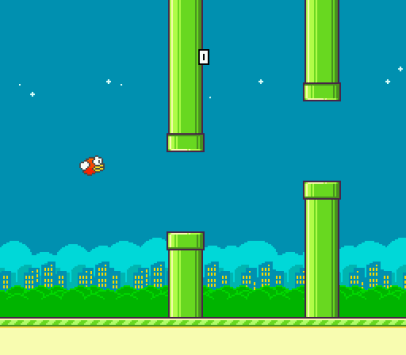
A.I. learns to play Flappy-Bird

A Neural Network Implementation with training via the backpropagation algorithm.

**The game environment:**

What are the important features?



**Rules**:

Flappy must avoid the pipes

Flappy must stay on the screen (can’t go off the top or bottom)

What are the important environment attributes/conditions?

* The position of the gap in the upcoming pipe
  + X and Y of bottom of upper pipe
  + X and Y of top of lower pipe
* The position of Flappy
  + X and Y of Flappy

Within each game loop Flappy must decide whether to flap or not.

So Flappy must have a Brain (Artificial Neural Network) which must decide to flap or not to flap.

So what should the architecture of our Brain (ANN) look like?

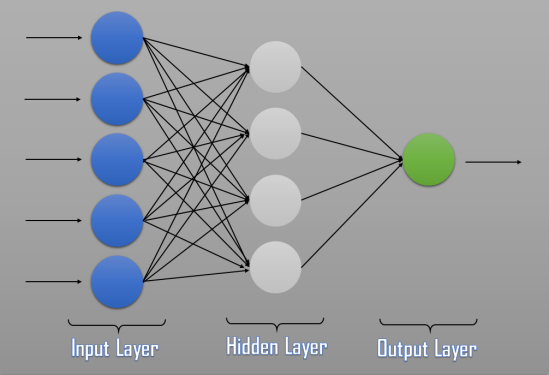
A 3 layer Feed Forward Neural Network.

First or Input Layer has 5 neurons

Second or hidden layer has 4 neurons

Third or output layer has 1 neuron.

If the output from the last neuron is of value greater than 0.5, Flappy jumps/flaps/moves up, and if less than 0.5, it goes down (due to gravity).



3-Layer Neural Network

The 5 inputs to the Neural Network are:

1. X – coordinate of the front – most “pillar”
2. Y – Coordinate of the lower part of the Upper “Pillar”
3. Y – Coordinate of the upper part of the Lower “Pillar”
4. X – Coordinate of the Bird
5. y – Coordinate of the Bird

**Activation Functions**:

For the first Layer of the Neural Network, the activation function is the ReLU (Rectified Linear Unit)

ReLu(x) = max(0, x)

For the second activation function we will use Sigmoid.

Another possibility would be the Softmax function, a wonderful activation function that turns numbers into probabilities that sum to one (normalisation).

Now we will be able to perform a Feedforward though the network.

So we need:

* **A Brain**
  + What attributes should it have?
    - The neural network structure (layers and neurons)
  + What methods should it have?
    - Feedforward
    - ReLu
    - Softmax (I don’t actually use this though)
    - Sigmoid
* Individual **Birds**
  + What attributes should a Bird have?
    - Its own Brain (ANN)
  + What methods should it have?
    - Update
    - Think/Decide

In the Game loop we call the Update method on each member (Bird) in the population. This in turn will call the Brird’s Think() function which executes the FeedForward() routine in the Brain.

**Training.**

We will use Backpropagation to find the best weights for the Brain’s neural network. Backpropagation is a supervised learning process, therefor we will need training data.

1. We will generate training data by playing the game and capturing the necessary state and user controls on each loop into a file “Training\_data.csv”.
   1. In Game.h
      1. set player = true
      2. set ai = false
      3. During game play press “C” to start capturing data
2. The program outputs the decision as either “glide” or “flap”. Since Tensorflow does not handle text, we need to find and replace the word glide with a 0 and flap with a 1. This way our model will learn to output a 0 or 1 which we will then interpret as glide or flap.
3. We will then use this training data to train a neural network using Tensoflow in Python. **You need to have Python and the Tensorflow library installed**. The resulting neural network weights are then exported to a file “flappy\_model.h5”.
   1. Source: **keras\_flappy-bird\_network.py**
4. We then convert the model to a txt file “flappy\_tf\_weights.csv”
   1. Source: **convert\_flappy\_model\_weights\_to\_csv.py**
5. These weights are then read into our Flappybird’s Brain which then takes control of the bird.
   1. Make sure flappy\_tf\_weights.csv is copied to the project folder
   2. In Game.h
      1. set player = false
      2. set ai = true
      3. set birdCount = 5 for example