy of the predictions for some specimens. Nevertheless, the chosen FFNN is expected to provide an accurate prediction of the anchor load capacity in the vast majority of the cases.

The testing of the chosen FFNN was performed with 16 experimental results that were not used in the training (i.e., never seen by the network). As shown in Fig. 14b, the accuracy in the testing is slightly higher than that in the training ($R^2 = 0.98$ versus $R^2 = 0.96$ from training) while the line is slightly more inclined ($y = 1.12x$ versus $y = 1.02x$ from training). This indicates that the FFNN predictions are slightly on the conservative side or in the favor of safety. These results show that the FFNN is able to make accurate predictions of the anchor load capacity while taking into account the effects of concrete cracking.



(a) (b)

NOTE: 1 kN = 0.225 kip

Fig. 14–(a) Results of the FFNN training; and (b) testing.

This FFNN is developed for adhesive anchors installed into steel-reinforced concrete slabs. A similar development can be followed to include specimens installed into other types of concrete including FRP-reinforced concrete. To that end, a new database containing a significant number of experimentally measured load capacities of anchors installed into FRP-reinforced concrete is needed. This database is to be used independently in a new training. Alternatively, the reinforcement type can be added as a discrete input parameter (assuming 0 or 1 values for steel or FRP, for example), in which case both databases are used together to train an FFNN that would be applicable for slabs with both reinforcement types.

Hybrid finite element-ANN methodology for the analysis of adhesive anchors in cracked concrete

To facilitate the use of the developed FFNN in the analysis of adhesive anchors installed in cracked concrete, a hybrid finite element-ANN methodology was developed. This methodology is based on the analysis of validated 3D high-fidelity NLFE models and combines the developed FFNN with the NLFE technique and analytical equations. An important feature of the methodology is the capability to consider the adverse effects of elevated temperatures, concrete cracking, and wind-induced beam bending when calculating the short-term static load capacities of post-installed adhesive anchors.

The methodology starts with the creation of a 2D NLFE model in Step 1 to simulate the pullout behavior of the anchors from the concrete. The thickness of the model is determined by applying a novel modeling method, the 'Equivalent cone method' (ECM). The ECM allows 2D models to be used instead of 3D ones when simulating the concrete breakout failure mode. This simplifies and speeds up the modeling and analysis processes. The ECM is shown to provide load capacities with accuracy of approximately 90% when compared to equivalent 3D models. Step 2 consists of a modification of the steel strength of the anchor to account for wind-induced beam bending effects in a system-level analysis. This effect is shown to potentially reduce the anchor load capacity by up to 62% in extreme cases (Almeida Jr and Guner, 2020). Step 3 includes the use of the developed FFNN to account for the bond damage caused by the concrete cracking. As the final step, Step 4 applies developed analytical equations to account for the degrading effects of elevated temperatures on the bond strength of the adhesive. This is found to reduce the load capacities of adhesive anchors by up to 70% at 82°C [179.6 °F]. Full details of the methodology's development and use (such as the creation of the NLFE model) are presented in Almeida Jr and Guner (2020) and educational spreadsheets and videos of the ECM and use of the developed FFNN are also available (Almeida Jr and Guner, 2019a; Almeida Jr and Guner, 2019b; Almeida Jr and Guner, 2019c; Almeida Jr and Guner, 2019d).

SUMMARY AND CONCLUSIONS

This study presented a review of the main artificial neural network (ANN) types and their applications in civil engineering and other fields; developed, trained and tested a deep feed-forward neural network (FFNN) to predict the load capacity of adhesive anchors installed into the concrete, and presented a hybrid methodology that combines the 2D nonlinear finite element (NLFE) technique with the developed FFNN to account for real-life adverse effects (such as concrete cracking, wind-induced beam bending and elevated temperature) in the analysis of adhesive anchors. The main conclusions drawn from this study are as follows:

- Artificial intelligence techniques, specifically FFNNs, are well suited for concrete anchorage analysis applications, as demonstrated in the past studies for uncracked concrete and in this study for cracked concrete.
- The developed FFNN successfully predicts the load capacity of adhesive anchors installed into cracked concrete, accounting for the bond damage caused by the cracks, with high accuracies and in a very short time (i.e., instantaneously).
- In order to establish a reliable prediction accuracy with FFNNs, a larger number of experimental data, with small variations between similar specimens, is needed.
- Hybrid NLFE-ANN methodologies can be used to account for complex and interacting effects, such as elevated temperatures and wind-induced beam bending, allowing for a more accurate load capacity prediction while saving significant computational effort as compared to stand-alone NLFE analyses.
- The developed FFNN can be expanded to fiber-reinforced concrete slabs, provided that sufficient experimental data is available.

REFERENCES

Adeli, H., "Neural Networks in Civil Engineering," *Computer-Aided Civil and Infrastructure Engineering*, V. 16, 2001, 126-142. <https://doi.org/10.1111/0885-9507.00219>

Akinduko, A. A.; Mirkes, E. M.; Gorban, A. N., "SOM: Stochastic Initialization versus Principal Components," *Information Sciences*, V. 364-365(10), 2016, 213-221. <https://doi.org/10.1016/j.ins.2015.10.013>

Albawi, S.; Mohammed, T. A., "Understanding of a Convolutional Neural Network," *The International Conference on Engineering and Technology*, Antalya, Turkey, 2017, 6 pp. <10.1109/ICEngTechnol.2017.8308186>

Almeida Jr., S. A.; Guner, S., "A Hybrid Methodology using Finite Elements and Neural Networks for the Analysis of Adhesive Anchors Exposed to Hurricanes and Adverse Environments," *Engineering Structures*, V. 212, 2020, 9 pp. <https://www.utoledo.edu/engineering/faculty/serhan-guner/docs/JP15_Almeida_Guner_2020.pdf>

Almeida Jr, S. A.; Guner, S. “Artificial Neural Network for Adhesive Anchors in Cracked Concrete,” Spreadsheet, University of Toledo, 2019a <https://www.utoledo.edu/engineering/faculty/serhan-guner/docs/4S_ANN_Adhesive_Anchors_Cracked_Concrete.xlsx>

Almeida Jr, S. A.; Guner, S. “Artificial Neural Network for Adhesive Anchors in Cracked Concrete,” Video, University of Toledo, 2019b <https://www.youtube.com/watch?v=6mAQPRJDQKM&feature=youtu.be>

Almeida Jr, S. A.; Guner, S. “ECM: Equivalent Cone Method,” Spreadsheet, University of Toledo, 2019c <https://www.utoledo.edu/engineering/faculty/serhan-guner/docs/5S_Equivalent_Cone_Method_ECM.xlsx>

Almeida Jr, S. A.; Guner, S. “Equivalent Cone Method (ECM) for Capturing 3D Cone Failure in 2D Analysis,” Video, University of Toledo, 2019d <https://www.youtube.com/watch?v=I3lpxFPtduM&feature=youtu.be>

Amidi, A.; Amidi, S., “VIP Cheatsheet: Convolutional Neural Networks,” *Stanford University*, 2018, Retrieved from <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks>

Amidi, A.; Amidi, S., “VIP Cheatsheet: Recurrent Neural Networks,” *Stanford University*, 2019, Retrieved from <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>

Amidi, A.; Amidi, S.; Vlachakis, D.; Megalooikonomou, V.; Paragios, N.; Zacharaki, E. I., “EnzyNet: Enzyme Classification using 3D Convolutional Neural Networks on Spatial Representation,” *Bioinformatics and Genomics*, PeerJ6:e4750, 2018, 11 pp. <https://doi.org/10.7717/peerj.4750>

Asanjan, A. A.; Yang, T.; Hsu, K.; Sorooshian S.; Lin, J.; Peng, Q., “Short-Term Precipitation Forecast Based on the PERSIANN System and LSTM Recurrent Neural Networks,” *Journal of Geophysical Research: Atmospheres*, V. 123(22), 2018, 12,543-12,563. <https://doi.org/10.1029/2018JD028375>

Ashour, A. F.; Alquedra, M. A., “Concrete Breakout Strength of Single Anchors in Tension using Neural Networks,” *Advances in Engineering Software*, V. 36, 2005, 87-97. <https://doi.org/10.1016/j.advengsoft.2004.08.001>

Chandwani, V.; Agrawal, V.; Nagar, R., “Applications of Soft Computing in Civil Engineering: A Review,” *International Journal of Computer Applications*, V. 81, 2013, 13-20. <10.5120/14047-2210>

Chen, J.; Li, Q.; Wang, H.; Deng, M., “A Machine Learning Ensemble Approach Based on Random Forest and Radial Basis Function Neural Network for Risk Evaluation of Regional Flood Disaster: A Case Study of the Yangtze River Delta, China,” *International Journal of Environmental Research and Public Health*, V.17(1), 49, 2020, 21 pp. <https://doi.org/10.3390/ijerph17010049>

Chen, Q.; Xu, J.; Koltun, V., “Fast Image Processing with Fully-Convolutional Networks,” *IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017, 10 pp. <10.1109/ICCV.2017.273>

Cheruku, R.; Edla., D. R.; Kuppili, V., “Diabetes Classification using Radial Basis Function Network,” *International Journal of Computational Intelligence Systems*, V. 10, 2017, 247-265. <https://doi.org/10.2991/ijcis.2017.10.1.17>

Cui, W.; Caracoglia, L., “Exploring Hurricane Wind Speed along US Atlantic Coast in Warming Climate and Effects on Predictions of Structural Damage and Intervention Costs,” *Engineering Structures*, V. 122, 2016, 209-255. <http://dx.doi.org/10.1016/j.engstruct.2016.05.003>

Dinan, T., “Projected Increases in Hurricane Damage in the United States: The Role of Climate Change and Coastal Development,” *Ecological Economics*, V. 138, 2017, 186-198. <http://dx.doi.org/10.1016/j.ecolecon.2017.03.034>

Emanuel, K. A., “Downscaling CMIP5 Climate Models Shows Increased Tropical Cyclone Activity over the 21st Century,” *The National Academy of Sciences of the United States of America*, V. 110(30), 2013, 12,219-12,224. <https://doi.org/10.1073/pnas.1301293110>

Gesoglu, M.; Güneyisi, E., “Prediction of Load-Carrying Capacity of Adhesive Anchors by Soft Computing Techniques,” *Materials and Structures*, V. 40, 2007, 939-951. <https://doi.org/10.1617/s11527-007-9265-6>

Goodfellow, I.; Bengio, Y.; Courville, A., “Deep learning,” *The MIT Press*, Ch. 9: Convolutional Neural Networks, Cambridge, MA, U.S.A., 2016, 800 pp. <https://www.deeplearningbook.org/contents/convnets.html>

Guo, J., "The Costs of Climate Change," *Earth and Environmental Science*, V. 120, 2017, 7 pp. <10.1088/1755-1315/120/1/012015>

Ilaboya, I. R.; Izinyon, O. C., "Modeling and Prediction of Sorption of Pb(II) and Mn(II) Ions from Aqueous Solution onto Acid Activated Shale using Statistical Design of Experiment (DOE) and Modular Neural Network (MNN)," *Journal of Civil and Environmental Engineering*, V. 17(1), 2019, 113-124. <https://dergipark.org.tr/en/pub/ijesa/issue/53333/489948>

Knutson, T. R.; Sirutis, J. J.; Vecchi, G. A.; Garner, S.; Zhao, M.; Kim, H.-S.; Bender, M.; Tuleya, R. E.; Held, I. M.; Villarini, G., "Dynamical Downscaling Projections of Twenty-First-Century Atlantic Hurricane Activity: CMIP3 and CMIP5 Model-Based Scenarios," *Journal of Climate*, V. 26(17), 2013, 6,591-6,617. <https://doi.org/10.1175/JCLI-D-12-00539.1>

Kulkarni, P. S.; Londhe, N. S.; Deo, M. C, "Artifical Neural Networks for Construction Management: A Review," *Journal of Soft Computing in Civil Engineering*, V. 1(2), 2017, 70-88. <10.22115/SCCE.2017.49580>

Liu, F., "Projections of Future US Design Wind Speeds due to Climate Change for Estimating Hurricane Losses," *Ph.D. Thesis*, Clemson University, Clemson, South Carolina, U.S.A., 2014. <https://tigerprints.clemson.edu/all_dissertations/1305>

Luckey, D.; Fritz, H.; Legatiuk, D.; Dragos, K.; Smarsly, K., "Artificial Intelligence Techniques for Smart City Applications," *International Conference on Computing in Civil and Building Engineering*, 18th ed., São Paulo, Brazil, 2020, 3-15. <https://doi.org/10.1007/978-3-030-51295-8_1>

Melin, P.; Monica, J. C.; Sanchez, D.; Castillo, O., "Multiple Ensemble Neural Network Models with Fuzzy Response Aggregation for Predicting COVID-19 Time Series: The Case of Mexico," *Healthcare*, V. 8(2), 181, 2020, 13 pp. <https://doi.org/10.3390/healthcare8020181>

Michalski, J., "Building Exposure Study in the State of Florida and Application to the Florida Public Hurricane Loss Model," *M.S. Thesis*, Florida Institute of Technology, Melbourne, Florida, U.S.A., 2016, 166 pp. <http://hdl.handle.net/11141/1074>

Mudd, L.; Wang, Y.; Letchford, C.; Rosowsky, D., "Assessing Climate Change Impact on the U.S. East Coast Hurricane Hazard: Temperature, Frequency, and Track," *Natural Hazards Review*, V. 15(3), 2014, 13 pp. <https://ascelibrary.org/doi/10.1061/%28ASCE%29NH.1527-6996.0000128>

Rosada, A.; Arliansyah, J.; Buchari, E., "Evaluation Pavement Deteriorating Condition on Surface Distress Index (SDI) Using Radial Basis Function Neural Networks (RBFNN)," *Symposium of Emerging Nuclear Technology and Engineering Novelty*, V. 1198(3), 2019, 7 pp. <10.1088/1742-6596/1359/1/012110>

Sakla, S. S. S.; Ashour A. F., "Prediction of Tensile Capacity of Single Adhesive Anchors using Neural Networks," *Computers and Structures*, V. 83, 2005, 1792-1803. <https://doi.org/10.1016/j.compstruc.2005.02.008>

Scarpiniti, M.; Comminiello, D.; Uncini A.; Lee, Y.-C., "Deep Recurrent Neural Networks for Audio Classification in Construction Sites," *European Signal Processing Conference*, 28th ed., Amsterdam, The Netherlands, 2020, 810-814. <10.23919/Eusipco47968.2020.9287802>

Skowron, M.; Wolkiewicz, M.; Orlowska-Kowalska, T.; Kowalski, C. T., "Application of Self-Organizing Neural Networks to Electrical Fault Classification in Induction Motors," *Applied Sciences*, V. 9(4), 2019, 22 pp. <https://doi.org/10.3390/app9040616>

Sun, Z.; Li, K.; Li, Z., "Prediction of Concrete Compressive Strength based on Principal Component Analysis and RBFNN," *Materials Science and Engineering*, V. 677(2), 2019, 9 pp. <10.1088/1757-899X/677/2/022045>

Winston, J. J.; Hemanth, D. J.; Angelopoulou, A.; Kapetanios, E., "Iris Image Recognition using Optimized Kohonen Self Organizing Neural Network," *International Conference on Imaging for Crime Detection and Prevention*, 9th ed., London, U.K., 2019, 6 pp. <https://dx.doi.org/10.1049/cp.2019.1171>