Development of a Neural Network-Based Program for Predicting the Compressive Strength of Concrete

1. Introduction

Problem Statement:

1. Traditional Method involves cube creation and destructive testing of specimens in UTM. Which is time consuming, generate huge waste and economically inefficiently
2. The program's adaptability for training with specific laboratory data further enhances its applicability.

Objectives:

- Develop a neural network program for accurate prediction of concrete's compressive strength.
- Implement a flexible program architecture for varying input variables and network parameters.
- Creating a portable ANN Program that can be use for any data training besides its utility for concrete strength prediction

The Compressive Strength and Mix Proportioning of Concrete

According to source [1], the 28-day compressive strength of concrete is influenced by seven key factors, which are;

1. Water
2. Cement
3. fine aggregate (sand)
4. coarse aggregate
5. blast furnace slag
6. fly ash
7. superplasticizer

The quantities of the above ingredients are used in a cubic meter of concrete mix. However, several other factors also affect the strength prediction but they are mostly considered are secondary factors. Consequently, the compressive strength of concrete can be represented as a mathematical function dependent on these seven variables:

$$y=f(x)$$

Here, x is defined as:

$$x=[x_1,x_2\ldots,x_7]^T$$

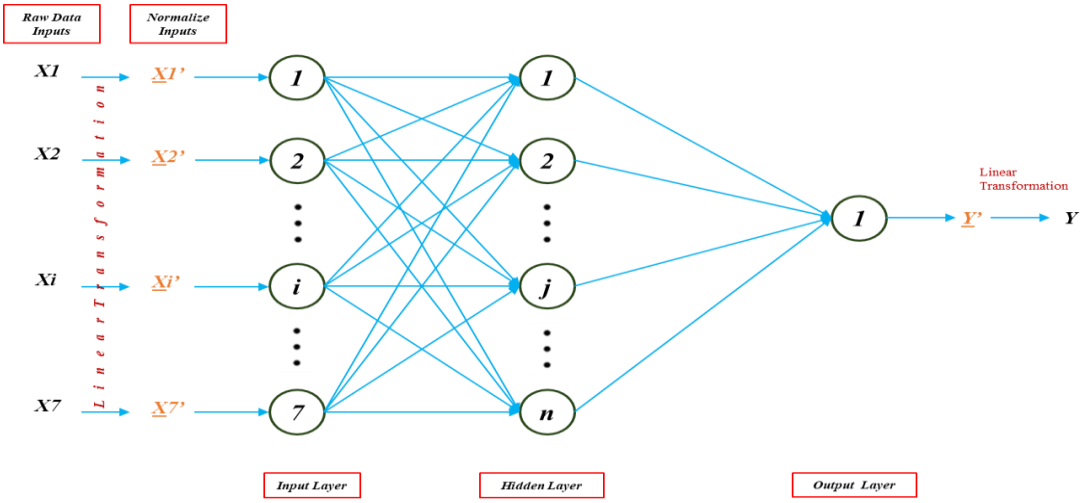
where x_1 to x_7 correspond to the mix proportioning factors, and y represents the compressive strength of the concrete

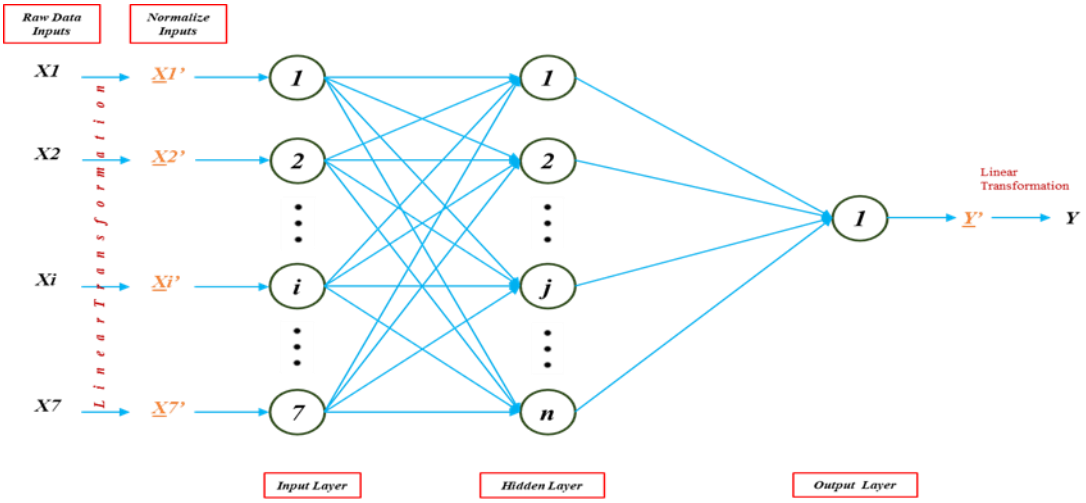
The Artificial Neural Network

1. Basic Architecture

The ANN model consist of

1. **An Input Layer** (7 Nodes): Can be vary as per requirement
2. **A Hidden Layer** (Nodes =70): Can be vary as per requirement
3. **An Output Layer** (Nodes = 1): Can be vary as per requirement





2. Mathematical Concept

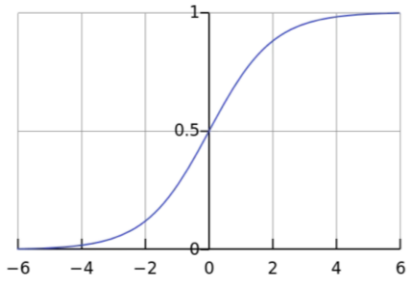
1. **Training Concept:** Back Propagation Algorithm with Gradient Descent Strategy of Error Minimization
2. **Error Cost Function:**

$$\text{Error} = \sum(Target - output)^2$$

3. **Activation Function:** Sigmoid Activation Function is used for activation

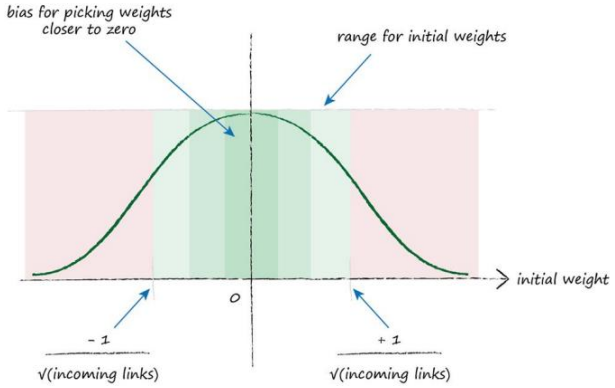
Sigmoid Function

$$A = \frac{1}{1+e^{-x}}$$



The activation function is smooth, nonlinear, and bounds outputs within the (0,1) range, preventing excessive activation values unlike linear function and counter the non-linearity of data [2].

4. **Weight Initializations:** Xavier Weight Initialization:
In this the weight is sampling from a normal distribution with a mean of zero and a standard deviation determined by the inverse of the square root of the number of incoming links to a node.



5. **Forward Propagation**

$x = a^{(1)}$ *Input layer*

$z^{(2)} = W^{(1)}x$ *neuron value at Hidden$_1$ layer*

$a^{(2)} = f(z^{(2)})$ *activation value at Hidden$_1$ layer*

$z^{(3)} = W^{(2)}a^{(2)}$ *neuron value Output layer*

$a^{(3)} = f(z^{(3)})$ *activation value at Outpu layer*

6. **Backward Propagation**

$$error_{output} = target - actual$$

$$error_{hidden} = w^{T}_{hidden\_output} \cdot error_{output}$$

$$error_{input} = w^{T}_{input\_hidden} \cdot error_{hidden}$$

$$\frac{\partial E}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} \Sigma_n (t_n - o_n)^2$$

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial o_k} \cdot \frac{\partial o_k}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k) \cdot \frac{\partial o_k}{\partial w_{jk}}$$

$$\frac{\partial E}{\partial w_{jk}} = -2(t_k - o_k) \cdot \frac{\partial}{\partial w_{jk}} sigmoid (\Sigma_j w_{jk} \cdot o_j)$$

$$new\ w_{jk} = old\ w_{jk} - \alpha \cdot \frac{\partial E}{\partial w_{jk}}$$

3. Data for Training and Testing

1. Data Collection

Data on actual concrete mix proportions was sourced from a previous thesis by [3]. To prevent over-training and over-fitting, the dataset was split into two sections: training and testing. The testing data, randomly selected and excluded from the training phase, consists of 15% of the total data. Consequently, the training set contains 85% of the data, amounting to 410 samples, while the testing set includes 72 samples.

2. Data Scaling

- As the data has large value but activation function gives output between 1 and 0. So to needs to be scaled between 1 and 0 [1] [2].
- For Smooth working of ANN data has been linearly transformed into the range of 0.01 to 1 by using following formula [1].

$$\xi_i = \frac{x_i - x_{i,\min}}{x_{i,\max} - x_{i,\min}}$$

$$\eta = \frac{y - y_{\min}}{y_{\max} - y_{\min}}$$

- Data Shifting from zero to 0.01 to avoid neurons dying.

4. Performance Check

1. **Root Mean Square Error :** In this approach the root mean square error is calculated of each epoch and plotted vs epoch. The graph visualize how the function is trained during the training function

2. Testing Performance Check

- To check the coefficient of efficiency (C.E.) to evaluate performance:

$$C.E. = 1 - E^2_{r.r.m.s.}$$

$$E_{r.r.m.s.} = \sqrt{\sum (y^{(p)} - y^{(a)})^2 / \sum (y^{(a)})^2}$$

Here, Y^p is predicted value and Y^a is actual value

3. Graph Between Actual value and predicted value

Equity Line : An equality line is plotted. To note that the predicted values are close to the linear fit, confirming an agreement with the concrete compressive strength values.

4. Programming Approach

- Object Neural Network Programming Approach is used to creating the Neural Network
- One class created with Three Method Named as (__init__(), training_fn() and testing_fn())
- **Libraries Used:**

```
1. import math
2. import numpy as np  # for array operation
3. import scipy.special  # for actication function
4. import matplotlib.pyplot # for plotting of data
```

- Instance of a Neural Network: Created For Training and Testing of compressive strength

```
input_nodes                    =                    7
hidden_nodes                   =                   70
output_nodes                    =                   1
learning_rate                  =                 0.01
#  creating  the  instance  of  concrete_neural  Network
n  =  concrete_NN(input_nodes,  hidden_nodes,  output_nodes,
learning_rate)
```

Model Results and Discussion

A promising outcome was revealed by neural network model developed to estimate the compressive strength of concrete. The model uses a structure with one output node that represents the compressive strength, and it employs seven input parameters – water, cement, fine aggregate (sand), coarse aggregate, blast furnace slag, fly ash and superplasticizer–. The number of nodes in the three layers of this network were 7,70 and 1 respectively with a learning rate of 0.01 over thousand epochs used for training the model to capture the intricate link between mix proportions and concrete strength.

The RMSE pattern over epochs shows a rapid decrease in error followed by no change, indicating that during training the model has learned well from input data without overfitting. The error reduction stops at 10%, suggesting that further reduction in the error rate could be achieved through more diverse datasets. As such, the model will keep on learning and refining its accuracy.

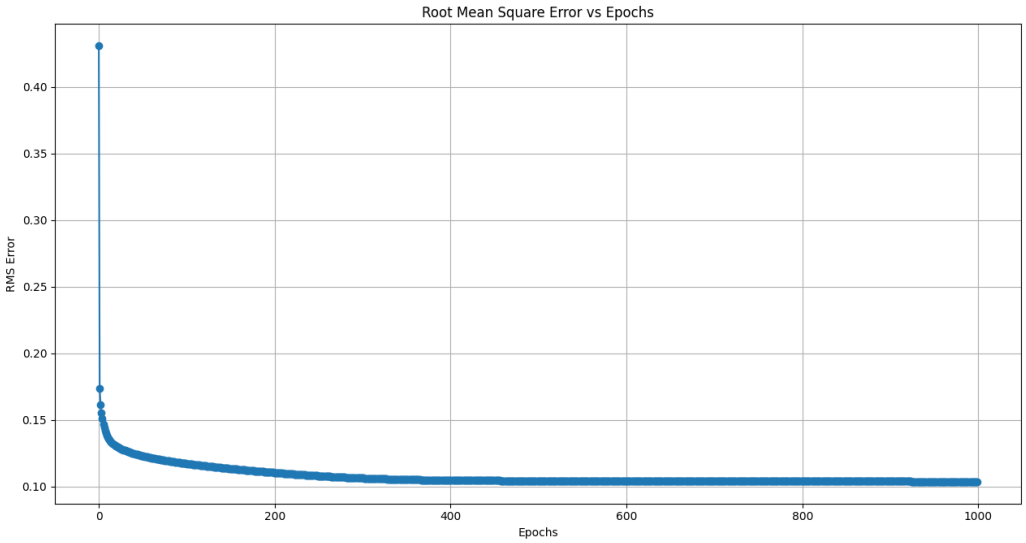
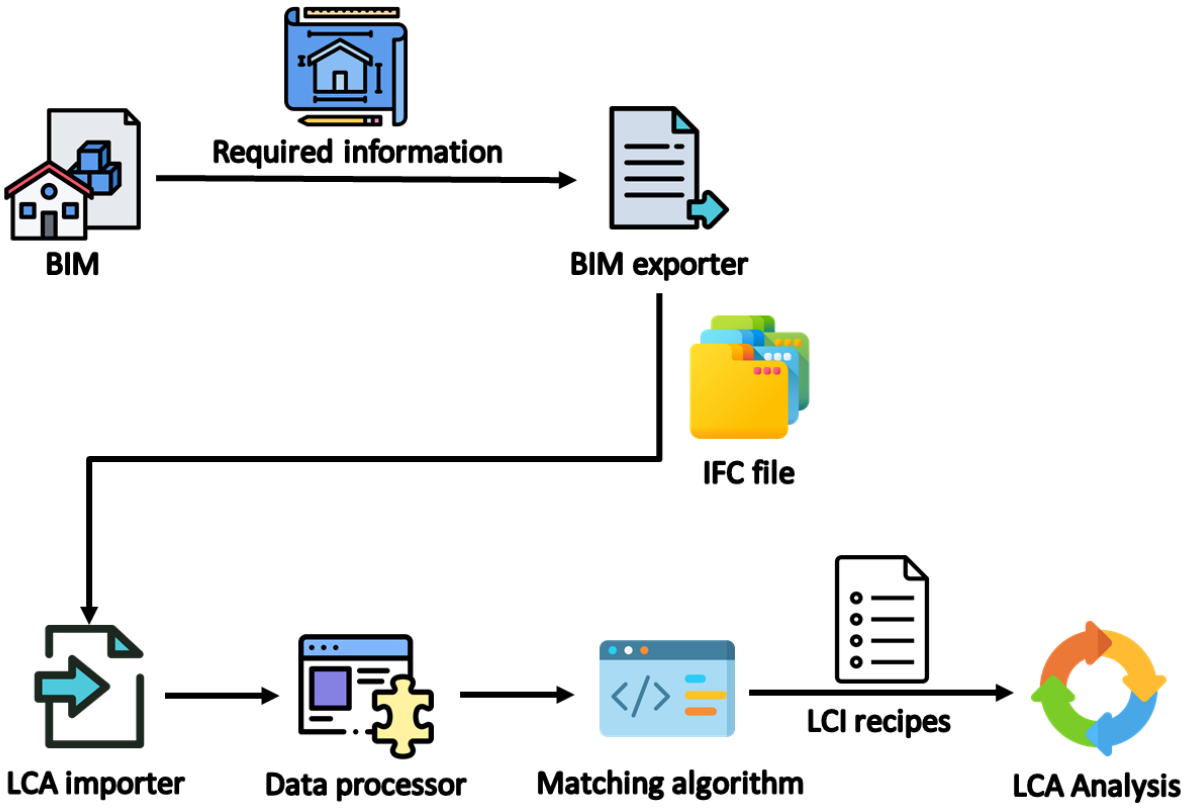


Figure 1: RMS_Loss Graph During Training

- While testing the model shows 95% performance in achieving the desired accuracy and further adjustment can be made by training and giving more experimental data.



IDLE Shell 3.11.7

File Edit Shell Debug Options Window Help

```
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec  4 2023, 19:24:49) [MSC v.1937 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\Final Submission\ANN_Program_VGU6PB.py
relative root-mean-square error 0.21795427269656614
The Model Performance is:  0.9524959350133109
>>>
```

Figure 2: Model Performance on 1000 Epoches

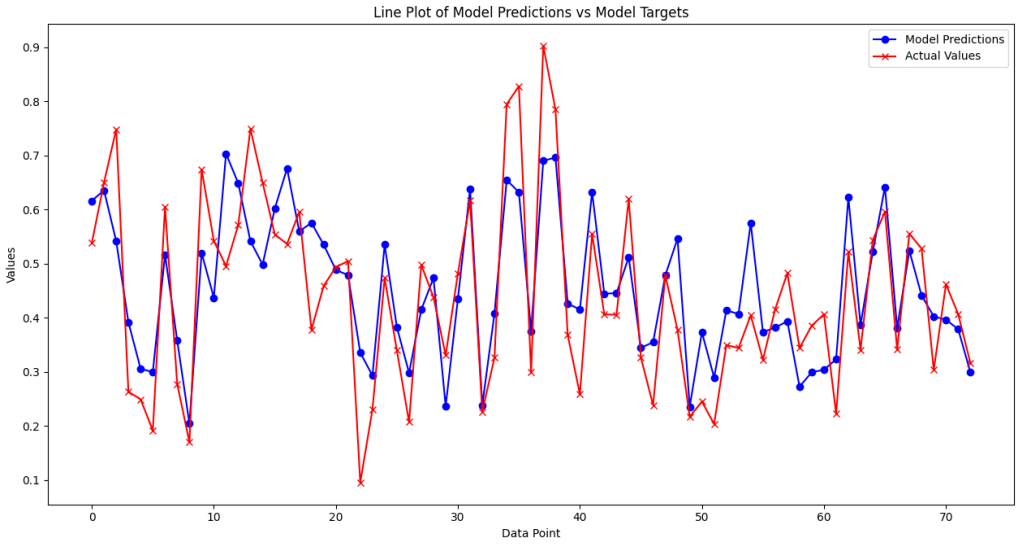


Figure 3: Graph Between Actual data and NN Predicted Data

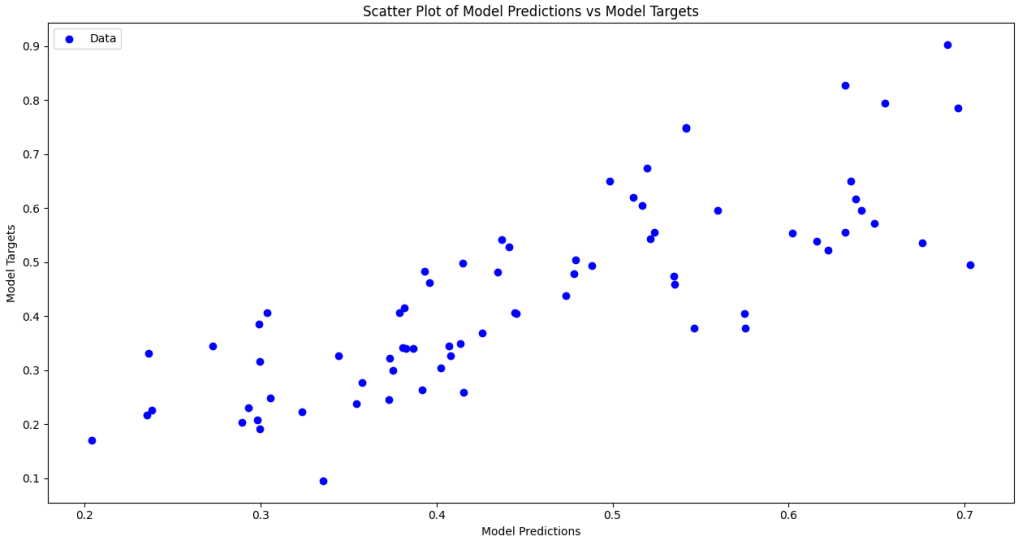


Figure 4: Graph Between Target Value and Predicted Value During Testing: Equity Line

References

[1]. Lin, C.-J.; Wu, N.-J. An ANN Model for Predicting the Compressive Strength of Concrete. Appl. Sci. 2021, 11, 3798. https:// doi.org/10.3390/app11093798

[2]. Rashid, T. (2016). Make Your Own Neural Network: A Gentle Journey Through the Mathematics of Neural Networks and Making Your Own Using the Python Computer Language.

[3]. Hao, C.-Y.; Shen, C.-H.; Jan, J.-C.; Hung, S.-K. A Computer-Aided Approach to Pozzolanic Concrete Mix Design. Adv. Civ. Eng. 2018, 2018, 4398017

[4]. Silva, V.P.; Carvalho, R.d.A.; Rêgo, J.H.d.S.; Evangelista, F., Jr. Machine Learning-Based Prediction of the Compressive Strength of Brazilian Concretes: A Dual-Dataset Study. Materials 2023, 16, 4977. https://doi.org/10.3390/ ma16144977

[5]. https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0