Assignment 3: Calculating Backward Pass Values in a Neural Network

Problem Statement 1

You are given a neural network with the following structure:

- Input layer: 3 units
- Hidden layer 1: 2 units, ReLU activation
- Hidden layer 2: 2 units, ReLU activation
- Output layer: 1 unit, no activation

The loss function used is the squared loss:

$$L = \frac{1}{2}(y - \hat{y})^2$$

You are provided with the following forward pass values, weights, and biases:

Forward Pass Values

$$\text{Input} : \mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$\text{Hidden Layer 1 Output} : \mathbf{a}^{(\mathbf{1})} = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

$$\text{Hidden Layer 2 Output} : \mathbf{a}^{(\mathbf{2})} = \begin{bmatrix} 6 \\ 7 \end{bmatrix}$$

$$\text{Output} : \hat{y} = 8$$

Weights and Biases

$$\mathbf{W^{(1)}} = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \end{bmatrix}, \quad \mathbf{b^{(1)}} = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$$
$$\mathbf{W^{(2)}} = \begin{bmatrix} 0.7 & 0.8 \\ 0.9 & 1.0 \end{bmatrix}, \quad \mathbf{b^{(2)}} = \begin{bmatrix} 0.3 \\ 0.4 \end{bmatrix}$$
$$\mathbf{W^{(3)}} = \begin{bmatrix} 1.1 & 1.2 \end{bmatrix}, \quad \mathbf{b^{(3)}} = 0.5$$

The true output value is:

$$y = 10$$

Instructions

Calculate the backward pass values at every point in the network. Specifically, find the gradients of the loss with respect to:

1. The output of the last layer ($\hat{y}$)
2. The input to the last layer (before applying ReLU in the last layer)
3. The output of ReLU in the second hidden layer
4. The input to the second hidden layer (before applying ReLU in the second hidden layer)
5. The output of ReLU in the first hidden layer
6. The input to the first hidden layer (before applying ReLU in the first hidden layer)

Problem Statement 2

Correct the following code snippet for the backward pass of the ReLU activation function:

```
def relu_backward(dA, Z):
    """
    Implementing the backward propagation for a single ReLU unit.

    Arguments:
    dA -- post-activation gradient, of any shape
    Z -- activation input, of the same shape as dA

    Returns:
    dZ -- gradient of the cost with respect to Z
    """

    dZ = np.array(dA, copy=True)  # Copying dA to dZ

    # When Z <= 0, setting dZ to 0
    dZ[Z <= 0] = 0

    return dZ
```

Explain the modification made to ensure proper implementation.