





DZone (/) > [AI Zone \(/artificial-intelligence-tutorials-tools-news\)](#) > A Very Basic Introduction to Feed-Forward Neural Networks

A Very Basic Introduction to Feed-Forward Neural Networks



(/users/3187203/beqari.html) by **Edvin Beqari** (/users/3187203/beqari.html) · Jan. 23, 18

· [AI Zone \(/artificial-intelligence-tutorials-tools-news\)](#) · Tutorial

 Like (2)  Comment (2)  Save  Tweet

Before we get started with our tutorial, let's cover some general terminology that we'll need to know.

General Architecture

- **Node:** The basic unit of computation (represented by a single circle)
- **Layer:** A collection of nodes of the same type and index (i.e. input, hidden, outer layer)
- **Connection:** A weighted relationship between a node of one layer to the node of another layer
- **W:** The weight of a connection
- **I:** Input node (the neural network input)
- **H:** Hidden node (a weighted sum of input layers or previous hidden layers)
- **HA:** Hidden node activated (the value of the hidden node passed to a predefined

x

- **OA:** Output node activated (the neural network output, the value of an output node passed to a predefined function)
- **B:** Bias node (always a contrant, typically set equal to 1.0)
- **e:** Total difference between the output of the network and the desired value(s) (total error is typically measured by estimators such as mean squared error, entropy, etc.)

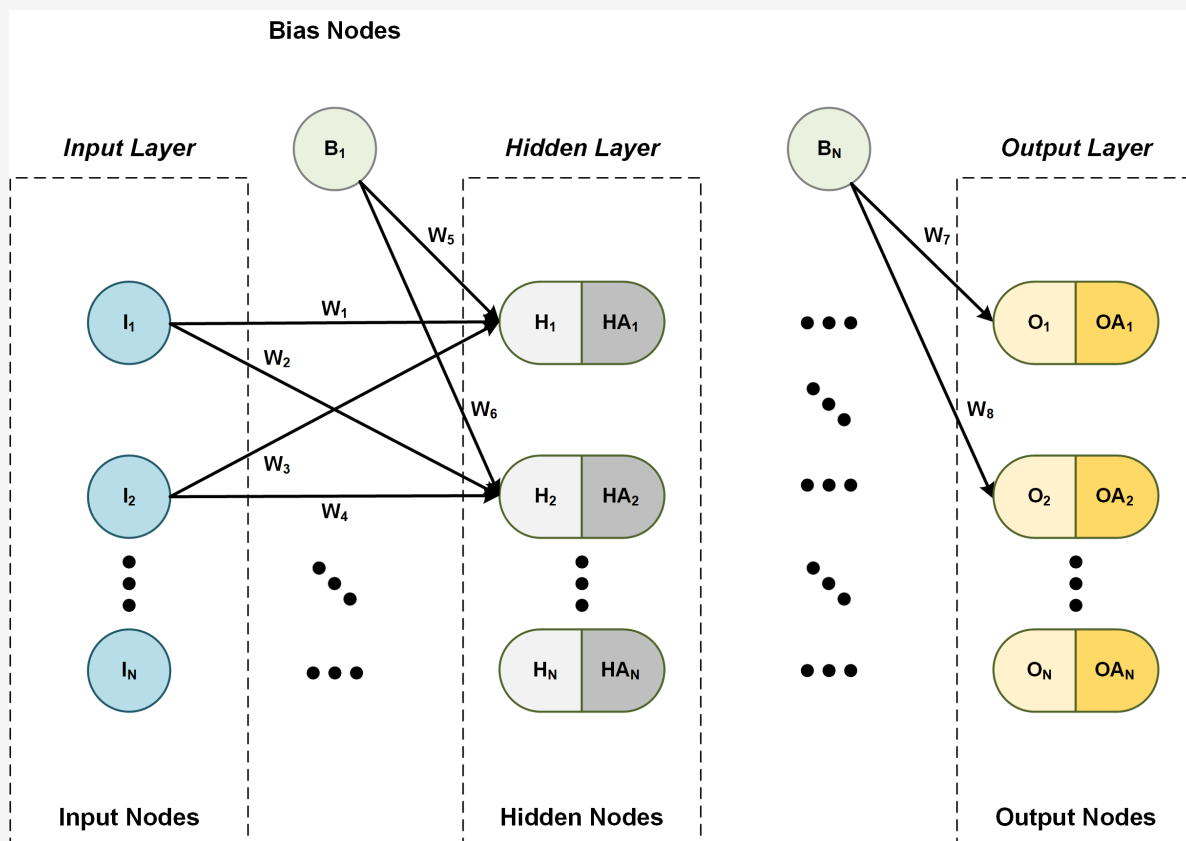


Figure 1: General architecture of a neural network

Getting straight to the point, neural network layers are independent of each other; hence, a specific layer can have an arbitrary number of nodes. Typically, the number of hidden nodes must be greater than the number of input nodes. When the neural network is used as a function approximation, the network will generally have one input and one output node. When the neural network is used as a classifier, the input and

output nodes will match the input features and output classes. A neural network must have at least one hidden layer but can have as many as necessary. The bias nodes are always set equal to one. In analogy, the bias nodes are similar to the offset in linear regression i.e. $y = mx + b$. How does one select the proper number of nodes and hidden number of layers? This is the best part: there are really no rules! The modeler is free to use his or her best judgment on solving a specific problem. Experience has shown that there are best practices such as selecting an adequate number of hidden layers, activation functions, and training methods; however, this is beyond the scope of this article.

As a user, one first has to construct the neural network, then train the network by iterating with known outputs (AKA desired output, expected values) until convergence, and finally, use the trained network for prediction, classification, etc.

The Math

Let's consider a simple neural network, as shown below.

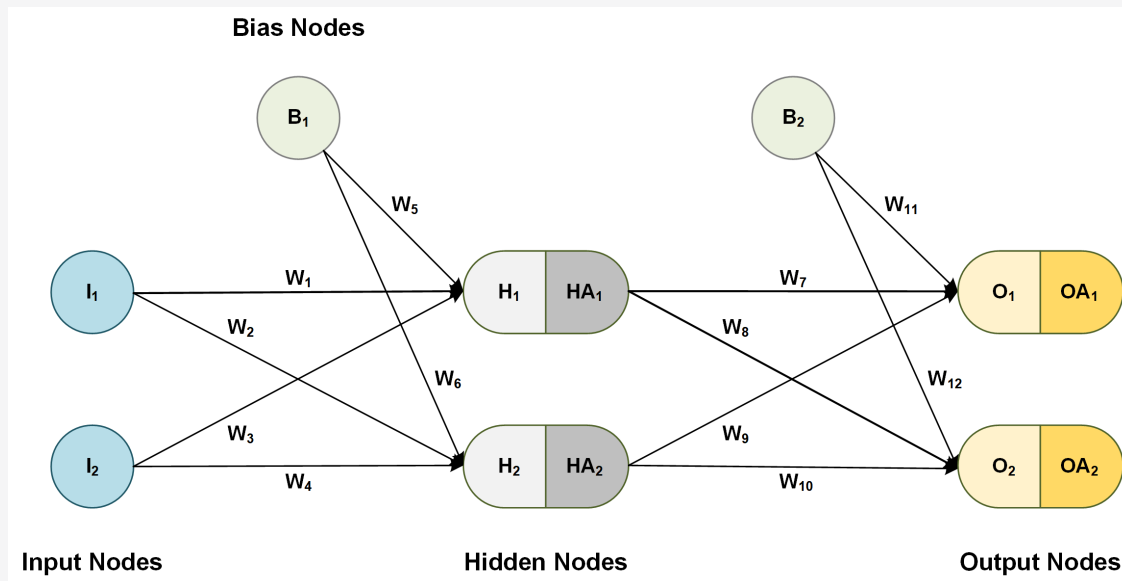


Figure 2: Example of a simple neural network

Step 1: Initialization

The first step after designing a neural network is initialization:

- Initialize all weights W_1 through W_{12} with a random number from a normal distribution, i.e. $\sim N(0, 1)$.
- Set all bias nodes $B_1 = B_2 = 1.0$.



Note: Keep in mind that the variance of the distribution can be a different value. This has an effect on the convergence of the network.


[\(/users/login.html\)](/users/login.html)

[\(/search\)](/search)
[REFCARDZ \(/refcardz\)](/refcardz)
[TREND REPORTS \(/trendreports\)](/trendreports)
[WEBINARS \(/webinars\)](/webinars)
[ZONES ▾](#)

Step 2: Feed-Forward

As the title describes it, in this step, we calculate and move forward in the network all the values for the hidden layers and output layers.

- Set the values of all input nodes.
- Calculate hidden node values.

$$H_1 = I_1W_1 + I_2W_3 + B_1W_5$$

$$H_2 = I_1W_2 + I_2W_4 + B_1W_6$$

- Select an activation function for the hidden layer; for example, the Sigmoid function:

$$S(x) = \frac{1}{1 + e^{-x}}$$

- Calculate hidden node activation values:

$$HA_1 = \frac{1}{1 + e^{-H_1}}$$

$$HA_2 = \frac{1}{1 + e^{-H_2}}$$

- Calculate output node values:

$$O_1 = HA_1W_7 + HA_2W_9 + B_2W_{11}$$

$$O_2 = HA_1W_8 + HA_2W_{10} + B_2W_{12}$$

- Select an activation function for the output layer; for example, the linear function:

$$y = f(x) = x$$

- Calculate output node activation values:

$$OA_1 = O_1$$



• Calculate the total error: if OA_i is the obtained output value for node i , then let y_i be the desired output.

$$e = \frac{1}{n} \sum_{i=1}^n (y_i - OA_i)^2$$

Note: Here, the error is measured in terms of the mean square error, but the modeler is free to use other measures, such as entropy or even custom loss functions..

After the first pass, the error will be substantial, but we can use an algorithm called backpropagation to adjust the weights to reduce the error between the output of the network and the desired values

Step 3: Backpropagation

This is the step where the magic happens. The goal of this step is to incrementally adjust the weights in order for the network to produce values as close as possible to the expected values from the training data.

Note: Keep in mind statistical principles such as overfitting, etc.

Backpropagation can adjust the network weights using the stochastic gradient decent optimization method.

$$W_i^{k+1} = W_i^k - \eta \frac{\partial e}{\partial W_i^k}$$

Where k is the iteration number, η is the learning rate (typically a small number), and:

$$\frac{\partial e}{\partial W_i^k}$$

...is the derivative of the total error with regards to to the weight adjusted.

In this procedure, we derive a formula for each individual weight in the network, including bias connection weights. The example below shows the derivation of the update formula (gradient) for the first weight in the network.

Let's calculate the derivative of the error e with regards to to a weight between the input and hidden layer, for example, W_1 using the calculus chain rule.

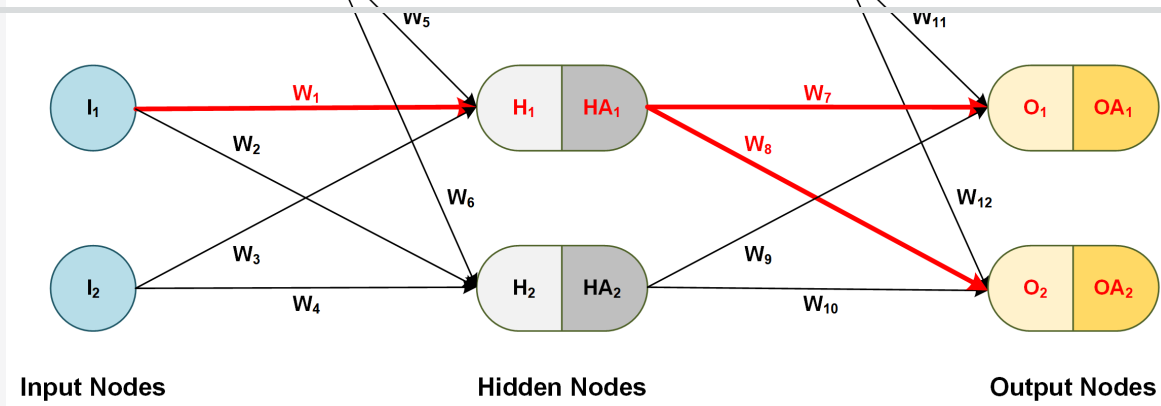


Figure 3: Chain rule for weights between input and hidden layer

$$\frac{\partial e}{\partial W_1} = \underbrace{\left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial H_1} \frac{\partial H_1}{\partial W_1} \right)}_{(1)} + \underbrace{\left(\frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial H_1} \frac{\partial H_1}{\partial W_1} \right)}_{(2)}$$

- Start from the very first activated output node and take derivatives backward for each node. For simplicity, one can think of a node and its activated self as two different nodes without a connection.

$$\frac{\partial e}{\partial O_1} = \frac{\partial \left(\frac{1}{n} \sum_{i=1}^2 (y_i - O_{A_i})^2 \right)}{\partial O_{A_1}} = \frac{\partial \frac{1}{n} [(y_1 - O_{A_1})^2 + (y_2 - O_{A_2})^2]}{\partial O_{A_1}} = -\frac{2}{n} (y_1 - O_{A_1})$$

- From the activated output bounce to the output node:

$$\frac{\partial O_{A_1}}{\partial O_1} = \frac{\partial O_1}{\partial O_1} = 1$$

- From the output node bounce to the first activated node of the last hidden layer:

$$\frac{\partial O_1}{\partial H_{A_1}} = \frac{\partial (H_{A_1} W_7 + H_{A_2} W_9 + B_2 W_{11})}{\partial H_{A_1}} = W_7$$

- From the activated hidden node, bounce to the hidden node itself:

- From the first hidden node, bounce to the weight of the first connection:

$$\frac{\partial H_1}{\partial W_1} = \frac{\partial (I_1 W_1 + I_2 W_3 + B_1 W_5)}{\partial W_1} = I_1$$

This concludes one unique path to the weight derivative — but wait... there is one additional path that we have to calculate. Refer to Figure 3, and notice the connections and nodes marked in red.

- Once again, start from the next activated output node and make your way backward by taking derivatives for each node. This time, we do not need to spell out every step.

$$\begin{aligned} \frac{\partial e}{\partial O_2} &= \frac{\partial \left(\frac{1}{n} \sum_{i=1}^2 (y_i - O_{A_i})^2 \right)}{\partial O_2} = \frac{\partial \left(\frac{1}{n} [(y_1 - O_{A_1})^2 + (y_2 - O_{A_2})^2] \right)}{\partial O_2} = -\frac{2}{n} (y_1 - O_{A_2}) \\ \frac{\partial O_2}{\partial O_2} &= \frac{\partial O_2}{\partial O_2} = 1 \\ \frac{\partial O_2}{\partial H_1} &= \frac{\partial (H_{A_1} W_8 + H_{A_2} W_{10} + B_2 W_{12})}{\partial H_1} = W_8 \\ \frac{\partial H_1}{\partial H_1} &= \frac{\partial \left(\frac{1}{1 + e^{-H_1}} \right)}{\partial H_1} = \left(\frac{1}{1 + e^{-H_1}} \right) \left(1 - \frac{1}{1 + e^{-H_1}} \right) \\ \frac{\partial H_1}{\partial W_1} &= \frac{\partial (I_1 W_1 + I_2 W_3 + B_1 W_5)}{\partial W_1} = I_1 \end{aligned}$$

- Finally, the total derivative for the first weight W1 in our network is the sum of the product the individual node derivatives for each specific path.

$$\frac{\partial e}{\partial W_1} = \underbrace{\left(\frac{\partial e}{\partial O_{A_1}} \frac{\partial O_{A_1}}{\partial O_1} \frac{\partial O_1}{\partial H_{A_1}} \frac{\partial H_{A_1}}{\partial H_1} \frac{\partial H_1}{\partial W_1} \right)}_x + \underbrace{\left(\frac{\partial e}{\partial O_{A_2}} \frac{\partial O_{A_2}}{\partial O_2} \frac{\partial O_2}{\partial H_{A_1}} \frac{\partial H_{A_1}}{\partial H_1} \frac{\partial H_1}{\partial W_1} \right)}_y$$

Note that we leave out the second hidden node because the first weight in the network does not depend on the node. This is clearly seen in Figure 3 above.

We follow the same procedure for all the weights one-by-one in the network.

Here is another example where we calculate the derivative of the error with regard to a weight between the hidden layer and the output layer:

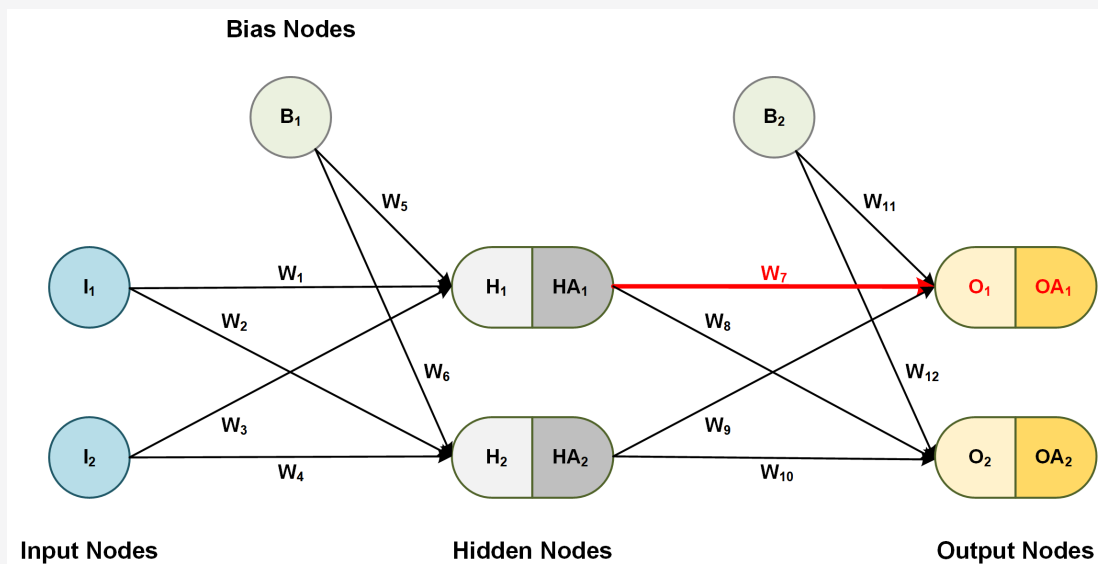


Figure 4: Chain rule for weights between hidden and output layer

- As in the previous step, start with the very first activated output weight in the network and take derivatives backward all the way to the desired weight, and leave out any nodes that do not affect that specific weight:

$$\frac{\partial e}{\partial OA_1} = \frac{\partial \left(\frac{1}{n} \sum_{i=1}^2 (y_i - OA_i)^2 \right)}{\partial OA_1} = \frac{\partial \left(\frac{1}{n} [(y_1 - OA_1)^2 + (y_2 - OA_2)^2] \right)}{\partial OA_1} = \frac{2}{n} (y_1 - OA_1)$$

$$\frac{\partial OA_1}{\partial O_1} = \frac{\partial O_1}{\partial O_1} = 1$$

$$\frac{\partial O_1}{\partial W_7} = \frac{\partial (HA_1 W_7 + HA_2 W_9 + B_2 W_{11})}{\partial W_7} = HA_1$$

- Lastly, we take the sum of the product of the individual derivatives to calculate the

$$\frac{\partial e}{\partial W_7} = \frac{\partial e}{\partial A_1} \frac{\partial A_1}{\partial W_7}$$

The same strategy applies to bias weights.

Once we have calculated the derivatives for all weights in the network (derivatives equal gradients), we can simultaneously update all the weights in the net with the gradient decent formula, as shown below.

$$\begin{bmatrix} W_1^{k+1} \\ W_2^{k+1} \\ \vdots \\ W_n^{k+1} \end{bmatrix} = \begin{bmatrix} W_1^k \\ W_2^k \\ \vdots \\ W_n^k \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial e}{\partial W_1^k} \\ \frac{\partial e}{\partial W_2^k} \\ \vdots \\ \frac{\partial e}{\partial W_n^k} \end{bmatrix}$$

Notes: Calculus Chain Rule

Why do we calculate derivatives for all these unique paths? Note that the backpropagation is a direct application of the calculus chain rule.

For example:

- Let's define the following functions:

$$\begin{aligned} x &= f(t) \\ y &= g(t) \\ z &= h(x, y) \end{aligned}$$

- If we need to take the derivate of z with regard to t, then by the calculus chain rule, we have:

$$\frac{\partial z}{\partial t} = \frac{\partial z}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial t}$$

Note that the total derivative of z with regard to t is the sum of the product of the



individual derivatives. What if t is also a function of another variable?



(/users/login.html)



(/search)

REFCARDS (/refcards) TREND REPORTS (/trendreports) WEBINARS (/webinars) ZONES ▾

$$t = k(s)$$

- Then, the derivate of z with respect to s , by the calculus chain rule, is the following:

$$\frac{\partial z}{\partial s} = \frac{\partial z}{\partial x} \frac{\partial x}{\partial t} \frac{\partial t}{\partial s} + \frac{\partial z}{\partial y} \frac{\partial y}{\partial t} \frac{\partial t}{\partial s}$$

Now, let's compare the chain rule with our neural network example and see if we can spot a pattern.

- Let's borrow the follow functions from our neural network example:

$$\begin{aligned} OA_1 &= f(O_1) \\ OA_2 &= g(O_2) \\ e &= h(OA_1, OA_2) \\ O_1 &= k(HA_1) \\ O_2 &= v(HA_1) \end{aligned}$$

- If we need to take the derivate of e , with respect to HA_1 , then by the chain rule, we have:

$$\frac{\partial e}{\partial HA_1} = \frac{\partial e}{\partial OA_1} \frac{\partial OA_1}{\partial O_1} \frac{\partial O_1}{\partial HA_1} + \frac{\partial e}{\partial OA_2} \frac{\partial OA_2}{\partial O_2} \frac{\partial O_2}{\partial HA_1}$$

The same pattern follows if HA_1 is a function of another variable.

At this point, it should be clear that the backpropagation is nothing more than the direct application of the calculus chains rule. One can identify the unique paths to a specific weight and take the sum of the product of the individual derivatives all the way to a specific weight.

Note: If you understand everything thus far, then you understand feedforward multilayer neural networks.

Neural networks with two or more hidden layers are called deep networks. The same rules apply as in the simpler case; however, the chain rule is a bit longer.

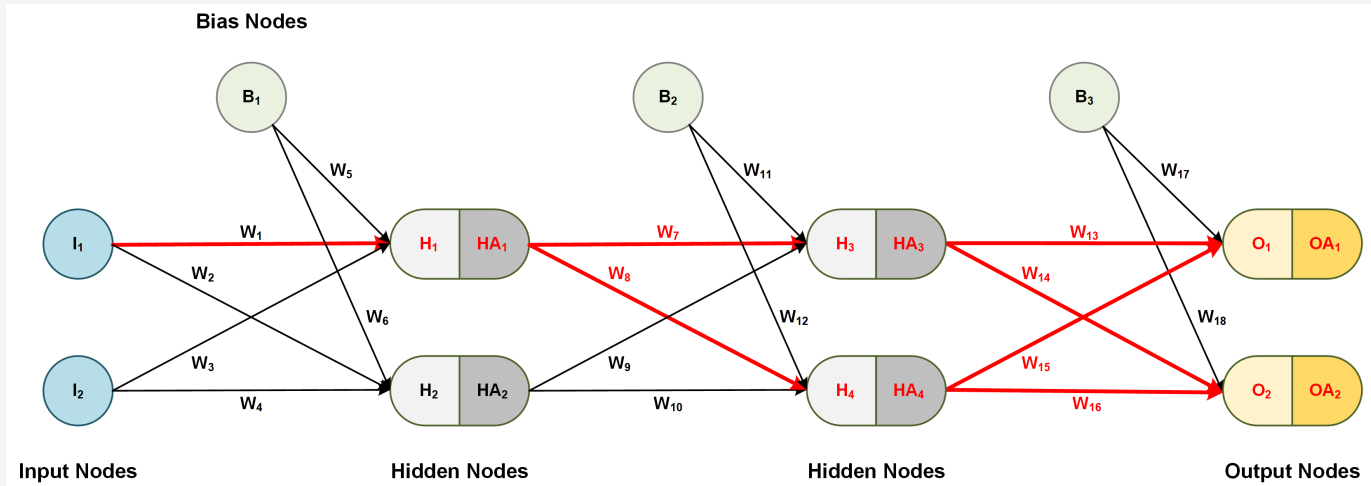
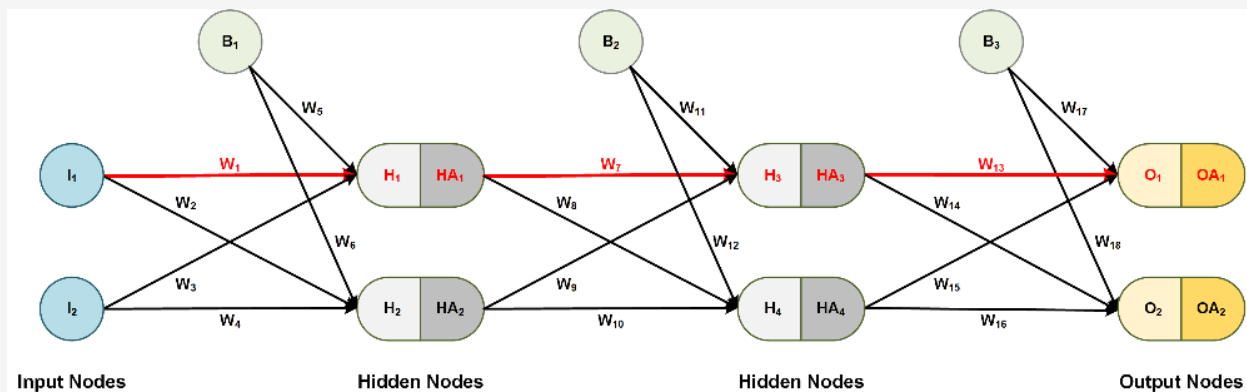


Figure 5: Chain rule for weights between input and hidden layer

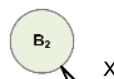
As an example, let's reevaluate the total derivative of the error with regard to W_1 , which is the sum of the product of each unique path from each output node, i.e. the sum of the products (paths 1-4).

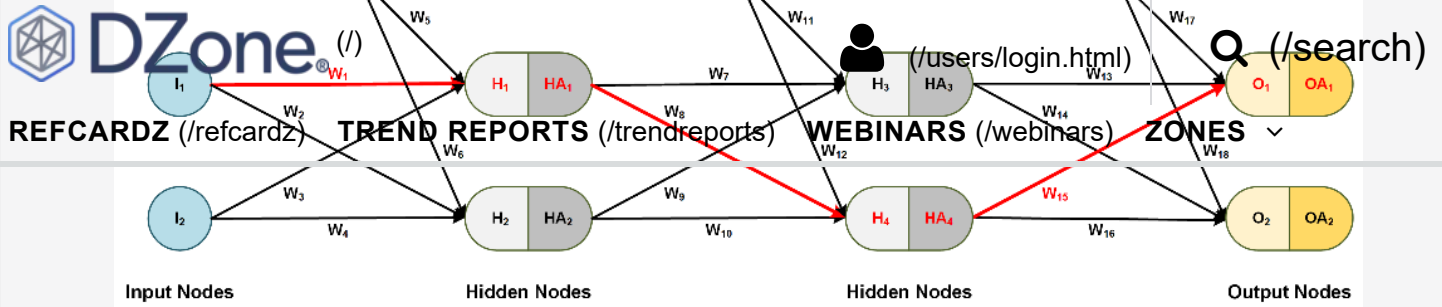
See example paths below.



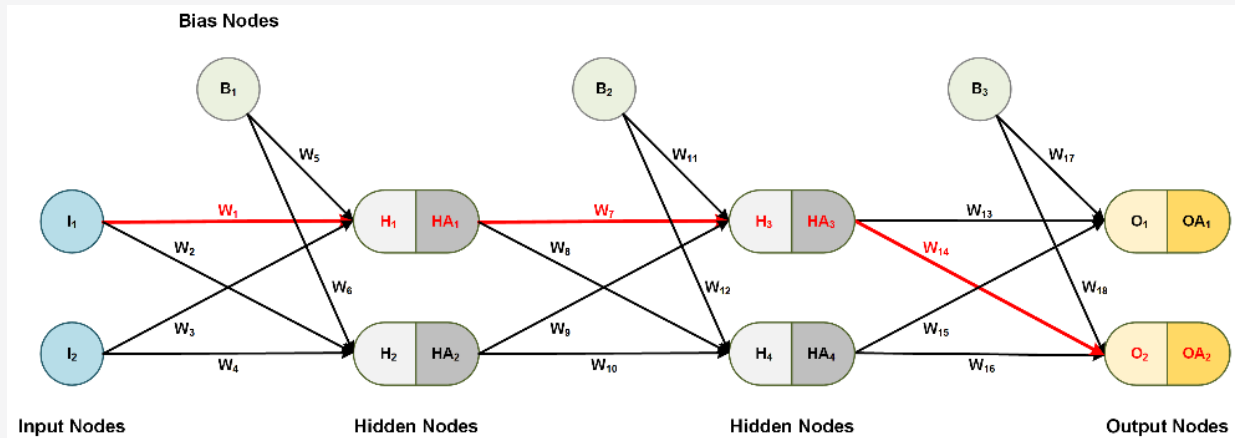
$$\frac{\partial e}{\partial W_1} = \left(\underbrace{\frac{\partial e}{\partial OA_1} \frac{\partial OA_1}{\partial O_1} \frac{\partial O_1}{\partial HA_3} \frac{\partial HA_3}{\partial H_3} \frac{\partial HA_1}{\partial H_1} \frac{\partial H_1}{\partial W_1}}_{(1)} \right)$$

Bias Nodes

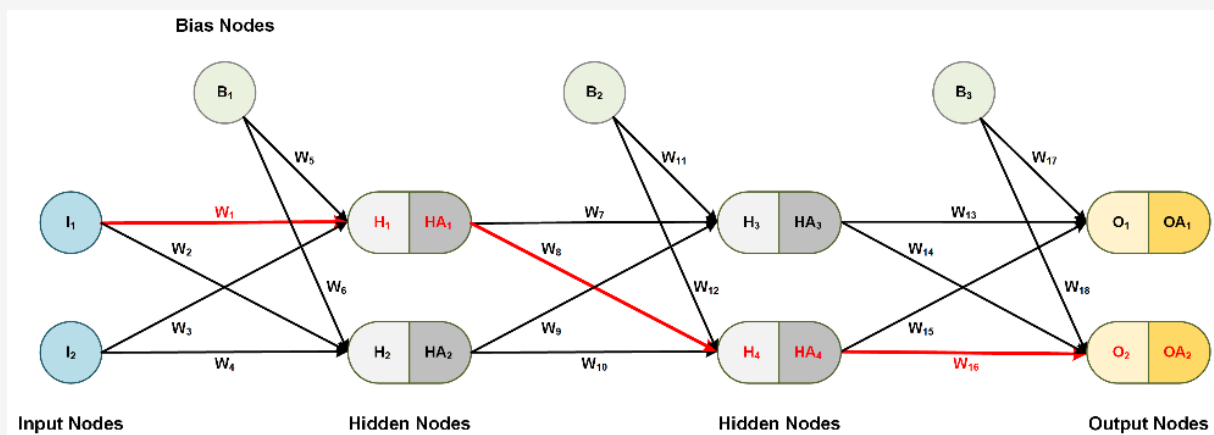




$$+ \left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial OA_1} \frac{\partial OA_1}{\partial H_1} \frac{\partial H_1}{\partial HA_1} \frac{\partial HA_1}{\partial W_1} \right) \quad (2)$$



$$+ \left(\frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial OA_2} \frac{\partial OA_2}{\partial H_2} \frac{\partial H_2}{\partial HA_2} \frac{\partial HA_2}{\partial W_2} \right) \quad (3)$$



$$+ \left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial OA_1} \frac{\partial OA_1}{\partial H_1} \frac{\partial H_1}{\partial HA_1} \frac{\partial HA_1}{\partial W_1} \right) \quad (4)_x$$



Next, we can factor the common terms, and the total derivative for W_1 is complete!

$$\frac{\partial e}{\partial W_1} = \left[\left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial H_3} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial H_3} \right) \frac{\partial H_3}{\partial H_1} \right] + \left[\left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial H_4} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial H_4} \right) \frac{\partial H_4}{\partial H_1} \right]$$

Notice something interesting here: each product factor belongs to a different layer. This observation will be useful later in the formulation.

Three Hidden Layers

Yet another example of a deep neural network with three hidden layers:

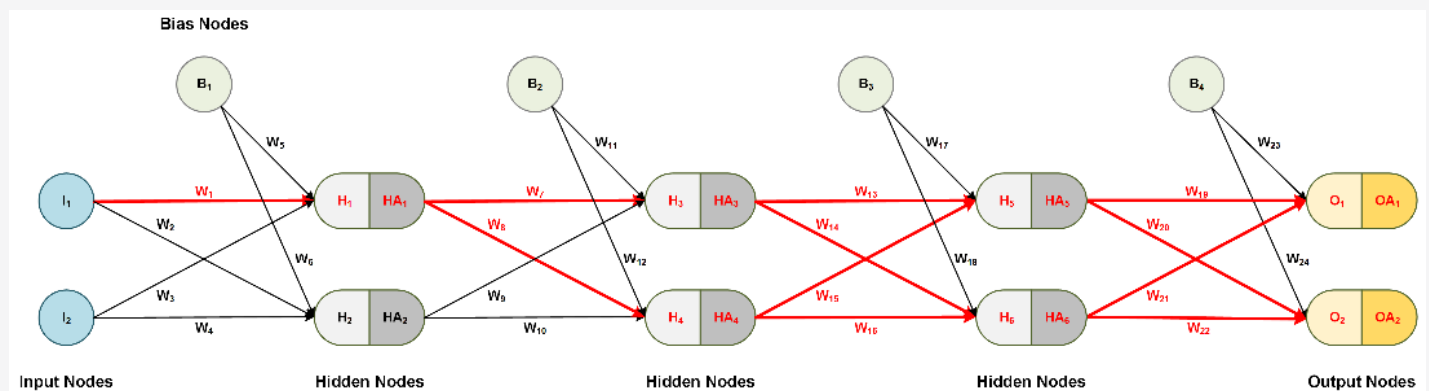


Figure 6: Chain rule for weights between input and hidden layer

Once again, the total derivative of the error e with regard to W_1 is the sum of the product of all paths (paths 1-8). Note that there are more path combinations with more hidden layers and nodes per layer.

$$\frac{\partial e}{\partial W_1} = \underbrace{\left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial H_3} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial H_3} \right) \frac{\partial H_3}{\partial H_1}}_{(1)} + \underbrace{\left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial H_4} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial H_4} \right) \frac{\partial H_4}{\partial H_1}}_{(2)} + \left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial H_5} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial H_5} \right) \frac{\partial H_5}{\partial H_1}$$

$$+ \underbrace{\left(\frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial H_6} \frac{\partial H_6}{\partial H_3} \frac{\partial H_3}{\partial H_1} \frac{\partial H_1}{\partial W_1} \right)}_{(4)}$$

$$+ \underbrace{\left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial H_5} \frac{\partial H_5}{\partial H_4} \frac{\partial H_4}{\partial H_1} \frac{\partial H_1}{\partial W_1} \right)}_{(5)}$$

$$+ \underbrace{\left(\frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial H_5} \frac{\partial H_5}{\partial H_4} \frac{\partial H_4}{\partial H_1} \frac{\partial H_1}{\partial W_1} \right)}_{(6)}$$

$$+ \underbrace{\left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial H_6} \frac{\partial H_6}{\partial H_4} \frac{\partial H_4}{\partial H_1} \frac{\partial H_1}{\partial W_1} \right)}_{(7)}$$

$$+ \underbrace{\left(\frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial H_6} \frac{\partial H_6}{\partial H_4} \frac{\partial H_4}{\partial H_1} \frac{\partial H_1}{\partial W_1} \right)}_{(8)}$$

- And again, we factor the common terms and re-write the equation below:

$$\frac{\partial e}{\partial W_1} = \left[\begin{aligned} & \left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial H_5} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial H_5} \right) \left(\frac{\partial H_5}{\partial H_4} \frac{\partial H_4}{\partial H_1} \right) \left(\frac{\partial H_1}{\partial W_1} \right) \\ & + \left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial H_6} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial H_6} \right) \left(\frac{\partial H_6}{\partial H_4} \frac{\partial H_4}{\partial H_1} \right) \left(\frac{\partial H_1}{\partial W_1} \right) \\ & + \left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial H_5} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial H_5} \right) \left(\frac{\partial H_5}{\partial H_4} \frac{\partial H_4}{\partial H_1} \right) \left(\frac{\partial H_1}{\partial W_1} \right) \\ & + \left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial H_6} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial H_6} \right) \left(\frac{\partial H_6}{\partial H_4} \frac{\partial H_4}{\partial H_1} \right) \left(\frac{\partial H_1}{\partial W_1} \right) \end{aligned} \right]$$

To efficiently program a structure, perhaps there exists some pattern where we can reuse the calculated partial derivatives.

REFCARDZ (/refcardz) **TREND REPORTS** (/trendreports) **WEBINARS** (/webinars) **ZONES** ▾

To illustrate the pattern, let's observe the total derivatives for W_1 , W_7 , W_{13} , and W_{19} in Figure 6 above. Note that these are not all the weights in the net but they are sufficient to make the point.

$$\frac{\partial e}{\partial W_1} = \left[\begin{aligned} & \left(\left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial A_1} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial A_1} \right) \left(\frac{\partial H_5}{\partial A_1} \right) \right) \left(\frac{\partial H_3}{\partial A_1} \right) \\ & + \left(\left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial A_1} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial A_1} \right) \left(\frac{\partial H_6}{\partial A_1} \right) \right) \left(\frac{\partial H_3}{\partial A_1} \right) \\ & + \left(\left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial A_1} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial A_1} \right) \left(\frac{\partial H_5}{\partial A_1} \right) \right) \left(\frac{\partial H_4}{\partial A_1} \right) \\ & + \left(\left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial A_1} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial A_1} \right) \left(\frac{\partial H_6}{\partial A_1} \right) \right) \left(\frac{\partial H_4}{\partial A_1} \right) \end{aligned} \right] \left(\frac{\partial H_1}{\partial W_1} \right)$$

Output Layer *Hidden Layer 3* *Hidden Layer 2* *Hidden Layer 1*

$$\frac{\partial e}{\partial W_7} = \left[\begin{aligned} & \left(\left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial A_1} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial A_1} \right) \left(\frac{\partial H_5}{\partial A_1} \right) \right) \left(\frac{\partial H_3}{\partial A_1} \right) \\ & + \left(\left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial A_1} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial A_1} \right) \left(\frac{\partial H_6}{\partial A_1} \right) \right) \left(\frac{\partial H_3}{\partial A_1} \right) \end{aligned} \right] \left(\frac{\partial H_3}{\partial W_7} \right)$$

Output Layer *Hidden Layer 3* *Hidden Layer 2*

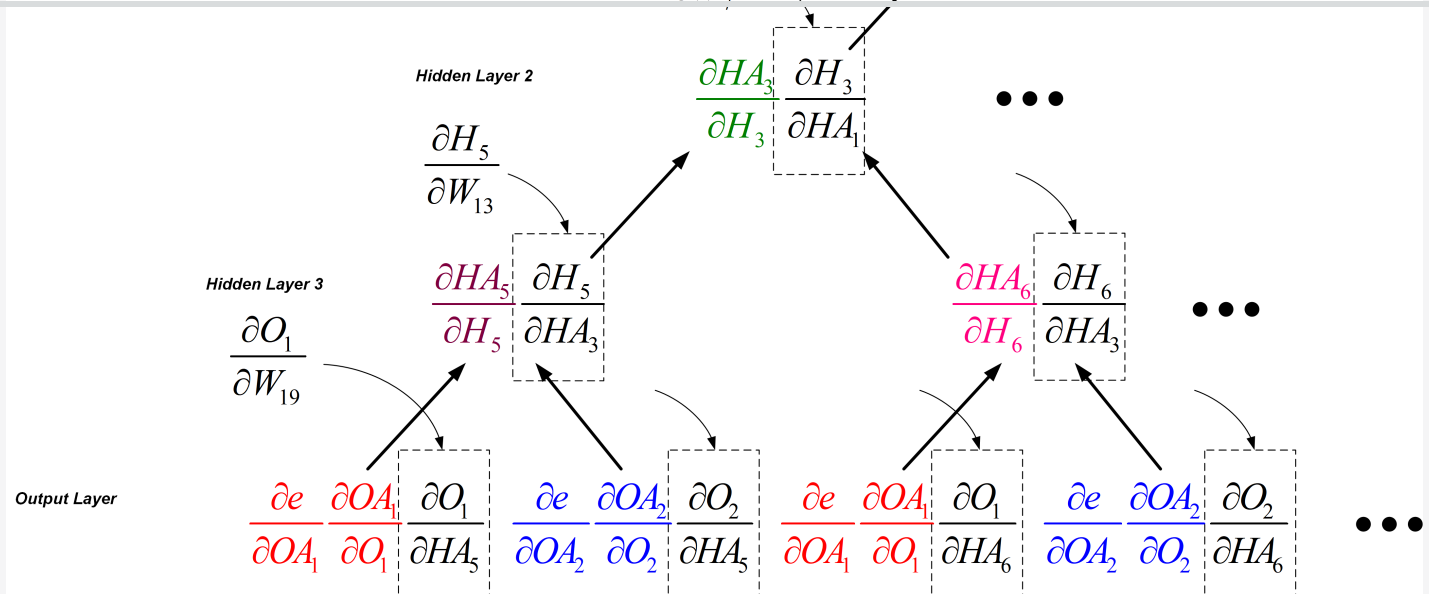
$$\frac{\partial e}{\partial W_{13}} = \left(\left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial A_1} + \frac{\partial e}{\partial O_2} \frac{\partial O_2}{\partial A_1} \right) \left(\frac{\partial H_5}{\partial A_1} \right) \right) \left(\frac{\partial H_5}{\partial W_{13}} \right)$$

Output Layer *Hidden layer 3*

$$\frac{\partial e}{\partial W_{19}} = \left(\frac{\partial e}{\partial O_1} \frac{\partial O_1}{\partial W_{19}} \right)$$

Output Layer

We can view the factored total derivatives for the specified weights in a tree-like form as shown below.



The rule to find the total derivative for a particular weight is to add the tree leaves in the same layer and multiply leaves up the branch.


For example, to find the total derivative for W_7 in Hidden Layer 2, we can replace (dH_3/dHA_1) with (dH_3/dW_{13}) and we obtain the correct formula. **Note:** We ignore the higher terms in Hidden Layer 1.

We can do the same for W_{13} , W_{19} , and all other weight derivatives in the network by adding the lower level leaves, multiplying up the branch, replacing the correct partial derivative, and ignoring the higher terms.

The advantage of this structure is that one can pre-calculate all the individual derivatives and then, use summation and multiplication as less expensive operations to train the neural network using backpropagation.

Hope this helps!

Neural Network Network Feed Forward (Control)

Published at DZone with permission of Edvin Beqari. [See the original article here.](#)  (Edvin Beqari)

Opinions expressed by DZone contributors are their own.

Popular on DZone

x

- [Maven Tutorial: Nice and Easy \[Video\] \(/articles/maven-nice-and-easy-video?fromrel=true\)](#)
- [Querying Kafka Topics Using Presto \(/articles/querying-kafka-topics-using-presto?fromrel=true\)](#)
- [How to Set Up and Run PostgreSQL Change Data Capture \(/articles/postgresql-change-data-capture?fromrel=true\)](#)

ABOUT US

[About DZone \(/pages/about\)](#)

[Send feedback \(mailto:support@dzzone.com\)](mailto:support@dzzone.com)

[Careers \(https://careers.dzzone.com/\)](https://careers.dzzone.com/)

[Sitemap \(/sitemap\)](#)

ADVERTISE

[Advertise with DZone \(https://advertise.dzzone.com\)](https://advertise.dzzone.com)

CONTRIBUTE ON DZONE

[Article Submission Guidelines \(/articles/dzzone-article-submission-guidelines\)](#)

[MVB Program \(/pages/mvb\)](#)

[Become a Contributor \(/pages/contribute\)](#)

[Visit the Writers' Zone \(/writers-zone\)](#)

LEGAL

[Terms of Service \(/pages/tos\)](#)

[Privacy Policy \(/pages/privacy\)](#)

CONTACT US





600 Park Offices Drive

Suite 300


Durham, NC 27709

[support@dzzone.com \(mailto:support@dzzone.com\)](mailto:support@dzzone.com)

+1 (919) 678-0300 (tel:+19196780300)

Let's be friends:    

[\(/pages/help\) \(/pages/help\) \(/pages/help\) \(/pages/help\) \(/pages/help\) \(/pages/help\) \(/pages/help\) \(/pages/help\) \(/pages/help\) \(/pages/help\)](#)




DZone

DZone.com is powered by

AnswerHub

A DEVADA SOFTWARE PRODUCT



[\(/users/login.html\)](/users/login.html)



[\(/search\)](/search)

[REFCARDZ \(/refcardz\)](/refcardz)

[TREND REPORTS \(/trendreports\)](/trendreports)

[WEBINARS \(/webinars\)](/webinars)

[ZONES](#) 