MODERN COMPUTER VISION

BY RAJEEV RATAN

Back Propagation

Back Propagation makes Neural Networks Trainable

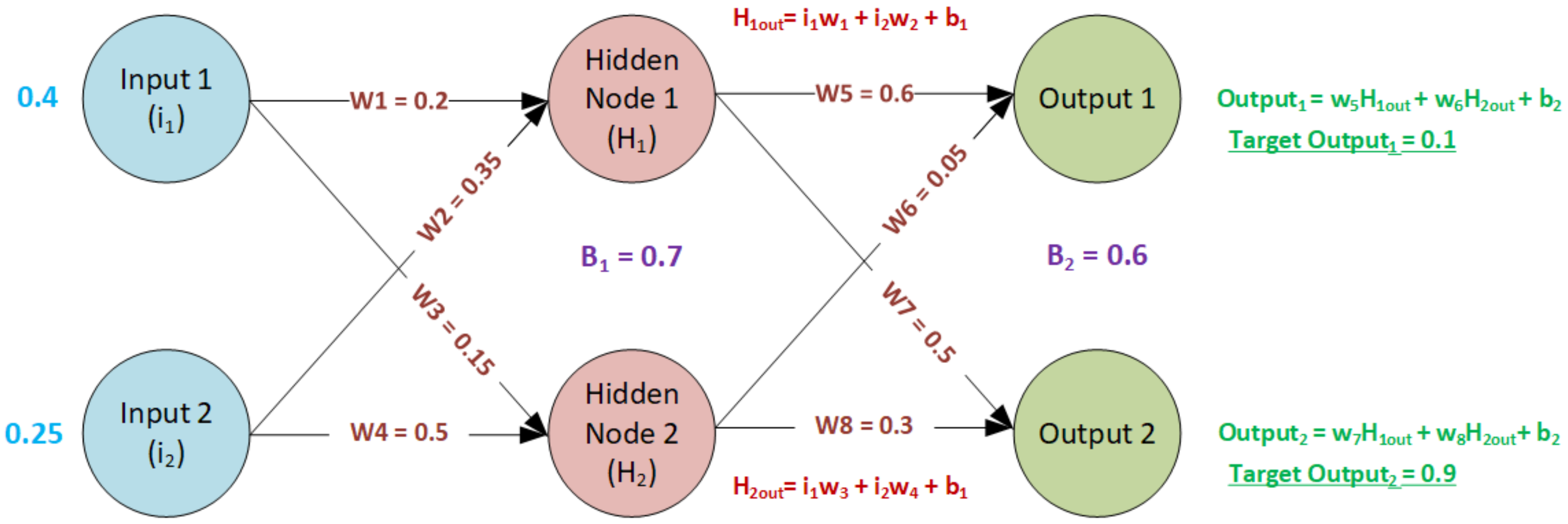
Back Propagation

This is what makes Neural Networks Trainable :)

- The importance of Back Propagation cannot be understated
- Using the loss, it tells us how much to change/update the gradients by so that we reduce the overall loss

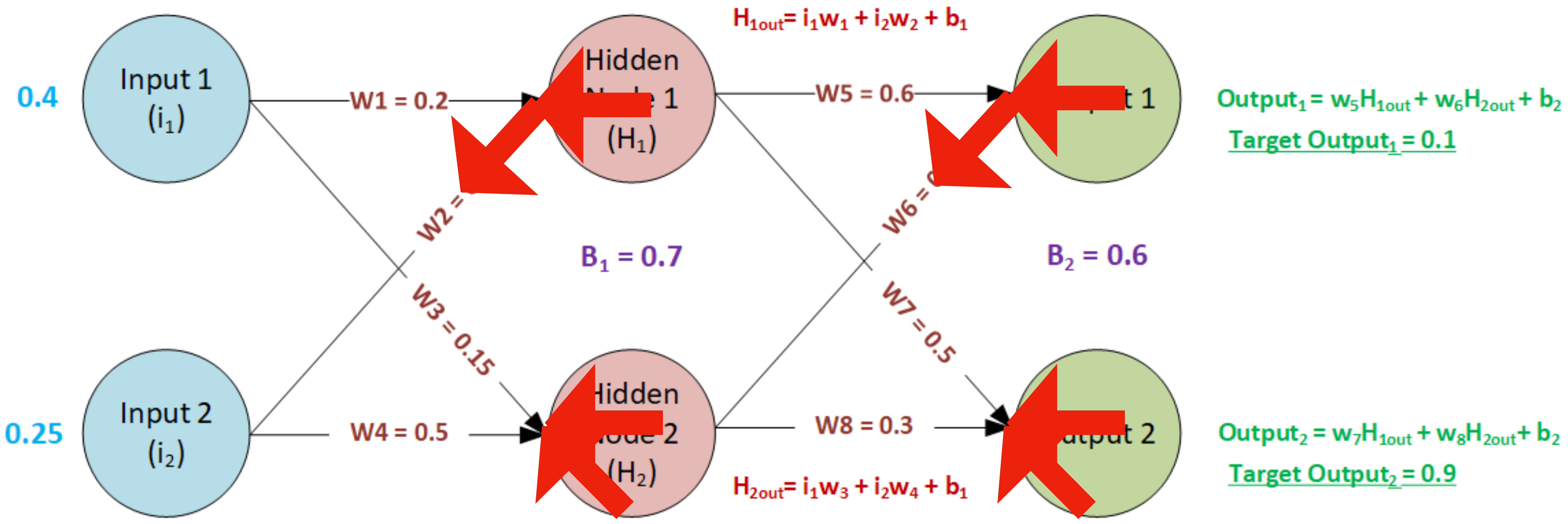
Back Propagation Example

Explained Using Neural Networks



- Let's look at a regular Neural Network above.
- Using the Loss value, Back Propagation can tells us whether a **small increase** of W_5 to 0.6001 or a **small decrease** 0.5999 will lead to a **reduction in the overall loss**

Back Propagation Example



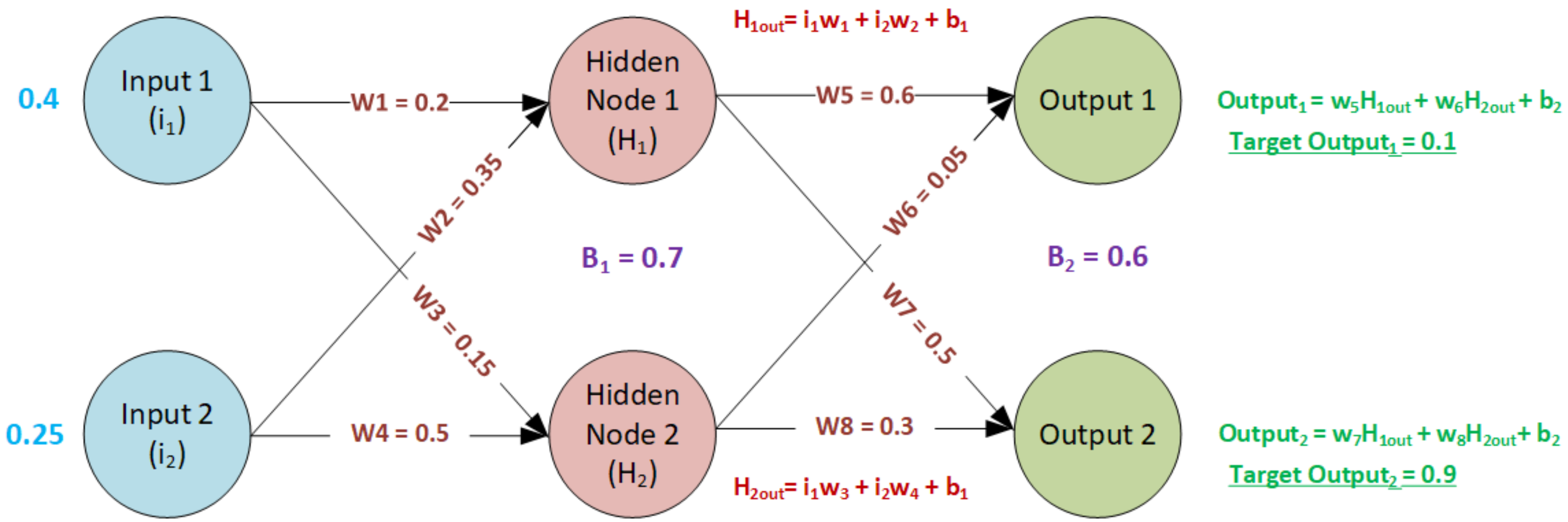
- Moving **right to left**
- Back Propagation gives us the new gradient or weight values for each node so that the overall loss is decreased
- This is done for all nodes

Back Propagation Process

- By **forward** propagating input data we can use back propagation to **lower** the weights to **lower the loss**
- **But**, this simply tunes the weights for that particular input (or batch of inputs)
- We improve **Generalisation** (ability to make good predictions on unseen data) by using all data in our training dataset
- By continuously changing the weights for each data input (or batch of images) we are lowering the overall loss for our training data.

What do our Weights or Gradients Look Like?

Let’s look at a Simple Neural Network



- The output from Hidden Node 1 is:
  - $H_1out = i_1w_1 + i_2w_2 + b_1$

For a Convolutional Neural Network

1	0	1	0	1
1	0	0	1	1
0	1	1	0	0
1	0	0	1	0
0	0	1	1	0

Input Image

*

0	1	0
1	0	-1
0	1	0

Filter or Kernel

=

2	1	-1
-1	1	3
2	1	1

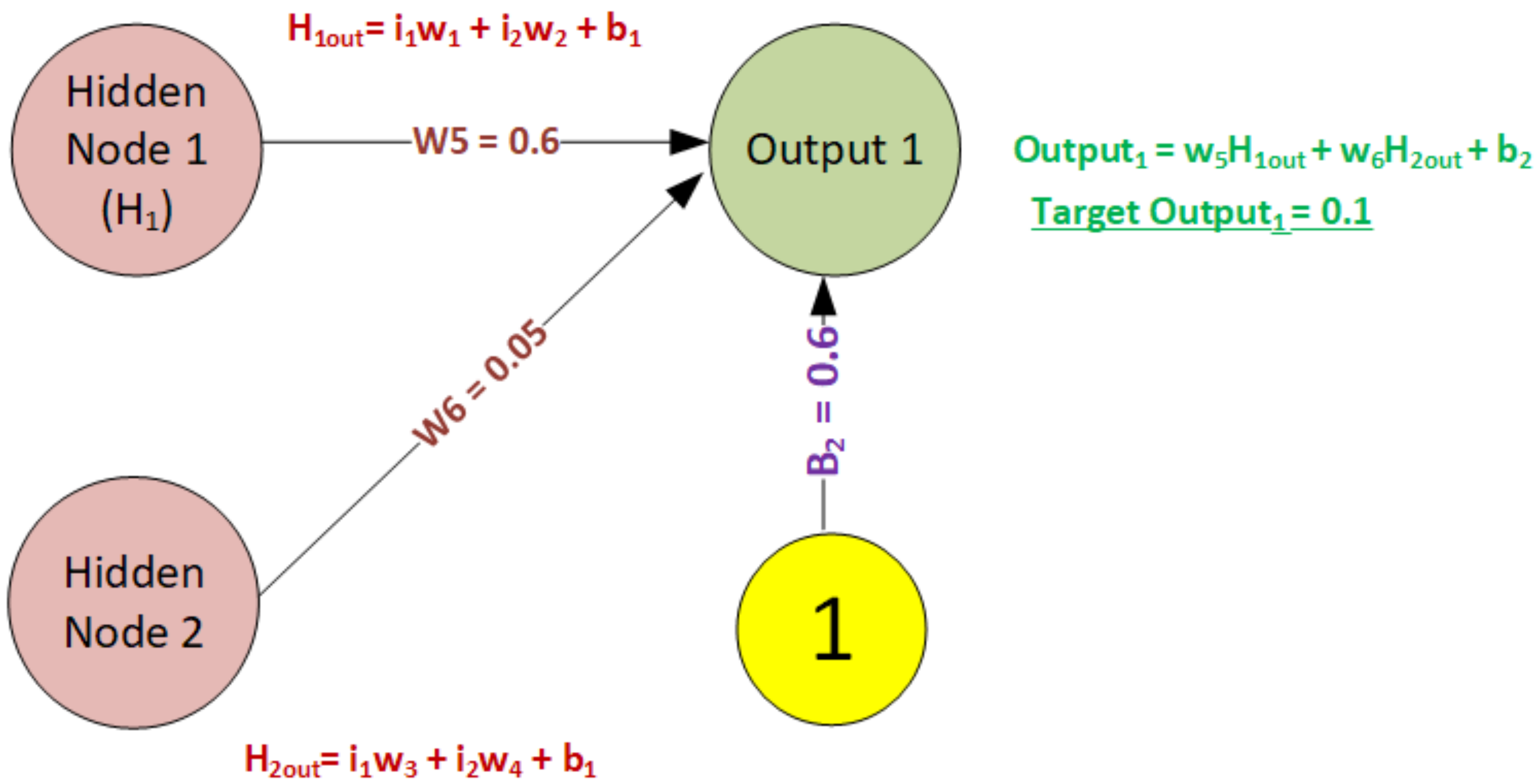
Output or Feature Map

- The values of our Filter/Kernel are the weights!

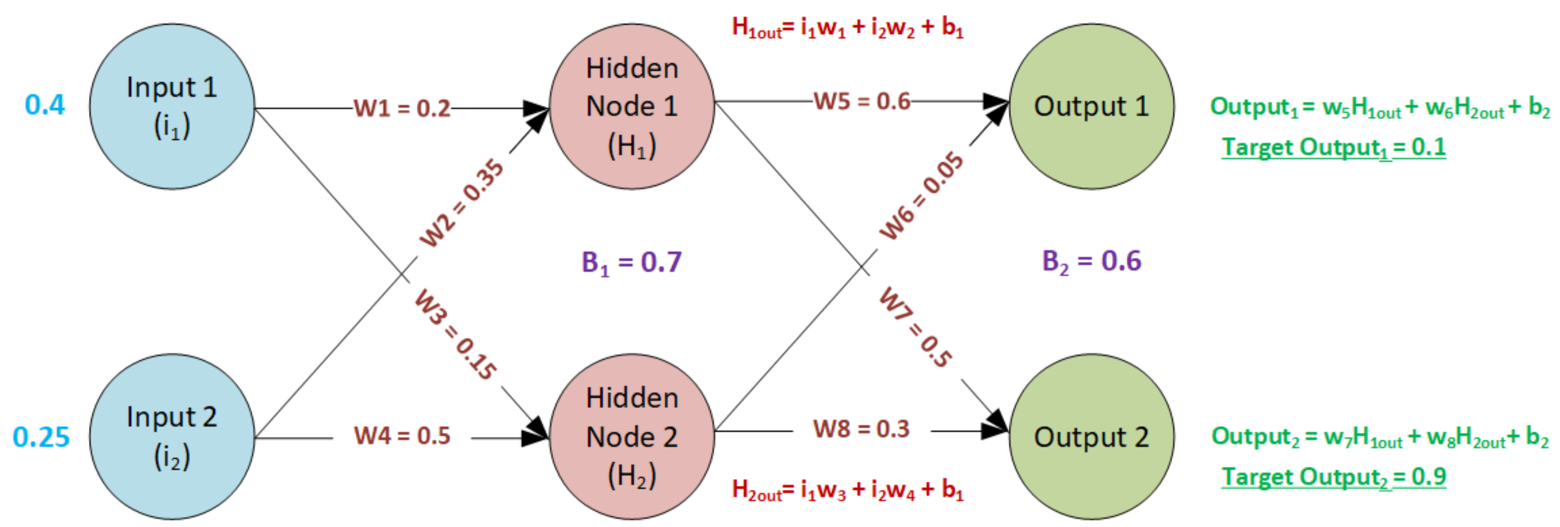
How does Back Propagation Work?

- **Chain Rule!**
- If we have two functions $y = f(u)$ and $u = g(x)$ then the derivative of y is:
- $\dfrac{dy}{dx} = \dfrac{dy}{du} \times \dfrac{du}{dx}$

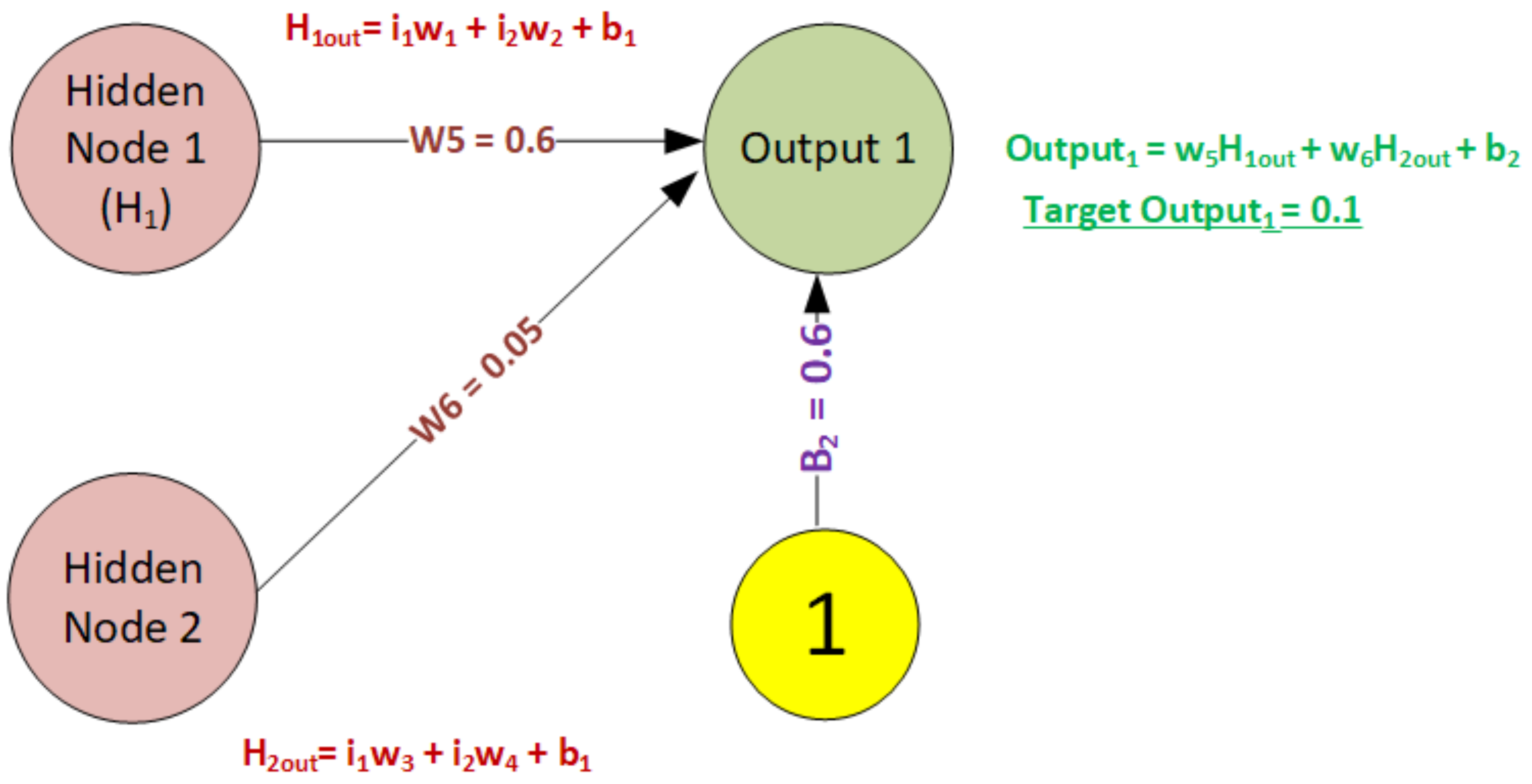
A Simple Back Propagation Example



- We want to know how much changing W_5 changes the **Total Error**.
- That is given by:
  - $$\frac{dE_T}{dW_5}$$
  - Where E_T is the sum of the error from Outputs 1 and 2 (see below)



A Simple Back Propagation Example



New $W_5 = -\lambda \times \dfrac{dE_T}{dW_5}$

- Note we introduced a new parameter λ
- λ is our learning rate
- It controls how a big a jump (positive or negative) we take when updating W_5
- Large learning rates train faster, but can get stuck in a Global Minimum
- Small learning rates train more slowly

MODERN COMPUTER VISION

BY RAJEEV RATAN

Next…

Gradient Descent