EVALUATING THE RELATIONSHIP BETWEEN CONCRETE STRENGTH AND MIX DESIGN PROPERTIES USING ARTIFICIAL NEURAL NETWORK (ANN) HYBRID ALGORITHMS

SİNAN KEFELİ

JUNE 2019

EVALUATING THE RELATIONSHIP BETWEEN CONCRETE STRENGTH AND MIX DESIGN PROPERTIES USING ARTIFICIAL NEURAL NETWORK (ANN) HYBRID ALGORITHMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCES OF
ÇANKAYA UNIVERSITY

BY
SİNAN KEFELİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF
MASTER OF SCIENCE
IN
CIVIL ENGINEERING
DEPARTMENT

JUNE 2019

Title of the Thesis: **Evaluating the Relationship between Concrete Strength and Mix Design Properties Using Artificial Neural Network (ANN) Hybrid Algorithms**

Submitted by **Sinan KEFELİ**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University.

Prof. Dr. Can ÇOĞUN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Nevzat YILDIRIM
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Seda YEŞİLMEN
Supervisor

Examination Date: 24.06.2019

Examining Committee Members

Prof. Dr. İsmail Özgür YAMAN	Middle East Technical Univ.	
Prof. Dr. Serhat KÜÇÜKALİ	Çankaya Univ.	
Assist. Prof. Dr. Seda YEŞİLMEN	Çankaya Univ.	

STATEMENT OF NON-PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Sinan KEFELİ

Signature: 

Date: 22.07.2019

ABSTRACT

EVALUATING THE RELATIONSHIP BETWEEN CONCRETE STRENGTH AND MIX DESIGN PROPERTIES USING ARTIFICIAL NEURAL NETWORK (ANN) HYBRID ALGORITHMS

KEFELİ, Sinan

M.Sc., Department of Civil Engineering

Supervisor: Assist. Prof. Dr. Seda YEŞİLMEN

June 2019, 65 pages

In this thesis, accurate prediction of concrete strength was investigated by using artificial neural network hybrid algorithms. To hybridize and tune ANN models, Particle Swarm Optimization was implemented. Optimization process was conducted step by step up to reaching predictions at a high level of accuracy. Activation functions, numbers of neuron in hidden layers, initial learning rate, solver and learning rate were subjected to optimization.

Keywords: Artificial Neural Network, Particle Swarm Optimization, Concrete, Strength Prediction, Hybrid Algorithms

ÖZ

BETON DAYANIMI VE KARIŞIM TASARIMI ÖZELLİKLERİNİN YAPAY SİNİR AĞI HİBRİT ALGORİTMALARI KULLANILARAK DEĞERLENDİRİLMESİ

KEFELİ, Sinan
Yüksek Lisans, İnşaat Mühendisliği Anabilim Dalı
Tez Yöneticisi: Dr. Öğr. Üyesi Seda YEŞİLMEN

Haziran 2019, 65 sayfa

Bu tezde, beton dayanımının doğru ve kesin tahmini yapay sinir ağı (YSA) hibrit algoritmaları kullanılarak incelenmiştir. YSA modellerini hibritleştirmek ve ayarlarını yapmak için Parçacık Sürü Optimizasyonu (PSO) kullanılmıştır. Optimizasyon süreci yüksek bir doğruluk seviyesindeki tahminlere ulaşıncaya kadar adım adım yürütülmüştür. Aktivasyon fonksiyonları, gizli katmanlardaki nöron sayıları, başlangıç öğrenme oranı, çözümleme algoritması ve öğrenme oranı optimizasyona konu edilmiştir.

Anahtar Kelimeler: Yapay Sinir Ağı, Parçacık Sürü Optimizasyonu, Beton Dayanımı Tahmini, Hibrit Algoritmalar

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Assist. Prof. Dr. Seda YEŞİLMEN for her supervision, special guidance, suggestions, and encouragement through the development of this thesis.

It is a pleasure to express my special thanks to my family for their valuable support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACO Ant Colony Optimization

ADALINE Adaptive Linear Neuron or Later Adaptive Linear Element

ADAM Adaptive Moment Estimation

AI Artificial Intelligence

ANFIS Adaptive Network-Based Fuzzy

ANN Artificial Neural Network

ARMA Autoregressive Moving Average Model

BFGS Broyden-Fletcher-Goldfarb-Shanno Algorithm

BP Back Propagation

CC Cascade Correlation

CF Correlation Factor

COD Coefficient of Determination

CUPSO Center-Unified Particle Swarm Optimization

EPS Expanded Polystyrene

FRP Fiber Reinforced Polymer

HMLP High Order Multilayer Perceptrons

HPC High Performance Concrete

L-BFGS Limited memory Broyden-Fletcher-Goldfarb-Shanno Algorithm

MAE Mean Absolute Error

MAPE Mean Absolute Percentage Error

MLR Multiple Linear Regression

MSE Mean-Square Error

MSS Maximum Surface Settlement

NADAM Nesterov-Accelerated Adaptive Moment Estimation

PSO Particle Swarm Optimization

RCPT Rapid Chloride Penetration Test

RELU Rectified Linear Unit

RMS Root Means Square

RMSE Root-Mean-Square Error

RMSPROP Root Mean Squared Prop

SCC Self Compacting Concrete

SGD Stochastic Gradient Descent

SVR Support Vector Regression

TANH Hyperbolic Tangent

UPSO Unified Particle Swarm Optimization

VMA Viscosity Modifying Agent

CHAPTER 1

INTRODUCTION

Strength of concrete depends on not only a few, but many factors such as mix design, curing conditions and mixing conditions, transporting, placing and testing [1], [2]. This is the reason for a will to have a complete model for concrete strength prediction. There has been many studies for proposing strength prediction [3], [4]. Predicting concrete strength with artificial neural network (ANN) models is more convenient in terms of accuracy since ANN models outperforms models on regression analysis or ANNs are more useful for practical purposes than mathematical models and observing changes in strength occurred by changes in mix design is very easy [2], [5].

Without the need for testing, having the advantage of the use of previous experiment, predicting concrete strength with ANN is a great advantage for many applications in civil engineering and for all parties involved in the process of designing, constructing, controlling and maintaining [2], [6].

The purpose of the thesis is to propose a strong method for predicting strength concrete and establish a foundation for future work in understanding the behavior of concrete as one of the most important materials in civil engineering.

The thesis comprises of 4 chapters including information about background of artificial neural networks, machine learning theory, metaheuristics and specifically Particle Swarm Optimization (PSO), methods that was used in the thesis and results of the methods. Conclusions and references are also provided.

Background chapter gives information about artificial neural networks, brief history of networks, ANN architecture, artificial neuron definition, activation functions, feedforward backpropagation, overfitting and underfitting, regularization,

learning algorithms, learning paradigms, applications of ANNs, particularly strength prediction with ANN, metaheuristics, PSO and hybrid intelligence with PSO. Methodology chapter includes medium of the study, dataset used, performance indicators used, steps of ANN optimization with PSO and the results of the methods. Conclusion chapter explains what is proposed in the thesis and underline the important points of the thesis. In the end, references are listed.

CHAPTER 2

BACKGROUND

2.1. Artificial Neural Network (ANN)

Artificial neural network (ANN) is a mathematical/computational model that emulate biological neural network, which is composed of interconnected set of artificial neurons that are the basic processing unit where computation occurs to answer challenging questions arisen in life in numerous areas [7]–[11]. An example ANN architecture is given in Fig. 1 below.

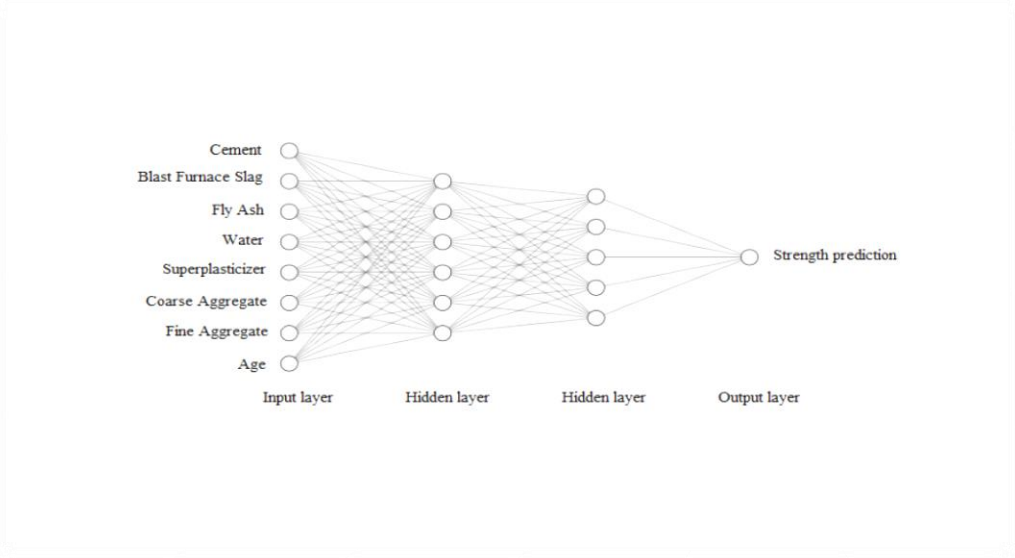

Figure 1: An Example ANN Architecture

Warren McCulloch and Walter Pitts stated the first ANN model and explained how neurons in brain function in 1943 [12]. They proposed a binary model in which neurons can only be on or off. Donald Hebb wrote a book named "The Organization of Behavior: A Neuropsychological Theory" stating that neural pathways are strengthened on time each time they are used [13]. This book established "Hebbian

Learning". Frank Rosenblatt was the first to compose an ANN and reached "perceptron" which assigned and updated weights therefore reduced error [14]. In 1960, Bernard Widrow and Marcian Hoff introduced two models named "ADALINE" and "MADALINE" which stands for adaptive linear element and multiple adaline [15]. Marvin Minsky is worth noting in the background of artificial intelligence and perceptrons. In 1954, he contributed to the related research with a doctorate dissertation named "Theory of Neural-Analog Reinforcement Systems and its Application to the Brain-Model Problem". In 1969, Marvin Minsky and Seymour Papert published a book named "Perceptrons: An Introduction to Computational Geometry" [16]. The book caused the curiosity on neural networks to decrease due to the claim of the limitations of perceptrons [17]. In 1982, John Hopfield published a paper opening a new and fresh look into artificial intelligence (AI), the paper renewed the interest to AI [18]. After his paper, neural networks became more popular and interests on AI started to grow.

ANNs comprise of layers which are named input layer, hidden layer(s) and output layer(s). Input layer takes data into network to transmit it to next layer for further processing. Hidden layer takes data from previous layer to process it in neurons to assign weights and compute the results of activation functions and transmit this result further to another hidden layer or to output layer [19].

Artificial neurons, neurons in short, mimics the function of biological neurons in brain. Neurons have three missions: multiplication, summation and activation; which be explained in a wider perspective. Inputs are assigned weights individually and they are multiplied with each other. After the "multiplication", weighted inputs are summed and bias is added, which is called "summation". The next mission is "activation": neurons apply an activation function to the sum of weighted inputs and bias to convert to an output activation and decide whether a signal is activated or not, or activation yields a value depending on the selection of activation function [7], [8]. An example artificial neuron is illustrated in Fig. 2.

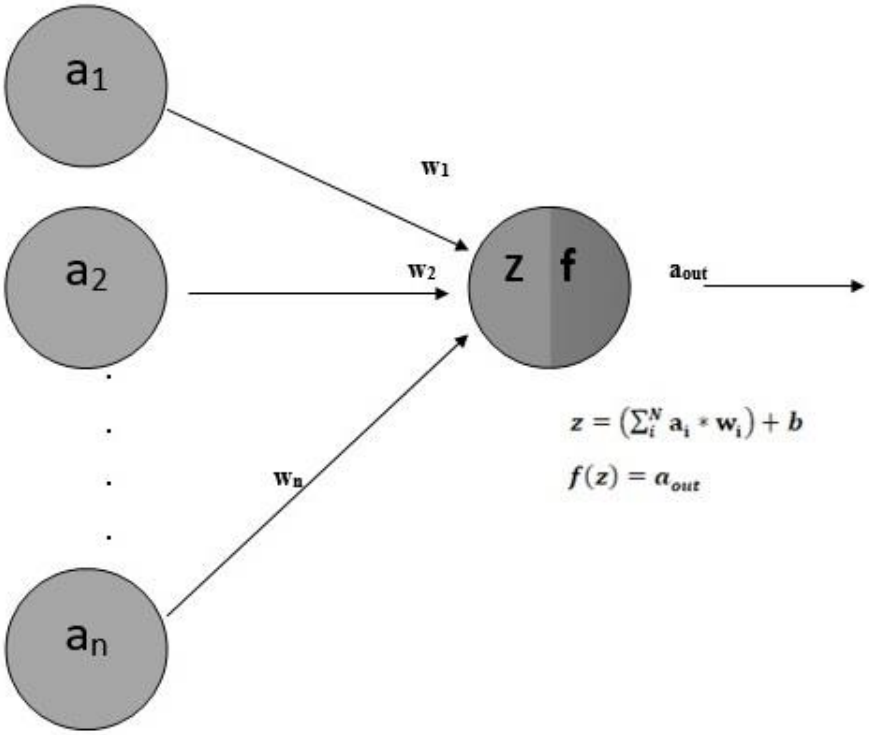

Figure 2: An Artificial Neuron

The mathematical relationship between weights, biases and activation function in a neuron or node is formulized in eq. 1 and eq. 2 as follows:

$$z = \left(\sum_{i}^{N} a_i * w_i\right) + b \tag{1}$$

$$f(z) = a_{out} \tag{2}$$

where a_i is the input from i^{th} input feature, w_i is the weight assigned to the i^{th} input feature, z is the weighted sum, f is the activation function, a_{out} is the output of the neuron and b is the bias. Bias is the term added to the weighted sum, which makes the operations of the neurons much more flexible and versatile [20].

Output of a neuron differs as discrete or continuous due to the selection of activation function. For instance, Heaviside step function results in binary-valued output whereas sigmoid function produces output values between 0 and 1. Activation functions are the functions for nodes which get input signals and emit output signals

that will be used as input signals at other nodes in next layer. Some of the most widely used activation functions are linear function, relu, sigmoid and tanh [21], [22].

Architecture or topology is how neurons are connected to one another, which can occur in many ways as it can be divided into two sub-groups: feedforward (acyclic) and recurrent (semi-cyclic) artificial neural networks.

In a feedforward backpropagation neural network, information flow from the input layer to the output layer from one layer to the next in a sequence, during this phase which is called forward propagation, outputs of neurons are computed and transmitted to the next unit that will take it as an input. After this phase of forward propagation, back propagation comes, where errors are computed in the reverse direction of forward propagation in a sequence from output layer to input layer [10]. At this point, it should be explained what error means. Error should be measured in a way to describe the difference between predictions and actual (real or experimental) values of data.

In a three-layer feedforward artificial neural network, which has an input layer, a hidden layer and an output layer, neurons in input layer takes data into network to transmit it to next layer which is hidden layer for further processing. Neurons in hidden layer takes data from neurons in input layer and assign weights individually to inputs, multiply with these weights and inputs and add bias to them. After the abovementioned "multiplication" and "summation" phases, neurons in hidden layer apply an activation function to the sum of weighted inputs and bias to pass a signal to the next layer, that is output layer. It is worth noting that information processing in one layer occurs at the same time rather than in a sequence, this phenomenon is called parallel processing or massive parallelism that provides great advance comparing to conventional sequential processing [8].

One of the main advantages of ANNs is to be the universal approximator, which means that ANNs are able to approximate any random function with any desired level of accuracy. Moreover, ANNs can be employed to a problem that has not an empirical model at all and ANN extracts the relationship hidden in data, which is called learning or training the data [22]–[24].

Accuracy is the degree or success to predictions of experimental results. Accuracy is measured with error between predicted values and experimental (actual) values. Assessing accuracy of a model is a very important issue in regression, machine learning and specifically ANN applications. To achieve this, dataset to be used is divided into different subsets: training, validation and test sets. As the names imply, training data is portion of data in which learning process happens. After having a model obtained, the model is tuned to get better predictions over various datasets. The portion of the dataset to be used to tune the model is validation data. It is separated from training set and only used to tune the model. There is a necessity to evaluate models on separate sets of data to have unbiased measure of accuracy [25]. Therefore, the model should be evaluated, namely tested with a different dataset that is held separate from training. This is test data.

When a model fits training data too well, error computed with that model becomes too small. However, when this "too good" model is tested with another set of data that is test data, it has a larger error than the one with training data. This is called overfitting. Overfitting means that the model obtained with training data fits training data too well, but it is not able to make good predictions on other sets of data.

In order to avoid overfitting; ANN architecture can be shrunk, model can be trained to a smaller number of epochs or regularization term can be implemented to cost function. The purpose of the regularization term is to apply a penalty to high values of weights [26]. General form of regularization for cost function is given in eq. 3 below [27].

$$\begin{pmatrix} Regularized \\ cost\ function \end{pmatrix} = \begin{pmatrix} Emprical \\ cost\ function \end{pmatrix} + \begin{pmatrix} Regularization \\ parameter \end{pmatrix} * (Regularizer) \quad (3)$$

Two common methods of regularization are explained here. Regression model which an L1 regularization term was added is called Lasso (Least Absolute Shrinkage and Selection Operator) Regression as it can be seen in eq. 4. If an L2 regularization term is added to cost function, model is called Ridge Regression as shown in eq. 5.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}(x_{ij}w_j)^2 + \lambda\sum_{j=1}^{p}|w_j| \quad (4)$$

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}(x_{ij}w_j)^2 + \lambda\sum_{j=1}^{p}w_j^2 \quad (5)$$

A learning algorithm is a systematic process to obtain patterns in data to fit a model to transform data from input to output [28]. In ANNs, there is a need to choose a learning algorithm to train a network. Among a diverse collection learning algorithms, one can be chosen according to the nature of problem. Artificial neural networks generally use a modified form of gradient descent algorithm, which is quite straight forward with the usage of derivative of cost function [9].

In ANN models, miscellaneous algorithms can be used. Among a diverse alternatives including Levenberg–Marquardt, Broyden-Fletcher-Goldfarb-Shanno (BFGS) Quasi Newton, limited memory BFGS (L-BFGS), One step secant, Bayesian regularization ad Gradient descent; L- BFGS and Gradient descent are examined for this study since those are two of the most effective, accurate and widely used algorithms.

L-BFGS was introduced by Nocedal [29]. The difference between BFGS and L-BFGS methods is the update in the matrix. Corrections of the matrix are stored and when the maximum number of corrections is reached, the oldest one is deleted and the newest one is added. L-BFGS method algorithm is given in equations 6-13 as follows [30]:

Let x_k denotes iterates, $s_k=x_{k+1}-x_k$ *and* $y_k=g_{k+1}-g_k$.

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T \quad (6)$$

where $\rho_k = \frac{1}{y_k^T s_k}$ (7)

and $V_k = I - \rho_k y_k s_k^T$ (8)

Step 1: Determine x_0, m, $0 < \beta' < 1/2$, $\beta' < \beta < 1$ and a symmetric and positive definite starting matrix H_0. Set k = 0.

Step 2

$$d_k = -H_k g_k \tag{9}$$

$$x_{k+1} = x_k + \alpha_k d_k \tag{10}$$

where α_k satisfies the Wolfe conditions:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \beta' \alpha_k g_k^T d_k \tag{11}$$

$$g(x_k + \alpha_k d_k)^T d_k \geq \beta g_k^T d_k \tag{12}$$

α_k=1 at first

Step 3 Update H_0 $\hat{m}$ + 1 times using the pairs $\{y_j, s_j\}_{j-\hat{m}}^{k}$,

$$\begin{aligned} H_{k+1} = {} & \left(V_k^T \dots V_{k-\hat{m}}^T\right) H_0 (V_{k-\hat{m}} \dots V_k) \\ & + \rho_{k-\hat{m}} \left(V_k^T \dots V_{k-\hat{m}+1}^T\right) s_{k-\hat{m}} s_{k-\hat{m}}^T (V_{k-\hat{m}+1} \dots V_k) \\ & + \rho_{k-\hat{m}+1} \left(V_k^T \dots V_{k-\hat{m}+2}^T\right) + s_{k-\hat{m}+1} s_{k-\hat{m}+1}^T (V_{k-\hat{m}+2} \dots V_k) \dots \\ & + \rho_k s_k s_k^T \end{aligned} \tag{13}$$

where $\hat{m} = min\{k, m-1\}$.

Step 4

$Set\ k := k + 1\ and\ go\ to\ Step\ 2.$

Gradient descent minimizes selected cost function which is a function of weights of neural network. It updates weights according to gradient of cost function. It is a first order optimization algorithm which means that it only takes into account the first derivative whereas there are second order algorithms that use the second derivative [31], [32].

There are three variants of gradient descent algorithm according to the amount of data used and a balance should be provided between the accuracy and the time to compute. These variants are Batch Gradient Descent (eq. 14), Stochastic Gradient Descent (SGD) (eq. 15) and Mini-Batch Gradient Descent (eq. 16).

$$w = w - \eta * \nabla_w J(w) \tag{14}$$

$$w = w - \eta * \nabla_w J\left(w; x^{(i)}; y^{(i)}\right) \tag{15}$$

$$w = w - \eta * \nabla_w J\left(w; x^{(i:i+n)}; y^{(i:i+n)}\right) \tag{16}$$

where *w* is weight, η is learning rate (step size), *x* and *y* are training examples, and *J* is cost function.

Also, there are different gradient descent optimization algorithms that can be implemented according to the nature of the problem for various challenges for convergence. These are can be listed as follows: Momentum, Nesterov-accelerated gradient, Adagrad, Adadelta, RMSprop, Adam, AdaMax, Nadam [31].

In this study, Adam is also examined as an option to be selected as a type of solver to train models. Adam is a first order optimization algorithm for stochastic objective functions, which needs little memory and was proposed by Kingma and Ba [33]. It actually benefits the advantages of two methods: AdaGrad [34] and RMSProp [35]. The name, Adam refers to adaptive moment estimation, which explains the computation of adaptive learning rates that comes from first and second moments of the gradients. Algorithm for Adam is as follows:

$m_0 \leftarrow 0$

$v_0 \leftarrow 0$

$t \leftarrow 0$

while θ_t not converged

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta f_t(\theta_t - 1)$ (Obtain gradients w.r.t. function f at timestep t)

$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (Update biased first moment estimate)

$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ (Update biased second raw moment estimate)

$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)

$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)

$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / \sqrt{\hat{v}_t} + \varepsilon$ (Update parameters)

end while

return θ_t

where α is step size, β_1, $\beta_2 \in [0, 1)$ are exponential decay rates for the moment estimates, $f(\theta)$ is stochastic objective function with parameters θ, θ_0 is initial parameter vector, g_t^2 is the elementwise square of $g_t \odot g_t$. Default values of α, β_1, β_2 and ε are 0.001, 0.9, 0.999 and 10^{-8}, respectively.

In the neural networks, weight matrix W keeps all the information in itself and learning can be defined as identifying the weights corresponding to the interconnection between neurons [9]. There are three significant types of learning paradigms: supervised, unsupervised, reinforced learning.

Supervised learning is a technique for machine learning to determine weights of an artificial neural network. Weights to be set are determined by training data which is composed of inputs and target values [7]. The concept of supervised learning is involved with finding a function from domain to target, namely from data to prediction. In the search of mapping that is led by data, cost function is difference between prediction figured out from mapping and actual values in data [9]. Supervised learning is used to solve problems that can be sub-divided into two groups: regression where output is a real number as an approximation and classification where output is a category [7], [9].

In unsupervised learning, unlike supervised learning, datasets are not labeled therefore network itself figure outs the patterns hidden in data among variables [7], [11]. Clustering is the most well-known application of unsupervised learning paradigm in a way that examples in a dataset are separated into different groups according to their similarities [36]. Unsupervised learning takes only independent variable matrix into account whereas supervised learning employs independent with dependent matrix, which makes a major difference between supervised learning [37].

As another paradigm of machine learning, in reinforcement learning, data is not given to artificial neural network. In the absence of training data, it is generated by an agent's interactions with environment to take actions to maximize long-term, namely cumulative reward. Discovering a policy is the principal target for what actions are to be taken to minimize the long-term (cumulative) cost due to usually unknown dynamics although dynamics of environments are not known but can be estimated in

some way. Environment can be described as Markov decision process (MDP) [7], [9], [38].

Artificial neural networks are used in a broad range of different disciplines. According to the complexity of a problem, network model should be adjusted to get satisfying results. For a complex and specific task, a simple model yields a weak and inaccurate results while using a complex model for a simple task ends up with overfitting, not learning. Experience plays a key role for adjusting ANN topology and configuration to get potent results [7].

Real-life applications of artificial neural networks cover wide range of areas for various purposes. The purposes of applications for accounting and finance, health and medicine, marketing, data mining, manufacturing and different branches of engineering and other many forecasting problems can be listed and grouped as: function approximation or regression analysis; classification problems like pattern recognition (radar systems, face identification, object recognition etc.) and sequence recognition (gesture, speech, handwriting recognition), novelty detection, and sequential decision-making; data processing for filtering, clustering, blind source separation, and compression; robotics in directing manipulators and computer numerical control [9], [39]. In the literature, there miscellaneous applications of ANN models, such as predicting the success of MBA students [40], forecasting travel demand [41], production scheduling decision making [42], predicting accident frequency [43]. ANN applications related to construction materials, specifically concrete and concrete strength prediction is examined under a separate section.

2.2. Strength Prediction with ANN

ANN is one of the most effective and accurate methods, although there are different ways to predict strength of concrete [44]. Strength prediction with ANN has the potential to eliminate the necessity of destructive tests which are costly and time-consuming. Being able to know or at least meaningfully predict the strength of concrete without a destructive test is a great advantage to adjust mix designs of various types of concrete.

In the literature there are many studies about predicting different types of strength for various types of concrete from normal concrete to high performance concrete or from self-compacting concrete to engineered cementitious composites.

In the cases mentioned here, there are differences in the architecture or types of ANNs. Also studies differ from each other in terms of activation function used. While some studies use linear equations, some studies benefits the advantages of tanh function and the others prefer sigmoid or relu. Naturally, studies have different datasets and input parameters such as mix design parameters or environmental conditions included in the input layer differ according to the relationships examined in the problem. In literature, there are numerous studies about property prediction with ANN models. In this thesis, strength prediction with ANN was reviewed and a part of leading examples were included.

In [5], an ANN model was composed and strength of high performance concrete was aimed to be predicted and the results were compared to the experiment results obtained in the laboratory. Satisfying results were obtained with 8 input features. The study showed that ANN models performed better than regression models. Indicating the relationship between predicted values and actual values ANN models provided R^2 scores in the range of 0.814- 0.922 whereas regression models obtained it in the range of 0.432-0.584 on the test data.

In [45], researchers proposed to build a multi-layer feed-forward neural network model that predicted 28-day compressive strength of concrete. In the study, 11 input features involved which were grade of cement, water/cement ratio, water content, cement content, maximum size of coarse aggregate, fine module of sand, sand/aggregate ratio, aggregate/cement ratio, slump, effect of admixtures and content of admixtures. The data used in the study was in two batches. In the first batch collected from literature, the maximum relative percentage error obtained was 5.86. The second batch of data was obtained from a mixing plant. In the second batch, the maximum relative percentage error was 12.81.

In [2], the researcher brought a new vision to predicting idea since he detected a difficulty to predict the strength of concrete at the same model for different ages of concrete when a single ANN model was used if the curing temperature of a day changed. To overcome this difficulty, the single ANN model was broken into pieces

according to the age of concrete. The study showed that all test patterns used provided the accuracy with the averaged R^2 scores above 0.9.

In [46], prediction of concrete strength was provided by the values of ultrasonic velocity and some mixture features. A multi-layered feed-forward neural network was constructed. Strength was predicted according to the features: amount of aggregate, nominal maximum, aggregate size, aggregate type, shape of aggregate and ultrasonic velocity. Coefficient of determination (R^2) of the ANN models proposed are 0.80, 0.84 and 0.90.

In [47], light weight concrete was the subject for strength prediction. Two ANN models that were feed-forward back propagation (BP) and cascade correlation (CC) were constructed to predict the strength values after 3, 7, 14, and 28 days of curing. As input parameters, 8 features of concrete were used, which were sand, water/cement ratio, light weight fine aggregate, light weight coarse aggregate, silica fume used in solution, silica fume used in addition to cement, superplasticizer, and curing period. The paper also provided information of ANN models according to the number of neuron in the hidden layers: as a trend, the graph demonstrated that as number of neurons increases, errors in predictions decreased to a point and then increased again. On the test data, models had absolute errors in the range of 0.042- 9.132 %.

In [6], compressive strength prediction of self-compacting concrete (SCC) and high performance concrete (HPC) with high volume fly ash was aimed. The researchers used the available data in the literature on SCC with normal volume since there was not adequate amount of data on SCC with high volume. Also, they used the same data to predict the strength of HPC. The study also aimed to predict slump flow of SCC. The ANN models included 10 input parameters which were cement content and ratios of water/cement, water/binder, water/powder, fine aggregate/powder, coarse aggregate/powder, high range water reducer/powder, VMA/powder, fly ash/binder and silica/binder where VMA was viscosity modifying admixture. The study used 300 rows of data, 270 of which were employed as the training data and the rest of it was the test data. R^2 scores for compressive strength of SCC, slump flow, compressive strength of HPC were 0.91, 0.82 and 0.84, respectively.

In [48], the researchers proposed a model to predict the compressive strength of recycled aggregate concrete. The model consisted of 14 input parameters, 1 hidden

layer with 16 neurons and an output layer with 1 output, with 168 instances of data, which was obtained from literature. The study showed that ANN were able to accurately predict the strength of recycled aggregate concrete. R^2 score, RMSE and mean absolute percentage error (MAPE) on the test data were 0.9955, 3.6804 and 1.6777, respectively.

One of the applications of ANN models was on the purpose to predict the compressive strength of fiber reinforced polymer (FRP)-confined concrete [49]. In the study, 213 instances of data were collected from various studies over years. The dataset was sub-divided into three portion: training data, validation data and test data. Learning algorithm of the models was Levenberg–Marquardt. The model architecture included 6 input parameters with one layer of hidden layer that had different number of neurons for different models trained. Input parameters consisted of diameter of the circular concrete specimen, height of the circular concrete specimen, the total thickness of FRP, the tensile strength of the FRP in the hoop direction, the compressive strength of the unconfined concrete and the elastic modulus of FRP. When it comes to comparing ANN models with empirical models, the study stated that the selected ANN model to predict compressive strength demonstrated an average error of 10 % whereas the models of Matthys et al. [50], Lam and Teng [51] and Mander et al. [52] showed average errors of 10.9%, 16.3%, and 16.5%, respectively with the randomly selected 113 data. Additionally, the percentage of the predicted values in the range of ±20 was more than 90, while it was less than 80% with the other models abovementioned.

In [53], modeling compressive strength of expanded polystyrene (EPS) bead lightweight concrete was examined with three methods: regression, ANN and adaptive network-based fuzzy (ANFIS). A dataset with 75 instances was used, which was divided into 64 and 11 as the training and the test data, respectively. The study indicated that ANN performed better when compared to regression and ANFIS. On test data, correlation factor (CF) and root means square (RMS) were obtained as 0.9783 and 3.4053, respectively with an ANFIS model while ANN reached 0.9937 and 1.9302.

Compressive strength of different types of concrete has been subjected to ANN models. An interesting example of that was the study conducted on concrete containing construction and demolition waste [54]. For this type of concrete, compressive

strengths at age of 3, 7, 28 and 91 days were predicted with ANNs with 17 input features. In the study, it was stated that the experimental test results were close to those obtained by ANN models. It was given that the percentage of data having an error of 7.5% or less was 60%. For both training and test dataset, ANN models showed R^2 scores of 0.928 and 0.971 for training and test data, respectively.

ANN models has been continuously compared to other techniques in many articles. In [55], ANN models were compared to multiple linear regression model (MLR) and adaptive neuro-fuzzy inference system (ANFIS) with a dataset of 173 samples. Having a detailed explanation in the paper, it was indicated that ANN and ANFIS models had the capability of predicting 28 days compressive strength of concrete whereas MLR models were not as successful as the others due to the non-linearity of the problem. In comparison, MLR, ANFIS and ANN models reached R^2 scores of 0.7456, 0.8212 and 0.9226 for predicted values vs. actual values on the test data.

2.3. Metaheuristics

Before the definition of metaheuristics, it is necessary to point out the meaning of the word itself. Meta means beyond or identifying something higher and heuristic means search, discover or find. Metaheuristics are optimization algorithms inspired from nature [56], [57]. Those algorithms are intended to be used to reach a solution below a tolerance for relatively short time [58]. However, it is not certain that to find an optimal solution with metaheuristic algorithms. Metaheuristics can provide different benefits. They are adaptable to different kind of problems in many ways. Inspirations behind metaheuristic algorithms have a broad range from flocking of birds or behavior of fireflies to cooling of a crystalline solid while some of those do not have such a background [58].

Metaheuristics have two main features in terms of algorithm behavior: intensification and diversification. Intensification refers to a concept for algorithm to search a solution locally while knowing that the region has already a solution that is good enough. Diversification, as the name implies, refers to breed miscellaneous solutions globally in solution space [59].

When it comes to classification of metaheuristic algorithms, they can be grouped due to the technique they discover an optimal solution after they assess candidate solutions from a pool of all solutions. There are three types of metaheuristics that can be named: local search, constructive and population-based algorithms. Moreover, as a fourth class of algorithms, there are various hybrid type algorithms that integrate assorted inspirations. Logic behind the class of local search algorithms is that finding an optimal solution takes place by iterating an only solution for many times and making small modifications on it. Simulated annealing can be an example of this class, which imitates the cooling of a crystalline solid. When an algorithm is not improving a candidate solution to get an optimal solution, but construct it by adding elements to a partial solution, this is called a constructive algorithm. Ant colony optimization (ACO) is an example for this class of algorithms. Population-based algorithms, one of which is Particle Swarm Optimization implemented in this study, discover candidates of optimal solution by integrating other candidate solutions from a pool of possible solution which is generally named swarm.

2.3.1. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) has been considered one of the best optimization techniques as a result of convergence capability, easy implementation and adoption and strong robustness [60]. PSO was first introduced by Kennedy and Eberhart in 1995, which imitates the social behavior of bird or fish (particle) flocking [61]. ANN and PSO application in engineering are well-known with a variety of studies in different aspects.

A candidate solution to the optimization problem corresponds to each particle in the swarm. In the algorithm, particles with n-dimensional problem are given random positions and velocities, and the objective function of each particle at its current position is recorded. Based on the results, the best known position for the a particle, termed as individual best position (p_{best}), and the whole particles' best-known position, which is called the global best position (g_{best}), can be located. The optimum solution is chased by combining p_{best} and g_{best} as they are updated by equations 17-20.

$$V_i(k\ +\ 1) =\ wV_i(k) + c_1 rand_1 \left(p_{best,i}(k) - X_i(k)\right) + c_2 rand_2 \left(g_{best}(k) -\ X_i(k)\right) \quad (17)$$

$$X_i(k\ +\ 1) =\ X_i(k) +\ V_i(k\ +\ 1) \quad (18)$$

$$p_{best}(k\ +\ 1) = \begin{cases} X_i(k\ +\ 1), F\left(X_i(k\ +\ 1)\right) <\ F\left(p_{best,i}(k)\right) \\ p_{best,i}(k), F\left(X_i(k\ +\ 1)\right) \geq\ F\left(p_{best,i}(k)\right) \end{cases} \quad (19)$$

$$g_{best} =\ min\{F\left(p_{best,0}(k), \dots, F\left(p_{best,n}(k)\right)\right\} \quad (20)$$

where *i* denotes particle index, *k* denotes iteration index, V_i represents velocity of the i^{th} particle, and *w* represents inertia weight, $rand_1$ and $rand_2$ are the two random variables within [0, 1]. c_1 and c_2 are cognitive and social positive acceleration constants [61], [62].

The pseudo code of the procedure is as follows:

randomly initialize population of particles

repeat

for each particle i of the population do

if $f(X_i(k))<f(p_{best,i}(k))$ then

$p_{best,i}(k)= X_i(k)$

end if

If $f(p_{best,i}(k))<f(g_{best}(k))$then

$g_{best}(k)= p_{best,i}(k)$

end if

end for

Update velocity and position of each particle according to eq. 17 and 18 until stop criteria being satisfied [61]. Flowchart of PSO algorithm is provided in Fig. 3 below.

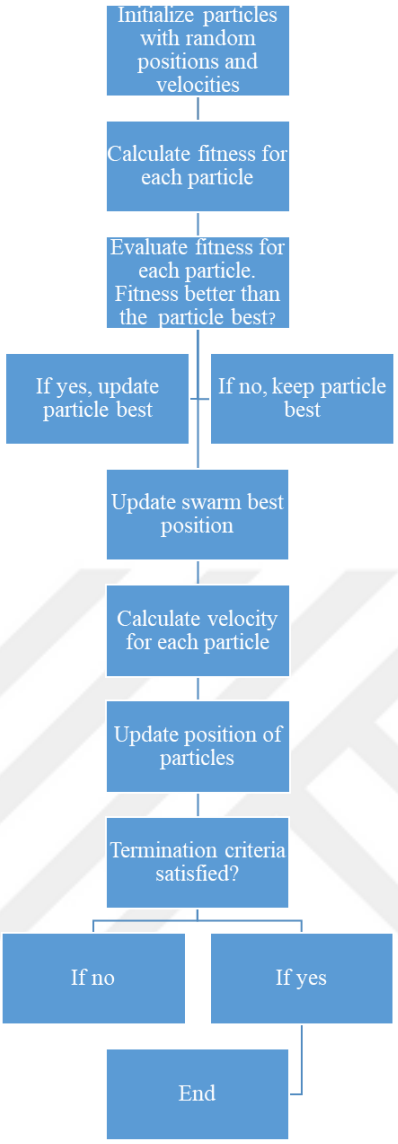

Figure 3: Flowchart of PSO algorithm

2.3.2. Hybrid Intelligence with PSO

Metaheuristics are used to hybridize models for fine tuning internal elements of ANNs. Particle Swarm Optimization (PSO) is used in many fields of engineering for making prediction and optimizing ANN parameters. Researchers from different areas have used various ways to hybridize artificial intelligence methods with PSO. Also, PSO has been used in different study areas of civil engineering. To give the examples of PSO implementation for different purposes, a few study will be mentioned.

Predicting outcomes of construction claims and litigation is one of those applications. In [63], PSO was implemented into ANN to replace the back propagation algorithm. In this implementation, data was extracted and sorted out in years 1991 to 2000 case to case and disputes and court decisions were linked to one another. As data, 1105 sets of cases were used and splitted into 3 portions. 550 of those were used as training data, 275 as test data and 280 were used to validate the model. After conducting a sensitivity analysis, which was used to determine the most relevant input features; 13 case input features were selected such as type of contract, parties involve and late payment. Binary format was used to identify the features. If one of those 13 features occurred in the case, the numerical value of occurrence is included as 1. If it did not occur, it took the value of 0. In the study, it was stated that the predictions made successfully were measured as up to 80 % while ANN with back propagation reached the prediction rate of 0.67 %. Additionally, the convergence speed was improved 33% as compared to ANN model with back propagation.

In [64], support vector machines (SVMs) were used to predict the streamflows of Swan River and St. Regis River in Montana, United States. Parameters of SVMs were optimized with PSO. The results obtained with the PSO-SVM hybrid model were compared to artificial neural networks (ANNs) and autoregressive moving average model (ARMA). The PSO-SVM hybrid model forecasted the values of one month lead with the correlation coefficient (R) of 0.86 whereas ANN and ARMA predicted with R of 0.82 and 0.76, respectively. Also, the three methods were compared for the mean relative absolute peak prediction error. PSO-SVM forecasted the values of peaks with a mean relative absolute peak prediction error of 0.248, while ANN and ARMA model predicted the peak values with errors of 0.266 and 0.355, respectively.

Another application of PSO-ANN model was the study [65]. In the study, maximum surface settlement (MSS) was predicted and the results were compared to the previously developed ANN model. Input features were horizontal to vertical stress ratio, cohesion and Young's modulus. The study did not use validation data where the dataset was splitted into two parts as training data and test data with portions of %80 and %20, respectively. The number of swarm particles was selected due to trial and error with the ANN model that had an architecture of 2-5-1, regarding that previously developed conventional ANN model had the architecture of 3-4-1. The comparison

showed that the PSO-ANN model was superior to ANN model. On the test data, the ANN model obtained the R^2 score of 0.940 and the selected PSO-ANN model reached the R^2 score of 0.968, which stated the superiority of hybrid PSO-ANN model as indicated in the study.

Slope stability during an earthquake is one of the most important subjects in geotechnical engineering. In [66], factor of safety on homogeneous slopes was predicted, which was predominantly affected by slope height, cohesion, peak ground acceleration, gradient and friction angle. The predictions were made by two intelligent systems: ANN and PSO-ANN hybrid models. A dataset of 699 instances was splitted into 5 subsets, which were also divided into two as training and test sets. As a performance indicator of the models obtained, R^2 scores of ANN and PSO-ANN models were 0.915 and 0.986, respectively. Considering the other performance indicator RMSE, the value of 0.057 was obtained by ANN whereas PSO-ANN obtained 0.022 at most.

In [67], a PSO-ANN hybrid model was aimed to be obtained to estimate ultimate bearing capacity of rock-socketed piles. The number of sample piles socketed in various rock types was 132. The model performance was not affected significantly due to the number of iterations performed, independently from what the swarm size was. Evaluating the training and test sets both, PSO-ANN model with the R^2 score of 0.932 outperforms the ANN model. The ANN model yielded the predictions at the accuracy with the R^2 score of 0.846. Only on test sets, PSO-ANN and ANN models obtained R^2 scores of 0.938 and 0.830, respectively. However, the researchers warned readers that the model was a good tool for preliminary design stage or should be used carefully.

In [68], unconfined compressive strength of soft rocks was predicted with PSO-ANN hybrid algorithm. On 40 good-quality soft rock samples, 4 different tests were conducted, which were unconfined compressive stress test, the Brazilian tensile strength test, point load index test and ultrasonic test. The dataset was splitted into two as training data and test data with portions of 80% and 20%, respectively. The best performing swarm size was determined as 125 with trial and error among the range of 25-300. ANN models with one hidden layer and two hidden layers were trained. The best performing architecture was determined with two hidden layers with 12 neurons

in each layer. For obtaining unconfined compressive stress, correlations with ultrasonic test, point load index test and the Brazilian tensile strength test yielded R^2 scores of 0.832, 0.958, and 0.821, respectively. The PSO-ANN hybrid model composed of two hidden layers with 12 neurons provided the R^2 score of 0.97, which outperforms the other correlations abovementioned.

In [69], PSO-ANN hybridization was used to predict unconfined compressive strength of cemented paste backfill. ANN was employed for prediction and PSO was used to tune the ANN models. Tailings type, cement-tailings ratio, solids content, and curing time were the input parameters of the ANN models. Number of instances in the dataset was 396, 80% of which was used as the training data and rest of it was used as test data. The study mentioned that PSO-ANN models provided cost and time saving, non-destructive and most importantly very accurate predictions as compared to other methods. The accuracy of the proposed model was indicated by the correlation coefficients of 0.969 and 0.979 on training and test data.

As a good example, in [70], ANN was hybridized with PSO to predict uniaxial compressive strength of rock samples. By implementing PSO to ANN, the researchers aimed to eliminate the disadvantage of getting stuck at a local minima or a slow rate of learning with a conventional ANN model. After training many hybrid PSO-ANN models, in the end, they obtained a model which performed much better than the average of conventional ANN models. When these two methods were compared, the PSO-ANN hybrid model obtained a success with the coefficient of determination (R^2) of 0.97 whereas ANN reached 0.71.

Another PSO-ANN hybrid model was used in mining engineering to predict strength of paste filling material [71]. Strength of paste filling material was examined as a function of concentration of fill, fly ash and cementing material with a dataset of 12 instances. Training and test data were divided as 9 and 3, respectively. The method provided predictions with maximum, minimum and average relative errors of 4.3%, 0.8% and 2.4%, respectively for early strength. For late strength, maximum, minimum and average relative errors were 3.7%, 0.5%, and 1.5%.The authors of the paper stated that they obtained relatively high errors since the training dataset did not have enough samples. In spite of the reason abovementioned, they found this method useful.

In [62], forecasting shear strength of squat reinforced concrete walls was conducted with PSO-ANN hybrid models and compared to previous models. The dataset used had 139 samples and it was divided into two part as training and test sets with portions of %80 and %20, respectively. The architecture of ANN models was 6-n-1. Since the researchers did not have specific technique for the number of neurons in the hidden layer, trial and error approach was used and eventually number of neurons was determined as 13. The only activation function used in ANN models was sigmoid function. On test data, R^2 scores of 9 PSO-ANN models ranged between 0.860- 0.975 while 9 ANN model made predictions within the interval of R^2 scores of 0.759- 0.881. Therefore, the study stated that the obtained PSO-ANN hybrid algorithm was a robust method.

Another endeavor for PSO application to ANN was to predict mechanical properties in fiber reinforced self-compacting concrete [72]. In the paper, a polynomial model was obtained with PSO using the data trained by a feed-forward multilayer perceptron ANN model for mechanical properties of fiber reinforced self-compacting concrete. This study also used %80-%20 training-test data portions with no validation data. It was stated that PSO-ANN hybrid model was able to model mechanical properties of fiber reinforced self-compacting concrete accurately. At the iteration number of 200, PSO-ANN hybrid model to generate a polynomial model predicted the compressive strength with the R^2 score of 0.999 while PSO without ANN predicted with 0.945.

In [73], PSO was implemented Support Vector Regression (SVR). The hybrid model was used to predict two outputs: compressive strength and rapid chloride penetration test (RCPT) values. Parameters of SVR were tuned by the application of PSO to obtain a better performing model. The results were compared to the results of adaptive neural-fuzzy inference system (ANFIS) method. Both methods were employed to the data with 100 instances. The dataset was splitted into training and test set with portions of 90% and 10%, respectively. The splitting was repeated five times and 5 training and test sets were generated. The swarm size was selected as 30 and number of iterations was selected as 100. On test sets, for predicting concrete compressive strength, R^2 scores of 0.941 and 0.823 were reached by SVR–PSO and

ANFIS on average, respectively. For the prediction of RCPT value, SVR–PSO with the R^2 score of 0.980 was better than ANFIS, while ANFIS obtained 0.974.

In [74], hybrid multilayer perceptrons (HMLP) was used with Center-Unified Particle Swarm Optimization (CUPSO) which is a combination of two variants: Unified PSO (UPSO) [75] and Center PSO (CPSO) [76] to predict strengths of concrete-type specimens. With this model, compressive strength of concrete, strength of deep beams and strength of squat walls were predicted with datasets having instances of 103, 62 and 62, respectively. The researcher stated that, as compared to traditional linear multilayer perceptrons, certain high-order HMLP models yielded more accurate results. On test data, the selected model predicted compressive strength of concrete, strength of deep beams and strength of squat walls with R^2 scores of 0.9756, 0.9772 and 0.9919, respectively.

CHAPTER 3

METHODOLOGY

For predicting compressive strength of concrete with PSO-ANN hybrid model accurately, the methodology explained below was followed. In the endeavor of accurate predictions, the best performing activation function, number of hidden layers, initial learning rate and L2 regularization term for weight updates were determined or optimized over individual intervals.

MLP Regressor of Scikit-learn library of Python programming language is used for ANN models [77]. By trial and error, in all PSO-ANN hybrid models, swarm size and number of iterations of PSO algorithm are set to 50 considering time consumption and efficiency, which is a common attitude for this kind of efforts [70], [78].

ANN is employed in the medium of Python for its functionality and flexibility in usage. Python is an object-oriented, high level programming language with its broad range of libraries, modules and packages. In Python, many different libraries can be utilized for many different purposes including machine learning, randomizing, mathematical operations, plotting, graph drawing and visualization, training/test data splitting, scaling and also dataframe and matrix operations. The libraries used in the codes developed for this thesis are listed in Table 1 below.

Table 1: Python libraries used in the thesis

Library	**Purpose**
Random	randomizing
Math	mathematical operations
Pandas	data manipulation and analysis
Numpy	scientific computing
pandas.plotting	plotting
matplotlib.pyplot	plotting
scikit-learn	machine learning
Pickle	saving and loading models
Time	time keeping

The dataset used in the thesis was donated to Machine Learning Repository, Center for Machine Learning and Intelligent Systems, University of California, Irvine by Prof. I-Cheng Yeh from Department of Information Management, Chung-Hua University on August 3, 2007. The dataset includes 1030 instances and 9 attributes, 8 of which are quantitative input variables and 1 of which is quantitative output variable. There is no missing attribute value. Table 2 showing the first 10 rows of the data can be seen below.

Table 2: First 10 rows of the data used

Cement (kg/m^3)	Blast Furnace Slag (kg/m^3)	Fly Ash (kg/m^3)	Water (kg/m^3)	SP* (kg/m^3)	Coarse Aggregate (kg/m^3)	Fine Aggregate (kg/m^3)	Age (day)	Concrete comp. strength (MPa)
540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99
540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89
332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27
332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05
198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30
266.0	114.0	0.0	228.0	0.0	932.0	670.0	90	47.03
380.0	95.0	0.0	228.0	0.0	932.0	594.0	365	43.70
380.0	95.0	0.0	228.0	0.0	932.0	594.0	28	36.45
266.0	114.0	0.0	228.0	0.0	932.0	670.0	28	45.85

SP: Superplasticizer

700 of total 1030 instances were used as training data while 175 instances were used as validation data and the remaining 155 were used as test data. In accordance to the explanation in the background chapter, the data was splitted into three portions as training data to train the data, validation data to tune models and test data to evaluate

accuracy of models. It is worth noting to clear the ambiguity about the existence of validation and test set simultaneously that it is necessary to have a different set of data which is not used for training or tuning models to evaluate accuracy of models without having overfitting [79].

As the name imply, training data is the portion of the data in which learning process happens. After having a model obtained, the model is tuned to get better predictions over datasets. The portion of the dataset to be used to tune the model is validation data. It is separated from training set and only used to tune the model. It is needed to evaluate models on separate sets of data to have unbiased measure of accuracy [25]. Therefore, the model should be evaluated, namely tested with a different dataset that is held separate from training. This is test data.

The data was normalized due to the eq. 21 below to be fit into the interval of [0,1] in order to avoid the effects of units of measurements. This normalization equation was used since Scikit-learn library employed it where the range of the normalization could be adjusted. The default interval of [0, 1] was selected. The statistical description of inputs and output is provided in Table 3 below.

$$X_{norm} = (X - X_{min}) - (X_{max} - X_{min}) \tag{21}$$

If ANN models train the data in too many iterations, the model may overfit on the data. Numbers of neurons in hidden layers after a certain threshold may yield overfitting, however very few numbers of neurons may demonstrate weak performance. The key for overfitting is checking model on test data. Prediction over test data gives the measurement of that whether the overfitting occurred or not. Performance of the model also depends on the number of instances of dataset. Less number of instances brings weaker models, but higher numbers of instances are more likely to provide more accurate predictions [80].

In this study, as performance indicators of the ANN-PSO hybrid algorithms, root-mean-square error (RMSE), mean absolute error (MAE) and coefficient of determination (COD) (R^2) are chosen. RMSE, MAE and COD (R^2) are given in equations 22, 23 and 24 below.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted\ Value_i - Actual\ Value_i)^2}{N}} \quad (22)$$

$$MSE = \frac{\sum_{i=1}^{N}(Predicted\ Value_i - Actual\ Value_i)^2}{N} \quad (23)$$

$$MAE = \frac{\sum_{i=1}^{N}|Predicted\ Value_i - Actual\ Value_i|}{N} \quad (24)$$

Table 3: Statistical description of inputs and output

Variables	Type	Maximum	Minimum	Mean	SD	Kurtosis	Skewness
Cement (kg/m3)	Input	540.00	102.00	281.17	104.51	-0.52	0.51
Blast Furnace Slag (kg/m3)	Input	359.40	0.00	73.90	86.28	-0.51	0.80
Fly Ash (kg/m3)	Input	200.10	0.00	54.19	64.00	-1.33	0.54
Water (kg/m3)	Input	247.00	121.75	181.57	21.36	0.12	0.07
Superplasticizer (kg/m3)	Input	32.20	0.00	6.20	5.97	1.41	0.91
Coarse Aggregate (kg/m3)	Input	1145.00	801.00	972.92	77.75	-0.60	-0.04
Fine Aggregate (kg/m3)	Input	992.60	594.00	773.58	80.18	-0.10	-0.25
Age (day)	Input	365.00	1.00	45.66	63.17	12.17	3.27
Concrete compressive strength (MPa)	Output	82.60	2.33	35.82	16.71	-0.31	0.42

In this study, PSO was used to tune the hyperparameters of ANN models. MLP regressor of Scikit-learn library has different parameters which can be optimized by an external algorithm. These are listed below on Table 4.

Table 4: MLP Regressor parameters and definitions

MLP Regressor parameters and definitions	
activation	Activation function for the hidden layer, it can be identity, logistic sigmoid, tanh or relu
alpha	L2 penalty (regularization term) parameter
batch_size	Minibatches' size for stochastic optimizers. When solver is 'lbfgs', it is not used.
beta_1	Exponential decay rate for estimates of first moment vector in adam, should be in [0, 1). Only used when solver='adam'
beta_2	Exponential decay rate for estimates of second moment vector in adam, should be in [0, 1). Only used when solver='adam'
early_stopping	Whether to use early stopping to terminate training when validation score is not improving. If set to true, it will automatically set aside 10% of training data as validation and terminate training when validation score is not improving by at least tol for two consecutive epochs. Only effective when solver='sgd' or 'adam'
epsilon	Value for numerical stability in adam. Only used when solver='adam'
hidden_layer_sizes	Number of neurons in hidden layers
learning_rate	Learning rate schedule for weight updates. It can be constant, invscaling or adaptive.
learning_rate_init	The initial learning rate used. It controls the step-size in updating the weights. Only used when solver='sgd' or 'adam'.
max_iter	Maximum number of iterations. The solver iterates until convergence (determined by 'tol') or this number of iterations.
momentum	Momentum for gradient descent update. Should be between 0 and 1. Only used when solver='sgd'
nesterovs_momentum	

power_t	The exponent for inverse scaling learning rate. It is used in updating effective learning rate when the learning_rate is set to 'invscaling'. Only used when solver='sgd'
random_state	State or seed for random number generator
shuffle	Whether to shuffle samples in each iteration. Only used when solver='sgd' or 'adam'.
solver	The solver for weight optimization. It can be 'lbfgs', 'sgd' or 'adam'.
tol	Tolerance for the optimization. When the loss or score is not improving by at least tol for two consecutive iterations, unless learning_rate is set to 'adaptive', convergence is considered to be reached and training stops.
validation_fraction	The proportion of training data to set aside as validation set for early stopping. Must be between 0 and 1. Only used if early_stopping is True
verbose	Whether to print progress messages to stdout.
warm_start	When set to True, reuse the solution of the previous call to fit as initialization, otherwise, just erase the previous solution.

Some of the most important parameters of MLP Regressor are explained here. Activation is the activation functions for the hidden layers. It can be one of those: identity function (eq. 25) logistic sigmoid function (eq. 26), hyperbolic tangent function (eq. 27) or rectified linear units (ReLU) function (eq. 28). In artificial neural networks, neurons send not only signals as on or off due to a threshold but also those can compute values within a range that is specified by the activation function employed [8]. Output of sigmoid function ranges within from 0 to 1 while hyperbolic tangent ranges from -1 to 1. ReLU takes values within the range from 0 to infinity.

$$f(x) = x \tag{25}$$

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{26}$$

$$\tanh(x) = \frac{e^{-x} - e^{-x}}{e^{-x} + e^{-x}} \tag{27}$$

$$f(x) = \begin{cases} 0 \; if \; x < 0 \\ x \; if \; x \geq 0 \end{cases} \tag{28}$$

"Solver" is the algorithm that updates weights of networks and can be selected one of these three alternatives explained in the background chapter: "lbfgs" which refers to limited memory quasi-Newton method for large scale optimization called L-BFGS [30], "sgd" which refers to Stochastic Gradient Descent and "adam" which refers to a stochastic gradient-based optimizer Adam stated by Kingma and Ba [33].

"Learning_rate" could be "constant", "invscaling" or "adaptive". If "constant" is chosen is kept constant and set to "learning_rate_init". "Invscaling" adjusts learning rate according to eq. 29.

$$effective\ learning\ rate = \frac{learning\_rate\_init}{t^{power\_t}} \tag{29}$$

"Adaptive" sets learning rate up to the point two iterations cannot reduce the training loss by minimum "tol". If "early_stopping" is "True", the learning rate is adjusted to one-fifth of itself. "Max_iter" is the maximum number of iterations. Solver iterates up to reaching "tol" or maximum number of iterations. "Tol" is the tolerance for convergence. "Learning_rate_init" is the initial learning rate used and it sets the step size for weight updates. Alpha is the L2 regularization parameter. For stochastic optimizers, batch size is the size for mini-batches and its value is determined due eq. 30. If lbfgs is selected as the solver, it is not used.

$$batch_{size} = min\left(200, n_{samples}\right) \tag{30}$$

Major MLP regressor parameters used in ANN models are given in Table 5. 20 ANN models are generated and the results are provided in Table 6 below. In the steps from 1 to 3, it was aimed to increase the prediction performance of ANN models by using PSO.

Table 5: Major MLP Regressor parameters in ANN models

Major MLP regressor parameters	Explanation	Possible parameter values, intervals or choices	Parameter values or choices
activation	activation function	identity, logistic, tanh, relu	Logistic
alpha	L2 penalty (regularization term) parameter	(0.00001, 0.001)	0.0001
hidden_layer_sizes	number of hidden layers	no defined interval	(20,20)
max_iter	Maximum number of iterations	no defined interval	200
solver	Solver	lbfgs, sgd, adam	Lbfgs
tol	Tolerance for convergence	0.0001 (default)	0.0001

Table 6: Performances of the ANN models

Model ID	RMSE		R^2 Score		MAE		Elapsed time
	Validation data	Test data	Validation data	Test data	Validation data	Test data	
ANN-1	8.67	8.22	0.75	0.74	6.60	6.16	0.43 sec.
ANN-2	6.90	6.56	0.84	0.83	4.90	4.66	0.37 sec.
ANN-3	7.42	7.83	0.82	0.77	5.57	5.63	0.45 sec.
ANN-4	7.21	7.19	0.83	0.80	5.26	5.11	0.51 sec.
ANN-5	7.62	7.39	0.81	0.79	5.50	5.45	0.43 sec.
ANN-6	8.09	7.16	0.78	0.80	5.78	5.24	0.41 sec.
ANN-7	6.14	7.78	0.88	0.77	4.73	5.17	0.49 sec.
ANN-8	6.72	6.54	0.85	0.84	5.09	4.86	0.56 sec.
ANN-9	7.58	7.34	0.81	0.79	5.80	5.70	0.53 sec.
ANN-10	7.28	6.37	0.83	0.84	5.47	4.80	0.49 sec.
ANN-11	8.16	7.25	0.78	0.80	6.10	5.51	0.49 sec.
ANN-12	6.38	8.17	0.87	0.74	4.88	5.31	0.55 sec.
ANN-13	5.92	6.68	0.88	0.83	4.31	4.96	0.49 sec.
ANN-14	6.31	6.23	0.87	0.85	4.86	4.69	0.42 sec.
ANN-15	8.03	7.17	0.79	0.80	5.87	5.22	0.46 sec.
ANN-16	6.23	7.65	0.87	0.78	4.80	5.23	0.42 sec.
ANN-17	6.94	7.71	0.84	0.77	5.27	5.66	0.42 sec.
ANN-18	6.27	7.23	0.87	0.80	4.52	5.11	0.42 sec.
ANN-19	8.98	8.87	0.73	0.70	6.63	6.33	0.40 sec.
ANN-20	7.82	7.52	0.80	0.78	5.89	5.98	0.44 sec.
AVERAGE	**7.23**	**7.34**	**0.83**	**0.79**	**5.39**	**5.34**	**0.46 sec.**

3.1. Step 1: Determining the best performing activation function for optimization

To obtain a well-established hybrid model, numbers of neurons in the two hidden layers should be determined. Since obtaining the optimal number of neurons in the two hidden layers can be achieved with the other parameters controlled or pre-determined, first of all, the best performing activation function was determined with PSO.

As a preliminary step, many combinations of swarm size and iterations were selected and employed. In the end, number of iterations and swarm size for PSO were set to 50 by trial and error.

There is a risk for underfitting or overfitting by the effect of numbers of neurons in the hidden layers if numbers of neurons were set to a too small or too large number. Therefore, numbers of neurons were initially set to 20 as a reasonable value from previous experience for both the hidden layers during the 80 runs of the algorithm.

After this, the best performing activation functions on average were determined to step up to determine the numbers of neurons in the hidden layers that will affect the performance of prospective ANN models to be composed.

PSO parameters that are c1, c2, w and w* that is the multiplier of w in each iteration are determined as 1.5, 2, 1 and 0.995, respectively by trial and error and also previous experience. Swarm particles generated in the PSO algorithm implemented in this section stores information about “position” which is the unknown but to be optimized, “velocity”, “cost” which is the value of the cost function used to evaluate the performance of ANN model, “best cost” which is the best cost value obtained among the iterations up to the latest iteration of the cost function, “model” which is stored as the current model, “best model” which is stored as the best model that yields best cost among the iterations up to the latest iteration and lastly “best position” which is the variable that yields the best cost.

In the further models to be explained later in this study, the information about the activation function was stored in the information cell of a particle, however since the best performing activation functions were selected during these runs and this information was stored in the ANN model, it was not stored in the particle. Major MLP

Regressor parameters used in construction of the ANN models used in this section are given in Table 7 below.

Table 7: Major MLP Regressor parameters in Step 1

Major MLP Regressor parameters	Explanation	Possible parameter values, intervals or choices	Parameter values or choices
activation	activation function	identity, logistic, tanh, relu	to be randomly selected
alpha	L2 penalty (regularization term) parameter	(0.00001, 0.001)	to be optimized
hidden_layer_sizes	number of hidden layers	no defined interval	(20,20)
max_iter	Maximum number of iterations	no defined interval	200
solver	Solver	lbfgs, sgd, adam	lbfgs
Tol	Tolerance for convergence	0.0001 (default)	0.0001

As the results can be seen in Tables 8,9,10 and 11, 80 PSO-ANN hybrid models were generated. While obtaining the possible best models by optimizing alpha over the interval of (0.00001, 0.001), RMSE, R^2 and MAE values of 4 different activation functions were computed for 20 models separately. Performances of models obtained with identity, logistic sigmoid, hyperbolic tangent (tanh) and rectified linear units (ReLU) functions were provided in tables 8, 9, 10 and 11, respectively. Average values of RMSE, R^2 and MAE were summarized in Table 12. Although identity function was the best choice in terms of time consumption, its prediction performance evaluated by these performance indicators was far behind the other three functions. The performances of the remaining three functions were in a tight competition. Regarding the results for both validation and test data, it was decided to step up the next phase for optimizing ANN models with numbers of neurons in the hidden layers with the

random usage of logistic sigmoid, hyperbolic tangent (tanh) and rectified linear units (ReLU) functions.

Table 8: Performances of the models with identity function

Identity Function							
Model ID	RMSE		R^2 Score		MAE		Elapsed time
	Validation data	Test data	Validation data	Test data	Validation data	Test data	
ID1	10.84	10.94	0.59	0.61	8.67	8.57	2 min. 55 sec.
ID2	10.84	10.94	0.59	0.61	8.67	8.57	3 min. 15 sec.
ID3	10.84	10.94	0.59	0.61	8.67	8.57	2 min. 45 sec.
ID4	10.84	10.94	0.59	0.61	8.67	8.57	2 min. 58 sec.
ID5	10.84	10.94	0.59	0.61	8.67	8.57	2 min. 55 sec.
ID6	10.84	10.94	0.59	0.61	8.67	8.57	2 min. 42 sec
ID7	10.84	10.94	0.59	0.61	8.67	8.57	2 min. 55 sec.
ID8	10.84	10.94	0.59	0.61	8.67	8.57	2 min. 47 sec.
ID9	10.84	10.94	0.59	0.61	8.67	8.57	2 min. 50 sec.
ID10	10.84	10.94	0.59	0.61	8.67	8.57	2 min. 55 sec.
ID11	10.86	11.07	0.57	0.61	8.35	8.66	2 min. 51 sec.
ID12	10.86	11.07	0.57	0.61	8.35	8.66	2 min. 43 sec.
ID13	10.86	11.07	0.57	0.61	8.35	8.66	2 min. 41 sec.
ID14	10.86	11.07	0.57	0.61	8.35	8.66	2 min. 56 sec.
ID15	10.86	11.07	0.57	0.61	8.35	8.66	3 min. 7 sec.
ID16	10.86	11.07	0.57	0.61	8.35	8.66	3 min. 11 sec.
ID17	10.86	11.07	0.57	0.61	8.35	8.66	3 min. 3 sec.
ID18	10.86	11.07	0.57	0.61	8.35	8.66	3 min. 4 sec.
ID19	10.86	11.07	0.57	0.61	8.35	8.66	3 min. 17 sec.
ID20	10.86	11.07	0.57	0.61	8.35	8.66	3 min. 1 sec.

Table 9: Performances of the models with logistic sigmoid function

Logistic Sigmoid Function							
Model ID	RMSE		R^2 Score		MAE		Elapsed time
	Validation data	Test data	Validation data	Test data	Validation data	Test data	
SIG1	5.24	6.95	0.84	0.81	5.85	4.87	13 min. 11 sec.
SIG2	5.13	5.86	0.76	0.87	5.84	4.19	13 min. 17 sec.
SIG3	5.32	6.28	0.90	0.85	5.08	4.30	13 min. 11 sec.
SIG4	5.47	6.99	0.87	0.81	5.39	5.04	12 min. 14 sec.
SIG5	5.21	6.83	0.83	0.82	4.93	4.87	13 min. 11 sec.
SIG6	5.35	6.15	0.79	0.85	6.15	4.51	13 min. 12 sec.
SIG7	5.37	7.16	0.78	0.80	4.63	4.74	13 min. 11 sec.
SIG8	5.30	6.19	0.85	0.85	6.39	4.53	13 min. 19 sec.
SIG9	5.29	6.39	0.84	0.84	4.81	4.61	13 min. 14 sec.
SIG10	5.42	5.61	0.85	0.88	5.75	4.00	13 min. 10 sec.
SIG11	5.27	7.58	0.81	0.81	4.74	5.48	24 min. 3 sec.
SIG12	5.57	7.30	0.83	0.83	5.19	5.33	22 min. 3 sec.
SIG13	5.49	6.96	0.86	0.84	4.80	5.06	17 min. 10 sec.
SIG14	5.21	7.57	0.83	0.81	5.24	5.65	17 min. 3 sec.
SIG15	5.43	7.89	0.86	0.80	5.43	5.82	16 min. 12 sec.
SIG16	5.38	8.10	0.88	0.79	5.43	6.06	17 min. 7 sec.
SIG17	5.25	7.19	0.87	0.83	4.84	5.13	17 min. 2 sec.
SIG18	5.46	7.70	0.84	0.81	5.13	5.72	17 min. 3 sec.
SIG19	5.33	7.37	0.85	0.82	5.35	5.45	19 min. 3 sec.
SIG20	5.44	6.80	0.85	0.85	4.66	5.00	20 min. 5 sec.

Table 10: Performances of the models with hyperbolic tangent (tanh) function

Hyperbolic Tangent Function							
Model ID	RMSE		R^2 Score		MAE		Elapsed time
	Validation data	Test data	Validation data	Test data	Validation data	Test data	
TANH 1	5.98	5.98	0.82	0.87	5.42	4.42	23 min. 49 sec.
TANH 2	6.08	5.97	0.81	0.87	5.58	4.60	24 min. 11 sec.
TANH 3	5.95	5.60	0.79	0.89	5.67	4.24	23 min. 42 sec.
TANH 4	5.95	5.88	0.81	0.88	6.44	4.45	23 min. 48 sec.
TANH 5	5.92	5.64	0.82	0.89	5.27	4.21	23 min. 39 sec.
TANH 6	6.03	6.39	0.84	0.85	5.12	4.99	23 min. 33 sec.
TANH 7	6.12	6.60	0.82	0.84	5.23	5.12	23 min. 44 sec.
TANH 8	5.93	5.92	0.82	0.87	5.46	4.58	23 min. 48 sec.
TANH 9	6.05	5.91	0.84	0.88	5.20	4.60	23 min. 46 sec.
TANH 10	5.74	5.91	0.79	0.87	5.92	4.62	23 min. 45 sec.
TANH 11	6.33	7.65	0.80	0.78	6.04	5.52	17 min. 51 sec.
TANH 12	6.22	7.55	0.81	0.79	5.20	5.53	18 min. 14 sec.
TANH 13	6.31	7.18	0.79	0.81	5.53	5.39	17 min. 28 sec.
TANH 14	6.31	7.18	0.79	0.81	5.53	5.39	17 min. 28 sec.
TANH 15	6.23	7.25	0.77	0.81	5.69	5.35	17 min. 6 sec.
TANH 16	6.35	7.34	0.78	0.80	5.20	5.42	16 min. 40 sec.
TANH 17	6.23	8.46	0.77	0.74	5.74	5.99	16 min. 43 sec.
TANH 18	6.10	7.07	0.68	0.82	5.71	5.12	17 min. 0 sec.
TANH 19	6.26	6.84	0.78	0.83	5.92	5.00	17 min. 2 sec.
TANH 20	6.06	6.76	0.77	0.83	6.90	4.92	16 min. 29 sec.

Table 11: Performances of the models with rectified linear units (ReLU) function

Rectified Linear Units (ReLU) Function							
Model ID	RMSE		R^2 Score		MAE		Elapsed time
	Validation data	Test data	Validation data	Test data	Validation data	Test data	
RELU 1	5.44	6.52	0.85	0.86	5.36	5.03	14 min. 39 sec.
RELU 2	5.45	6.75	0.85	0.85	5.13	5.18	14 min. 25 sec.
RELU 3	5.33	7.10	0.85	0.83	4.87	5.35	14 min. 42 sec.
RELU 4	5.44	6.47	0.79	0.86	4.98	4.83	14 min. 34 sec.
RELU 5	5.45	6.54	0.86	0.85	4.85	5.11	14 min. 37 sec.
RELU 6	5.27	6.54	0.86	0.85	4.48	4.91	14 min. 44 sec.
RELU 7	5.49	6.94	0.86	0.84	5.04	5.15	14 min. 49 sec.
RELU 8	5.29	6.35	0.85	0.86	4.68	4.90	14 min. 42 sec.
RELU 9	5.75	7.16	0.84	0.80	5.55	5.53	9 min. 46 sec.
RELU 10	5.72	7.38	0.87	0.79	5.45	5.74	9.0 min. 52 sec.
RELU 11	5.80	6.90	0.80	0.83	5.72	5.40	9 min. 59 sec.
RELU 12	5.83	6.56	0.77	0.85	5.53	5.11	10 min. 3 sec.
RELU 13	5.66	6.99	0.80	0.83	5.86	5.44	10 min. 6 sec.
RELU 14	5.83	6.86	0.79	0.83	5.68	5.24	10 min. 12 sec.
RELU 15	5.72	7.06	0.80	0.82	5.63	5.63	11 min. 17 sec.
RELU 16	5.66	6.75	0.82	0.84	5.37	5.20	11 min. 22 sec.
RELU 17	5.67	6.87	0.78	0.83	4.85	5.36	10 min. 1 sec.
RELU 18	5.71	6.79	0.79	0.84	5.08	5.31	9 min. 47 sec.
RELU 19	5.71	6.72	0.78	0.84	5.11	5.18	9 min. 41 sec.
RELU 20	5.72	6.83	0.80	0.83	5.59	5.26	9 min. 51 sec.

Table 12: Average values of RMSE, R^2 Score and MAE of the models in Step 1

Activation function	RMSE		R^2 Score		MAE		Elapsed time
	Validation data	Test data	Validation data	Test data	Validation data	Test data	
Identity	10.85	11.01	0.58	0.61	8.51	8.61	2 min 57 sec.
Logistic Sigmoid	5.35	6.94	0.84	0.83	5.28	5.02	15 min 54 sec.
Tanh	6.11	6.65	0.79	0.84	5.64	4.97	20 min. 29 sec.
ReLU	5.60	6.80	0.82	0.84	5.24	5.24	11 min. 57 sec.

3.2. Step 2: PSO-ANN Hybrid Models Optimizing Numbers of Neurons in Hidden Layers

Numbers of neurons in hidden layers play a key role for accuracy of ANN models. To reach accurately predicting models, PSO algorithm was implemented into ANN. Major MLP Regressor parameters used in construction of the ANN models used in optimization with numbers of neurons in the hidden layers are given in Table 13.

Table 13: Major MLP Regressor parameters used in Step 2

Major MLP Regressor parameters	Explanation	Possible parameter values, intervals or choices	Parameter values or choices
activation	activation function	identity, logistic, tanh, relu	to be randomly selected
Alpha	L2 penalty (regularization term) parameter	no defined interval	0.00047388
hidden_layer_sizes	number of hidden layers	[20,35]	to be optimized
max_iter	Maximum number of iterations	no defined interval	150
solver	Solver	lbfgs, sgd, adam	lbfgs
Tol	Tolerance for convergence	0.0001 (default)	0.00001

As in the previous section, number of iterations and swarm size for PSO were set to 50. Swarm particles generated in the PSO algorithm implemented in this section stores information about activation function since it was randomly selected by the swarm particles, in addition to the information stored explained in the previous section.

Interval for generating particles with positions of numbers of hidden layers was set to [20,35]. If numbers of neurons are set to too small, there is a potential for underfitting, whereas if those are too large, in reverse, it carries a risk for overfitting or inefficiency. Therefore, from previous experience and trial and error, interval of [20, 35] was decided. The results given by the 3 performance indicators which were calculated on two sets of data, validation and test data, confirmed the fitness of the interval selection.

Totally, 20 PSO-ANN hybrid models were generated, which had model names from HL1 to HL20. As can be seen from Table 14 below, 6 of those had the numbers of neurons of (20,35), just on the boundaries of the interval. Moreover, among the 20 models generated, the best model was from those 6 models whose numbers of neurons were (20,35). Therefore, for the next step, numbers of neurons in hidden layer were determined as (20,35).

The scatter plot for HL7 which was the best model in this section demonstrated the relationship between experimental test outputs (actual values) and the predicted values of the model in Fig. 4 below. Iterations of PSO algorithm vs. errors in terms of RMSE is provided in Fig. 5 below.

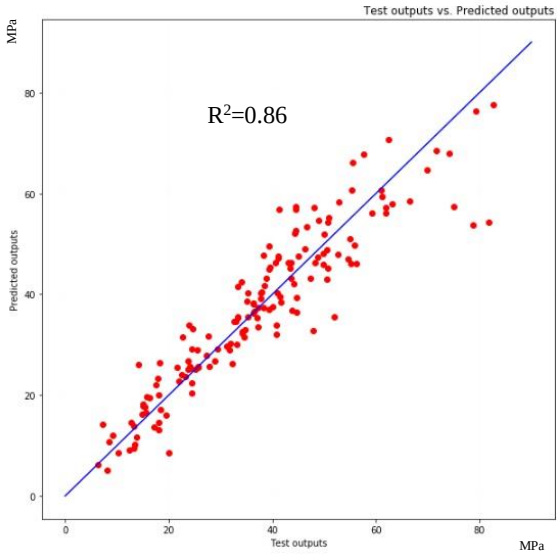

Figure 4: Test Outputs vs. Predicted Outputs for PSO-ANN Model HL7

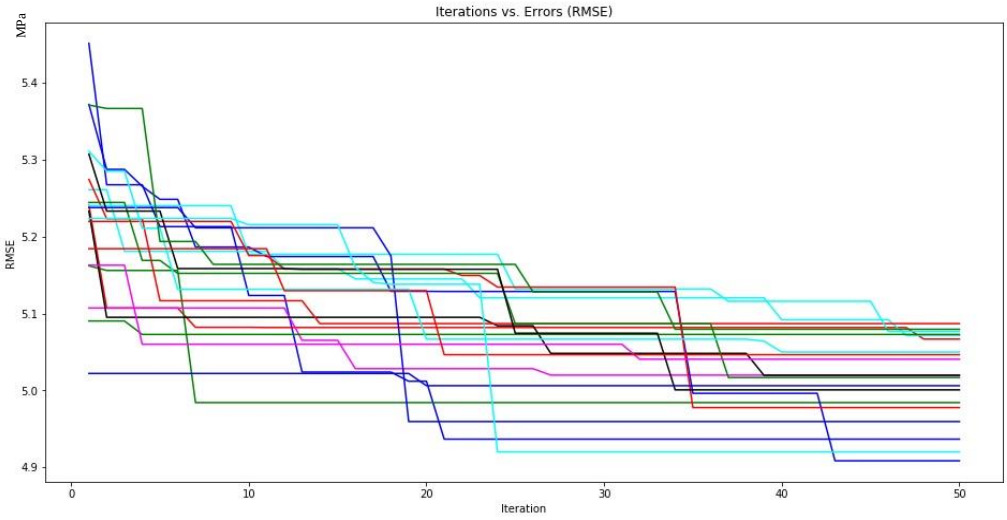

Figure 5: Iterations vs. RMSE for PSO-ANN models HL1-HL20

Table 14: Performances of the PSO-ANN models HL1-HL20

Model ID	Number of neurons	RMSE		R2 Score		MAE		Elapsed time
		Validation data	Test data	Validation data	Test data	Validation data	Test data	
HL1	[20, 35]	4.94	6.92	0.87	0.83	4.86	4.74	13 min. 36 sec.
HL2	[22, 23]	5.07	6.22	0.86	0.86	5.08	4.61	11 min. 50 sec.
HL3	[20, 20]	4.98	6.60	0.88	0.85	4.62	4.73	11 min. 0 sec.
HL4	[20, 35]	5.07	6.88	0.90	0.83	4.66	4.88	13 min. 4 sec.
HL5	[20, 32]	5.02	6.89	0.88	0.83	4.86	5.26	14 min. 50 sec.
HL6	[32, 20]	5.02	7.10	0.89	0.82	4.65	5.46	13 min. 11 sec.
HL7	[20, 35]	4.91	6.30	0.90	0.86	4.17	4.63	14 min. 15 sec.
HL8	[25, 32]	5.09	6.54	0.88	0.85	5.43	5.07	14 min. 44 sec.
HL9	[21, 26]	5.07	7.15	0.90	0.82	5.31	5.26	12 min. 47 sec.
HL10	[28, 25]	5.08	7.05	0.86	0.82	4.32	5.49	13 min. 22 sec.
HL11	[25, 35]	4.96	6.76	0.85	0.84	5.06	5.10	12 min. 35 sec.
HL12	[27, 20]	4.98	6.86	0.89	0.83	4.75	5.09	13 min. 34 sec.
HL13	[29, 20]	5.02	6.69	0.88	0.84	5.37	5.29	13 min. 16 sec.
HL14	[20, 35]	5.05	6.71	0.87	0.84	5.09	5.11	12 min. 25 sec.
HL15	[20, 20]	5.04	6.91	0.85	0.83	4.45	5.09	11 min. 16 sec.
HL16	[21, 22]	5.00	6.90	0.87	0.83	4.83	5.14	11 min. 2 sec.
HL17	[27, 23]	5.01	6.77	0.87	0.84	4.56	5.06	12 min. 50 sec.
HL18	[20, 35]	5.05	6.92	0.87	0.83	5.54	5.15	13 min. 27 sec.
HL19	[35, 20]	5.08	7.01	0.88	0.83	4.35	5.40	13 min. 4 sec.
HL20	[20, 35]	4.92	6.47	0.85	0.85	4.97	4.70	14 min. 11 sec.
AVERAGE		**5.02**	**6.78**	**0.87**	**0.84**	**4.85**	**5.06**	**13 min. 1 sec.**

3.3. Step 3: Final PSO-ANN Hybrid Models

3.3.1. PSO-ANN Hybrid Models Optimizing Alpha (L2 Regularization term) with L-BFGS as Solver and Optimized ANN Architecture

Major MLP Regressor parameters used in construction of the PSO-ANN hybrid models used in optimization with number of neurons in hidden layers is given in Table 15 below.

Table 15: Major MLP Regressor parameters used in Step 3 first part

Major MLP Regressor parameters	Explanation	Possible parameter values, intervals or choices	Parameter values or choices
activation	activation function	identity, logistic, tanh, relu	to be randomly selected
alpha	L2 penalty (regularization term) parameter	no defined interval	to be optimized
hidden_layer_sizes	number of hidden layers	no defined interval	(20,35)
max_iter	Maximum number of iterations	no defined interval	150
solver	Solver	lbfgs, sgd, adam	lbfgs
tol	Tolerance for convergence	0.0001 (default)	0.00001

Number of iterations and swarm size for PSO were set to 50. There is no specific interval for generating particles with positions of alpha values. The interval for positions of alpha values was set to [0.00001, 0.001]. Alpha value is very important for preventing models from overfitting and helps models be optimized and predict target values accurately. Since there is no certain rule for that, the interval abovementioned is determined with experience.

In this section, total 60 PSO-ANN models were generated. These models had numbers of neurons 20 and 35 for the first and the second hidden layers, respectively. While these 60 models shared the same properties, except randomly selected or optimized items, those differed in terms of their activation function. Although activation functions employed in the previous section were selected randomly for 20 models, the endeavor in this section gave the opportunity to generate more models with the best choice of numbers of neurons. Logistic sigmoid, hyperbolic tangent and relu functions were contributed in the hybrid model equally per function. Tables 16,

17, 18 and 19 showing the performances of PSO-ANN models with logistic sigmoid, hyperbolic tangent (tanh) and rectified linear units (ReLU) functions are provided below.

Table 16: Performances of the PSO-ANN models with logistic sigmoid function in Step 3 first part

Model ID	Activation function	RMSE		R^2 Score		MAE		Elapsed time
		Validation data	Test data	Validation data	Test data	Validation data	Test data	
L1	log. sig.	5.14	6.04	0.85	0.87	5.11	4.58	18 min. 0 sec.
L2	log. sig.	5.34	6.16	0.87	0.86	4.74	4.52	17 min. 49 sec.
L3	log. sig.	5.38	6.32	0.85	0.85	5.17	4.57	17 min. 36 sec.
L4	log. sig.	5.28	6.48	0.82	0.85	5.29	4.93	17 min. 38 sec.
L5	log. sig.	5.27	6.06	0.84	0.87	4.71	4.46	17 min. 47 sec.
L6	log. sig.	5.38	6.42	0.86	0.85	4.84	4.41	17 min. 37 sec.
L7	log. sig.	5.32	6.64	0.86	0.84	5.26	4.88	17 min. 35 sec.
L8	log. sig.	5.38	6.47	0.88	0.85	4.79	4.63	17 min. 46 sec.
L9	log. sig.	5.35	6.34	0.87	0.85	4.54	4.49	17 min. 58 sec.
L10	log. sig.	5.39	6.68	0.87	0.84	4.99	4.93	17 min. 41 sec.
L11	log. sig.	5.31	6.27	0.87	0.86	5.00	4.68	17 min. 52 sec.
L12	log. sig.	5.34	6.38	0.87	0.85	5.30	4.83	17 min. 48 sec.
L13	log. sig.	5.37	5.56	0.85	0.89	5.49	4.00	17 min. 47 sec.
L14	log. sig.	5.38	6.20	0.86	0.85	4.58	4.57	18 min. 57 sec.
L15	log. sig.	5.29	6.30	0.85	0.85	5.15	4.63	17 min. 32 sec.
L16	log. sig.	5.35	6.28	0.87	0.86	5.07	4.55	17 min. 28 sec.
L17	log. sig.	5.39	6.69	0.87	0.84	4.64	4.90	17 min. 25 sec.
L18	log. sig.	5.38	6.08	0.87	0.86	5.04	4.65	17 min. 26 sec.
L19	log. sig.	5.39	5.53	0.87	0.89	4.66	4.04	17 min. 25 sec.
L20	log. sig.	5.36	6.47	0.87	0.85	4.70	4.90	17 min. 40 sec.

Table 17: Performances of the PSO-ANN models with hyperbolic tangent (tanh) function in Step 3 first part

Model ID	Activation function	RMSE		R^2 Score		MAE		Elapsed time
		Validation data	Test data	Validation data	Test data	Validation data	Test data	
T1	tanh	4.92	6.50	0.89	0.86	4.86	4.97	16 min. 17 sec.
T2	tanh	4.84	6.89	0.88	0.84	4.64	5.32	16 min. 30 sec.
T3	tanh	4.85	6.80	0.90	0.85	4.38	5.10	17 min. 52 sec.
T4	tanh	4.65	6.87	0.88	0.84	4.72	5.15	17 min. 48 sec.
T5	tanh	4.84	7.17	0.85	0.83	4.62	5.35	18 min. 5 sec.
T6	tanh	4.84	6.80	0.88	0.85	4.40	5.25	17 min. 50 sec.
T7	tanh	4.93	6.63	0.88	0.85	4.26	4.87	17.min.59 sec.
T8	tanh	4.84	6.31	0.89	0.87	5.24	4.79	18 min. 0 sec.
T9	tanh	4.99	6.65	0.89	0.85	4.25	5.12	16 min. 50 sec.
T10	tanh	4.82	7.28	0.87	0.82	4.56	5.47	16 min. 14 sec.
T11	tanh	4.80	6.94	0.88	0.84	4.51	5.26	16 min. 47 sec.
T12	tanh	4.92	6.77	0.87	0.85	4.59	5.07	17 min. 31 sec.
T13	tanh	4.94	6.58	0.91	0.86	4.44	4.92	20 min. 29 sec.
T14	tanh	4.86	6.88	0.90	0.84	3.99	5.12	16 min. 24 sec.
T15	tanh	4.78	6.76	0.87	0.85	5.22	5.06	17 min. 55 sec.
T16	tanh	4.95	7.03	0.88	0.84	4.48	5.22	17 min. 49 sec.
T17	tanh	4.91	6.65	0.88	0.85	4.46	5.04	17min. 12 sec.
T18	tanh	4.86	6.50	0.89	0.86	4.54	4.89	17 min. 39 sec.
T19	tanh	4.88	6.72	0.88	0.85	4.55	5.09	17 min. 44 sec.
T20	tanh	4.85	6.94	0.85	0.84	4.54	5.26	17 min. 1 sec.

Table 18: Performances of the PSO-ANN models with rectified linear units (ReLU) function in Step 3 first part

Model ID	Activation function	RMSE		R^2 Score		MAE		Elapsed time
		Validation data	Test data	Validation data	Test data	Validation data	Test data	
R1	relu	6.19	6.55	0.84	0.86	5.30	4.84	14 min. 54 sec.
R2	relu	6.09	7.20	0.82	0.83	5.56	5.31	17 min. 0 sec.
R3	relu	6.06	6.27	0.82	0.87	5.16	4.49	15 min. 34 sec.
R4	relu	6.25	6.52	0.84	0.86	5.35	4.89	14 min. 30 sec.
R5	relu	5.91	6.12	0.84	0.88	5.34	4.60	12 min. 59 sec.
R6	relu	6.26	6.28	0.83	0.87	5.14	4.68	12 min. 48 sec.
R7	relu	6.24	6.37	0.85	0.87	5.55	4.71	12 min. 40 sec.
R8	relu	6.18	6.28	0.83	0.87	5.54	4.58	12 min. 39 sec.
R9	relu	6.18	6.30	0.85	0.87	5.46	4.66	12 min. 49 sec.
R10	relu	5.97	7.00	0.83	0.84	5.48	5.37	12 min. 41 sec.
R11	relu	6.10	6.25	0.82	0.87	5.39	4.44	12 min. 41 sec.
R12	relu	6.19	6.31	0.82	0.87	5.15	4.72	12 min. 55 sec.
R13	relu	6.24	6.44	0.83	0.87	5.13	4.62	12 min. 50 sec.
R14	relu	6.22	6.36	0.78	0.87	5.52	4.64	12 min. 43 sec.
R15	relu	6.24	6.54	0.82	0.86	5.21	4.94	12 min. 43 sec.
R16	relu	6.28	6.48	0.82	0.87	5.05	4.80	12 min. 55 sec.
R17	relu	6.30	6.57	0.84	0.86	6.20	5.04	12 min. 46 sec.
R18	relu	6.23	6.08	0.82	0.88	5.79	4.46	12 min. 57 sec.
R19	relu	5.98	6.50	0.84	0.86	5.60	4.62	12min. 44 sec.
R20	relu	6.14	6.17	0.83	0.88	5.33	4.60	12 min. 51 sec.

Table 19: Average values of RMSE, R2 Score and MAE of the models in Step 3 first part

Activation function	RMSE		R^2 Score		MAE		Elapsed time
	Validation data	Test data	Validation data	Test data	Validation data	Test data	
log. sig.	5.34	6.27	0.86	0.86	4.95	4.61	17 min. 45 sec.
tanh	4.86	6.78	0.88	0.85	4.56	5.12	17 min. 30 sec.
relu	6.16	6.43	0.83	0.87	5.41	4.75	13 min. 20 sec.

Performance of the models are given in the Tables 16, 17, 18 and 19. On average, models with logistic sigmoid and hyperbolic tangent functions performed better than models with relu function, while relu demonstrated the best time consumption by far. If R^2 scores on data are considered among all models; L13, L19, T8, T13 which also has the highest R^2 score, T18, R5, R18 and R20 outshine. The plot showing iterations vs errors (RMSE) of 8 selected model is provided in Fig. 6. The scatter plots showing test outputs vs. predicted outputs for L13, L19, T8, T13, T18, R5, R18 and R20 are provided in figures 7, 8, 9, 10, 11, 12, 13, 14 below.

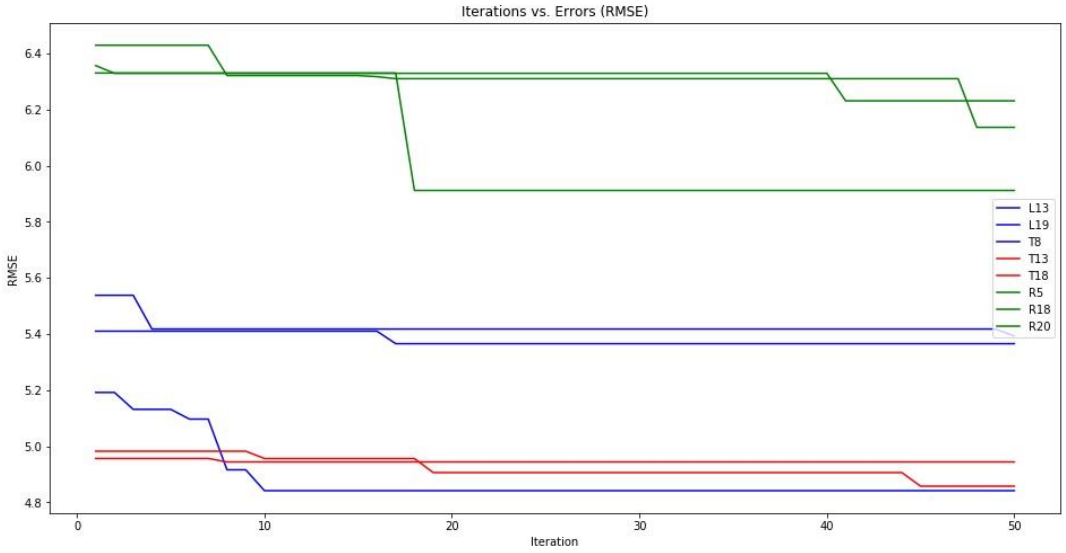

Figure 6: Iteration vs RMSE for 8 selected PSO-ANN models

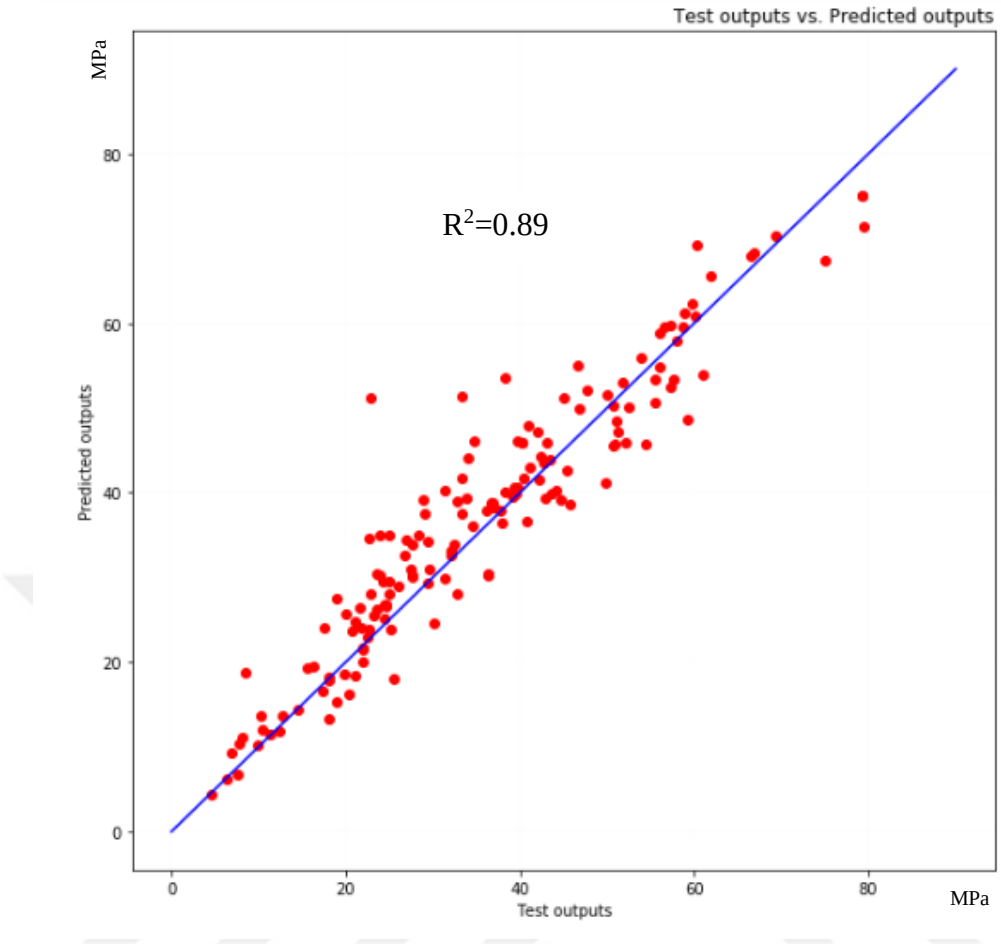

Figure 7: Test Outputs vs. Predicted Outputs for PSO-ANN Model L13

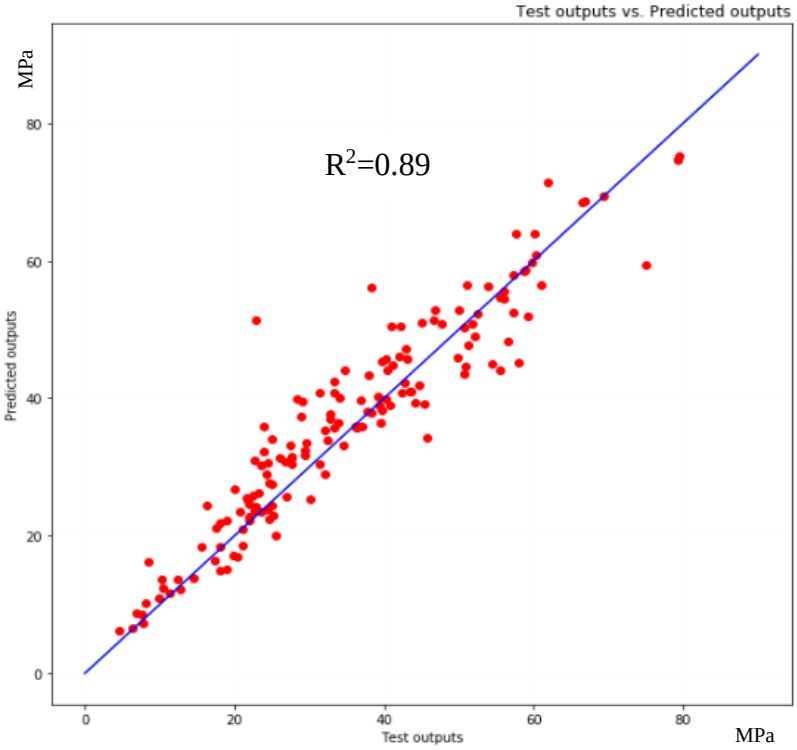

Figure 8: Test Outputs vs. Predicted Outputs for PSO-ANN Model L19

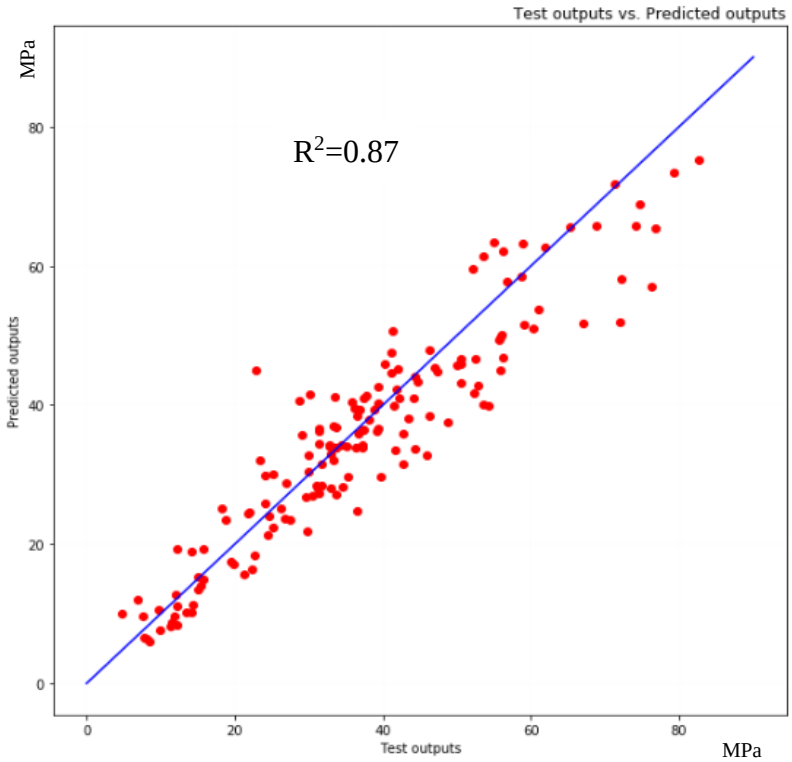

Figure 9: Test Outputs vs. Predicted Outputs for PSO-ANN Model T8

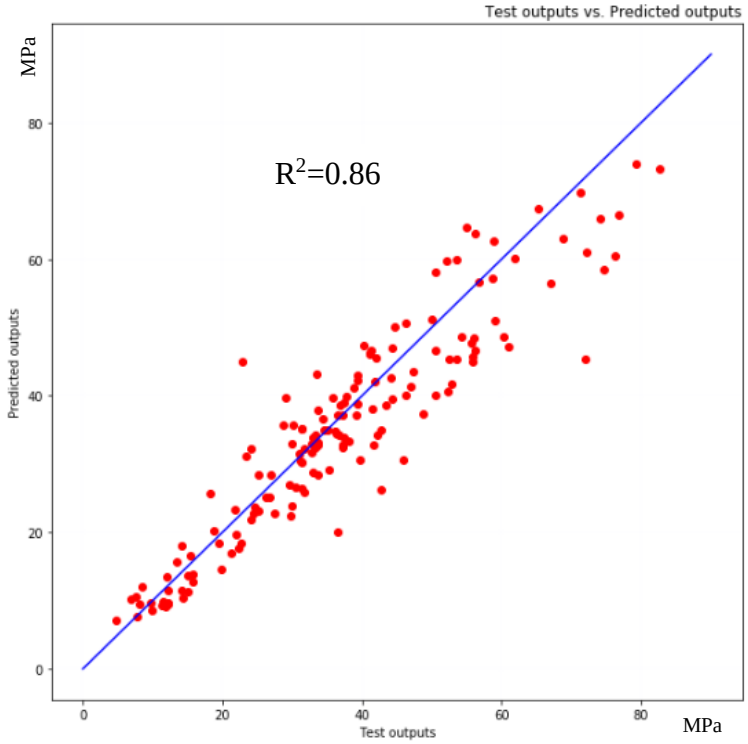

Figure 10: Test Outputs vs. Predicted Outputs for PSO-ANN Model T13

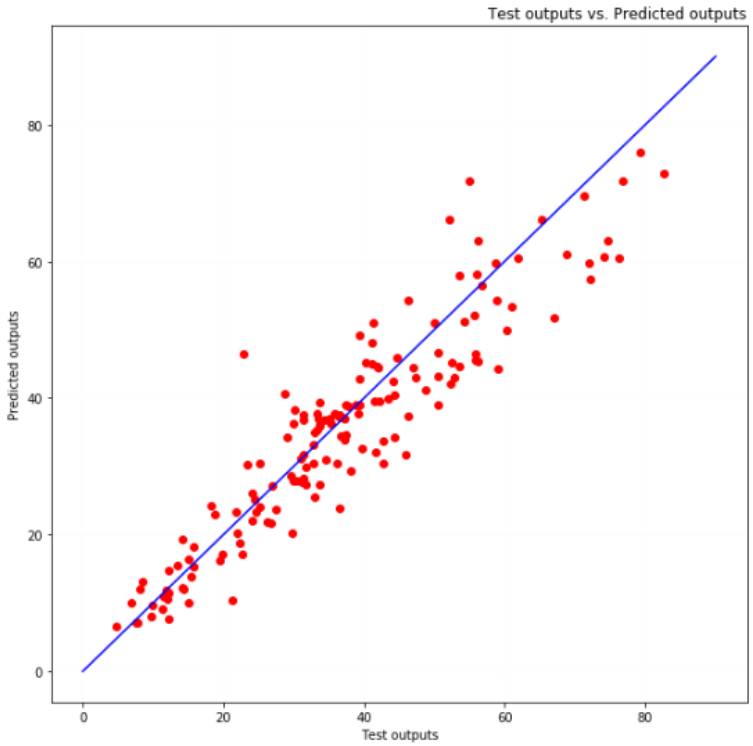

Figure 11: Test Outputs vs. Predicted Outputs for PSO-ANN Model T18

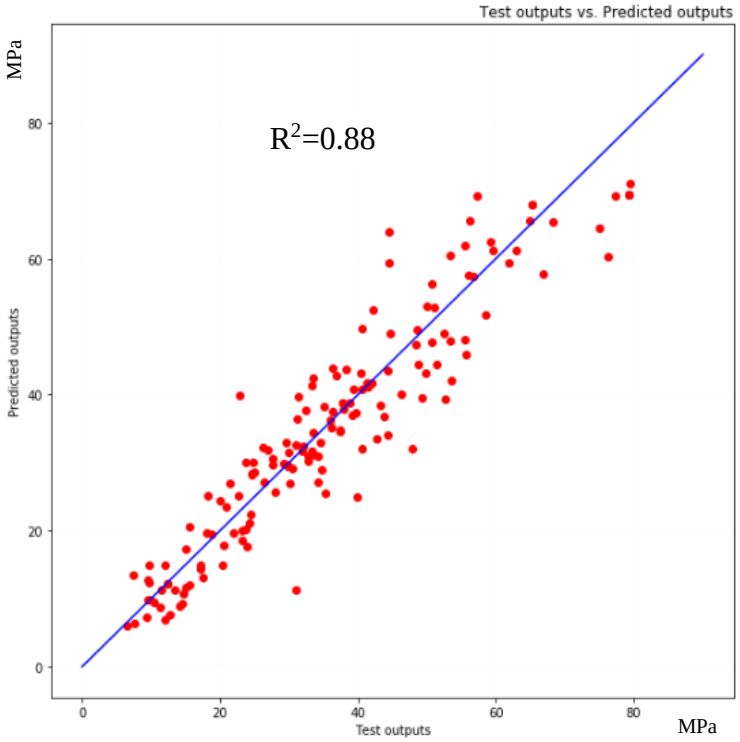

Figure 12: Test Outputs vs. Predicted Outputs for PSO-ANN Model R5

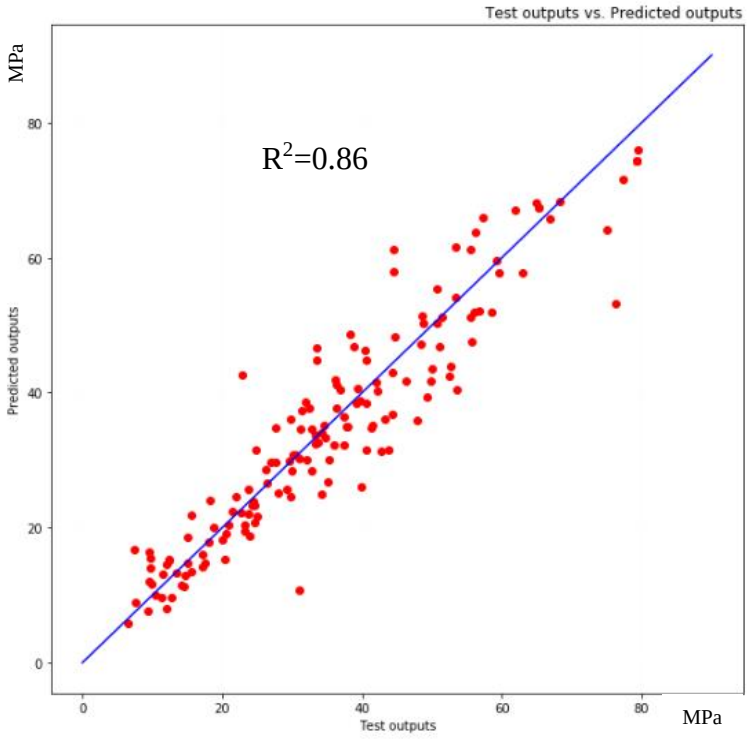

Figure 13: Test Outputs vs. Predicted Outputs for PSO-ANN Model R19

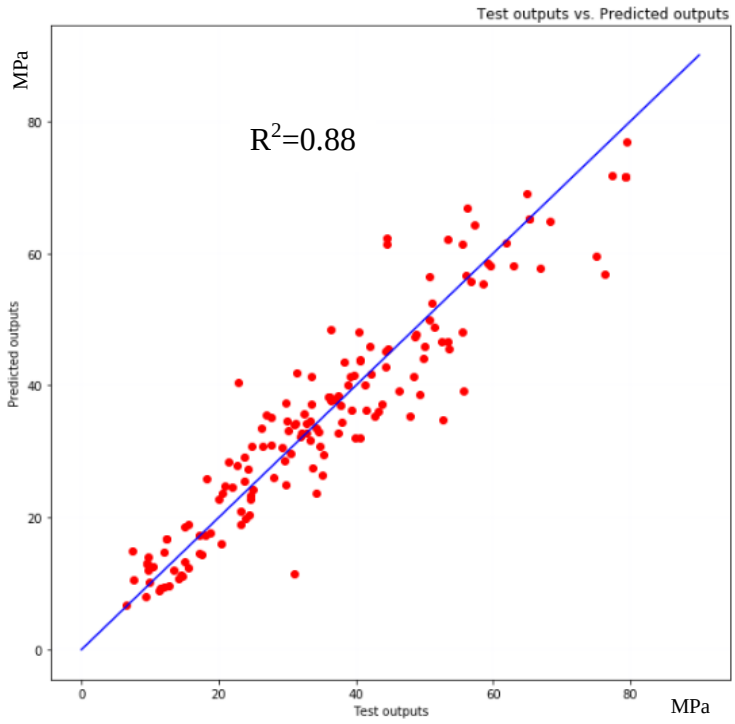

Figure 14: Test Outputs vs. Predicted Outputs for PSO-ANN Model R20

3.3.2. PSO-ANN Hybrid Models Optimizing Alpha (L2 Regularization term) and Initial Learning Rate with Stochastic Gradient Descent (SGD) and Optimized ANN Architecture

Major MLP Regressor parameters used in construction of the PSO-ANN hybrid models in optimizing alpha (L2 Regularization term) and initial learning rate with Stochastic Gradient Descent (SGD) as solver are given in Table 20.

Table 20: Major MLP Regressor parameters used in Step 3 second part

Major MLP Regressor parameters	Explanation	Possible parameter values, intervals or choices	Parameter values or choices
activation	activation function	identity, logistic, tanh, relu	to be randomly selected
alpha	L2 penalty (regularization term) parameter	no defined interval	to be optimized

hidden_layer_sizes	number of hidden layers	no defined interval	(20,35)
solver	Solver	sgd, adam	sgd
learning rate	Learning rate schedule	constant, invscaling, adaptive	constant
learning_rate_init	Initial learning rate	[0.000001, 0.001]	to be optimized
max_iter	Maximum number of iterations	no defined interval	150
tol	Tolerance for convergence	0.0001 (default)	0.00001

Number of iterations and swarm size for PSO were set to 50. There is no specific interval for generating particles with positions of alpha values was set to [0.000001, 0.001]. Alpha value is used in optimization again. Since there is no certain rule for that, the interval abovementioned was expanded with respect to the previous section. Initial learning rate also used the same interval for optimization.

10 PSO-ANN models were generated. These models had numbers of neurons 20 and 35, respectively for the first and the second hidden layers because it was decided that among the other possibilities this pair of numbers for neurons performed better. Additionally, activation function was randomly selected by swarm particles from the best performing three activation functions.

Quick convergence of SGD solver can be seen obviously in the models. However, the performance of SGD was far behind the performance of solver L-BFGS in all three performance indicators as can be seen in Table 21. The best performing model SGD6 among the models with Stochastic Gradient Descent (SGD) can be seen in Fig. 15.

Table 21: Performances of the PSO-ANN models with solver SGD

Model ID	Activation function	RMSE		R^2 Score		MAE		Elapsed time
		Validation data	Test data	Validation data	Test data	Validation data	Test data	
SGD1	relu	7.25	7.58	0.67	0.80	8.06	5.83	13 min. 14 sec.

SGD2	relu	7.10	7.66	0.64	0.79	7.97	5.74	12 min. 52 sec.
SGD3	log. sig.	6.89	8.34	0.66	0.75	8.45	6.61	15 min. 53 sec.
SGD4	relu	7.34	8.12	0.66	0.77	8.18	6.23	12 min. 4 sec.
SGD5	tanh	7.72	9.06	0.68	0.71	8.02	7.25	18 min. 53 sec.
SGD6	relu	7.37	7.49	0.67	0.80	8.06	5.66	13 min. 48 sec.
SGD7	relu	7.28	8.56	0.62	0.74	7.37	6.72	12 min. 15 sec.
SGD8	relu	7.21	7.61	0.64	0.79	8.10	5.81	12 min. 27 sec.
SGD9	log. sig.	7.15	8.16	0.68	0.76	8.10	6.35	15 min. 52 sec.
SGD10	tanh	7.16	8.05	0.66	0.77	8.15	6.11	16 min. 48 sec.

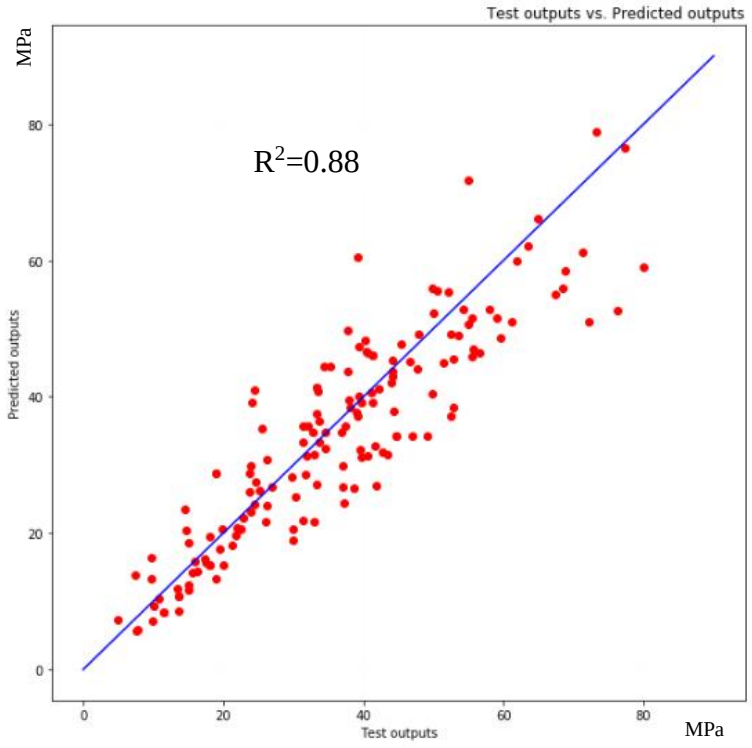

Figure 15: Test Outputs vs. Predicted Outputs for PSO-ANN Model SGD6

CHAPTER 4

CONCLUSIONS

In this thesis, the relationship between concrete strength and mix design properties was evaluated with artificial neural network (ANN) hybrid algorithms. PSO algorithm and ANN were brought together and hybridized to generate model making more accurate predictions of concrete strength. Particle Swarm Optimization (PSO) was selected from metaheuristic algorithms due to its convergence capability, easy implementation and adoption, and robustness [60].

ANN was constructed in the medium of Python. Python provides flexibility for implementing PSO to ANN with its wide range of libraries for different purposes. Python libraries of random, math, pandas, numpy, pandas.plotting, matplotlib.pyplot, scikit-learn, pickle and time were used for the purposes of randomizing, mathematical operations, data manipulation and analysis, scientific computing, plotting, machine learning, saving and loading models, and time keeping, respectively.

In this thesis, PSO-ANN and ANN models were employed on a dataset which was donated by Prof. I-Cheng Yeh from Department of Information Management, Chung-Hua University to Machine Learning Repository, Center for Machine Learning and Intelligent Systems, University of California, Irvine on August 3, 2007.

The dataset was normalized to be fit into the interval of [0,1] and splitted into three portions as training, validation and test sets. For the models obtained, root-mean-square error (RMSE), mean absolute error (MAE) and coefficient of determination (COD) (R^2) were selected as performance indicators.

Activation functions of identity, logistic sigmoid, hyperbolic tangent and rectified linear unit functions were employed in the ANN models. Among the options for solver in the ANN structure, Limited memory Broyden-Fletcher-Goldfarb-Shanno Algorithm (L-BFGS) and Stochastic Gradient Descent (SGD) were used.

In total, 170 different models were generated with different variables to increase the prediction power of ANN models. In the pursuit of obtaining the most accurate hybrid PSO-ANN model, the performances of the activation functions were tested, at first. Logistic sigmoid, hyperbolic tangent (tanh) and rectified linear units (relu) functions were very close to one another while identity function was far behind in terms of accuracy, but it worked very quickly when compared to the others.

After determining that logistic sigmoid, hyperbolic tangent and rectified linear unit functions were worth using in the next steps of hybridizing, numbers of neurons in the hidden layers were optimized as a major part of obtaining the best performing PSO-ANN model since it directly affected the success of the models and it was a key factor for underfitting or overfitting. When the models were generated for the hidden layers, 6 of those gave the same numbers of neurons. Considering the pool of possibilities, it demonstrated a strong tendency to reach an optimized ANN architecture around the determined values of numbers of neurons in the hidden layers. For the two hidden layers, 20 for the first hidden layer and 35 for the second layer were determined as the best choices due the values of performance indicators.

Once the architecture of the ANN was determined, it was the task to find the correct combination of other variables. For this, L-BFGS and SGD as solvers, alpha and initial learning rate as continuous variables, and activation functions were used in various combinations. Since the performances of activation functions were very close one another, it was not very easy to pick up one of those, but they helped increasing the randomization of candidate solutions. It is seen that L-BFGS outperformed SGD. The best predicting models were obtained from PSO-ANN hybrid models optimizing alpha (L2 Regularization term) with L-BFGS as solver. The highest R^2 score obtained from all models is 0.91.

When evaluating the success of the predictions made, it can be said that it is directly related to the numbers of instances and input features of data. The method proposed in the thesis is also steady in predictions. The predictions do not fluctuate in a large interval of values of performance indicators. In the future, in the case of having a large set of data in terms of both samples and input features for different types of

concrete under different circumstances, more accurate predictions can easily be made with the method proposed in this thesis.

REFERENCES

[1] W. Ji-Zong, N. Hong-Guang, and H. Jin-Yun, "The application of automatic acquisition of knowledge to mix design of concrete," *Cem. Concr. Res.*, vol. 29, no. 12, pp. 1875–1880, 1999.

[2] S.-C. Lee, "Prediction of concrete strength using artificial neural networks," *Eng. Struct.*, vol. 25, no. 7, pp. 849–857, 2003.

[3] S. Popovics, *History of mathematical model for strength development of Portland cement concrete*, vol. 95. 1998.

[4] L. M. Snell, J. Van Roekel, and N. D. Wallace, "Predicting early concrete strength," *Concr. Int.*, vol. 11, no. 12, pp. 43–47, 1989.

[5] I.-C. Yeh, "Modeling of strength of high-performance concrete using artificial neural networks," *Cem. Concr. Res.*, vol. 28, no. 12, pp. 1797–1808, 1998.

[6] B. K. R. Prasad, H. Eskandari, and B. V. V. Reddy, "Prediction of compressive strength of SCC and HPC with high volume fly ash using ANN," *Constr. Build. Mater.*, vol. 23, no. 1, pp. 117–128, 2009.

[7] A. Krenker, "Introduction to the Artificial Neural Networks," J. Bešter, Ed. Rijeka: IntechOpen, 2011, p. Ch. 1.

[8] C.-M. Kuan, "Artificial Neural Networks BT - The New Palgrave Dictionary of Economics," London: Palgrave Macmillan UK, 2017, pp. 1–12.

[9] D. M. D'Addona, "Neural Network BT - CIRP Encyclopedia of Production Engineering," T. I. A. for Produ, L. Laperrière, and G. Reinhart, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 1–9.

[10] P.-N. Tan, "Neural Networks BT - Encyclopedia of Database Systems," L. Liu and M. T. Özsu, Eds. New York, NY: Springer New York, 2016, pp. 1–5.

[11] S. L. Prime, "Neural Networks BT - Encyclopedia of Personality and Individual Differences," V. Zeigler-Hill and T. K. Shackelford, Eds. Cham:

Springer International Publishing, 2017, pp. 1–4.

[12] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[13] D. O. Hebb, “The organization of behavior. A neuropsychological theory,” 1949.

[14] F. Rosenblatt, “Principles of neurodynamics. perceptrons and the theory of brain mechanisms,” Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[15] B. Widrow and M. E. Hoff, “Adaptive switching circuits,” Stanford Univ Ca Stanford Electronics Labs, 1960.

[16] M. Minsky and S. Papert, “Perceptron: an introduction to computational geometry,” *MIT Press. Cambridge, Expand. Ed.*, vol. 19, no. 88, p. 2, 1969.

[17] “Introduction to artificial neural networks,” in *Proceedings Electronic Technology Directions to the Year 2000*, 1995, pp. 36–62.

[18] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” *Proc. Natl. Acad. Sci.*, vol. 79, no. 8, pp. 2554–2558, 1982.

[19] X.-S. Zhang, “Introduction to Artificial Neural Network BT - Neural Networks in Optimization,” X.-S. Zhang, Ed. Boston, MA: Springer US, 2000, pp. 83–93.

[20] J. Zupan, “Introduction to artificial neural network (ANN) methods: what they are and how to use them,” *Acta Chim. Slov.*, vol. 41, p. 327, 1994.

[21] W. S. Sarle, “Neural networks and statistical models,” 1994.

[22] I. A. Basheer and M. Hajmeer, “Artificial neural networks: fundamentals, computing, design, and application,” *J. Microbiol. Methods*, vol. 43, no. 1, pp. 3–31, 2000.

[23] R. Hecht-Nielsen, “Neurocomputing Addison-Wesley,” *Reading, MA*, 1990.

[24] D. Svozil, V. Kvasnicka, and J. Pospichal, “Introduction to multi-layer feed-

forward neural networks," *Chemom. Intell. Lab. Syst.*, vol. 39, no. 1, pp. 43–62, 1997.

[25] M. Kuhn and K. Johnson, *Applied predictive modeling*, vol. 26. Springer, 2013.

[26] M. Hellström and J. Behler, "Neural Network Potentials in Materials Modeling BT - Handbook of Materials Modeling : Methods: Theory and Modeling," W. Andreoni and S. Yip, Eds. Cham: Springer International Publishing, 2018, pp. 1–20.

[27] S. Haykin, "Neural networks and learning machines,–3rd ed., Copyright by Pearson Education," *Inc., Up. Saddle River, New Jersey*, vol. 7458, 2009.

[28] S. Lahmiri, "On simulation performance of feedforward and NARX networks under different numerical training algorithms," in *Handbook of Research on Computational Simulation and Modeling in Engineering*, IGI Global, 2016, pp. 171–183.

[29] J. Nocedal, "Updating quasi-Newton matrices with limited storage," *Math. Comput.*, vol. 35, no. 151, pp. 773–782, 1980.

[30] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Program.*, vol. 45, no. 1, pp. 503–528, 1989.

[31] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv Prepr. arXiv1609.04747*, 2016.

[32] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[33] D. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*. 2014.

[34] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[35] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA Neural networks Mach.*

Learn., vol. 4, no. 2, pp. 26–31, 2012.

[36] P.-N. Tan, M. Steinbach, and V. Kumar, "Introduction to data mining: Pearson addison wesley," *Boston*, 2005.

[37] R. Tauler, B. Walczak, and S. D. Brown, *Comprehensive chemometrics: chemical and biochemical data analysis*. Elsevier, 2009.

[38] A. A. Benczúr, L. Kocsis, and R. Pálovics, "Reinforcement Learning, Unsupervised Methods, and Concept Drift in Stream Learning BT - Encyclopedia of Big Data Technologies," S. Sakr and A. Zomaya, Eds. Cham: Springer International Publishing, 2018, pp. 1–8.

[39] M. Paliwal and U. A. Kumar, "Neural networks and statistical techniques: A review of applications," *Expert Syst. Appl.*, vol. 36, no. 1, pp. 2–17, 2009.

[40] B. C. Hardgrave, R. L. Wilson, and K. A. Walstrom, "Predicting graduate student success: A comparison of neural networks and traditional techniques," *Comput. Oper. Res.*, vol. 21, no. 3, pp. 249–263, 1994.

[41] M. C. M. De Carvalho, M. S. Dougherty, A. S. Fowkes, and M. R. Wardman, "Forecasting travel demand: a comparison of logit and artificial neural network methods," *J. Oper. Res. Soc.*, vol. 49, no. 7, pp. 717–722, 1998.

[42] W. Remus and T. Hill, *Neural network models of managerial judgment*. 1990.

[43] L.-Y. Chang, "Analysis of freeway accident frequencies: negative binomial regression versus artificial neural network," *Saf. Sci.*, vol. 43, no. 8, pp. 541–557, 2005.

[44] T. J. Griinke, "Development of an artificial neural network (ANN) for predicting tribological properties of kenaf fibre reinforced epoxy composites (KFRE)," 2013.

[45] H.-G. Ni and J.-Z. Wang, "Prediction of compressive strength of concrete by neural networks," *Cem. Concr. Res.*, vol. 30, no. 8, pp. 1245–1250, 2000.

[46] G. Trtnik, F. Kavčič, and G. Turk, "Prediction of concrete strength using ultrasonic pulse velocity and artificial neural networks," *Ultrasonics*, vol. 49,

no. 1, pp. 53–60, 2009.

[47] M. M. Alshihri, A. M. Azmy, and M. S. El-Bisy, “Neural networks for predicting compressive strength of structural light weight concrete,” *Constr. Build. Mater.*, vol. 23, no. 6, pp. 2214–2219, 2009.

[48] Z.-H. Duan, S.-C. Kou, and C.-S. Poon, “Prediction of compressive strength of recycled aggregate concrete using artificial neural networks,” *Constr. Build. Mater.*, vol. 40, pp. 1200–1206, 2013.

[49] H. Naderpour, A. Kheyroddin, and G. G. Amiri, “Prediction of FRP-confined compressive strength of concrete using artificial neural networks,” *Compos. Struct.*, vol. 92, no. 12, pp. 2817–2829, 2010.

[50] S. Matthys, H. Toutanji, K. Audenaert, and L. Taerwe, “Axial load behavior of large-scale columns confined with fiber-reinforced polymer composites,” *ACI Struct. J.*, vol. 102, no. 2, p. 258, 2005.

[51] L. Lam and J. G. Teng, “Strength models for fiber-reinforced plastic-confined concrete,” *J. Struct. Eng.*, vol. 128, no. 5, pp. 612–623, 2002.

[52] J. B. Mander, M. J. N. Priestley, and R. Park, “Theoretical stress-strain model for confined concrete,” *J. Struct. Eng.*, vol. 114, no. 8, pp. 1804–1826, 1988.

[53] A. Sadrmomtazi, J. Sobhani, and M. A. Mirgozar, “Modeling compressive strength of EPS lightweight concrete using regression, neural network and ANFIS,” *Constr. Build. Mater.*, vol. 42, pp. 205–216, 2013.

[54] A. T. A. Dantas, M. Batista Leite, and K. de Jesus Nagahama, “Prediction of compressive strength of concrete containing construction and demolition waste using artificial neural networks,” *Constr. Build. Mater.*, vol. 38, pp. 717–722, 2013.

[55] F. Khademi, M. Akbari, S. M. Jamal, and M. Nikoo, “Multiple linear regression, artificial neural network, and fuzzy logic prediction of 28 days compressive strength of concrete,” *Front. Struct. Civ. Eng.*, vol. 11, no. 1, pp. 90–99, 2017.

[56] F. Glover, “Future paths for integer programming and links to artificial intelligence,” *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, 1986.

[57] F. W. Glover and G. A. Kochenberger, *Handbook of metaheuristics*, vol. 57. Springer Science & Business Media, 2006.

[58] K. Sörensen and F. W. Glover, “Metaheuristics,” *Encycl. Oper. Res. Manag. Sci.*, pp. 960–970, 2013.

[59] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: Overview and conceptual comparison,” *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, 2003.

[60] F. Xie, Q. Wang, and G. Li, “Optimization research of FOC based on PSO of induction motors,” in *2012 15th International Conference on Electrical Machines and Systems (ICEMS)*, 2012, pp. 1–4.

[61] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in *MHS’95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[62] X. L. Chen, J. P. Fu, J. L. Yao, and J. F. Gan, “Prediction of shear strength for squat RC walls using a hybrid ANN–PSO model,” *Eng. Comput.*, vol. 34, no. 2, pp. 367–383, 2018.

[63] K. W. Chau, “Application of a PSO-based neural network in analysis of outcomes of construction claims,” *Autom. Constr.*, vol. 16, no. 5, pp. 642–646, 2007.

[64] C. Sudheer, R. Maheswaran, B. K. Panigrahi, and S. Mathur, “A hybrid SVM-PSO model for forecasting monthly streamflow,” *Neural Comput. Appl.*, vol. 24, no. 6, pp. 1381–1389, 2014.

[65] M. Hasanipanah, M. Noorian-Bidgoli, D. J. Armaghani, and H. Khamesi, “Feasibility of PSO-ANN model for predicting surface settlement caused by tunneling,” *Eng. Comput.*, vol. 32, no. 4, pp. 705–715, 2016.

[66] B. Gordan, D. J. Armaghani, M. Hajihassani, and M. Monjezi, “Prediction of

seismic slope stability through combination of particle swarm optimization and neural network," *Eng. Comput.*, vol. 32, no. 1, pp. 85–97, 2016.

[67] D. J. Armaghani, R. S. N. S. Bin Raja, K. Faizi, and A. S. A. Rashid, "Developing a hybrid PSO–ANN model for estimating the ultimate bearing capacity of rock-socketed piles," *Neural Comput. Appl.*, vol. 28, no. 2, pp. 391–405, 2017.

[68] E. T. Mohamad, D. J. Armaghani, E. Momeni, and S. V. A. N. K. Abad, "Prediction of the unconfined compressive strength of soft rocks: a PSO-based ANN approach," *Bull. Eng. Geol. Environ.*, vol. 74, no. 3, pp. 745–757, 2015.

[69] C. Qi, A. Fourie, and Q. Chen, "Neural network and particle swarm optimization for predicting the unconfined compressive strength of cemented paste backfill," *Constr. Build. Mater.*, vol. 159, pp. 473–478, 2018.

[70] E. Momeni, D. J. Armaghani, M. Hajihassani, and M. F. M. Amin, "Prediction of uniaxial compressive strength of rock samples using hybrid particle swarm optimization-based artificial neural networks," *Measurement*, vol. 60, pp. 50–63, 2015.

[71] Q.-L. Chang, H.-Q. Zhou, and C.-J. Hou, "Using particle swarm optimization algorithm in an artificial neural network to forecast the strength of paste filling material," *J. China Univ. Min. Technol.*, vol. 18, no. 4, pp. 551–555, 2008.

[72] H. Mashhadban, S. S. Kutanaei, and M. A. Sayarinejad, "Prediction and modeling of mechanical properties in fiber reinforced self-compacting concrete using particle swarm optimization algorithm and artificial neural network," *Constr. Build. Mater.*, vol. 119, pp. 277–287, 2016.

[73] S. S. Gilan, H. B. Jovein, and A. A. Ramezanianpour, "Hybrid support vector regression–Particle swarm optimization for prediction of compressive strength and RCPT of concretes containing metakaolin," *Constr. Build. Mater.*, vol. 34, pp. 321–329, 2012.

[74] H.-C. Tsai, "Predicting strengths of concrete-type specimens using hybrid multilayer perceptrons with center-unified particle swarm optimization,"

Expert Syst. Appl., vol. 37, no. 2, pp. 1104–1112, 2010.

[75] K. E. Parsopoulos, "UPSO: A unified particle swarm optimization scheme," *Lect. Ser. Comput. Comput. Sci.*, vol. 1, pp. 868–873, 2004.

[76] Y. Liu, Z. Qin, Z. Shi, and J. Lu, "Center particle swarm optimization," *Neurocomputing*, vol. 70, no. 4–6, pp. 672–679, 2007.

[77] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[78] M. Hajihassani, D. J. Armaghani, H. Sohaei, E. T. Mohamad, and A. Marto, "Prediction of airblast-overpressure induced by blasting using a hybrid artificial neural network and particle swarm optimization," *Appl. Acoust.*, vol. 80, pp. 57–67, 2014.

[79] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[80] L. Shi, S. T. K. Lin, Y. Lu, L. Ye, and Y. X. Zhang, "Artificial neural network based mechanical and electrical property prediction of engineered cementitious composites," *Constr. Build. Mater.*, vol. 174, pp. 667–674, 2018.