**1.Artificial neural network from wikipedia**

[**https://en.wikipedia.org/wiki/Artificial\_neural\_network**](https://en.wikipedia.org/wiki/Artificial_neural_network)

**2.Chapter 10. Neural Networks of *the nature of code***

[**http://natureofcode.com/book/chapter-10-neural-networks/**](http://natureofcode.com/book/chapter-10-neural-networks/)

**3.**[**Neural Networks and Deep Learning**](http://neuralnetworksanddeeplearning.com/index.html)

[**http://neuralnetworksanddeeplearning.com/index.html**](http://neuralnetworksanddeeplearning.com/index.html)

**4.Neural Net in C++ Tutorial**

[**https://vimeo.com/19569529**](https://vimeo.com/19569529)

**5.Neural Net in C++ Tutorial on Vimeo**

[**https://www.youtube.com/watch?v=KkwX7FkLfug**](https://www.youtube.com/watch?v=KkwX7FkLfug)

**6.AI : Neural Network for beginners**

[**http://www.codeproject.com/Articles/16419/AI-Neural-Network-for-beginners-Part-of**](http://www.codeproject.com/Articles/16419/AI-Neural-Network-for-beginners-Part-of)

**private void train\_network(double[] target)  
{  
 *//get momentum values (delta values from last pass)* double[] delta\_hidden = new double[nn.NumberOfHidden + 1];  
 double[] delta\_outputs = new double[nn.NumberOfOutputs];  
  
 *// Get the delta value for the output layer* for (int i = 0; i < nn.NumberOfOutputs; i++)  
 {  
 delta\_outputs[i] =  
 nn.Outputs[i] \* (1.0 - nn.Outputs[i]) \* (target[i] - nn.Outputs[i]);  
 }  
 *// Get the delta value for the hidden layer* for (int i = 0; i < nn.NumberOfHidden + 1; i++)  
 {  
 double error = 0.0;  
 for (int j = 0; j < nn.NumberOfOutputs; j++)  
 {  
 error += nn.HiddenToOutputWeights[i, j] \* delta\_outputs[j];  
 }  
 delta\_hidden[i] = nn.Hidden[i] \* (1.0 - nn.Hidden[i]) \* error;  
 }  
 *// Now update the weights between hidden & output layer* for (int i = 0; i < nn.NumberOfOutputs; i++)  
 {  
 for (int j = 0; j < nn.NumberOfHidden + 1; j++)  
 {  
 *//use momentum (delta values from last pass),* *//to ensure moved in correct direction* nn.HiddenToOutputWeights[j, i] += nn.LearningRate \* delta\_outputs[i] \* nn.Hidden[j];  
 }  
 }  
 *// Now update the weights between input & hidden layer* for (int i = 0; i < nn.NumberOfHidden; i++)  
 {  
 for (int j = 0; j < nn.NumberOfInputs + 1; j++)  
 {  
 *//use momentum (delta values from last pass),* *//to ensure moved in correct direction* nn.InputToHiddenWeights[j, i] += nn.LearningRate \* delta\_hidden[i] \* nn.Inputs[j];  
 }  
 }  
}**