**1. Introduction**

**In complex industrial processes like spray drying of coconut milk, accurate prediction and control of parameters are essential for product quality.  
Traditional optimization methods often struggle with non-linear, high-dimensional data.  
To address this, hybrid models combining Genetic Algorithms (GA) and Neural Networks (NN) are used.**

* **Neural Networks (NNs) are powerful function approximators capable of modeling complex relationships.**
* **Genetic Algorithms (GAs) are global optimization techniques inspired by the process of natural evolution.**

**By using GAs to optimize the weights of an NN, we can achieve better performance, faster convergence, and avoid getting trapped in local minima.**

**2. Genetic Algorithm (GA) Overview**

**A Genetic Algorithm is a stochastic search and optimization technique based on the principles of natural selection and genetics.  
It operates on a population of possible solutions, improving them over generations.**

**Genetic Algorithm (GA) is a bio-inspired optimization technique based on the principles of natural selection and genetics.**

**It mimics evolution — the survival of the fittest — to find the best solution to a given problem.**

**Main Steps in Genetic Algorithm**

1. **Initialization:**
   * **Randomly create an initial population (set of solutions).**
2. **Fitness Evaluation:**
   * **Evaluate how "good" each solution is (fitness).**
3. **Selection:**
   * **Choose the best solutions for reproduction.**
4. **Crossover (Recombination):**
   * **Combine two parents to create offspring (new solutions).**
5. **Mutation:**
   * **Introduce small random changes to offspring to maintain diversity.**
6. **Elitism:**
   * **Retain a few best solutions without any changes.**
7. **Termination:**
   * **Stop after a fixed number of generations or if a good enough solution is found.**

**Why Genetic Algorithm?**

* **Works without gradient information (good for complex, non-differentiable problems).**
* **Good for global optimization (not easily stuck in local optima).**
* **Flexible to handle discrete, continuous, or even mixed variable problems.**
* **Suitable for multi-objective optimization.**

**Types of Problems Solved by GA**

* **Function optimization.**
* **Scheduling (e.g., job-shop scheduling).**
* **Machine learning (e.g., neural network training).**
* **Engineering design (e.g., bridge design optimization).**
* **Robotics (e.g., path planning).**
* **Bioinformatics (e.g., gene selection).**

**3. Neural Network (NN) Overview**

**Artificial Neural Networks are computational models inspired by the human brain.  
They consist of layers of neurons connected by weights.**

* **The weights determine how input signals are transformed into outputs.**
* **Training a neural network involves adjusting the weights to minimize prediction error.**

**However, training with methods like gradient descent can be slow or stuck in local optima, especially in complex datasets.**

**Thus, Genetic Algorithms are used to find better initial weights or optimize weights directly.**

**4. Hybrid GA–NN Modeling**

**In the Hybrid GA–NN model:**

* **A Genetic Algorithm is first used to optimize the weights of the Neural Network.**
* **The optimized weights are then used to improve the prediction capability of the Neural Network.**
* **This hybridization enhances convergence speed, prediction accuracy, and the generalization ability of the network.**

**This approach is particularly useful in spray drying processes, where multiple non-linear variables affect the outcome (e.g., inlet temperature, feed flow rate, atomizer speed).**

**5. Application to Spray Drying of Coconut Milk**

**Spray drying is a method of producing a dry powder from a liquid by rapidly drying with a hot gas.  
In coconut milk spray drying, controlling parameters like temperature, airflow, and concentration is crucial.**

**The Hybrid GA–NN model is applied as follows:**

1. **Collect process data (input: spray drying parameters; output: product quality).**
2. **Set up a Neural Network structure to model the process.**
3. **Use a Genetic Algorithm to optimize the weights of the Neural Network.**
4. **Train and validate the NN with optimized weights.**
5. **Predict and optimize spray drying outcomes like moisture content, powder yield, and product quality.**

**By using GA-optimized networks, better and more reliable predictions of spray drying behavior are achieved.**

**6. Working of the Genetic Algorithm**

**The Genetic Algorithm proceeds through the following steps:**

**a) Initialization**

* **Create an initial population of random candidate solutions.**
* **Each candidate represents a set of neural network weights.**

**b) Fitness Evaluation**

* **Evaluate each candidate's fitness using a fitness function.**
* **In this case, the fitness reflects how accurately the neural network predicts the spray drying results.**

**c) Selection**

* **Select the best individuals based on fitness.**
* **Better candidates have a higher chance to reproduce.**

**d) Crossover**

* **Combine two parent solutions to produce new offspring.**
* **This allows the mixing of good traits from parents.**

**e) Mutation**

* **Introduce small random changes to offspring.**
* **Helps maintain genetic diversity and avoid premature convergence.**

**f) Elitism**

* **Preserve the best solutions directly into the next generation.**
* **Ensures that the quality of solutions never degrades.**

**g) Iteration**

* **Repeat fitness evaluation, selection, crossover, mutation, and elitism for several generations.**

**The process continues until a satisfactory solution (optimized weights) is found.**

**7. Fitness Function**

**The fitness function measures how good a candidate solution (set of weights) is.**

**In this case:**

* **Fitness is higher if the neural network prediction error is low.**
* **Simulated fitness function encourages weights to be close to ideal values (e.g., near 0.5).**

**Thus, candidates that enable the neural network to predict better have higher fitness and are more likely to survive and reproduce.**

**8. Advantages of Hybrid GA–NN Approach**

* **Global Search: GA avoids local minima by searching globally.**
* **Faster Convergence: Good weight initialization speeds up neural network training.**
* **Higher Accuracy: GA-optimized weights result in better generalization and predictive performance.**
* **Robustness: Works well even with complex, non-linear, or noisy data.**

**🧠 Conclusion**

**In this practical, the Genetic Algorithm is successfully used to optimize the parameters (weights) of a Neural Network model applied to spray drying of coconut milk.  
This hybrid approach overcomes the limitations of traditional neural network training, providing a powerful tool for modeling and optimizing industrial processes.**

**🎯 Short Summary Points**

* **GA is inspired by natural evolution.**
* **NN models complex relationships between inputs and outputs.**
* **GA optimizes NN weights to improve performance.**
* **Hybrid GA–NN models are applied to coconut milk spray drying.**
* **Fitness evaluates prediction accuracy.**
* **Evolutionary operators: Selection, Crossover, Mutation.**
* **Results: Better prediction, faster convergence, global optimization.**

**Algorithm**

**Optimization of Neural Network Weights Using Genetic Algorithm for Spray Drying of Coconut Milk**

**Step 1: Import Necessary Libraries**

* **Import libraries for numerical operations, plotting, and random number generation (like numpy, matplotlib, random).**

**Step 2: Define Parameters for Genetic Algorithm**

* **Set the values for:**
  + **Population size: Number of candidate solutions.**
  + **Number of generations: How many iterations the GA should run.**
  + **Crossover rate: Probability of performing crossover between two individuals.**
  + **Mutation rate: Probability of mutation in an individual.**
  + **Elite size: Number of best individuals to directly pass to next generation.**

**Step 3: Define Neural Network Structure**

* **Decide the architecture of the Neural Network:**
  + **Number of input nodes.**
  + **Number of hidden layer nodes.**
  + **Number of output nodes.**

**Step 4: Initialize the Population**

* **Randomly generate a population.**
* **Each individual (chromosome) is a vector of real numbers representing all weights of the neural network.**

**Step 5: Define the Fitness Function**

* **Evaluate each individual’s fitness.**
* **Fitness is inversely proportional to the error (lower error → higher fitness).**
* **Calculate the output of the neural network for each set of weights.**
* **Compute how close the output is to the expected result.**

**Step 6: Genetic Operators**

* **Selection: Select the fittest individuals.**
* **Crossover: Perform crossover between selected individuals to create new offspring.**
* **Mutation: Randomly mutate offspring to maintain genetic diversity.**
* **Elitism: Keep the best individuals unchanged.**

**Step 7: Iterate Over Generations**

* **Repeat fitness evaluation, selection, crossover, mutation, and elitism for a defined number of generations.**
* **Monitor and store the best fitness value over generations.**

**Step 8: Output the Best Solution**

* **After all generations, select the best set of weights.**
* **Plot the fitness over generations for visualization.**

**Real-Life Application of This Practical**

**Now specifically connecting your practical:  
(Optimizing Neural Network Weights using GA for Spray Drying of Coconut Milk)**

**Practical Context:**

* **Spray Drying Process:**
  + **It's a method to produce dry powder from a liquid (like coconut milk) by rapidly drying with a hot gas.**
  + **Important parameters: Inlet temperature, feed flow rate, atomizer speed.**
  + **Desired output: Optimum moisture content, powder quality, energy efficiency.**

**Why Use Neural Network Here?**

* **Spray drying is a complex, nonlinear process.**
* **Traditional mathematical modeling is very hard.**
* **Neural networks can learn the behavior from experimental data (input → output relation).**

**Why Optimize Neural Network with Genetic Algorithm?**

* **Neural networks typically need good weight initialization to perform well.**
* **Traditional training (like gradient descent) can:**
  + **Get stuck in local minima.**
  + **Be sensitive to initial weights.**
* **Genetic Algorithm provides:**
  + **Better weight initialization.**
  + **Global search for best performance.**
  + **More accurate and reliable prediction of moisture content or quality based on process parameters.**

**🌍 Other Real-Life Applications Similar to Your Practical**

| **Field** | **Application** |
| --- | --- |
| **Food Industry** | **Optimize drying conditions for milk powder, coffee, and egg powder** |
| **Pharmaceutical Industry** | **Optimize spray drying for drugs (to maintain bioavailability)** |
| **Chemical Engineering** | **Model and optimize chemical reactors using GA-optimized neural networks** |
| **Agriculture** | **Optimize drying of fruits and vegetables** |
| **Material Science** | **Predict and optimize material properties after processing** |
| **Environmental Science** | **Model complex pollution dispersion processes** |

**✨ Short Summary:**

* **Genetic Algorithm is inspired by natural evolution to solve complex optimization problems.**
* **In your project, it optimizes the neural network so that it better predicts or controls the spray drying process of coconut milk.**
* **It has huge real-world importance in food, pharma, chemical, agricultural, and material processing industries.**

**1. Import Libraries**

python

CopyEdit

import numpy as np

import matplotlib.pyplot as plt

import random

* **numpy**: Used for numerical operations like arrays, dot products.
* **matplotlib.pyplot**: Used to plot graphs (fitness vs generations).
* **random**: Used for random number generation (mutation, crossover points).

**2. Genetic Algorithm Parameters**

python

CopyEdit

population\_size = 100

num\_generations = 50

crossover\_rate = 0.8

mutation\_rate = 0.1

elite\_size = 5

* **population\_size**: 100 candidate solutions (chromosomes).
* **num\_generations**: Run the GA for 50 cycles (iterations).
* **crossover\_rate**: 80% chance that crossover occurs between selected parents.
* **mutation\_rate**: 10% chance of mutation in offspring.
* **elite\_size**: Top 5 individuals automatically carried to next generation.

**3. Neural Network Architecture**

python

CopyEdit

input\_size = 3

hidden\_size = 5

output\_size = 1

* **input\_size**: 3 inputs to the neural network (e.g., inlet temp, feed flow, atomizer speed).
* **hidden\_size**: 5 neurons in the hidden layer.
* **output\_size**: 1 output (e.g., moisture content).

**4. Population Initialization**

python

CopyEdit

num\_weights = (input\_size \* hidden\_size) + (hidden\_size \* output\_size)

population = [np.random.rand(num\_weights) for \_ in range(population\_size)]

* **num\_weights**: Total number of weights in the network (input→hidden + hidden→output).
* Each individual is a **random vector** of size num\_weights between 0 and 1.

**5. Fitness Function**

python

CopyEdit

def fitness\_function(weights):

target\_value = 0.5

fitness = 1 / (np.mean(np.abs(weights - target\_value)) + 0.0001)

return fitness

* **weights**: Input set of weights for an individual.
* **target\_value**: Ideal weight (assumed 0.5 for demonstration).
* **fitness**: Higher if weights are closer to 0.5.
* **+ 0.0001**: Avoids division by zero error.

**6. Selection**

python

CopyEdit

def select\_parents(population, fitness\_scores):

sorted\_indices = np.argsort(fitness\_scores)[::-1]

selected = [population[i] for i in sorted\_indices[:elite\_size]]

while len(selected) < population\_size:

selected.append(random.choices(population, weights=fitness\_scores, k=1)[0])

return selected

* Sort individuals based on fitness (higher is better).
* Directly select **elite\_size** best individuals.
* Fill the rest of the next generation by **probabilistic selection** based on fitness.

**7. Crossover**

python

CopyEdit

def crossover(parent1, parent2):

if random.random() < crossover\_rate:

point = random.randint(1, len(parent1)-1)

child1 = np.concatenate((parent1[:point], parent2[point:]))

child2 = np.concatenate((parent2[:point], parent1[point:]))

return child1, child2

else:

return parent1.copy(), parent2.copy()

* **Single-point crossover**:
  + Choose a random point.
  + Swap genes (weights) after that point between two parents to produce two children.

**8. Mutation**

python

CopyEdit

def mutate(individual):

for i in range(len(individual)):

if random.random() < mutation\_rate:

individual[i] = np.random.rand()

return individual

* Randomly mutate each weight with a **10% chance**.
* Mutation helps maintain diversity in population and avoid local minima.

**9. Main GA Loop**

python

CopyEdit

best\_fitness\_over\_time = []

for generation in range(num\_generations):

fitness\_scores = [fitness\_function(individual) for individual in population]

best\_fitness\_over\_time.append(max(fitness\_scores))

selected\_parents = select\_parents(population, fitness\_scores)

next\_generation = []

for i in range(0, population\_size - elite\_size, 2):

parent1 = selected\_parents[i]

parent2 = selected\_parents[i+1]

child1, child2 = crossover(parent1, parent2)

child1 = mutate(child1)

child2 = mutate(child2)

next\_generation.extend([child1, child2])

population = selected\_parents[:elite\_size] + next\_generation

* For each generation:
  + **Evaluate fitness** of all individuals.
  + **Record the best fitness** for plotting.
  + **Select parents**.
  + **Perform crossover** and **mutation** to produce next generation.
  + **Apply elitism**: Keep best individuals unchanged.

**10. Output the Best Solution**

python

CopyEdit

best\_individual = population[np.argmax([fitness\_function(individual) for individual in population])]

print("Best weights:", best\_individual)

* After all generations, find and print the best individual.

**11. Plot Fitness Progress**

python

CopyEdit

plt.plot(best\_fitness\_over\_time)

plt.xlabel('Generation')

plt.ylabel('Best Fitness')

plt.title('Fitness Over Generations')

plt.show()

* Visualizes how the **best fitness** improved over generations.

**🎯 Final Key Points**

* The Genetic Algorithm optimizes the Neural Network’s weights.
* Fitness rewards individuals whose weights are close to ideal.
* Selection, crossover, mutation, and elitism work together to evolve better solutions.
* The best fitness improves gradually over generations.