1. What a Neural Network is and what it can do

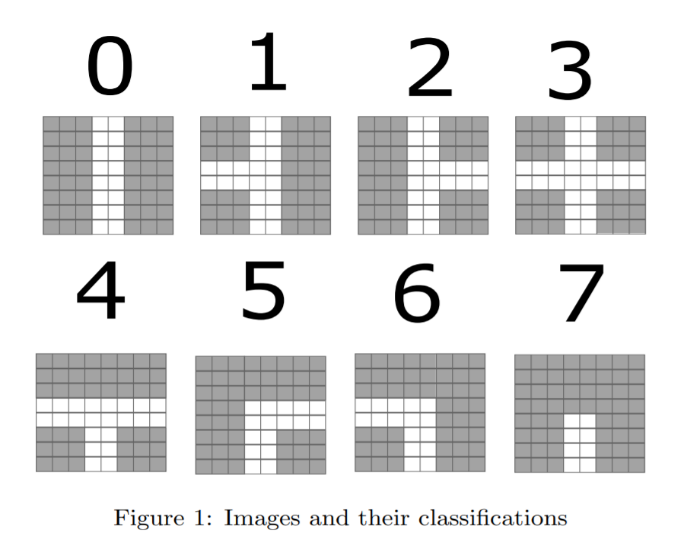
A neural network is a form of artificial intelligence (AI) that ties closely the workings of a brain. The network consists of neurons that are aligned in layers that communicate with each other. The neural network that this manual cover consists one two layers, one hidden layer and one output layer. The hidden layer consists of 16 neurons, while the output layer has 8 neurons. The output layer is bounded by the amount of possible outputs the network has. A neuron network is different that of a simple AI. Instead of programmed like a conventional AI is a neural network is trained through the process of providing training data that it can alter itself until it provides the correct outputs for this training data. After sufficient training is complete you can then feed data into the network and it will determine (based off its training) what the input is. A neural network can do amazing things such as pattern identification, letter identification, image identification, and all sorts of other things provided you can supply enough training data to the network itself so it can learn.

1. What your program can do

This program consists of a rather simple neural network that simply (by default) takes in 8x8 patterns and will identify what kind of pattern it is based off 8 static patterns.

1. Which input data your program can take

This neural work can take 64 inputs (changeable by request) of integers that can be identified into 8 different categories. These categories need to be defined by the index of the input data output target which is -10 is black and + 10 is white. In the default case the input data is every single cell in an 8 \* 8 grid for one of the base Figure 1.



4. What should be changed in your code if input data are different from what

you used (number of inputs, number of outputs changed)

Let’s say this input pattern is base number 4. This means the target array would be {0, 0, 0, 0, 1, 0, 0, 0}, see how index 4 is the only number one? The working dataset needs to be the same, minus the target output array. This is the default configuration. If you would like to increase the number of inputs the neural network can take, or the number of outputs is possible you would need to change the variables nInputs and nOutputs respectively to achieve this. The program should take care of the rest. If your input dataset is of a different size to the default 750 you should change the nPatterns variable to the correct number. To achieve good results from this program you should provide an enough and quality training data to provide the network with training. If you find that while you have done the above but still fail to see results you may have to change specific values inside of the program such as the InitNet values, the number of hidden neurons (nHidden variable) or the learning rate . The key to good results is patience and a lot of fine tuning to certain values.

5. What the customer should do to achieve good results from the Neural

network (all the things you changed when trying to achieve convergence)

The network consistently scores 100% detection rate on the given training dataset. The program first begins by loading the training dataset and the working dataset, along with initializing all beginning biases and neuron weights to a random value between -0.75 and 0.75. The network takes 64 inputs (one for each tile of the input patterns), has 16 hidden neurons and has 8 output neurons (one for each possible tile). Upon training initialization, the program will select a random row from the training dataset and tune the weighted values and biases in the network based off the direction the total network error value is going. This is done through the use of the backpropagation algorithm which calculates the derivative of each weighted value and bias which is then passed onto another function which then changes the values based off the derivative which changes the learning rate (to ensure we don’t overshoot and get to as close to the lowest error value as possible). The program then calculates the total error average across the entire training data, this is done by first cycling through each pattern in the training data, calculating the output/error for every neuron in the output layer. This is then all added together and divided by the number of patterns to get the average error across all training inputs. We then repeat these 5,000 times unless the total average error ever falls below 10.0, where we break the loop. After that we run through the training dataset one last time and for each row we calculate if the network outputted the correct answer and then form a detection rate based off how many the network got correct. The checking of the working dataset is trivial, it simply just loads each row and calculates the output which the user will then have to determine if the network is correct or not, or whatever system this network may be implemented into the initial values before training. These values were concluded through trial and error and fine tuning. The learning rate and the random values of minimum and maximum was simply played around with until an acceptable consistency was found in the detection rate.

6. You can make program interface more user-friendly

1.Make the training dataset (750 patterns based off base patterns) that is used to train the neural network and reports back to the user with the detection rate after training.

2. Make the work dataset (750 patterns based off the base patterns) and reports the outputs back to the user.

3. user provide the training dataset file and used it to train the neural network and then reports back to the user with the detection rate of the dataset based off the training.

4. user provided working dataset that returns to the user the output of the neural network.