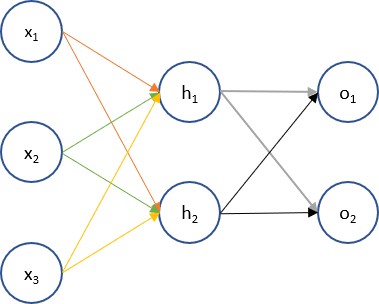
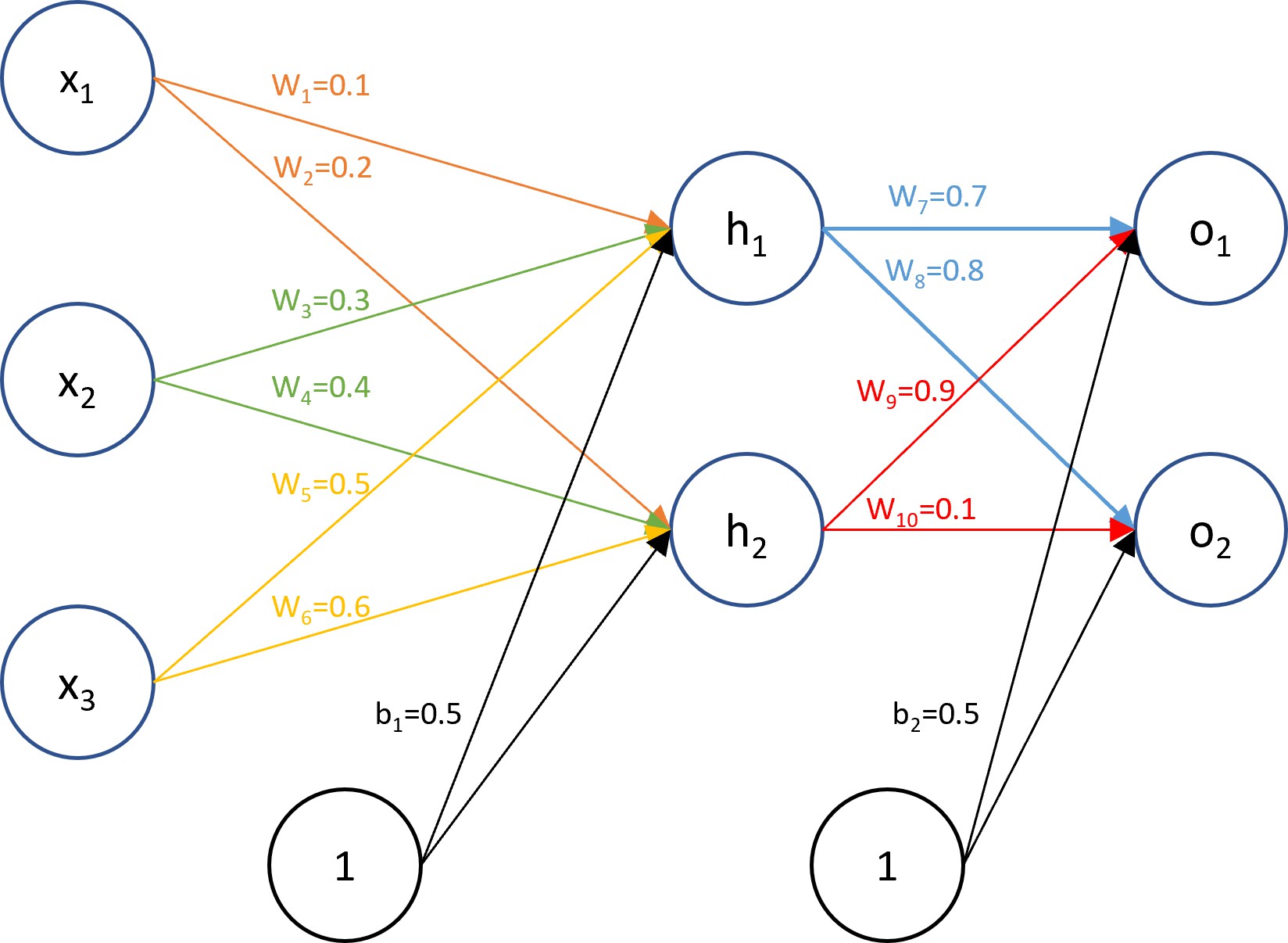
Backpropagation by Hand

******

Assume we have the above network to be trained. The following are the (very) high level steps to train this network.

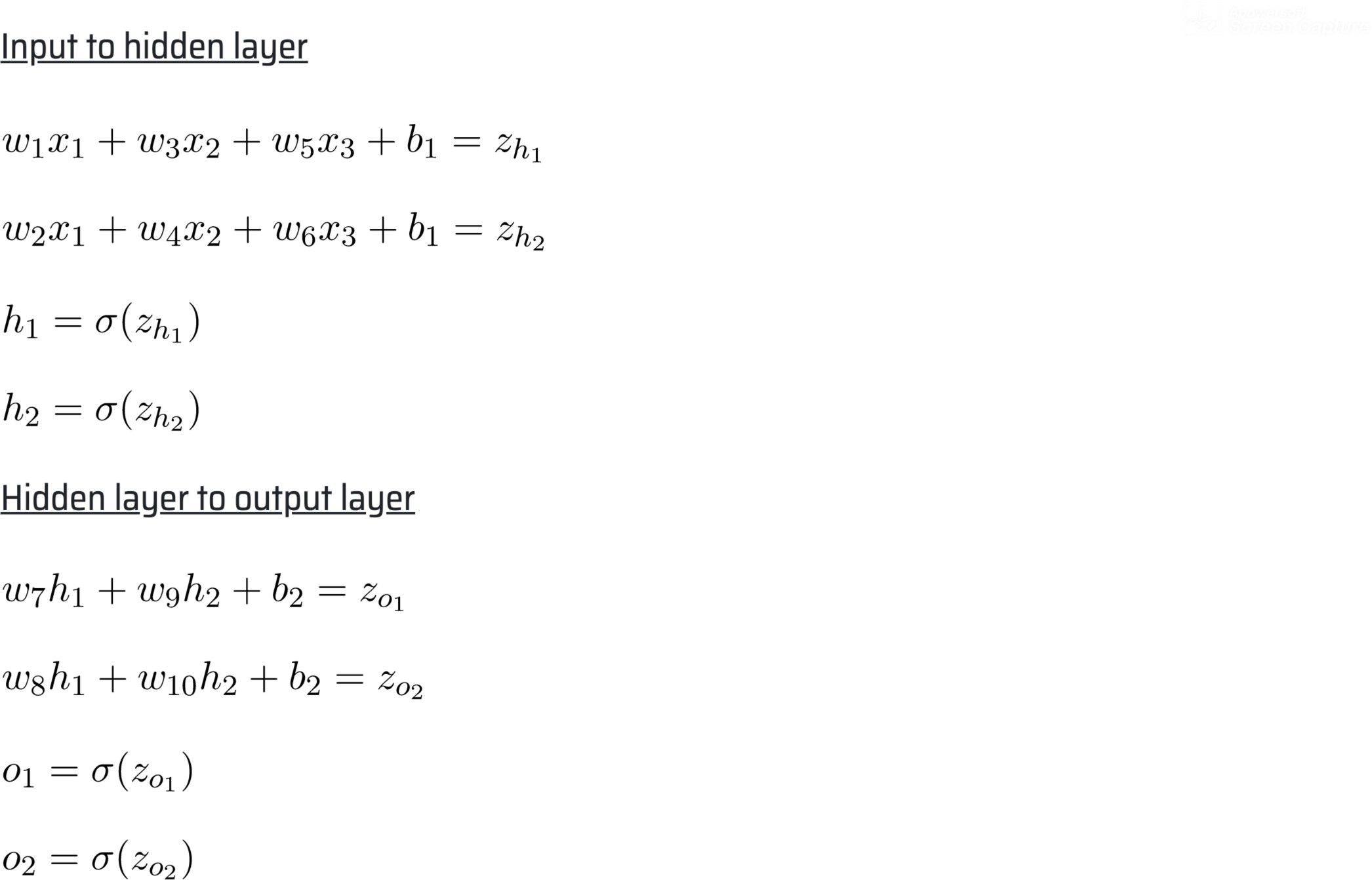
1. Initialize weights for the parameters we want to train
2. Forward propagate through the network to get the output values
3. Define the error or cost function and its first derivatives
4. Backpropagate through the network to determine the error derivatives
5. Update the parameter estimates using the error derivative and the current value
6. **Initialize**

****

The input and target values for this problem are x1=1, x2=4, x3=5 and t1=0.1, t2=0.05. Initialize weights as shown in the diagram. Generally, you will assign them randomly but for calculation purposes, I’ve chosen these numbers.

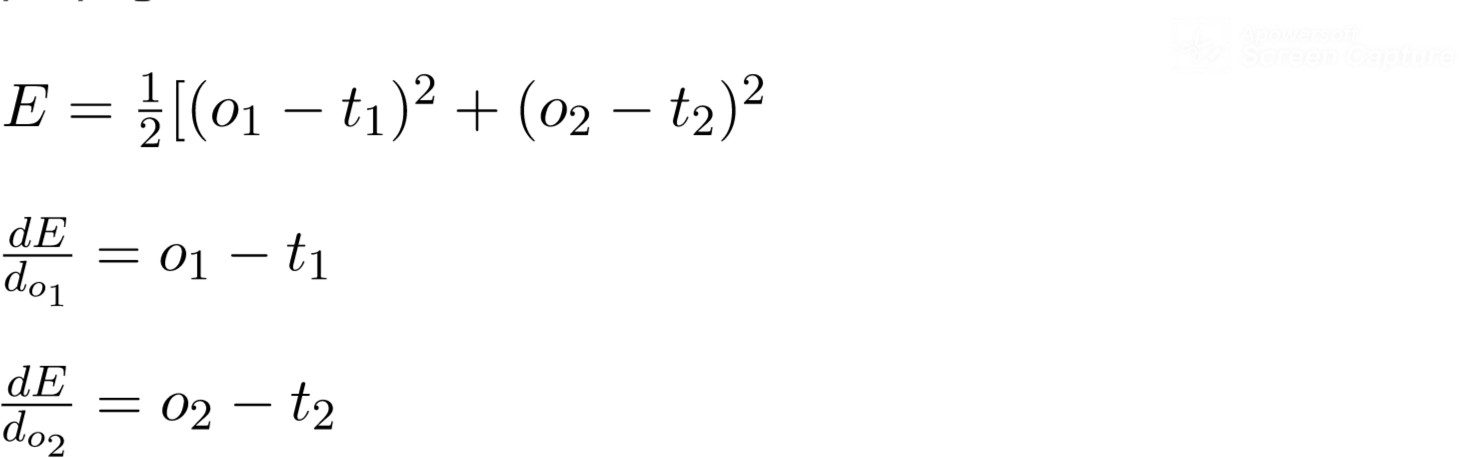
# Forward Propagate

Mathematically, we have the following relationships between nodes in the networks. For the input and output layer, let’s use the convention of denoting zh1, zh2, zo1, and zo2 to denote the value before the activation function is applied and the notation of h1, h2, o1, and o2 to denote the values after application of the activation function.



We can use the formulas above to forward propagate through the network. Calculate the numerical values.

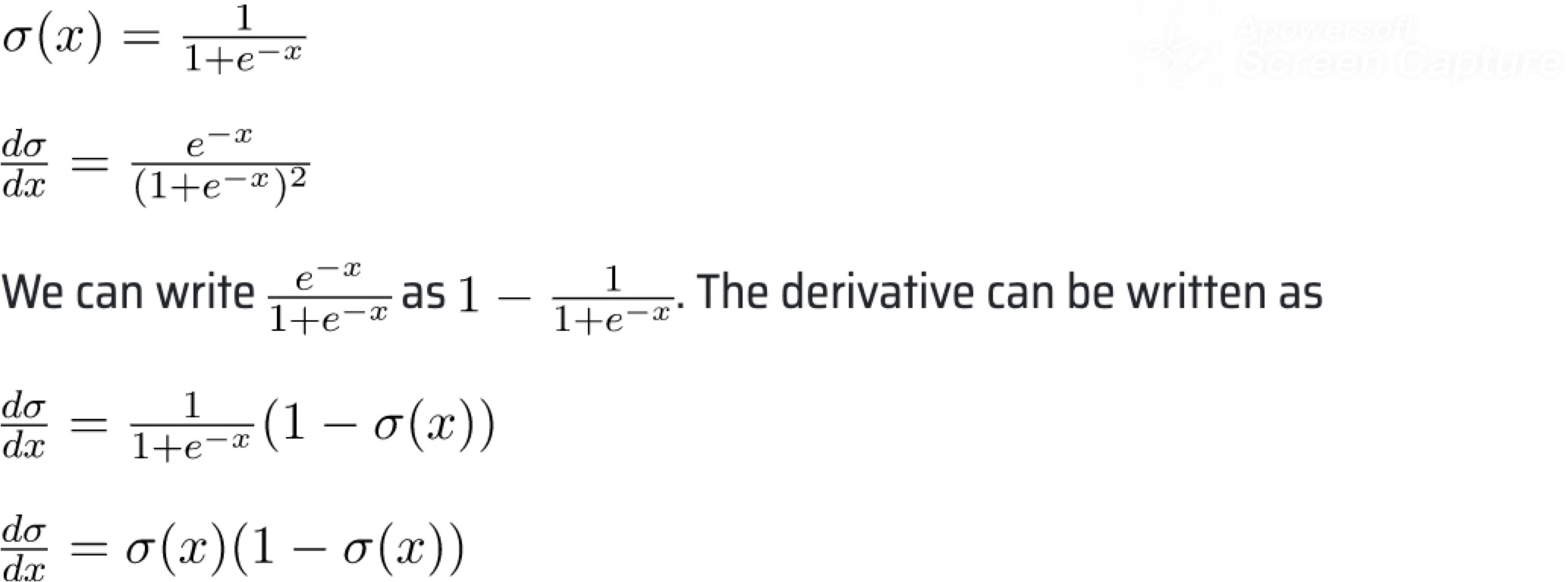
# Sum of Squares Error

Define the sum of squares error using the target values and the results from the last layer from forward propagation.

# BackPropagate

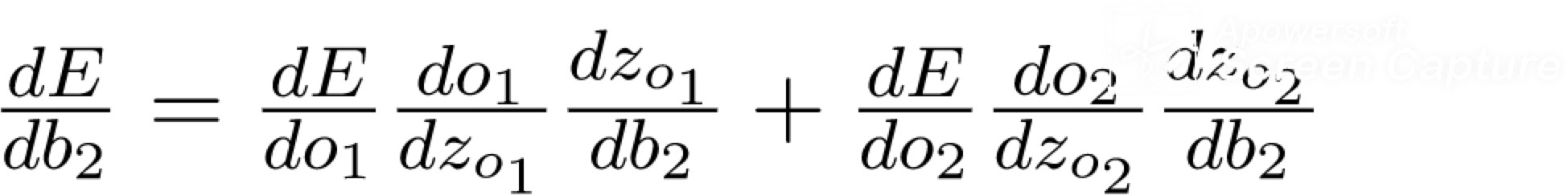
We are now ready to backpropagate through the network to compute all the error derivatives with respect to the parameters. Note that although there will be many long formulas, we are not doing anything fancy here. We are just using the basic principles of calculus such as the chain rule.

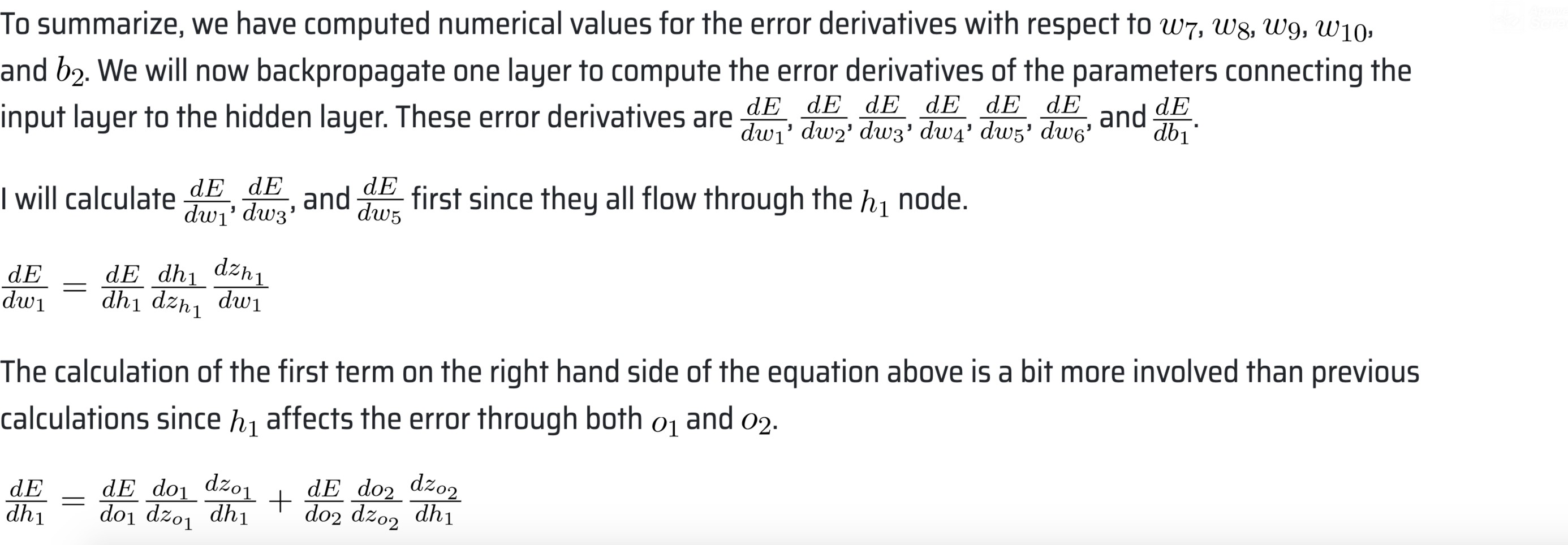
First, derivative of a Sigmoid function:



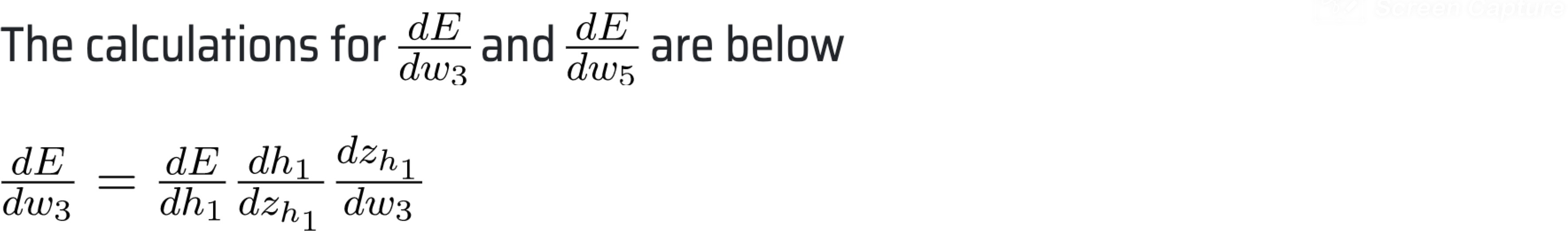
Similarly find the other three and find the numerical values w.r.t w8, w9, w10:

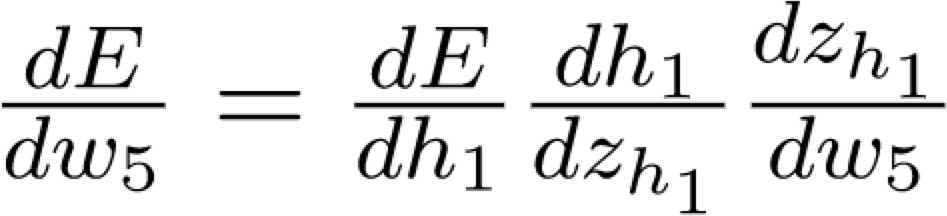
The error derivative of b2 is a little bit more involved since changes to b2 affect the error through both o1 and o2.

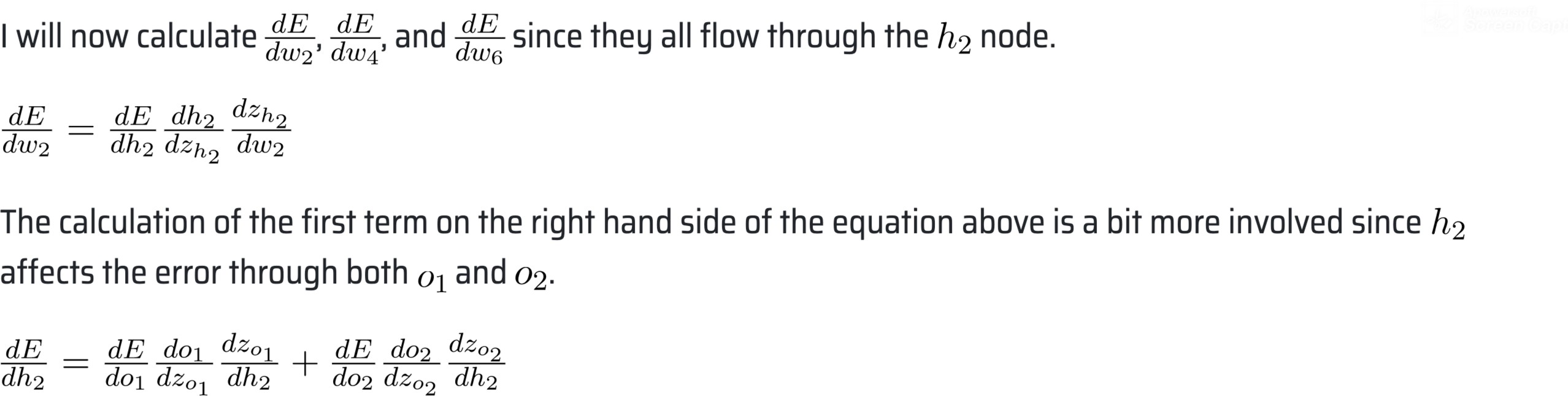


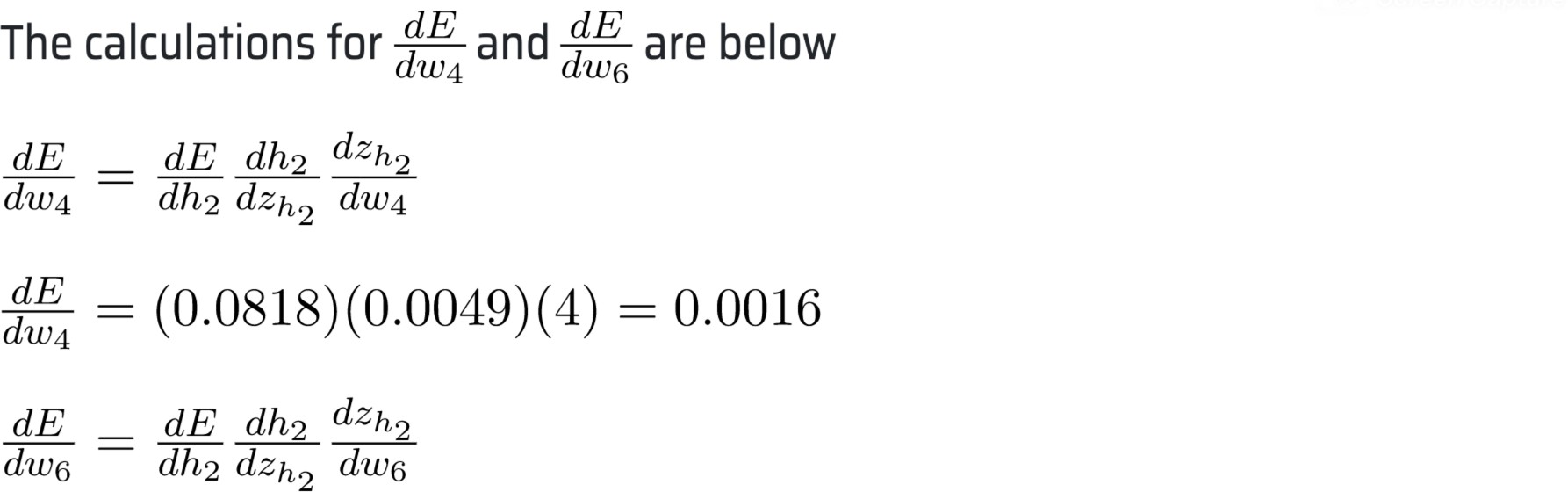
**Propagate to the previous layer first w.r.t to h1 and then h2**

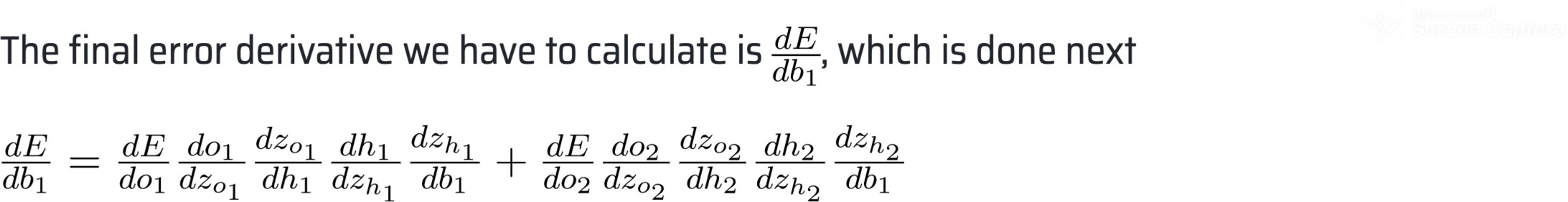
Proceed with the numerical values for the error derivatives above. These derivatives have already been calculated above or are similar in style to those calculated above.





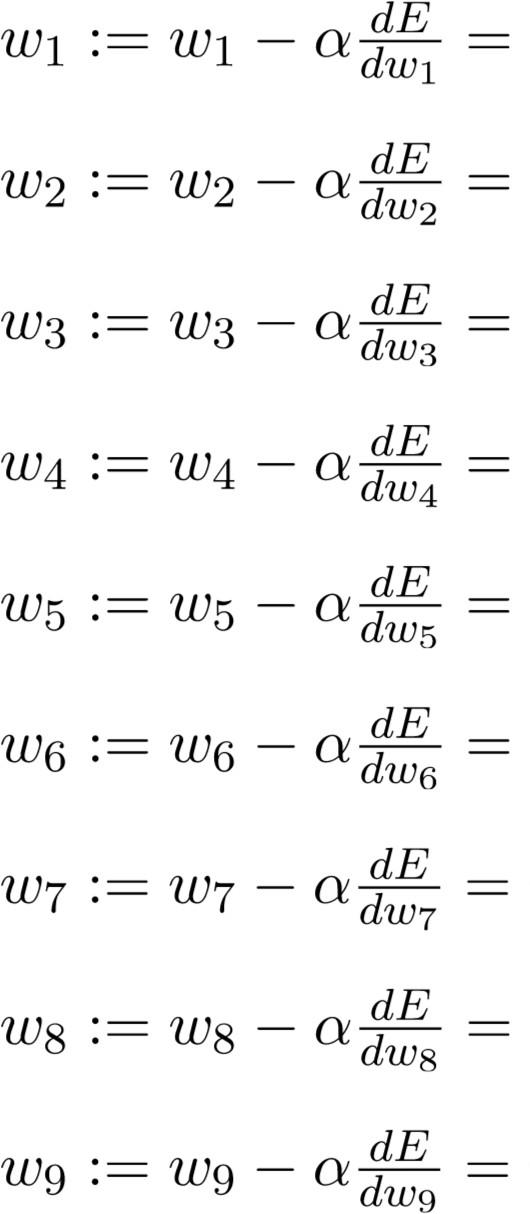
Find these and other derivates mentioned above w.r.t w2, w4, w6. Some hint below:

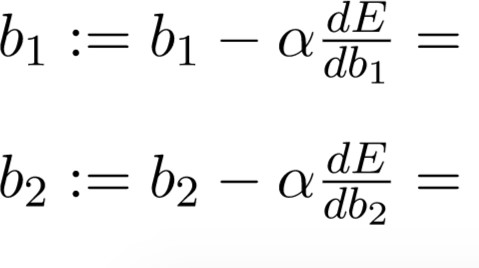
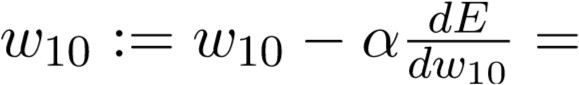




# Update the weights and bias

We now have all the error derivatives and we’re ready to make the parameter updates after the first iteration of backpropagation. We will use the learning rate of \alpha = 0.01





Repeat the iterations starting from the beginning.