# Artificial Neural Network (ANN) and Particle Swarm Optimization (PSO) Integration Report

## 1. Design and Implementation of the Artificial Neural Network (ANN)

The ANN implemented in this project is structured to handle regression tasks to find concrete strength, utilizing a customized architecture designed through various classes and methods to facilitate both flexible configuration and efficient learning.

### ANN Design

Architecture  
The Artificial Neural Network (ANN) in this project is designed to be flexible, allowing for customization of its structure through user-defined input on the number of layers, nodes per layer, and activation functions. This modular approach is critical for testing various network configurations to achieve optimal performance. Each layer of the network, from the input through the hidden layers to the output, plays a specific role in determining the accuracy and generalization of the model.

For the hidden layers, I applied activation functions like ReLU (Rectified Linear Unit) and tanh, which help introduce non-linearity, allowing the network to learn complex patterns within the data. The output layer, however, employs a linear activation function instead of the sigmoid function typically used in classification tasks. This choice is better suited to regression problems, as it allows the model to predict a continuous output rather than a probability, aligning with the project's objective to predict numerical values accurately.

### Key Classes and Methods

The ANNBuilder class serves as a neural network constructor, allowing the user to define the architecture of the network, including the number of layers, nodes per layer, and activation functions. The build method in this class initializes the network’s structure and assigns weights and biases to each layer, providing flexibility for the user to experiment with different network configurations.

The ActivationFunctions class defines various activation functions, such as ReLU, sigmoid, tanh, and linear. These functions are crucial for introducing non-linearity into the network, enabling it to learn complex patterns. The ANNBuilder class uses this class to assign activation functions to different layers of the network based on the user’s preferences, allowing for customization depending on the task.

The forward method performs the forward pass through the network by propagating input data from the input layer through the hidden layers, applying activation functions at each step. This method enables the network to make predictions by using the current weights and biases. The optimization process, such as using PSO, refines the model by adjusting these parameters to minimize the error and improve prediction accuracy..

## 2. Design and Implementation of the Particle Swarm Optimization (PSO)

PSO was implemented to optimize the weights and biases of the ANN, targeting minimized error rates in prediction. This design leverages multiple parameters and swarm behaviors that evolve over iterations to achieve the best possible results.

### PSO Design

### PSO, or Particle Swarm Optimization, is a heuristic optimization algorithm inspired by the social behaviors observed in nature, such as the movement of birds in a flock or fish in a school. Each particle in the swarm represents a potential solution in the search space, and it adjusts its position based on its own best experience and the experiences of neighboring particles. This cooperative search for the best solution allows PSO to effectively explore complex problem spaces.

### In the context of optimizing an Artificial Neural Network (ANN), PSO operates by treating each particle as a candidate set of weights and biases for the network. During each iteration, the particles update their positions based on two main factors: their own best-known position (best solution found so far) and the best-known positions of neighboring particles (collaborative search). The swarm collectively converges to a global optimal or near-optimal solution, which, in the case of an ANN, improves the model's performance by minimizing the error (e.g., Mean Absolute Error or Mean Squared Error). Through multiple iterations, the PSO algorithm gradually refines the weights and biases to find the optimal configuration for the network.

### PSO has proven effective in tasks like optimizing the weights of neural networks, as it avoids the local minima problems that are common in traditional gradient-based optimization methods.

### Key PSO Parameters and Configurations

## - Swarm size: This defines how many particles (candidate solutions) are present in the swarm. A larger swarm increases the diversity of the search, improving the chances of finding the global optimum. However, it can slow down convergence due to the increased number of particles needing to be processed. Smaller swarms are faster but may not explore the search space thoroughly, potentially getting stuck in local optima. In practical applications, choosing the right swarm size is a balance between exploration and computation efficiency. We tested with swarm size of 50, 100 and 200.

## - Max iterations: The maximum number of iterations determines how long the swarm will continue to search for an optimal solution. If the number of iterations is too small, the algorithm may not have enough time to converge, while too many iterations may lead to excessive computation without significant improvements. The choice of iterations depends on the problem's complexity and the convergence speed of the swarm. We tested with max iterations of 50, 100 and 200.

## - Alpha, Beta, Gamma, Delta, and Epsilon: These parameters control the movement of particles in the search space. They influence how much the particles are attracted to their own best position (alpha), the best position of neighboring particles (beta), and how much randomness is introduced to avoid premature convergence (gamma, delta, epsilon). Tuning these hyperparameters helps in striking a balance between exploration (randomness) and exploitation (local search), which is crucial for the algorithm's efficiency. For example, adjusting alpha affects the degree of personal attraction, while beta controls the influence of neighbors.

## 3. Integrating PSO with ANN

Connecting the PSO and ANN components was accomplished by using PSO to search for optimal ANN weights and biases. This integration occurs as follows:  
  
1. Particle Representation: In PSO, each particle represents a potential solution in the search space, which in this case corresponds to a set of weights and biases for the ANN. Each particle's position is a vector that holds these weights and biases, and the particle’s velocity controls how its position (solution) changes across iterations.

2. Fitness Evaluation: The fitness of each particle is evaluated based on its ability to minimize the error in the ANN. For each iteration, the weights and biases assigned by the particle are used to perform a forward pass on the training data, and the resulting predictions are compared to the true values. The Mean Absolute Error (MAE) is calculated to quantify the fitness, with lower MAE values indicating better performance. This process helps to identify which particles are contributing better solutions.

3. Optimization Loop: Over multiple iterations, the PSO algorithm iteratively updates the positions (weights and biases) of particles based on their previous best positions and the best positions found by their neighbors. The aim is to minimize the MAE by adjusting the weights and biases, effectively optimizing the ANN. The particle that yields the lowest MAE represents the best solution, and its weights and biases are used to initialize the final trained ANN model. The performance of the model is then evaluated on the test data to assess its generalization capability.

## 4. Experimental Setup and Results

### Setup

- Data: The concrete dataset was used, with inputs normalized between 0 and 1 using MinMaxScaler. A linear activation function was used in the output layer due to the regression nature of the problem.  
- Splitting: Data was split into a training set (70%) and a test set (30%) to evaluate generalization performance.  
- Evaluation Metric: Mean Absolute Error (MAE) was chosen as the evaluation metric for both training and test data, with lower MAE indicating better model performance.  
- Parameter Selection: Various PSO configurations were tested by adjusting key parameters like swarm size, max iterations, and the hyperparameters (alpha, beta, gamma, delta, epsilon). Each combination of these settings impacted the convergence speed and final optimization results. After extensive testing, the configuration that yielded the lowest Mean Absolute Error (MAE) was identified, indicating the optimal balance between exploration and exploitation within the search space. By fine-tuning these parameters, the PSO algorithm effectively minimized the error in the ANN’s predictions, achieving the best performance on both training and test datasets. This iterative process helps in refining the model's weights and biases, leading to improved generalization to unseen data.Results

The table below summarizes the best ANN and PSO configurations based on MAE:

|  |  |  |  |
| --- | --- | --- | --- |
| **ANN Topology** | **PSO Parameters (Swarm Size, Iterations)** | **MAE Train** | **MAE Test** |
| [8, 10, 5, 1] | [200, 500, 0.5, 1.5, 1, 1, 1] | 4.69 | 5.12 |
| [8, 10, 8, 1] | [200, 500, 0.5, 1.5, 1, 1, 1] | 4.97 | 5.5 |
| [8, 10, 5, 1] | [50, 50, 1, 1.5, 1.5, 1, 1.2] | 11.87 | 11.63 |
| [8, 15, 12, 9, 5 ,1] | [200, 500, 0.5, 1, 1, 1, 1.2] | 5.09 | 5.31 |
| [8, 10, 5, 1] | [50, 50, 0.5, 0.5, 0.5, 0.5, 0,8] | 25.2 | 24.1 |

The above graph shows MAE over PSO iterations for the best configuration, indicating that PSO successfully minimizes MAE as iterations progress.

## 5. Reflection on the Impact of Parameters on Performance

Variations in ANN topology and PSO parameters have a significant impact on the Mean Absolute Error (MAE) for both training and testing sets.  
  
1. ANN Topology:

- Increasing the complexity of the ANN topology, such as adding more layers or nodes, can lower the MAE to a certain extent. For example, the topology [8, 10, 5, 1] achieved a test MAE of 5.12, whereas a slightly more complex topology, [8, 10, 8, 1], resulted in a higher MAE of 5.5. However, adding excessive complexity can have diminishing returns. The topology [8, 15, 12, 9, 5, 1] had a test MAE of 5.31, showing that too many layers or nodes can overcomplicate the model and possibly lead to overfitting, thereby degrading performance.

2. PSO Parameters:

- Swarm Size and Iterations: A larger swarm size and more iterations tend to improve optimization. For example, a swarm size of 200 and 500 iterations resulted in a lower test MAE of 5.12, compared to a smaller configuration of 50 swarm size and 50 iterations, which produced a higher MAE of 11.63. This suggests that larger swarms and more iterations provide a better search space exploration, increasing the likelihood of finding an optimal solution.

- Cognitive and Social Components: The cognitive and social components of the PSO algorithm (which govern how particles adjust their positions) must be balanced carefully. When these values were set too low, such as [50, 50, 0.5, 0.5, 0.5, 0.5, 0.8], the optimization process was poor, resulting in a high MAE of 24.1. This indicates that an appropriate balance between individual and collective learning is critical for effective optimization.  
In summary, moderate ANN complexity paired with well-tuned PSO parameters—particularly higher swarm size, iterations, and balanced component values—provides the best performance by lowering MAE effectively.