***NEURAL NETWORKS***

Contact email: [onedragutin@gmail.com](mailto:onedragutin@gmail.com)

Forum post: <https://forum.unity3d.com/threads/neural-networks-for-unity.470754/>

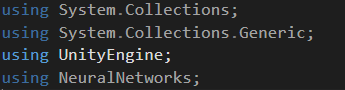
Youtube channel: <https://www.youtube.com/channel/UC9qxDA0szpbPZ1dJgKwg5fA>

MNIST Image database: <http://yann.lecun.com/exdb/mnist/>

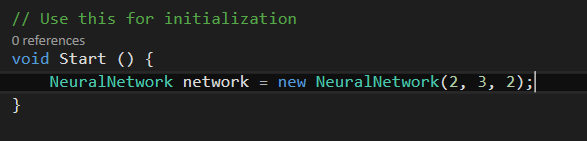
***HOW TO USE:***

After reading this section you should learn the basics about how to create your own neural network then train it and ask the network for predictions based on learned data.

We will start by creating a new script. In order to create neural network we will use script that come with this package, but first we need to include namespace called **NeuralNetworks** which will allow us to reference these scripts like so:



Main class which can now be used to create neural network is called **NeuralNetwork**,butbefore we can create instance of this class we need to tell how much neurons each layer will have. There is limited number of layers, only 3. First layer is for the input, second layer is not directly changeable (this layer is called hidden layer) and there is last layer, third layer, which is output layer and this layer of neurons serve as output of the network.

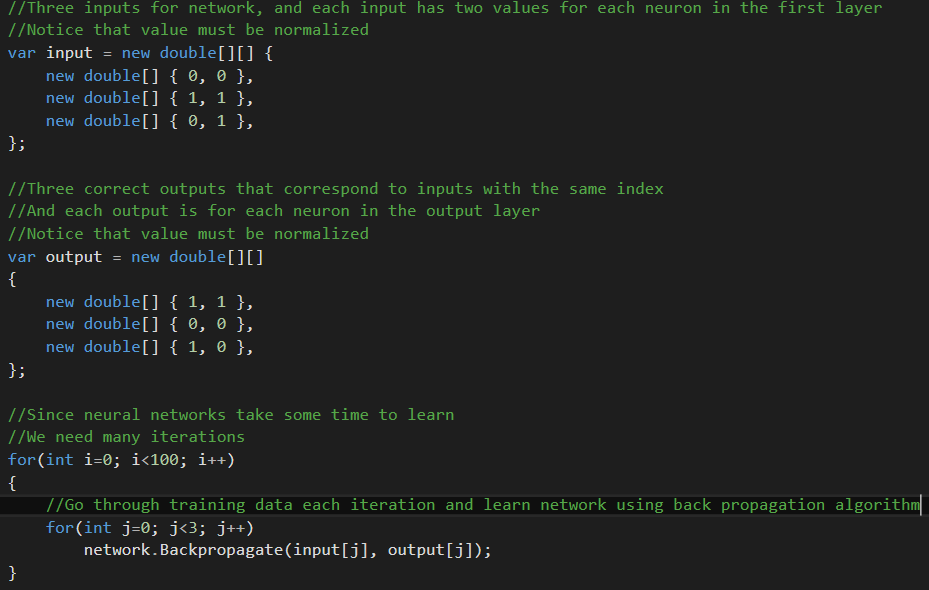


As you can see my network has two input neurons, three hidden neurons and two output neurons.

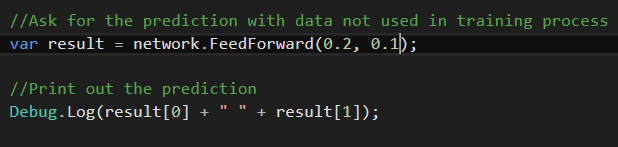
This network does not know anything since we haven’t yet trained it. Depending on the training data and the possibly outcome you may want to change size of the each layer.

Now let’s create some training data which we will use to train network. In my case I’ll use just three training data with very basic logic for network to learn. Logic is: if the input is zero output should be 1 and vice-versa. When training you must tell the network which input produces which output.

Now we are ready to use backpropagate method which is part of the neural network class to train the network. Backpropagate method requires input and output normalized values which must be the same size of the input and output layer. Here is the code:



Now it is the time when we should test if network has learned anything. This time we will ask our network for a prediction with input data which was not included before. That is: 0.2 and 0.1



Network prediction will be printed on screen. After running the code this is the result I got:



As you can see network Is not 100% certain. But it thinks that value for both output neurons is around 1. Which is correct, at least that’s what we learned it to think.

Here are some facts:

More neurons in layers = slower to learn, learns more complex stuff

More training data = better prediction, slower to learn

***Setting up:***

You can check out two demo scenes. First one is called NumberRecognition and the other one is called XOR. Scenes are located in /NeuralNetworks/Demos/ folder.

***Number recognition demo:***

This demo shows how to use neural network to recognize digits. It is using 28\*28 input neurons for each pixel, 30 hidden neurons and 10 output neurons indicating digits from zero up to ten.

Once you start demo you will be able to see that network prediction is bad, this is because network needs to be trained. In order to train network you must have data for training, in this case we need images of different digits.

We can use MNIST database of images, you can find link at the top of the document where you can download images and their labels. There is already a class for loading these images inside of unity you just need to change file path in NumberRecognition.cs

To start training process click train button, training can take up to 1 hour or longer depending on how well you want to train network.

Once the network is trained you can save it so in the future you can just load already trained network.

***XOR demo:***

This is most basic demo and perhaps you should start learning neural networks with this demo. You can change 4 train data. Each train data contains A and B values which should be normalized values between zero and one, this can be changed once you start demo.

Also each train data contains target value, so for values A and B there is always some TARGET which network should learn.

Later when you train it you can test it with custom input data, this time without giving network TARGET value, since this is what the network should tell us based on what it learned.

By default network training data is:

A B OUT

1 1 1

0 1 0

1 0 0

0 0 1

So after some training network should adapt to give correct answer.

***In future I plan to add more demos and tutorials so make sure to contact me if you have any ideas or questions. Thank you!***