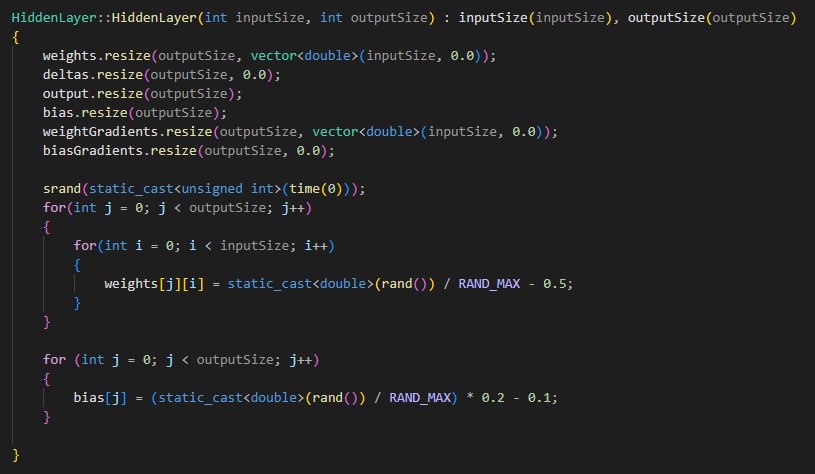
Neural Network Process:

1. Initialize network with random weights and biases

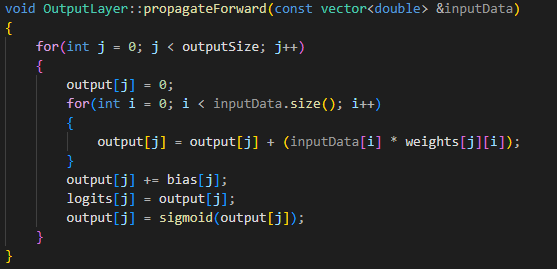
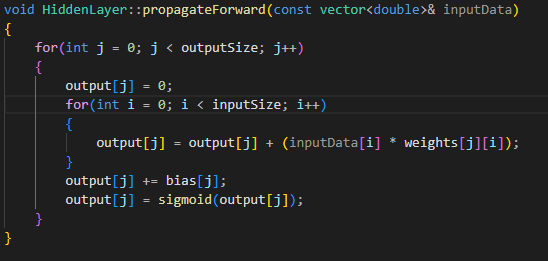


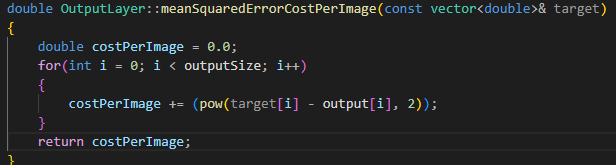


2. For EACH training image :

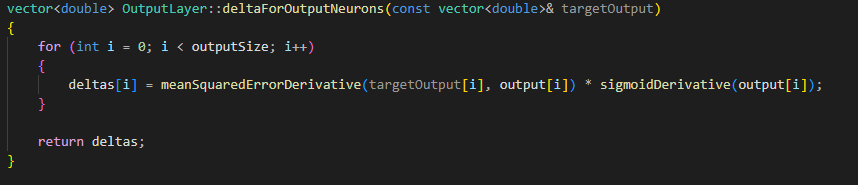
2a. Compute activations for the entire network (Forward Pass).

For EACH neuron’s activated output per layer:

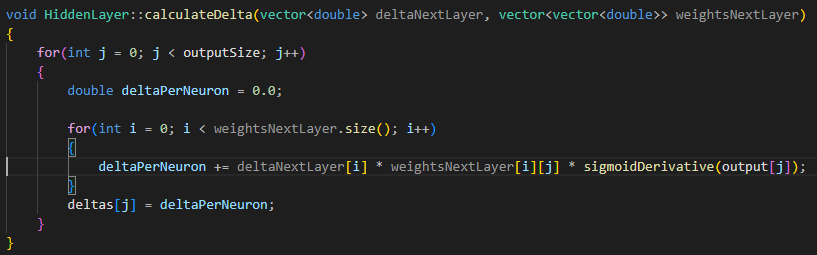


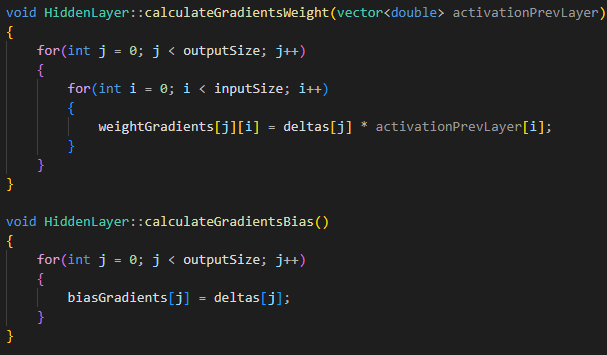
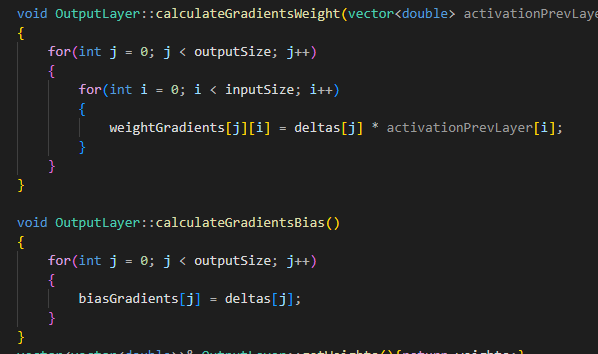
2b. Compute cost to calculate how different the predicted output compared to the true value using Mean Squared Error formula:

2c. Compute delta**(δ)** for neurons in the output layer using the chain rule in calculus.

**δ** represents the sensitivity of each neuron with respect to cost. It can be computed using the values we already have in the output layer from the chain rule.

2d. Using the weights and **δ** computed from the output layer, compute **δ** for ALL neurons in previous layer/s using:

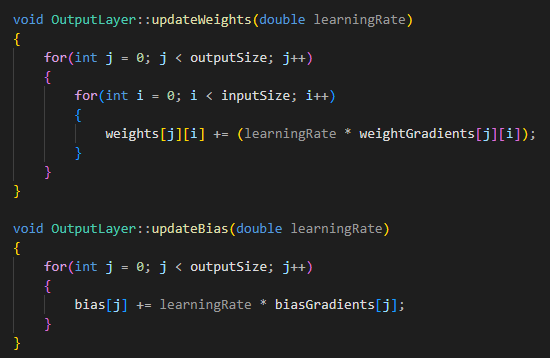


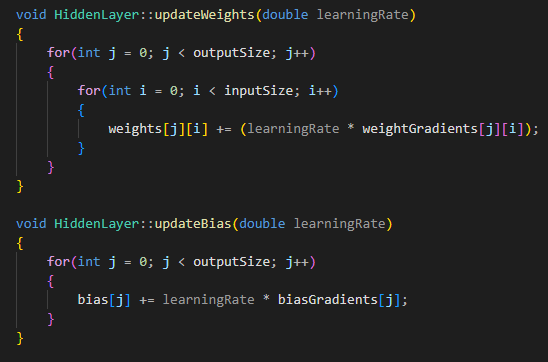
2e. Compute gradient for each weight and biases for the entire network using each layer’s respective **δ**s:

**&**

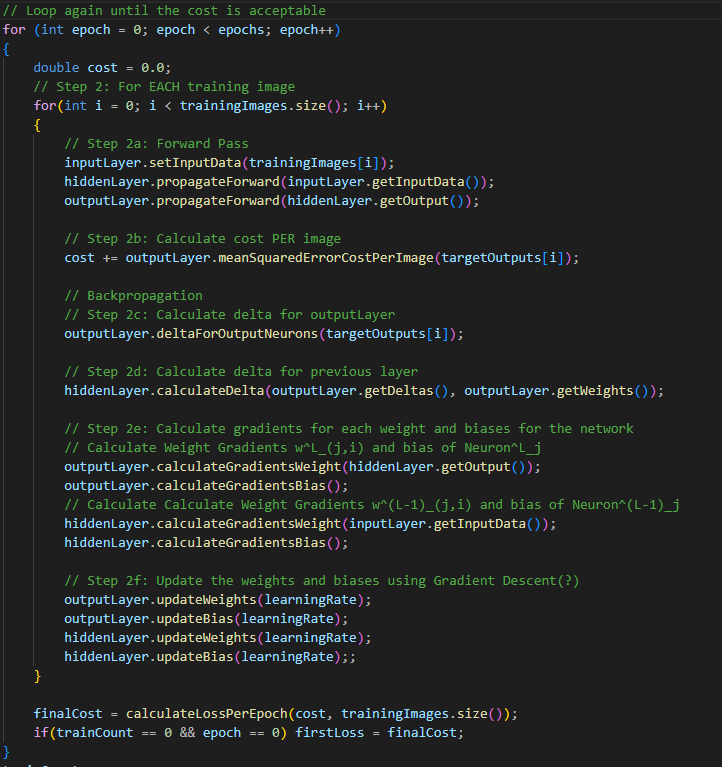
2f. Update weights and biases on the entire network using Gradient Descent(?) using:

**&**





**Repeat step 2 until the cost is acceptable.**

Implementation:

Result 1: No noise, Result 2: Heavy noise (1 random pixel)

