**Block 1:**

i1 and i2 are the input variables. We have two hidden layers h1 and h2. h1 and h2 represent the hidden layer neurons in a neural network. They are computed as weighted sums of input values. The **activation function** applied to the hidden layer neurons is the **sigmoid function** (σ), which squashes the output between 0 and 1. Next, we compute the **output layer** values **o1** and **o2** using the hidden layer activations and we apply the sigmoid function to the output layer to get the final output activations a\_o1 and a\_o2. The **error** for each output is calculated as the squared difference between the target value (t1 or t2) and the actual output (a\_o1 or a\_o2). Finally, the **total error** is the sum of individual errors. The weights (w1 to w8) are learned during training to minimize the total error. The sigmoid activation ensures that the output is within a reasonable range for classification or regression tasks.

**Block 2:**

∂E\_total/∂w5: This represents the rate of change of the total error with respect to the weight w5. It can be simplified as **∂E1/∂w5**.

**∂E1/∂w5** is the partial derivative of the error **E1** with respect to the weight **w5**., and we can compute this by applying chain rule for derivative.

**Block 3:**

Now that we have calculated **∂E\_total/∂w5,** let’s compute the derivative with respect to w6, w7 and w8 also.

**Block 4:**

In this step we go one step further to calculate the **∂E\_total** with respect to the non-linear functions in the network.

**∂E\_total/∂a\_h1** represents the rate of change of the total error with respect to the hidden layer activation **a\_h1.** These partial derivatives help us understand how changes in the hidden layer activations impact the overall error. The weights (w5, w6, w7, w8) determine how much each output contributes to the error at the hidden layer.

**Block 5:**

In this set of equations, we are calculating the impact of w1, w, w3 and w4 on the E\_total.

**Block 6:**

Finally we can understand the impact of all the initial weights on E\_total. For example **∂E\_total/∂w1 can be computed as follows:**

* The first term corresponds to the contribution from output neuron 1:

((a\_01 - t1) \* a\_o1 \* (1 - a\_o1) \* w5

* The second term corresponds to the contribution from output neuron 2:

(a\_02 - t2) \* a\_o2 \* (1 - a\_o2) \* w7

* Finally, we multiply these terms by the contribution from the hidden layer neuron 1

a\_h1 \* (1 - a\_h1) \* i1

These equations describe the forward pass and backward pass in a neural network, including hidden layer computations, activation functions, output layer calculations, and error terms. The goal is to adjust weights to minimize the total error between predicted and target values.