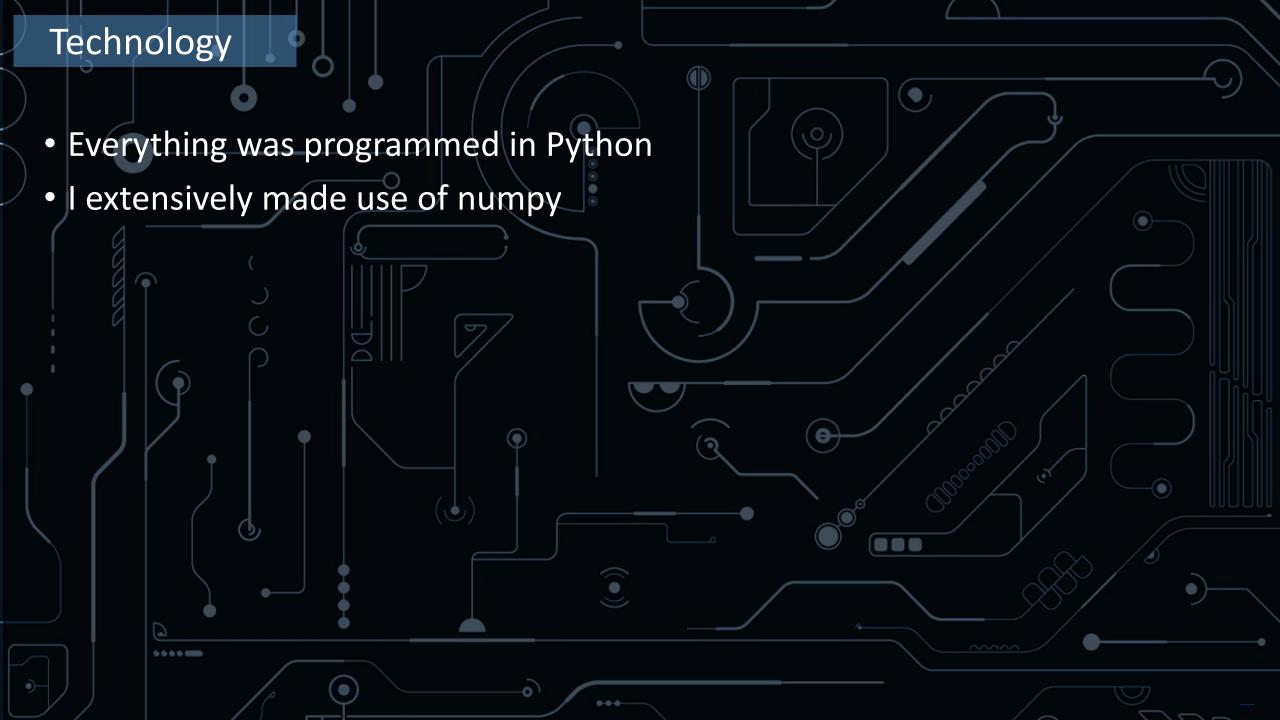


Game Problems

- Games provide fascinating problems for computers to solve
- They can produce intractably large game search spaces so more clever ways of approximating the right solution need to be found
- An issue with many machine learning algorithms is that for even the slightest domain change (i.e. size) training has to be redone

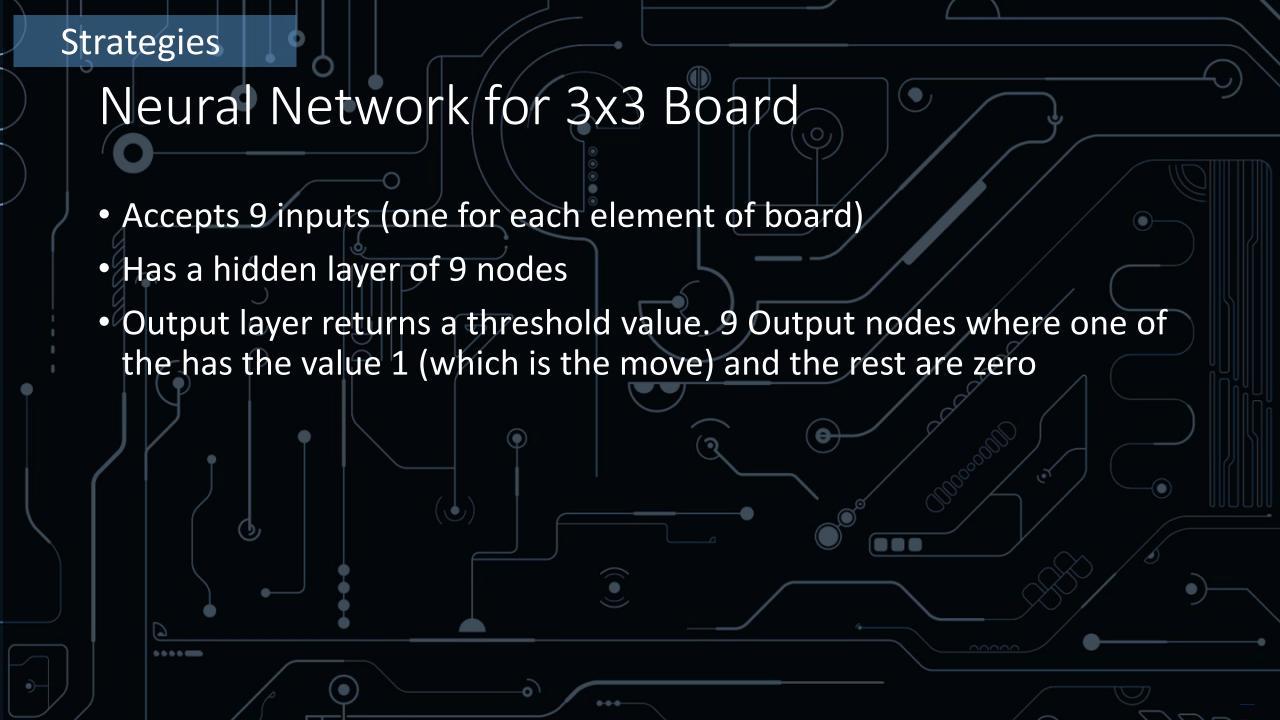


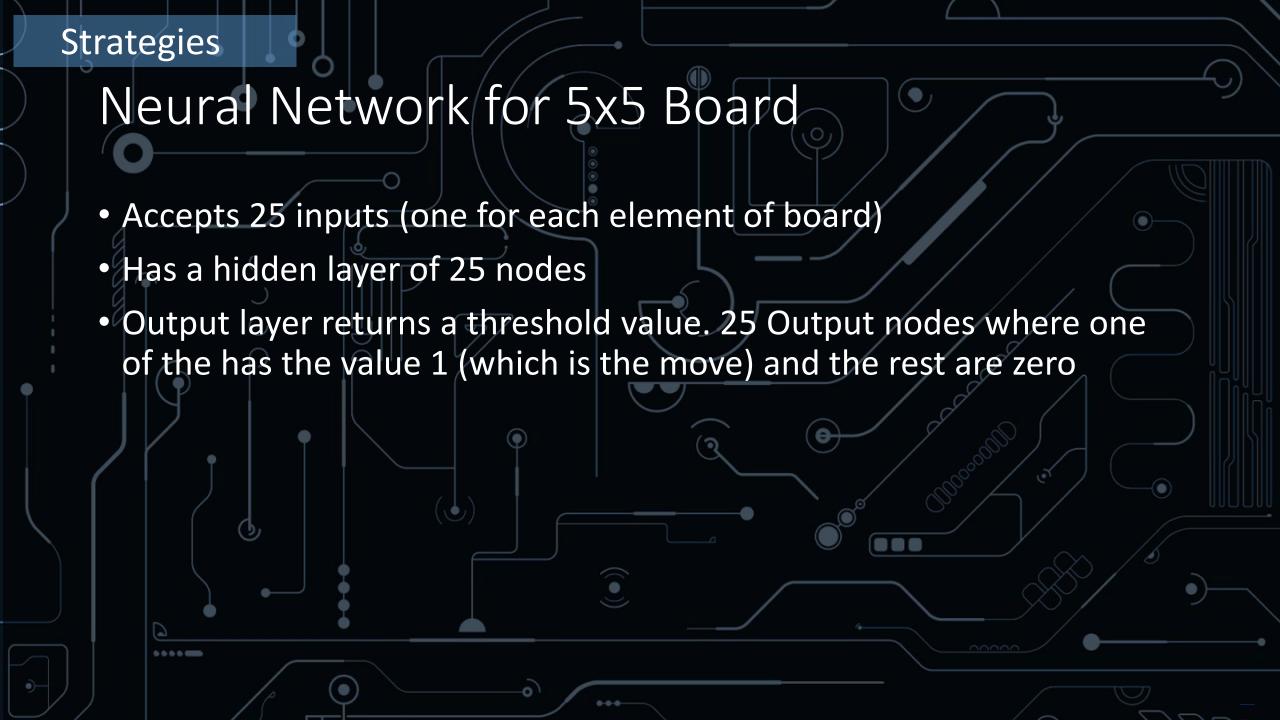












Strategies

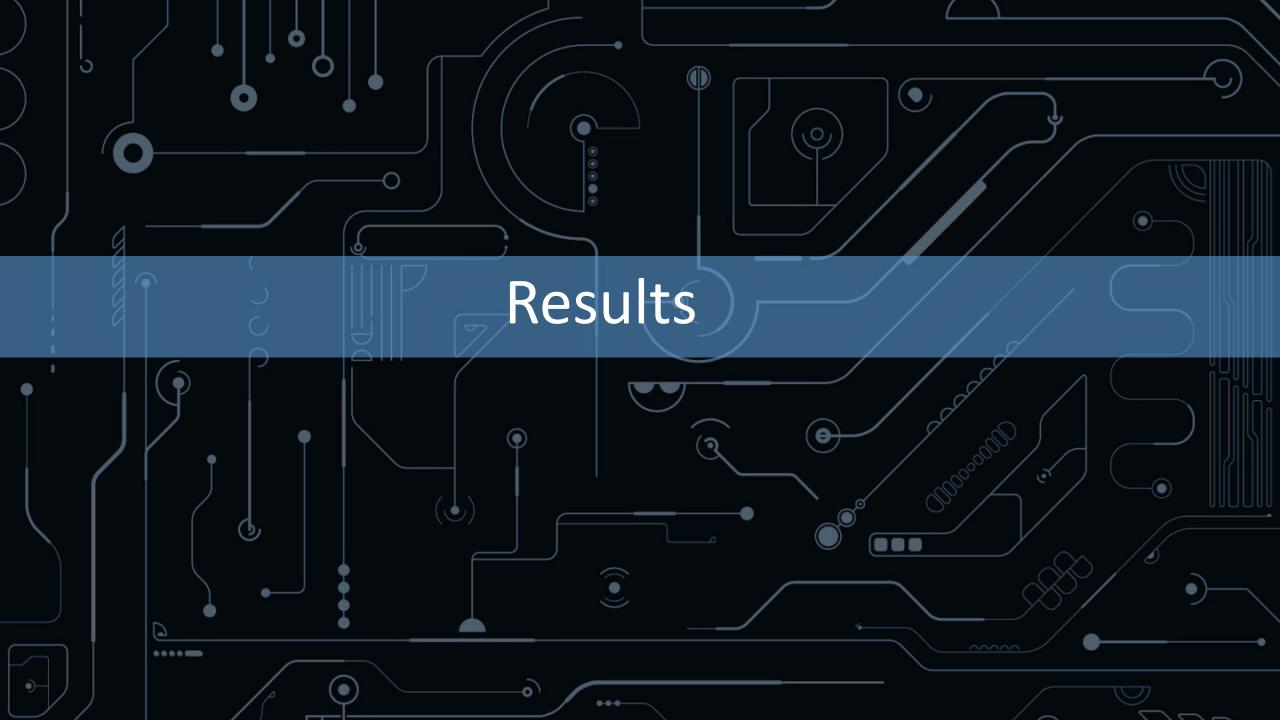
Attempt to make an Al which can play any sized board

- I attempted to use principal components analysis to make any sized board being played have the same inputs and output for a neural network.
- I failed because PCA does not lend itself to scaling up components.
 These approximations mean nothing. Also the dataset generated was not very good because I relied on random games for data. There was not enough control.

Strategies

Attempt to make an Al which can play any sized board

- I moved on to trying to use an auto-encoder to produce an intermediary set of inputs for a general neural network agent.
- Here each board would need its own auto encoder
- I failed again because I was trying to use the same bad dataset I had generated. I should have generated a new dataset which was just every state the board can be in. I could then train the auto-encoder better. I do believe this method will still produce terrible results because the accuracy of the compression was 70% which was not good enough.



Results

3x3 Neural Network

- Over an averaged set of games versus the random agent the network performed terribly.
- It lost 59% of the time
- Drew 40% of the time
- And won only 1% of the time
- Two reasons this happened:
- The dataset used was actually not good as it generated based off random games where the move may have been bad but that player still won.
- The network structure was not complex enough to approximate the optimal move function

Results

5x5 Neural Network

- Over an averaged set of games versus the random agent the network performed terribly.
- It lost 65% of the time
- Drew 34% of the time
- And won only 1% of the time
- Two reasons this happened:
- The dataset used was actually not good as it generated based off random games where the move may have been bad but that player still won.
- The network structure was not complex enough to approximate the optimal move function – the 5x5 board probably needs eve more data and a larger network than the 3x3 due to have much more decision states.

