Delhi Technological University

Swarm Optimization & Evolutionary Computing
(CO-423)

Flappy Bird Using Neural Networks And Genetic Algorithm Optimization

SUBMITTED BY:

Rashid Saifi (2K17/IT/94)
Shashank Bhardwaj (2K17/IT/111)
Shubham Chaudhary (2K17/IT/117)

SUBMITTED TO:

Ms. Pratima Sharma
Dept. of Computer Engineering

November, 2020

Acknowledement

We would like to express our sincere gratitude to all those who provided us with the opportunity to complete this report.

First of all, we would like to thank our parents for giving us encouragement, enthusiasm, and invaluable assistance. Without their support, we might not have been able to complete this project properly.

Secondly, we would like to thank Mr. Yogesh Singh, Vice-Chancellor of Delhi Technological University for managing the college and the curriculum and for his supervision and support.

Thirdly, we would like to thank Ms. Pratima Sharma, Professor in the Department of Computer Engineering, who provided us the opportunity for writing this report and for her constant guidance, comments, and suggestions throughout the development of the project.

Finally, we extend our gratitude to all other unnamed people who helped us in various ways to complete this report.

Thanks and Regards

Rashid Saifi

Shashank Bhardwaj

Shubham Chaudhary

Certificate

This is to certify that the technical report entitled “Flappy-Bird Using Neural Networks and Genetic Algorithm Optimization” is a record of the bonafide work carried out by Rashid Saifi (2K17 / IT/ 94), Shashank Bhardwaj (2K17 / IT/ 111) and Shubham Chaudhary (2K17/IT/117) Delhi Technological University during the academic year 2020-2021.

MS PRATIMA SHARMA
PROFESSOR
DEPT. OF COMPUTER ENGINEERING

Abstract

The aim of this Flappy Bird game is to program an artificial intelligence game controller using neural networks and a genetic algorithm

Hence, we want to create an AI robot which can learn how to optimally play the Flappy Bird game. As a result, our little bird should be able to fly safely through a number of barriers. In the best scenario, it should never die.

This program teaches birds how to flap optimally in order to fly safely through barriers as long as possible.

Table of Contents

Introduction

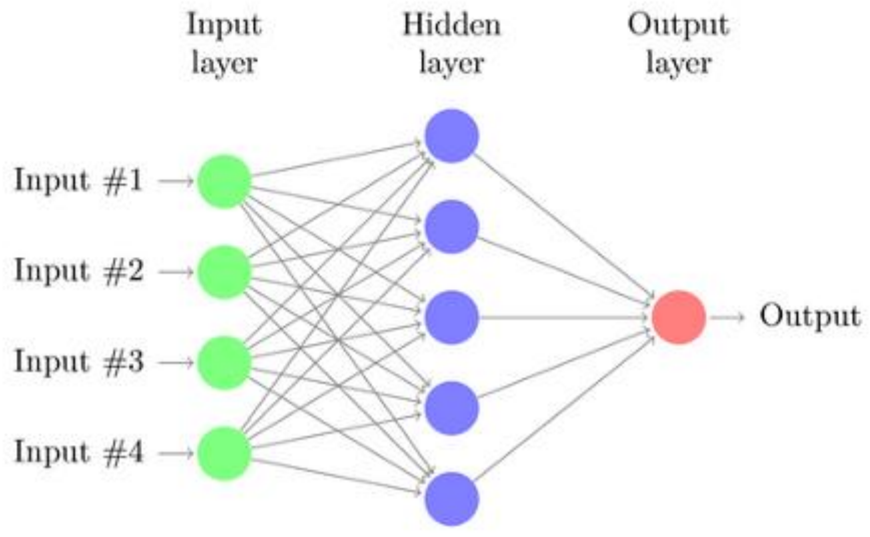
Flappy Bird is a mobile game developed by Vietnamese video game artist and programmer Dong Nguyen (Vietnamese: *Nguyễn Hà Đông*), under his game development company dotGears. The game is a side-scroller where the player controls a bird, attempting to fly between columns of green pipes without hitting them. Nguyen created the game over the period of several days, using a bird protagonist that he had designed for a cancelled game in 2012.

The game was released in May 2013 but received a sudden rise in popularity in early 2014. *Flappy Bird* received poor reviews from some critics, who criticized its high level of difficulty, alleged plagiarism in graphics and game mechanics, while other reviewers found it addictive. At the end of January 2014, it was the most downloaded free game in the App Store for iOS. During this period, its developer said that *Flappy Bird* was earning $50,000 a day from in-app advertisements as well as sales.

In this project we aim to build an automation architecture which would using neural networks and genetic algorithm optimization for the flappy bird game, which points in the direction of building better automatic machines hence building a better future.

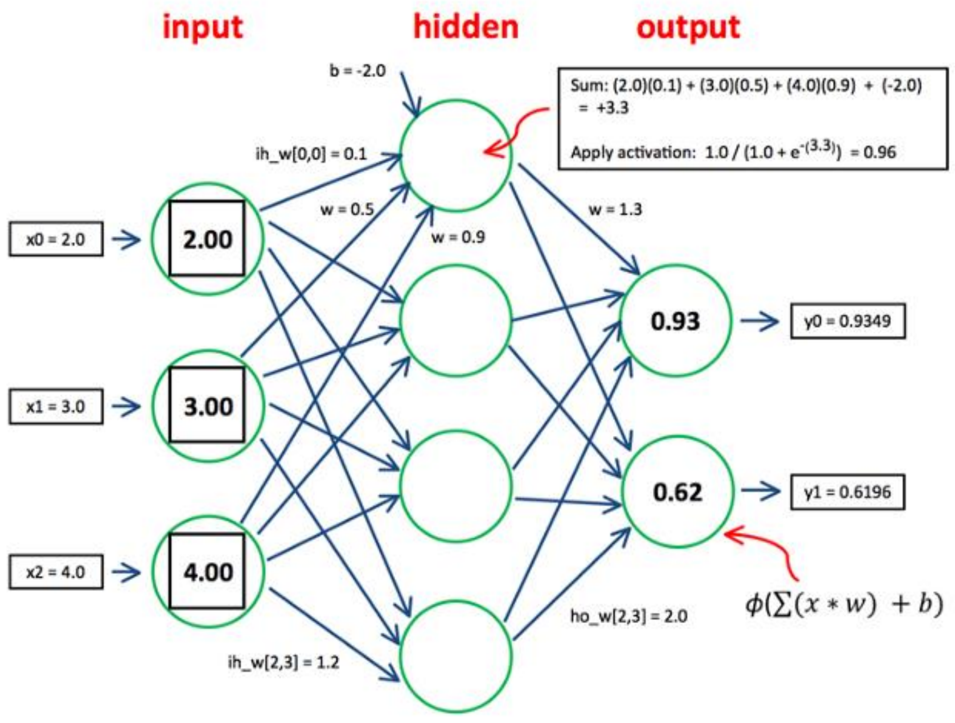
What are Neural Networks?

In programming, artificial neural network is computational model/algorithm for machine learning that is inspired by structure and functional aspects of biological neural networks.



Every neural network has one input layer, one output layer and one or more hidden layers. Each circle represents a neuron and a neuron has a connection to every neuron in the next layer.

Each connection has weight value and each neuron has a bias value. For example, below diagram shows what’s happening in the neural network.



Training neural network means finding and adjusting weight and bias values that gives the best output that we want.

<u>Basic Idea</u>

The Main concept is based on:-

1. Creating the initial population randomly
2. Learning as the Game is being played
3. Applying natural evolution to form the next improved generation
4. Each population as 10 units (Birds)
5. The initial population consisted of birds with random neural networks.

The Main Concept of Machine Learning

The main concept of machine learning implemented in this program is based on the neuro-evolution form. It uses evolutionary algorithms such as a genetic algorithm to train artificial neural networks. Here are the main steps:

1. create a new population of 10 units (birds) with a **random neural network**
2. let all units play the game simultaneously by using their own neural networks
3. for each unit calculate its **fitness** function to measure its quality as:

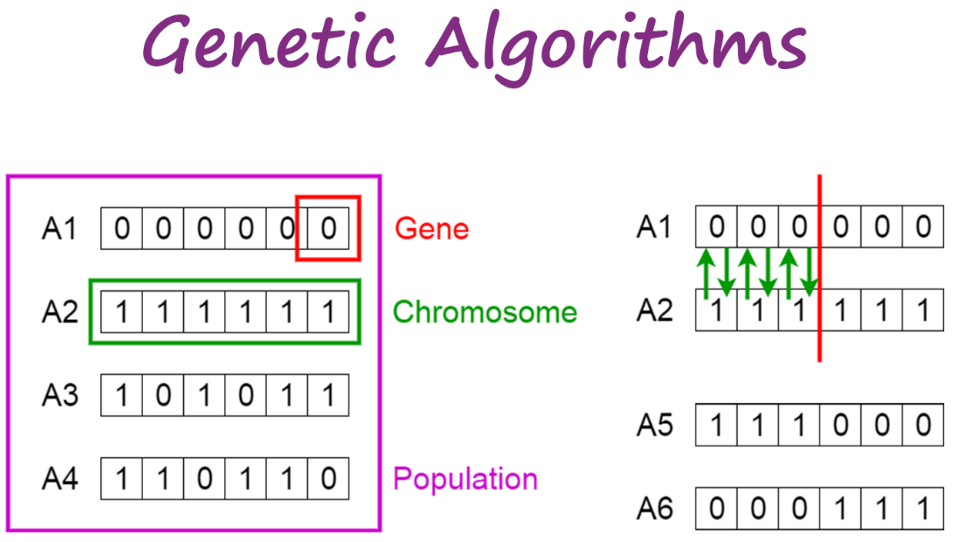
   fitness = total travelled distance - distance to the closest gap

4. when all units are killed, evaluate the current population to the next one using **genetic algorithm operators** (selection, crossover and mutation) as follows:

5. go back to the step 2

Genetic Algorithm Optimization

A **genetic algorithm** is a search heuristic that is inspired by Charles Darwin’s theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.



Notion of Natural Selection

The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found.

This notion can be applied for a search problem. We consider a set of solutions for a problem and select the set of best ones out of them.

Five phases are considered in a genetic algorithm.

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

Implementation of Genetic Algorithm:

1. Create initial population of 10 units (birds) with random neural networks

2. Let all units play the game simultaneously by using their own neural networks

3. For each unit calculate its fitness function to measure its quality

4. When all units died, evaluate the current population to the next one by using genetic operators

5. Go back to the step 2

Genetic Algorithm Code

The genetic algorithm is implemented in g enetic.js file which consists of the following class:

GeneticAlgorithm Class, the main class to handle all genetic algorithm operations. It needs two parameters: *max_units* to set a total number of units in population and *top_units* to set a number of top units (winners) used for evolving population. Here are its essential functions:

- ***reset()*** to reset genetic algorithm parameters
- ***createPopulation()*** to create a new population
- ***activateBrain()*** to activate the AI neural network of an unit and get its output action according to the inputs
- ***evolvePopulation()*** to evolve the population by using genetic operators (selection, crossover and mutations)
- ***selection()*** to select the best units from the current population
- ***crossOver()*** to perform a single point crossover between two parents
- ***mutation()*** to perform random mutations on an offspring

Replacement Strategy

1. Sort the units of the current population by their fitness ranking
2. Select the top 4 units (winners) and pass them directly on to the next population
3. Create 1 offspring as a crossover product of two best winners
4. Create 3 offsprings as a crossover products of two random winners
5. Create 2 offsprings as a direct copy of two random winners
6. Apply random mutations on each offspring to add some variations

Neural Network Architecture

To play the game, each unit (bird) has its own neural network consisted of the next 3 layers:

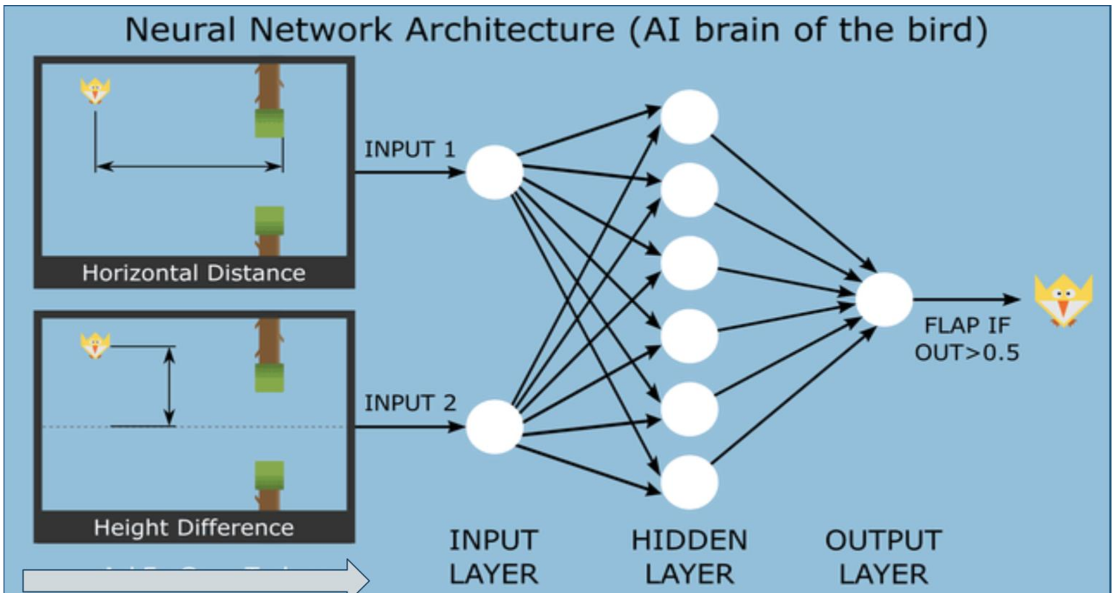
1. an input layer with 2 neurons presenting what a bird sees:

2. 1) horizontal distance between the bird and the closest gap
3. 2) height difference between the bird and the closest gap

4. a hidden layer with 6 neurons

5. an output layer with 1 neuron used to provide an action as follows:

   if output > 0.5 then flap else do nothing

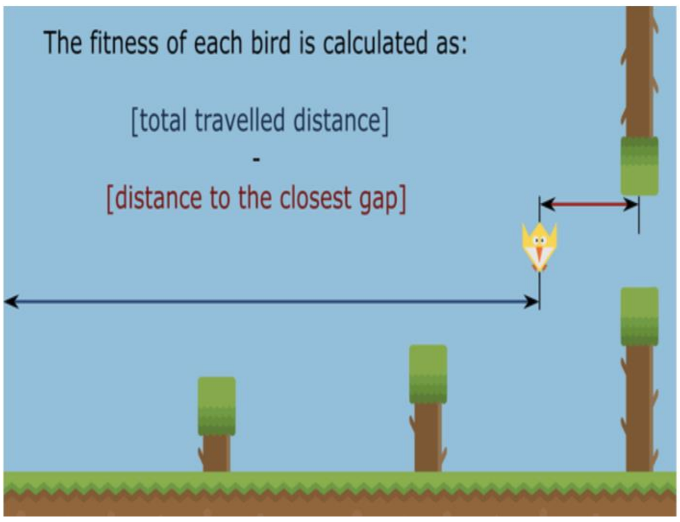


Fitness Function

Generally, the fitness function is the metrics to measure quality of an object. While we have a quality measure for each bird, we can select the fittest units and use them to reproduce the next population.

In this project, we reward a bird equally to its travelled distance. Also, we penalize it by its current distance to the closest gap. So in that way, we are making a difference between birds which travelled the same distance.

To conclude, our fitness function is the difference between the total distance covered by a bird and its current distance to the closest gap.


The fitness of each bird is calculated as:
[total travelled distance]
-
[distance to the closest gap]

Libraries Used

Synaptic.min.js

Synaptic is a javascript neural network library for node.js and the browser, its generalized algorithm is architecture-free, so you can build and train basically any type of first order or even second order neural network architectures.

This library includes a few built-in architectures like multilayer perceptrons, multilayer long-short term memory networks (LSTM), liquid state machines or Hopfield networks, and a trainer capable of training any given network, which includes built-in training tasks/tests like solving an XOR, completing a Distracted Sequence Recall task or an Embedded Reber Grammar test, so you can easily test and compare the performance of different architectures.

The algorithm implemented by this library has been taken from Derek D. Monner's paper:

A generalized LSTM-like training algorithm for second-order recurrent neural networks

There are references to the equations in that paper commented through the source code.

creature.js

```
var synaptic = require('synaptic');
this.network = new synaptic.Architect.Perceptron(40, 25, 3);
```

world.js

```
creatures.forEach(function(creature)
{
    // move
    var input = [];
    for (var i in creatures)
    {
      input.push(creatures[i].location.x);
      input.push(creatures[i].location.y);
      input.push(creatures[i].velocity.x);
      input.push(creatures[i].velocity.y);
    }
    var output = creature.network.activate(input);
    creature.moveTo(output);

    // learn
    var learningRate = .3;
    var target = [
      targetX(creature),
      targetY(creature),
      targetAngle(creature)];
    creature.network.propagate(learningRate, target);
});
```

Phaser.min.js:

Phaser is a 2D game framework used for making HTML5 games for desktop and mobile.It is a free software and developed by Photon Storm.

Phaser uses both a Canvas and WebGL renderer internally and can automatically swap between them based on browser support. This allows for fast rendering across desktop and mobile. It uses the Pixi.js library for rendering.

Games can be compiled to iOS, Android and native desktop apps via 3rd party tools like Apache Cordova and phonegap.

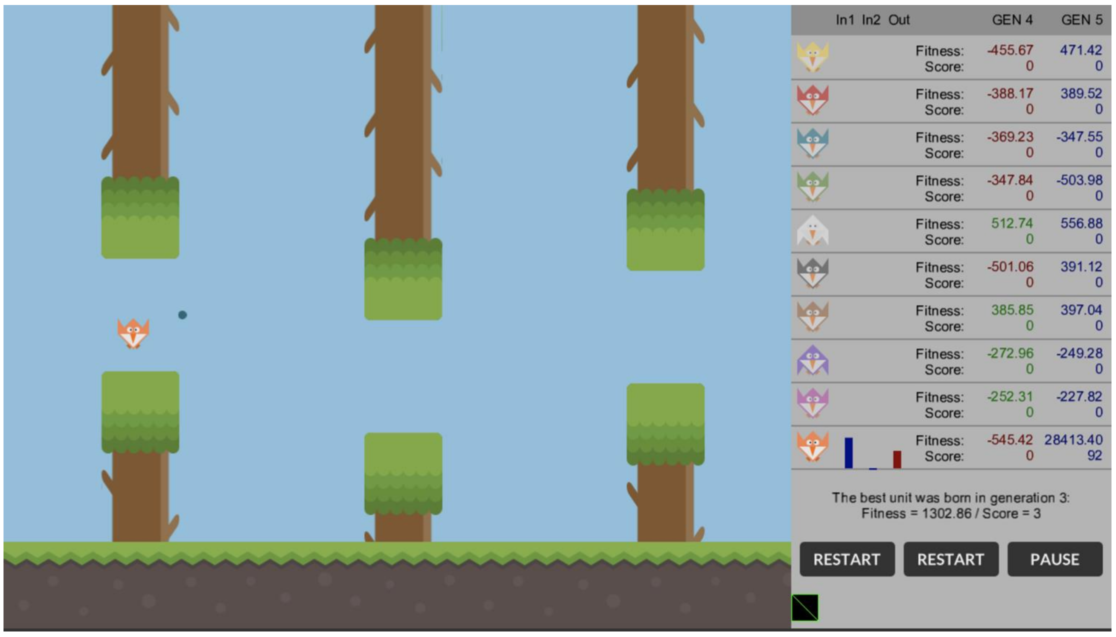
Whilst you can wrap your game into a native app using tools such as Cordova and Phonegap the game itself is never compiled. The 'game' is simply run as JavaScript in a bundled browser. This means performance is nothing like a native compiled app.

The Interface

The entire game logic is implemented in gameplay.js file. It consists of the following classes:

- **App.Main**, the main routine with the following essential functions:
  - *preload()* to preload all assets
  - *create()* to create all objects and initialize a ne w genetic algorithm object
  - *update()* to run the main loop in which the Flappy Bird game is played by using AI neural networks and the population is evolved by using genetic algorithm
  - *drawStatus()* to display information of all units
- **TreeGroup Class,** extended Phaser Group class to represent a moving barrier. This group contains a top and a bottom Tree sprite.
- **Tree Class**, extended Phaser Sprite class to represent a Tree sprite.
- **Bird Class**, extended Phaser Sprite class to represent a Bird sprite.
- **Text Class**, extended Phaser BitmapText class used for drawing text.

Gameplay Snapshot:



In1 In2 Out		GEN 4	GEN 5
	Fitness: Score:	-455.67 0	471.42 0
	Fitness: Score:	-388.17 0	389.52 0
	Fitness: Score:	-369.23 0	-347.55 0
	Fitness: Score:	-347.84 0	-503.98 0
	Fitness: Score:	512.74 0	556.88 0
	Fitness: Score:	-501.06 0	391.12 0
	Fitness: Score:	385.85 0	397.04 0
	Fitness: Score:	-272.96 0	-249.28 0
	Fitness: Score:	-252.31 0	-227.82 0
	Fitness: Score:	-545.42 0	28413.40 92

The best unit was born in generation 3:
Fitness = 1302.86 / Score = 3

References

1. https://docs.idew.org/video-game/project-references/phaser-introduction
2. https://en.wikipedia.org/wiki/Flappy_Bird
3. https://caza.la/synaptic/#/
4. https://medium.com/towards-artificial-intelligence/main-types-of-neural-networks-and-its-applications-tutorial-734480d7ec8e?source=friends_link&sk=24cb7c440bf6831b13b28bbc0437099b
5. https://www.sciencedirect.com/topics/engineering/genetic-algorithm