- X: Input data matrix with dimensions $(n_{\text{samples}}, n_{\text{features}})$.
- $W^{(1)}, W^{(2)}, W^{(3)}$: Weight matrices for the first hidden layer, second hidden layer, and output layer, respectively.
- $b^{(1)}, b^{(2)}, b^{(3)}$: Bias vectors for the first hidden layer, second hidden layer, and output layer, respectively.
- σ: Activation function (e.g., sigmoid, ReLU).
- α: Learning rate.

Now, let's define the steps of backpropagation:

1. **Forward Pass**:
   - Calculate the activations of the first hidden layer:
     $Z^{(1)} = XW^{(1)} + b^{(1)}$
     $A^{(1)} = \sigma(Z^{(1)})$
   - Calculate the activations of the second hidden layer:
     $Z^{(2)} = A^{(1)}W^{(2)} + b^{(2)}$
     $A^{(2)} = \sigma(Z^{(2)})$
   - Calculate the output of the network:
     $Z^{(3)} = A^{(2)}W^{(3)} + b^{(3)}$
     $A^{(3)} = \sigma(Z^{(3)})$
2. **Backward Pass**:
   - Compute the error at the output layer:
     $\delta^{(3)} = (A^{(3)} - Y) \cdot \sigma'(Z^{(3)})$
   - Compute the error at the second hidden layer:
     $\delta^{(2)} = \delta^{(3)}W^{(3)T} \cdot \sigma'(Z^{(2)})$
   - Compute the error at the first hidden layer:
     $\delta^{(1)} = \delta^{(2)}W^{(2)T} \cdot \sigma'(Z^{(1)})$
3. **Update Weights and Biases**:
   - Update weights and biases for the output layer:
     $W^{(3)} \leftarrow W^{(3)} - \alpha \cdot A^{(2)T} \cdot \delta^{(3)}$
     $b^{(3)} \leftarrow b^{(3)} - \alpha \cdot \text{mean}(\delta^{(3)}, \text{axis} = 0)$
   - Update weights and biases for the second hidden layer:
     $W^{(2)} \leftarrow W^{(2)} - \alpha \cdot A^{(1)T} \cdot \delta^{(2)}$
     $b^{(2)} \leftarrow b^{(2)} - \alpha \cdot \text{mean}(\delta^{(2)}, \text{axis} = 0)$
   - Update weights and biases for the first hidden layer:
     $W^{(1)} \leftarrow W^{(1)} - \alpha \cdot X^T \cdot \delta^{(1)}$
     $b^{(1)} \leftarrow b^{(1)} - \alpha \cdot \text{mean}(\delta^{(1)}, \text{axis} = 0)$

This is a basic implementation of backpropagation for a neural network with two hidden layers. Remember to initialize the weights and biases randomly and to choose an appropriate activation function and learning rate based on your problem. Also, consider using techniques like regularization and momentum for better training performance and avoiding overfitting.