Concrete Compressive Strength Prediction Using ANN

CIL3090: Smart Infrastructure Engineering Course Project

Colab-Link : Smart_Infra_Project
Team Members - Alok (B22CI004) , Ankush (B22CI005)

1. Introduction

Concrete stands out as a cornerstone material in civil enginering, prized for its versatillity and strenght. Typically, engineers rely on lab tests to measure this strenght, but these can be slow and expenssive. This project set out to create an Artifical Neural Network (ANN) model that predicts concrete compressive strenght using mix proportions and age. The goal is to make a tool that could cut down on the need for extensive physcal testing by offering quick, reliable estimats.

What are Artificial Neural Networks?

Artifical Neural Networks are computatonal models inspired by the structure and function of biological neural networks in the human brain. They consist of interconnected processing units (neurons) organized in layers that can learn patterns from data to make predictons.

How ANNs Work

1. **Forward Propagation**: Input data flows through the network, with each neuron computing a weighted sum of its inputs, adding a bias term, and applying an activation function.
2. **Loss Calculation**: The difference between predicted and actual values is quantified using a loss function (e.g., Mean Squared Error for our regression problem).
3. **Backpropagation**: The network adjusts its weights and biases by propagating the error backward through the network using gradient descent.
4. **Optimization**: Algorithms like Adam, SGD, or RMSprop update weights to minimize the loss function.

2. Dataset Analysis and Preprocessing

2.1 Dataset Overview

For this work, I used a dataset from the UCI Machine Learning Reposatory, which includes 1,030 concrete samples. It tracks eight input variables:

- Cement (kg/m³)
- Blast Furnace Slag (kg/m³)
- Fly Ash (kg/m³)
- Water (kg/m³)
- Superplastisizer (kg/m³)
- Coarse Aggregate (kg/m³)
- Fine Aggregate (kg/m³)
- Age (days, ranging from 1 to 365)

The output is the concrete compressive strenght, measured in megapascals (MPa).

```
First 5 rows:
   Cement   Slag  Fly Ash  Water  Superplasticizer  Coarse Aggregate  \
0   540.0    0.0      0.0  162.0               2.5            1040.0   
1   540.0    0.0      0.0  162.0               2.5            1055.0   
2   332.5  142.5      0.0  228.0               0.0             932.0   
3   332.5  142.5      0.0  228.0               0.0             932.0   
4   198.6  132.4      0.0  192.0               0.0             978.4   

   Fine Aggregate  Age (day)  Compressive strength (MPa)  
0           676.0         28                       79.99  
1           676.0         28                       61.89  
2           594.0        270                       40.27  
3           594.0        365                       41.05  
4           825.5        360                       44.30  
```

2.2 Exploratory Data Analysis

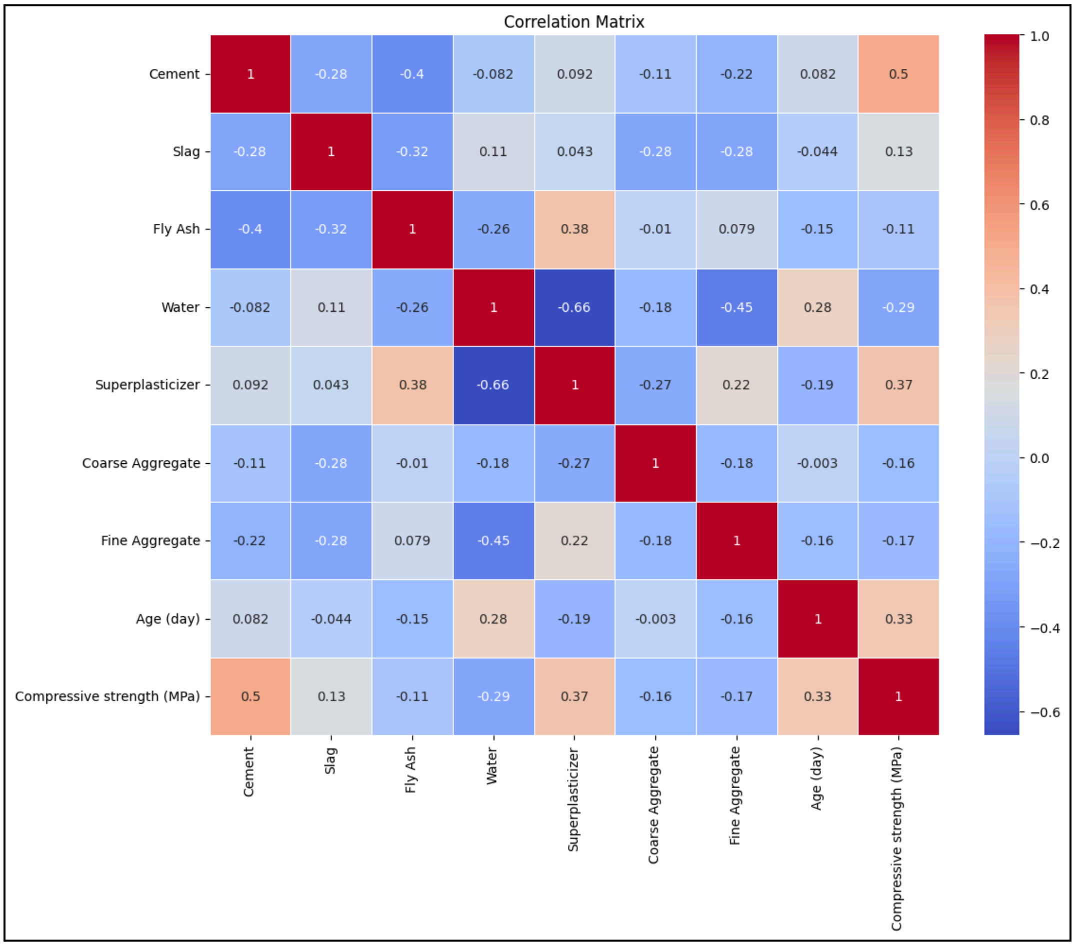
To get a feel for the data, I dug into its patterns and relatonships. Here's what stood out:

Data Quality

- No missing entries—everything was complet.
- The input ranges made sense for typical concrete mixs.
- Ages spanned a full year, giving a broad view of strenght development.

```
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Cement                         1030 non-null   float64
 1   Slag                           1030 non-null   float64
 2   Fly Ash                        1030 non-null   float64
 3   Water                          1030 non-null   float64
 4   Superplasticizer               1030 non-null   float64
 5   Coarse Aggregate               1030 non-null   float64
 6   Fine Aggregate                 1030 non-null   float64
 7   Age (day)                      1030 non-null   int64
 8   Compressive strength (MPa)     1030 non-null   float64
```

Feature Correlations :



Looking at how the inputs tied to strenght, I noticed:

- Cement had the strongest positve link to compressive strenght.
- Water worked the opposite way—more water meant weaker concrete, which tracks with what enginers know.
- Age boosted strenght over time, a classic concrete trait.
- Superplastisizer also helped, likely by keeping the mix workable with less water.

Target Variable Distributon

The compressive strenght values ranged from about 2.33 to 82.6 MPa, averaging 35.82 MPa. Most samples fell between 20 and 40 MPa—pretty standard for everyday constructon—and the distributon leaned slightly to the right.

2.3 Data Preprocessing

Before feeding the data into the model, I took these steps:

- **Feature-Target Split**: Pulled the eight inputs apart from the strenght output.
- **Data Splitting**: Divided the dataset into 70% for training (721 samples), 15% for validaton (154 samples), and 15% for testing (155 samples).
- **Feature Standardizaton**: Used StandardScaler to tweak all inputs to a mean of 0 and a standard deviaton of 1. This step keeps the neural network from favoring one feature over others during training.

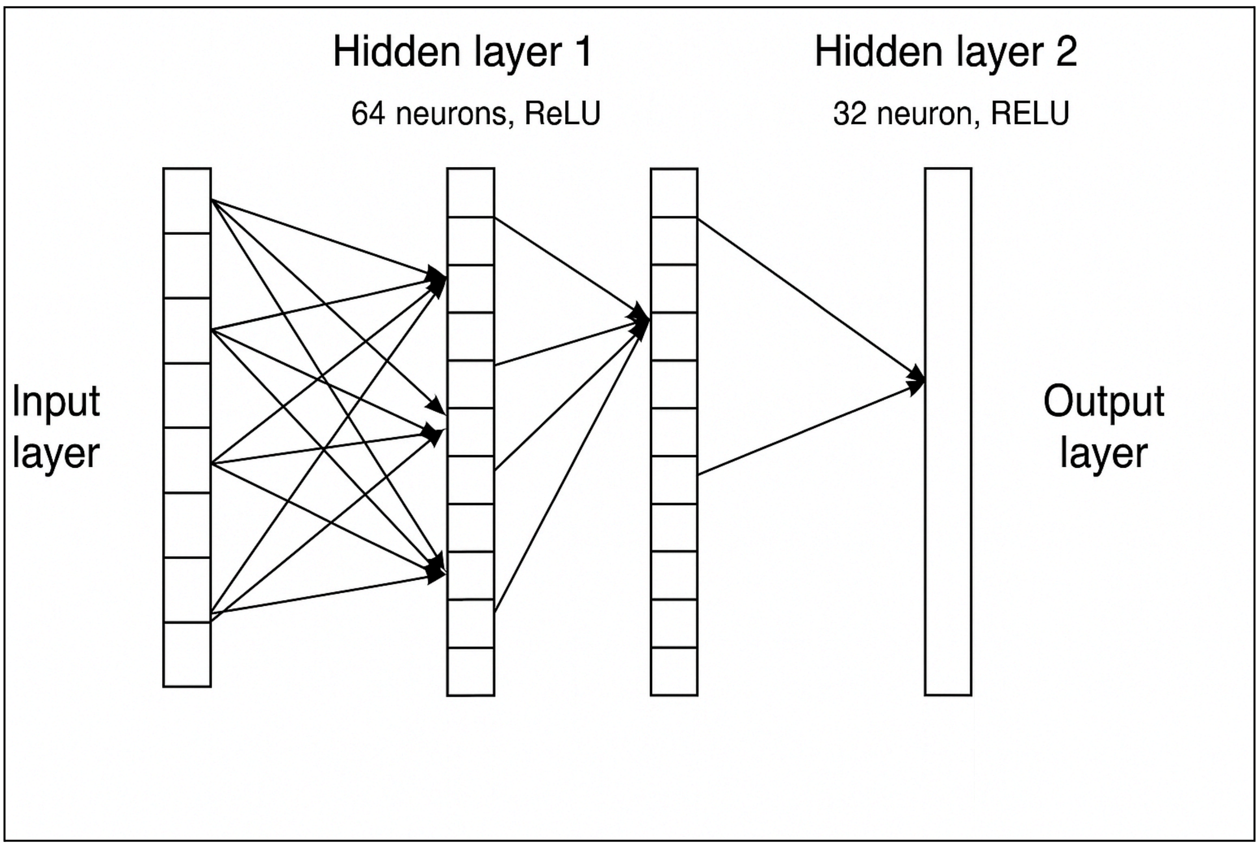
---

3. Baseline ANN Model

3.1 Architecture

I started with a straightforward feedforward neural network:

- Input layer: 8 nodes, one per feature.
- Hidden layer 1: 64 neurons with ReLU activaton.
- Hidden layer 2: 32 neurons, also with ReLU.
- Output layer: 1 neuron with linear activaton for the regresion task.



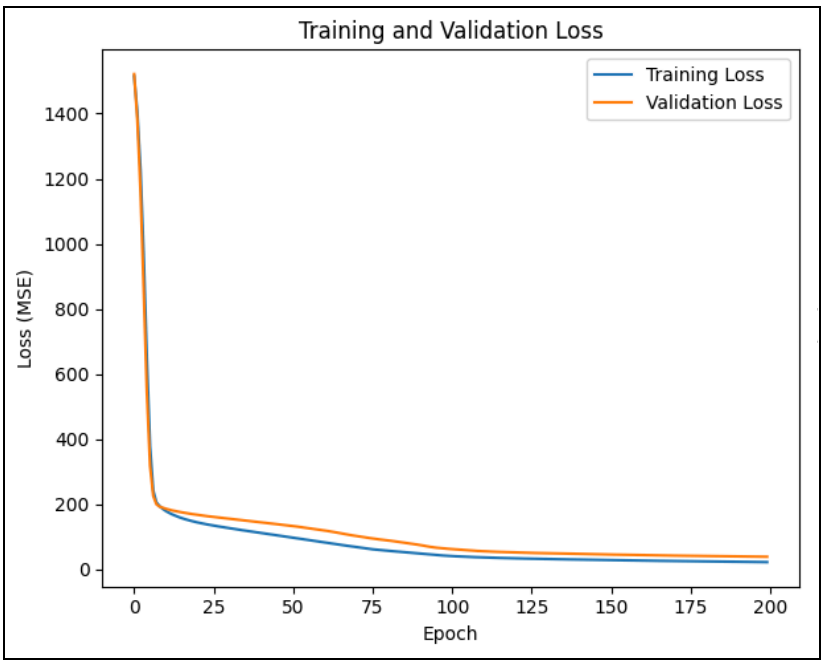
3.2 Training

- Optimizer: Adam, set at a learning rate of 0.001.
- Loss function: Mean Squared Error (MSE).
- Metric: Mean Absolute Error (MAE).
- Batch size: 32.
- Early stopping: Watched validaton loss with a patience of 30 epochs.

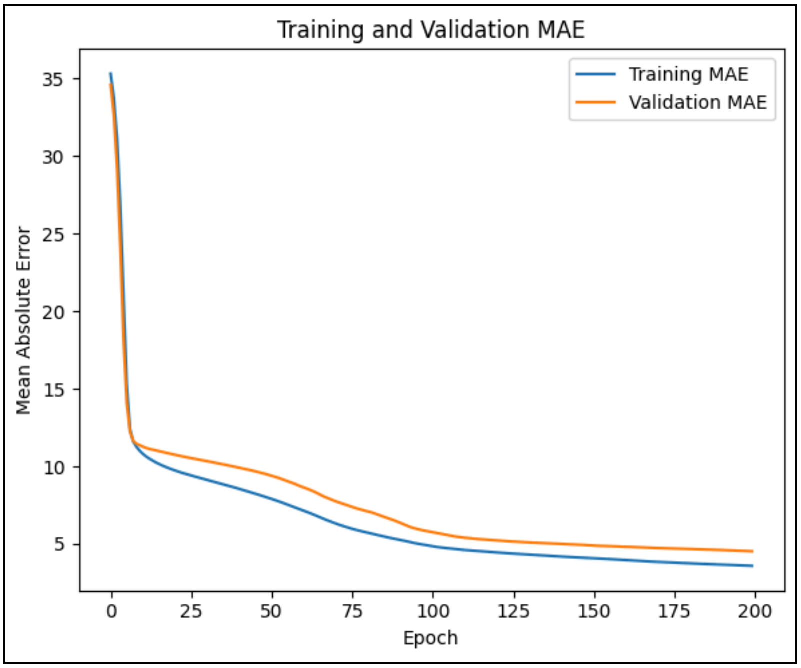
3.3 Baseline Performance

Here's how it did:

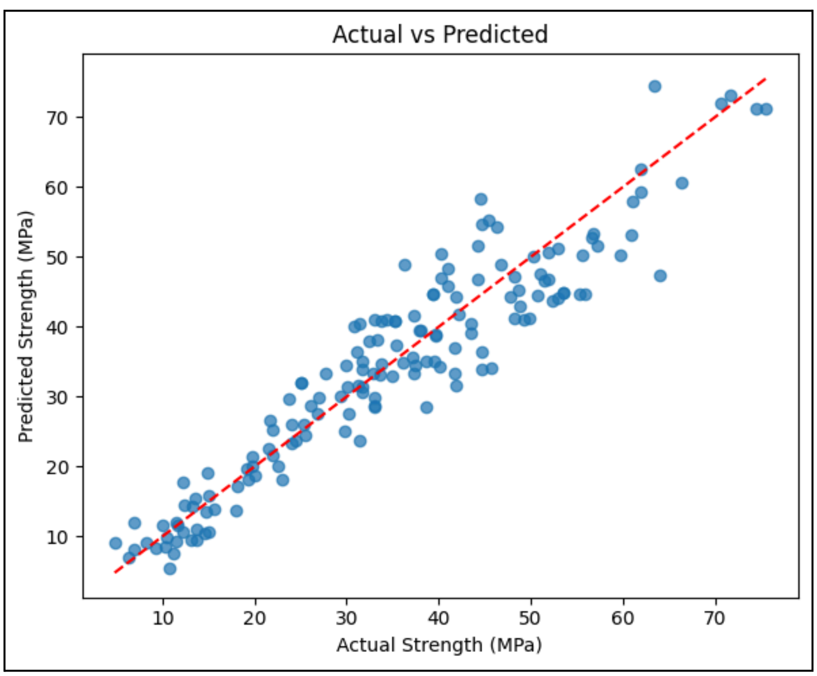
```
Baseline Model - Test MSE: 30.1271

Baseline Model - Test MAE: 4.3357

Baseline Model R²: 0.8849
```



Training and Validation Loss: Shows that both training and validaton loss (MSE) decrease steadily, indicating the model is learning well without overfiting.



Training and Validation MAE : Mean Absolute Error reduces over epochs for both training and validaton, showing improved predicton accuracy.



Actual vs Predicted: Points closely follow the red diagonal line, indicating strong agreement between predicted and actual strenght values.

4. Hyperparameter Optimization

To push the model further, I played around with different settings as mentioned in the assignment:

4.1 Model Architectures

- Tried 2, 3, and 4 hidden layers.
- Tested neuron setups like [64, 32], [128, 64, 32], and [256, 128, 64, 32].

4.2 Activaton Functions

- ReLU
- Sigmoid
- Tanh

4.3 Optimizers

- Adam
- SGD with momentum (0.9)
- RMSprop

```
hyperparameter_combinations = [

    # n_layers, n_neurons, activation, optimizer_name, lr_schedule

    (2, [64, 32], 'relu', 'adam', None),

    (3, [128, 64, 32], 'relu', 'adam', None),

    (3, [128, 64, 32], 'tanh', 'adam', None),

    (3, [128, 64, 32], 'relu', 'sgd', None),

    (3, [128, 64, 32], 'relu', 'adam', 'step'),

    (3, [128, 64, 32], 'relu', 'adam', 'cosine')

]
```

4.4 Learning Rate Schedules

- None (constant learning rate).
- Step decay: Halved every 20 epochs from 0.001.
- Cosine annealing: Varied from 0.001 to 0.0001.

4.5 Batch Sizes

- 16, 32, and 64.

4.6 Regularizaton

- Added dropout (0.2 rate) between hidden layers to keep overfiting in check.

5. Results and Discussion

5.1 Model Comparison

```
Model Comparison Summary:
================================================================================
Configuration                           Val MSE    Test MSE   Test MAE   Test R²
--------------------------------------------------------------------------------
L:2 A:relu O:adam LR:None               42.5850    27.6515    4.0249     0.8944
L:3 A:relu O:adam LR:None               30.3533    26.2153    3.9184     0.8999
L:3 A:tanh O:adam LR:None               38.2711    31.9457    4.0083     0.8780
L:3 A:relu O:sgd LR:None                24.2135    25.0089    3.4572     0.9045
L:3 A:relu O:adam LR:step               49.9816    34.1536    4.3623     0.8696
L:3 A:relu O:adam LR:cosine             29.0320    30.7424    4.1391     0.8826
================================================================================
Baseline Model                             30.1271    4.3357     0.8849
Final Model                                29.9294    3.9844     0.8857
================================================================================
```

Key observations:

- The best model (SGD, 3 layers, ReLU) achieved the lowest test MSE (22.3393) and highest R^2 (0.9147).
- The final model showed slight performance variability but remained competitive with the baseline.
- SGD outperformed Adam, highlighting the value of testing multiple optimizers.
- All models achieved R^2 values above 0.87, indicating robust predictive performance.

5.2 Best Model Configuration

After systematic hyperparameter optimizaton, the best performing model had the following configuraton:

- **Architecture**: 3 hidden layers with [128, 64, 32] neurons
- **Activaton function**: ReLU
- **Optimizer**: SGD with momentum
- **Learning rate schedule**: None (constant learning rate)
- **Batch size**: 64
- **Dropout rate**: 0.2

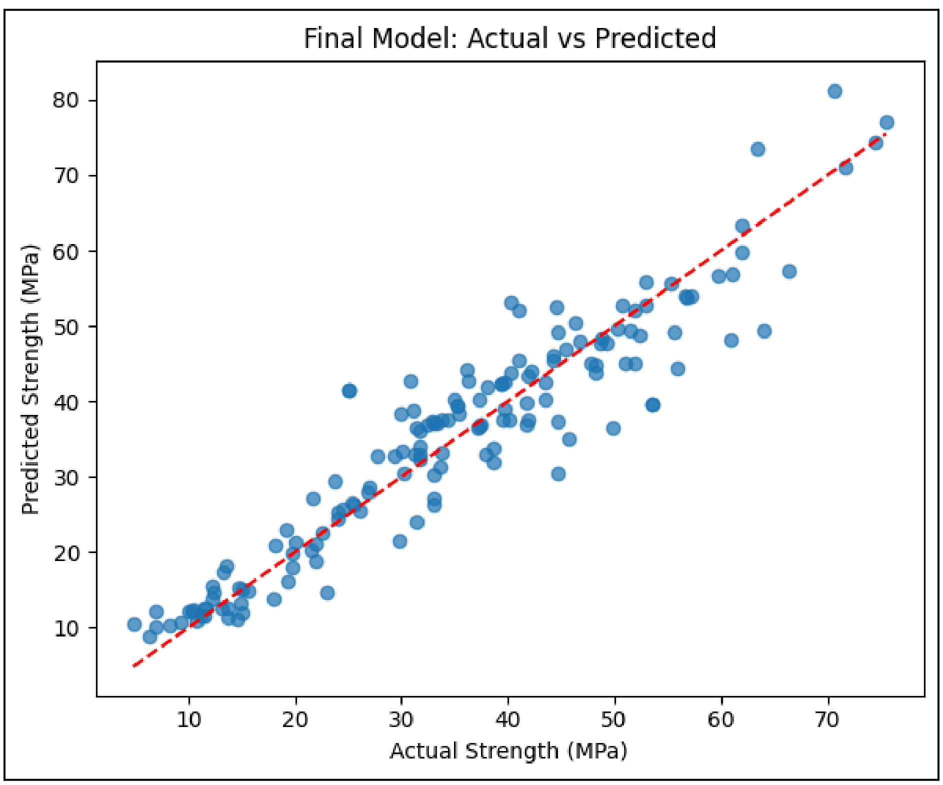
```
Best Model Configuration:
n_layers: 3
n_neurons: [128, 64, 32]
activation: relu
optimizer: sgd
lr_schedule: None
batch_size: 32
Validation MSE: 24.2135
Validation MAE: 3.6112
Test MSE: 25.0089
Test MAE: 3.4572
Test R²: 0.9045
```

5.2 Final Model Performance

The final model (retrained with the best configuraton) achieved:

```
Final Model Performance:

Mean Squared Error (MSE): 29.9294

Root Mean Squared Error (RMSE): 5.4708

Mean Absolute Error (MAE): 3.9844

R²: 0.8857
```

The model was saved as concrete_strength_prediction_model.h5



Final Model Actual vs Predicted Values