Super Mario Neural Network input design

in super mario bro's they array with the screen tiles is stored in the following ram addresses:

0x0500-0x069F

This is a total of 415 memory addresses. If we were to take all these memory addresses as inputs so one input node for each address we would have 415 inputs and if each input is mapped to each output (Outputs on the network will be the six buttons on the NES controller) we would have six weights per input with would amount to 2490 weights, this is without any hidden layers on the network.

With all 415 memory addresses only some are going to have a value as most of the space on the super mario screen is empty space. When we are playing mario as a human we generaly dont look to far ahead of where mario is going, we only look at what is an immediate treat to mario ie. A enemy withen 4 tiles infront or a hole in the floor infront. Because of this we can say our network only needs to know about a space of four tiles in all directions this reduces the number of inputs significantly.

When taking in the inputs for pre processing we look to the tile that contains Mario (the ram addresses with a value representing mario) and then only select tiles in a four tile radius around him.

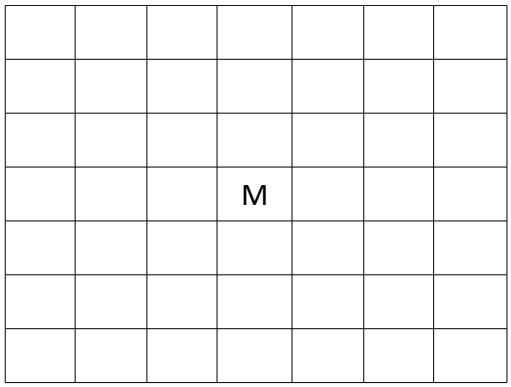
				M				

M = mario's position on screen (mario will always be in the center)

So now have a 9x9 grid with a total number of inputs = 81
with six weights each we have a total number of weights exculding any hidden layer = 486

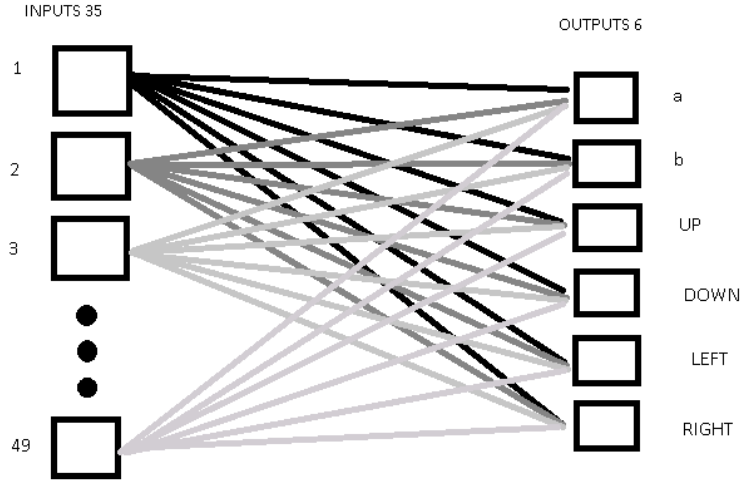
Still a lot of weights!

So lets only look at a radius of 3



7 x 7 = 49 inputs
49 * 6 = 294 weights
I think this is a much better number for the inputs!

Diagram of how the network would look including its weights :



Thersholds for each connection will just be a extra weight on each input so the total number of weights would actually be:
49 * 7 = 343 weights

INPUT VALUES

When reading the values of the ram addresses convert the value to a better number to work with. The values of our inputs would be 0,1,2,3 where:
0 = empty space
1 = mario
2 = statonary object (Ground,Platform)
3 = moving object (enemys, moving platforms)

When we read in the actual ram values convert it to one of the above depending on the value. For example a ram address with the value 81 in binary equals a tile of a breakable block this is statonary object so asign it the value of 2.

NOTE:
The center of the array will always be 1 representing mario as our array is all tiles in a 3 tile radius of this marios tile. (would this be ok?!)

Learning weights!

For learning weights we will use a genetic algorithm with its fitness function being how far left mario has moved over how fast he it is taking him

POTENTAL PROBLEM: how do we pass this value to the algorithm do we have to include this as a input in the network or can this be a seprate input outside of the network??

TODO: NEED TO READ OVER GENETIC ALGORITHMS!!! SEE MARKS NOTES....

Building the hidden layers:

At the begin of the networks runs there will be no hidden nodes/layers.
From marks notes: Design a function to measure interference(more then likely there is already a algorithm out there to do so)
Everytime with our genetic algorithm when we generate a new species we measure the interference of the previous specie and if the interfecene is high increase the number of hidden units(this will be a random value greater the number of hidden units in the previous specie) or if the interference is to low reduce the number of hidden units(again this value is a random number less then the number of units in previous species)