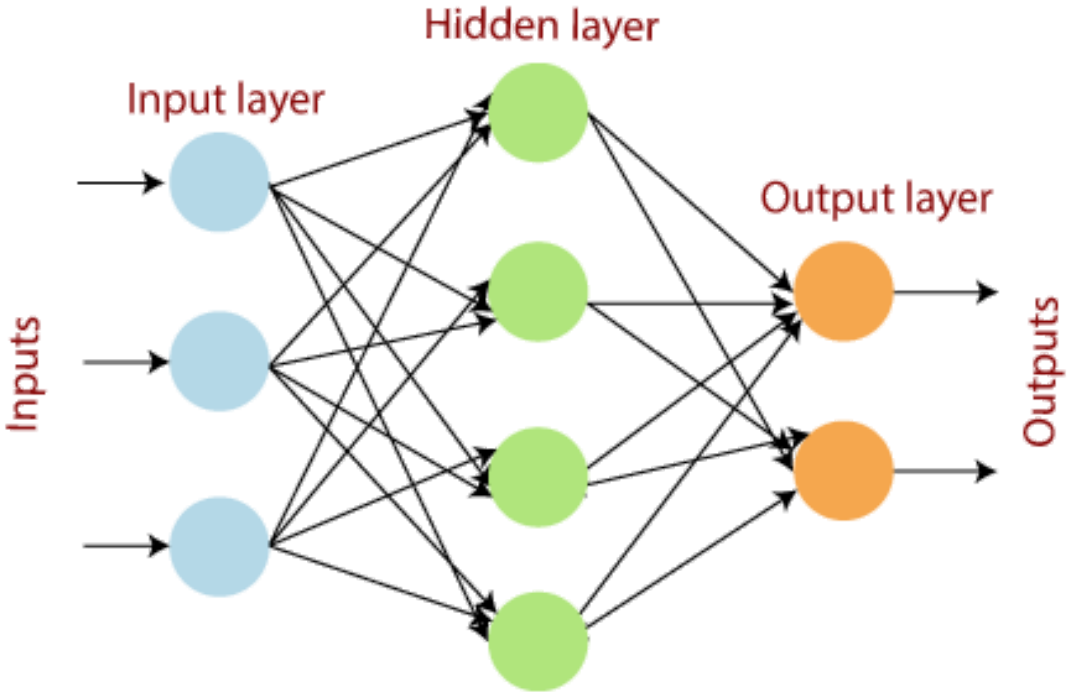
This is not comprehensive. This is not clean. This may even be misleading. I'm just getting my thoughts out. Just a heads up.

I decided a while back to create AI.

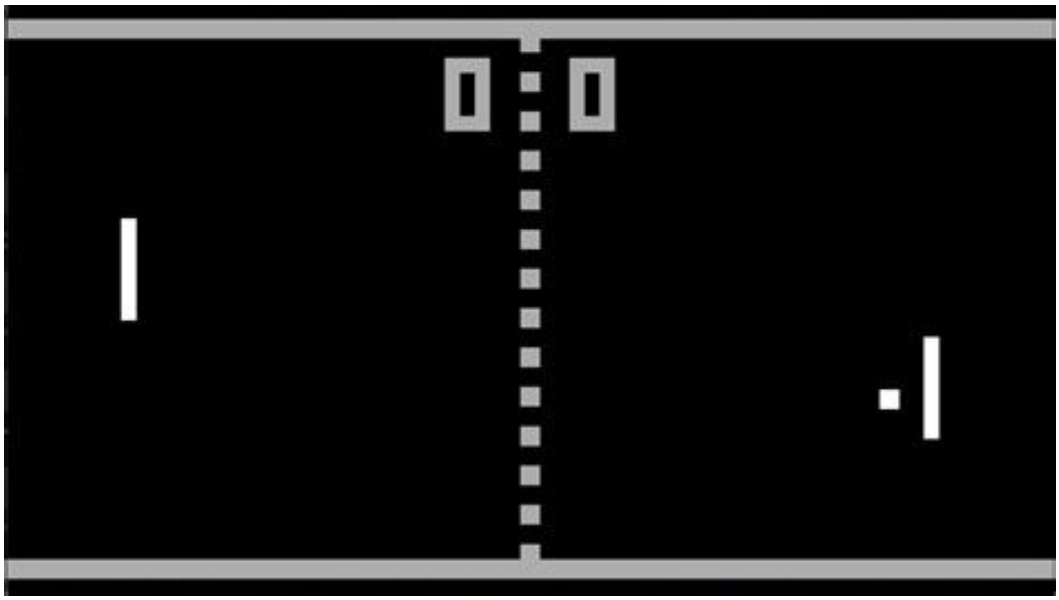
Fortunately, AI is not complicated. The simple ones at least certainly aren't. The kind I'm making is literally just a bunch of math functions composed into each other in a way that makes it possible to adjust each of them to turn the whole into literally any function you want. You can think of it as a neural network with information flowing from inputs through various simple intermediate steps to output.



Each of those dots, lets call them neurons, is a single function, like maybe a simple line, but it's composed with so many other similar lines that it can be used to model any function we like. Well, I can try.

If I were a sane person, there are places I can go to get premade AI to train for whatever I want. Unfortunately, I don't like other people or their work (generally), so I'm just going to do this the hard way. This will mean a lower quality AI, but it'll be mine, inside and out.

Being able to model any math function we'd like using those smaller units is essential to how AI works.



This is pong. To play pong, the AI needs to decide what to do at each moment in time, but all it can do is output numbers. However, you control a paddle by either moving up, moving down, or holding still, so you could just represent this with a single number between -1 and 1:

```
   1 > output >  0.5   |- GO UP
 0.5 > output > -0.5   |- STAY STILL
-0.5 > output > -1     |- GO DOWN
```

You'd just need to program the AI's environment so that having a number in any of these ranges as the result of its output neuron will cause it to take the corresponding action. Actually, we could take it a step further and map the output to something that'll allow it to modulate its speed by selecting different values from -1 to 1, with negative values going down and positive values going up. I'm keeping it simple for now though, so ignore that last sentence if it's confusing at all.

Then there's the matter of getting the AI to "see" what's going on. You as a person have eyes, presumably, and if you were playing Pong, your eyes would detect the colors of the pixels and pass that along to your brain. We can do a little better for the AI though. It only really needs to know a few things: Where its paddle is on the screen, where the ball is on the screen, and

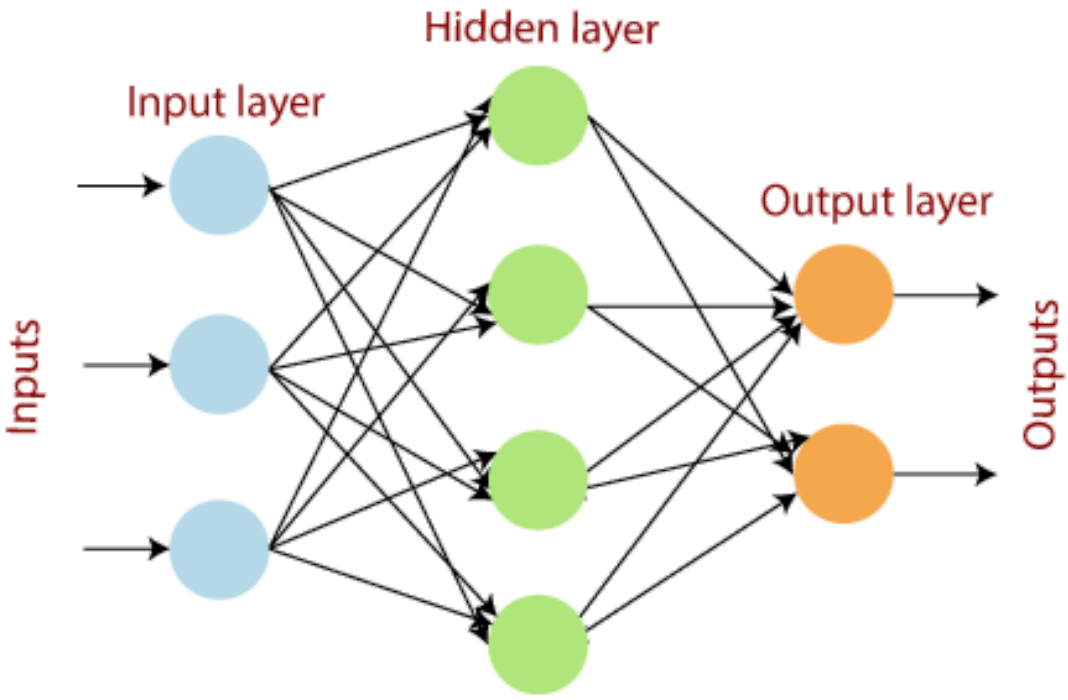
which direction the ball is moving. It doesn't need to know where the other paddle is, because only one of two things can happen when the ball reaches the other side: Either the other player hits the ball, and the other side of the game might as well be a solid wall, or the other player misses, and the AI doesn't need to do anything anymore. In either case, the AI might as well treat the ball as if it's going to come back.

So the inputs are:

```
(paddle x, paddle y)
(ball x, ball y)
(ball speed x, ball speed y)
```

This means 6 different input neurons will be needed.

There's still the issue of that junk in between the input and output neurons:



That in-between stuff, the hidden layer(s), are where the magic happens. Try thinking of that whole middle part as a single black box, a single function that takes the input neurons as inputs, and outputs an output that will move the paddle according to those inputs. Information passes from the inputs to the middle part, through the middle part where it's transformed, and off to the outputs. The black box looks at the state of the game, somehow transforms that information from the game in a way that produces an output. It looks, thinks, then acts.

This black box, this magical function is why we needed to be able to compose together simple functions into literally anything else. As it turns out, we can create that middle part by running the values of the inputs through a series of simple functions composed into each other.

Now this still feels pretty futile. We don't really know what exactly those middle functions are supposed to be, so we guess. We make them all the same type of function, like maybe a line, but we give them all different slopes so that each neuron can provide a slightly different transformation.

We have two problems here.

Firstly, lines actually really suck for the function that a neuron represents, because you can't actually get anything other than a line by composing them. We'll use a modified sigmoid function, which, keep in mind, is literally just a base function for the neurons to represent.

Next problem: the neuron functions are basically random right now. If we stuck the AI into a game of Pong, we have no idea what it'll do. It doesn't even know if it's doing well.

It was at this point that I got bored. Training the AI so that it isn't a mess of garbled nonsense is a bit tricky, and it involves this thing called backpropagation. Backpropagation is a complicated bit of math that I understand in theory but would probably have a hell of a time programming.

So, I decided to keep things simple by cutting out the most complicated part: the training. I'm lazy, so I'm going to take some advice from nature and just let the behavior evolve on its own. I think that's a thing that works anyway. A large fraction of the world's population would disagree with me on that, but whatever, they're stuck manually intelligently designing their AI.

Evolution in real life takes millions of years. Here though, we only have a couple of variables, and computers are fast, so it shouldn't take that long. What we could do for this scenario is randomly create 100 Pong AI networks, with all of their middle functions completely randomized, and just let them play against each other. When it's clear that one is losing, we'll take him out of the game. Once we're down to 50 AI players, we'll run another round with those 50. We'll continue doing this until we're down to 2 AI players. These winning AI players will each have their neural networks copied into 49 new players, except each player will have a slightly randomly modified version of that neural network. We'll then have 100 players whose neural networks are based on the previous winners.

It will be largely luck in the beginning. Maybe the first 2 winners won't really do much but sit there and wait. All it takes though for there to be an improvement though, is one of them needs to just get lucky enough to hit the ball back. Once that happens, they'll be more likely to make it into the top 2. If the top 2 players are all lucky ball hitters, all of the new players will be based on lucky ball hitters. From there, new strategies will be randomly found, like following the

ball, and the successful strategies will win and be passed onto the next round of players. It's survival of the fittest.

After a while, they'll be serious pong players developed out of the sheer fact that only the best survived. If another AI randomly develops a better strategy, it'll rise to the top and then it will be the new standard for AI Pong players.

* *

Pong is just an example. The actual thing I'm planning to program for AI to deal with is slightly simpler and slightly more interesting, in my opinion.