Assignment 1: Implementing a Deep Neural Network for Regression from Scratch

Objective:
In this assignment, you will build a deep neural network from scratch using NumPy for matrix operations to solve a regression problem. The network will consist of multiple hidden layers with ReLU activation, and the final output layer will use a linear activation function. You will implement key components like forward propagation, activation functions, loss functions (Sum squared error), and backward propagation. Gradient descent will be used as the optimizer.

Problem Statement:

You are tasked with implementing a fully connected deep neural network that can predict continuous values from input data. The deep neural network will consist of an input layer, multiple hidden layers with ReLU activation, and an output layer with a linear activation function. You will also implement the Sum Squared Error (SSE) loss function and use gradient descent for optimization.

Part 1: Understanding the Code Skeleton

Study the following components provided in the skeleton code:

1. **Activation Functions:**
    - relu(): The Rectified Linear Unit (ReLU) function.
    - relu_derivative(): The derivative of the ReLU function.
    - linear_activation(): The linear activation function or any activation function for the output layer (used for regression).
2. **Loss Function:**
    - sum_squared_error_loss(): The Sum Squared Error (SSE) loss function to measure the model's performance.
    - sum_squared_error_derivative(): The derivative of the SSE loss for backpropagation.
3. **Forward Propagation:**
    - forward_propagation(): A function that computes the output of the neural network by passing input data through each layer using matrix multiplication and activation functions.
4. **Backward Propagation:**
    - backward_propagation(): A function that computes the gradients needed to update the weights of the network using backpropagation.

5. **Parameter Initialization:**
    - init_params(): A function that initializes the weights and biases for each layer of the network.

Part 2: Tasks

1. **Implement Forward Propagation and Backward Propagation:**

Complete the missing parts of the forward_propagation() and backward_propagation() functions in the provided code skeleton. Specifically, you need to:

- Perform the matrix multiplications and activation functions for each layer.
- Use a **linear activation function** in the final output layer, suitable for regression.
- Implement the correct formulas for computing the gradients of weights and biases during backward propagation.

2. **Implement the Training Loop:**

Using the provided code structure, implement a function to train the deep neural network using gradient descent. You will need to:

- Calculate the forward pass through the network.
- Compute the loss using the Sum Squared Error (SSE) loss function.
- Perform backward propagation to compute gradients.
- Update the weights using gradient descent.

3. **Experiment with Hyperparameters:**

Modify the following hyperparameters and observe how they affect the performance of the model:

- Number of hidden layers.
- Number of neurons in each layer.
- Learning rate.
- Number of training iterations (epochs).

Write a short report (1-2 paragraphs) discussing how the choice of these hyperparameters influences the model's ability to predict the target values accurately.

4. **Test Your Model:**

Test your model on a regression dataset (examples below). Evaluate your model's performance using Mean Squared Error (MSE) and R-squared metrics. Report your findings.

You may select one of the following datasets for testing your model:

1. Boston Housing Prices

2. California Housing Prices
3. Auto MPG Dataset
4. Concrete Compressive Strength
5. Bike Sharing Demand
6. Air Quality Dataset
7. Generate synthetic data or use the provided real-world datasets to test the model's performance.

Part 3: Deliverables

1. The complete Python code for the deep neural network, including:
    - Implemented forward_propagation() and backward_propagation() functions.
    - The training loop to optimize the network's parameters.
2. A brief report on the impact of different hyperparameters on the model's performance.
3. The final MSE and R-squared metrics on the test dataset.

Question:

In your own words, discuss the advantages and disadvantages of ReLU activation in the context of training deep neural networks for regression tasks.

Grading Criteria:

- Correct implementation of forward propagation (25%)
- Correct implementation of backward propagation (25%)
- Training loop implementation and model convergence (25%)
- Report on hyperparameter tuning (15%)
- Model testing and accuracy (10%)

Deadline: Sunday, November 3rd