**Requirements Documents**

**Project Name: Brain-ish**

**Name: Cody Harkins**

**Description:**

**An Artificial Neural Network C++ library. Each instance of the Neural Network class will be able to learn with user provided data by changing the weights between nodes during the training phase (may use recursive derivative function). After training the instance should be able to take in a similar group of inputs and provide some accurate output(s). I would also like it to include many different types of Artificial Neural Network structures.**

**Purpose:**

**I want to make an easy to use Artificial Neural Network library that is useful, not just a simple 3 layer Neural Network. It should be able to be used for many projects along the lines of classifying (sorting into groups), incrementing (math), and deciding (which option is better).**

**End Users:**

**Artificial Neural Network Hobbyists, Machine Learning Engineers, and AAA game makers (stretch).**

**Scope:**

**A many style Neural Network library. Each node connecting to other nodes, but each style with its own connection organization.**

**The end result of this project is not a specific task for an Artificial Neural Network, although I could present some simple tasks to show that it works.**

**Stretch goal: A web page GUI to create an Artificial Neural Network and train and test it with user or provided data. Memory nodes, directed paths.**

**Functionality/Scope:**

**C++, windows, downloadable from web, maybe backend of web page.**

**Many constructors for creating an Artificial Neural Network instance.**

**Two training methods (at least): one that splits up user provided data for user and trains it, and the other takes in data from user and trains with all of it. Connections (coefficients) between nodes change value in order to get closer to correct answers (learn). Get correct answers from different data not trained with, but in the same style.**

**It is not:**

**A human, an evil learning machine, going to rule the world as a dictator.**

**It might not:**

**Think about things without input.**

**Use Cases:**

**Happy Case (Pt1):**

* **A Neural Network hobbyist finds the zip download online and downloads it.**
* **Inside he finds a short readme document, the code files and the executable Artificial Neural Network file.**
  + **He looks at the readme which tells him about the file reading and creation of a neural network. He then separates input data and output data.**

**Happy Case (Pt2):**

* **He runs the executable, it runs in a terminal. It asks for a list of numbers separated by spaces first the # of layers followed by the layer sizes to create the Artificial Neural Network. He types in a short list “4 783 16 16 10{ENTER}”.**
* **The Program asks, yes/no, if he has split up training and testing data.**
  + **Answer: No.**
  + **It asks for a file path to the input data: “../AI#input.csv{ENTER}”.**
  + **It asks for a path to the output data file: “../expected#output.csv{ENTER}”.**

**Happy Case (Pt3):**

* **It asks if it should train till it gets the right answers: “yes{ENTER}”.**
  + **The program passes in a set from the AI#input.csv as start values for the 783 start nodes, they multiply by the weights to the next layer and propagate till the end.**
    - **It first would return garbage values, but as it changes the weights the between nodes it will return values that are not so garbage.**
  + **The program trains the network for a while, finally printing the iteration count 1,657,032 and then the test inputs along with their outputs. The user sees the answers are slightly off, but acceptable.**

**Less happy Case (Pt1):**

* **Charlie, a machine learning engineer, finds the download. He usually uses python and doesn’t have a C++ Integrated Development Environment.**
* **He needs to get an IDE or relearn how to compile on the command line.** 
  + **He goes with an IDE, it’s a big download and install so that takes about twenty-five minutes.**
* **He opens up the project which goes fine and runs it as-is. It starts creates an Artificial Neural Network, trains it with hardcoded data and tests it with other hardcoded data to show that it works.**
* **Then Charlie starts trying to modify the program to take in a file and train on that, but forgets to change the shape of the Neural Network to account for the input output number change.**
  + **It prints the reason that it crashes “Input/Output features don’t match the shape of the Artificial Neural Network.”**

**Less Happy Case (Pt2):**

* **He fixes the network shape and runs the program, but it doesn’t seem to be ending.** 
  + **This is because it is set to run till it gets the right answers and the input data he provided is difficult to learn on.**
* **So he stops it to debug and finds the other training function that only trains a certain number of loops, and makes a small change to call that training function.**
* **It runs this time printing after every 100,000 loops to print the network error and it stops at the given 3,000,000 loops.**
  + **Then runs the test part that prints the inputs and the outputs the network makes.**
  + **Charlie sees that the network is getting some correct and is completely wrong on other answers.**

**Less Happy Case (Pt3):**

* **He starts taking his time looking through the code to try and understand my ideas for the Artificial Neural Network.**