Implementing a MADALINE neural network for the AND function using the MRI algorithm

1. Importing necessary libraries

We begin by importing the `numpy` library required for the numerical operations involved in our analysis.

```
import numpy as np
```

2. Defining the MADALINE class

Next, we define a class encapsulating the functionality of a MADALINE neural network for binary classification.

- It initialises with a specified number of input features, randomly assigning weights and bias.
- It includes an activation function that outputs 1 for non-negative inputs and 0 otherwise.
- The `predict` method computes the network's output by calculating the weighted sum of inputs plus the bias, followed by applying the activation function.
- The `train` method employs the Modified Random Initialisation (MRI) algorithm to update the weights and bias based on the error between predicted and actual outputs during multiple training epochs, providing real-time feedback on the training progress.

```
class MADALINE:
  def __init__(self, input_size):
    self.input_size = input_size
    self.weights = np.random.rand(input_size)
    self.bias = np.random.rand(1)

  def activation(self, x):
    return 1 if x >= 0 else 0

  def predict(self, inputs):
    linear_output = np.dot(self.weights, inputs) + self.bias
    return self.activation(linear_output)

  def train(self, training_inputs, labels, epochs=10, learning_rate=0.1):
    for epoch in range(epochs):
      for inputs, label in zip(training_inputs, labels):
        prediction = self.predict(inputs)
        error = label - prediction

        if error != 0:
          self.weights += learning_rate * error * inputs
          self.bias += learning_rate * error
        print(f'Epoch: {epoch+1}, Inputs: {inputs}, Prediction: {prediction}, Error: {error}')
```

3. Defining the training data for the AND function

Now, we generate the training dataset for the AND function with inputs and corresponding labels. The inputs consist of all possible combinations of two binary values (0 and 1). The labels represent the output of the AND logic gate, which is 1 only when both inputs are 1.

```
def generate_training_data():
  training_inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
  labels = np.array([0, 0, 0, 1])
  return training_inputs, labels
```

4. Testing the MADALINE network

Finally, we define the main execution block which:

- calls the `generate_training_data` function to get the training inputs and labels,
- initialises the MADALINE network with 2 input features,
- trains the network for a specified number of epochs and learning rate, and,
- tests the trained network by predicting the output for each input in the training set and printing the results.

```
if __name__ == "__main__":
  training_inputs, labels = generate_training_data()
  madaline = MADALINE(input_size=2)
  madaline.train(training_inputs, labels, epochs=10, learning_rate=0.1)

  print('\nTesting the MADALINE network:')
  for inputs in training_inputs:
    output = madaline.predict(inputs)
    print(f'Input: {inputs}, Predicted output: {output}')
```

```
Epoch: 1, Inputs: [0 0], Prediction: 1, Error: -1
Epoch: 1, Inputs: [0 1], Prediction: 1, Error: -1
Epoch: 1, Inputs: [1 0], Prediction: 1, Error: -1
Epoch: 1, Inputs: [1 1], Prediction: 1, Error: 0
Epoch: 2, Inputs: [0 0], Prediction: 1, Error: -1
Epoch: 2, Inputs: [0 1], Prediction: 0, Error: 0
Epoch: 2, Inputs: [1 0], Prediction: 1, Error: -1
Epoch: 2, Inputs: [1 1], Prediction: 0, Error: 1
Epoch: 3, Inputs: [0 0], Prediction: 0, Error: 0
Epoch: 3, Inputs: [0 1], Prediction: 1, Error: -1
Epoch: 3, Inputs: [1 0], Prediction: 0, Error: 0
Epoch: 3, Inputs: [1 1], Prediction: 0, Error: 1
Epoch: 4, Inputs: [0 0], Prediction: 0, Error: 0
Epoch: 4, Inputs: [0 1], Prediction: 1, Error: -1
Epoch: 4, Inputs: [1 0], Prediction: 1, Error: -1
Epoch: 4, Inputs: [1 1], Prediction: 0, Error: 1
Epoch: 5, Inputs: [0 0], Prediction: 0, Error: 0
Epoch: 5, Inputs: [0 1], Prediction: 0, Error: 0
Epoch: 5, Inputs: [1 0], Prediction: 1, Error: -1
Epoch: 5, Inputs: [1 1], Prediction: 0, Error: 1
Epoch: 6, Inputs: [0 0], Prediction: 0, Error: 0
Epoch: 6, Inputs: [0 1], Prediction: 1, Error: -1
Epoch: 6, Inputs: [1 0], Prediction: 0, Error: 0
Epoch: 6, Inputs: [1 1], Prediction: 1, Error: 0
Epoch: 7, Inputs: [0 0], Prediction: 0, Error: 0
Epoch: 7, Inputs: [0 1], Prediction: 0, Error: 0
Epoch: 7, Inputs: [1 0], Prediction: 0, Error: 0
Epoch: 7, Inputs: [1 1], Prediction: 1, Error: 0
Epoch: 8, Inputs: [0 0], Prediction: 0, Error: 0
Epoch: 8, Inputs: [0 1], Prediction: 0, Error: 0
Epoch: 8, Inputs: [1 0], Prediction: 0, Error: 0
Epoch: 8, Inputs: [1 1], Prediction: 1, Error: 0
Epoch: 9, Inputs: [0 0], Prediction: 0, Error: 0
Epoch: 9, Inputs: [0 1], Prediction: 0, Error: 0
Epoch: 9, Inputs: [1 0], Prediction: 0, Error: 0
Epoch: 9, Inputs: [1 1], Prediction: 1, Error: 0
Epoch: 10, Inputs: [0 0], Prediction: 0, Error: 0
Epoch: 10, Inputs: [0 1], Prediction: 0, Error: 0
Epoch: 10, Inputs: [1 0], Prediction: 0, Error: 0
Epoch: 10, Inputs: [1 1], Prediction: 1, Error: 0

Testing the MADALINE network:
Input: [0 0], Predicted output: 0
Input: [0 1], Predicted output: 0
Input: [1 0], Predicted output: 0
Input: [1 1], Predicted output: 1
```