

CREATING AN ARTIFICIAL GAME PLAYER FOR MODIFIED STICK RUNNING USING NEUROEVOLUTION OF AUGMENTING TOPOLOGIES (NEAT)

Presentor: Aldrich S. Goco
Adviser: Rozano S. Maniaol



Stick Running

Stick Running is an endless running game platform game. It is a single player game wherein the human player controls the game character to jump in order to avoid all of the of the varying obstacles as the game character attempt to run as far as possible. Once the game character run into any obstacle then the game will be over.



Artificial Intelligence in Video Games

- Rule-Based (Ex. IF-THEN)
- Machine Learning



Machine Learning in Video Games

- Reinforcement Learning
- Genetic Algorithm



Genetic Algorithm

- Inspired by biological evolution
- Iterative process that discovers the fitness landscape via mutations to discover optima

Key Components:

- Generate Fitness

Fitness function

- Select top organism “Survival of the fittest”

- Replicate

With chance of mutation



NEUROEVOLUTION OF AUGMENTING TOPOLOGIES (NEAT)

Genetic Algorithm + Neural Network

Fixed Topologies

- Search space is too large
- Unnecessary Complexity

Augmenting Topologies

- Necessary complexity
- Purposeful motivation behind topology changes



Genome / Neural Network

Key Components:

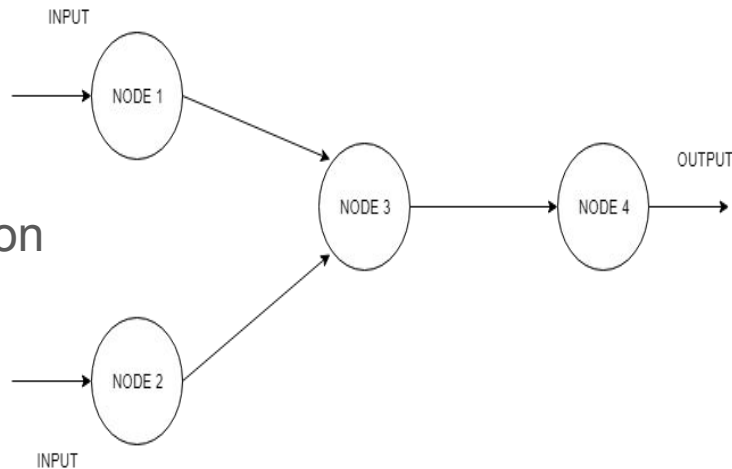
Node Gene / Neuron

Receives input

Calculates output via Activation Function

Connection Gene / Edge

Internode , weighted connection



NEURAL NETWORK

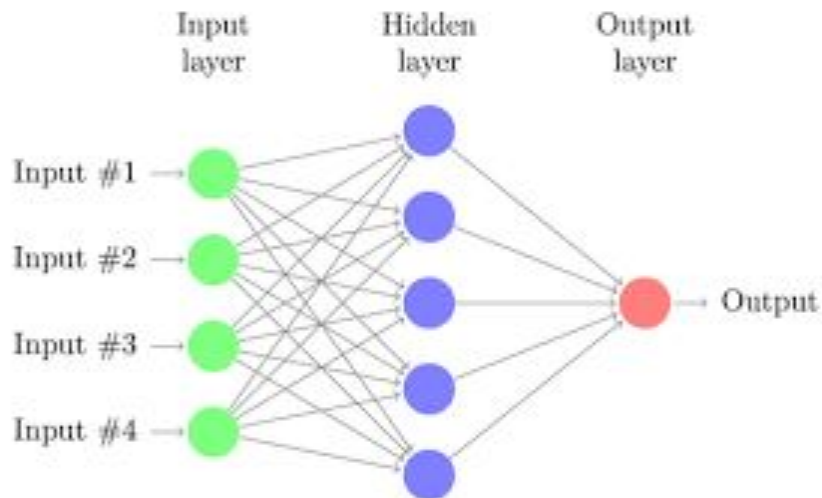
Features

-Layers

Input

Hidden

Output



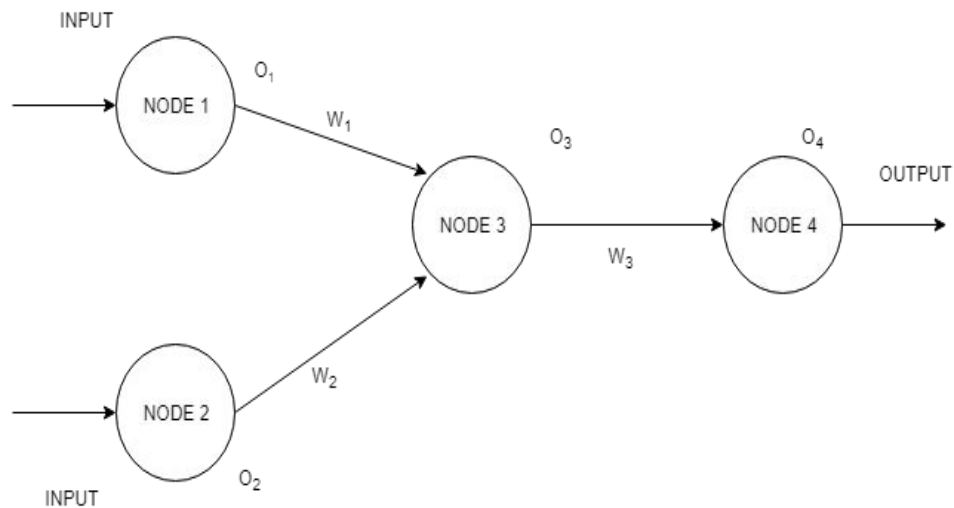
NEURAL NETWORK

$O_1 = \text{Norm}(\text{Input1})$

$O_2 = \text{Norm}(\text{Input2})$

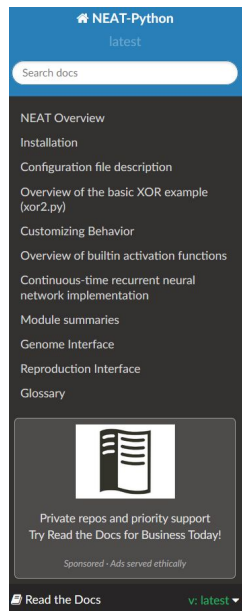
$O_3 = \text{Sigm}(O_1 * w_1 + O_2 * w_2)$

$O_4 = \text{Sigm}(O_3 * w_3)$



NEAT-Python Library

- Neural Network
- Evaluation through NEAT algorithm
- Statistics



Docs » Welcome to NEAT-Python's documentation!

[Edit on GitHub](#)

Welcome to NEAT-Python's documentation! 🐍

NEAT is a method developed by Kenneth O. Stanley for evolving arbitrary neural networks. NEAT-Python is a pure Python implementation of NEAT, with no dependencies other than the Python standard library.

Note

Some of the example code has other dependencies; please see each example's README.md file for additional details and installation/setup instructions for the main code for each. In addition to dependencies varying with different examples, visualization of the results (via [visualize.py](#) modules) frequently requires [graphviz](#) and/or [matplotlib](#). TODO: Improve README.md file information for the examples.

Support for HyperNEAT and other extensions to NEAT is planned once the fundamental NEAT implementation is more complete and stable.

For further information regarding general concepts and theory, please see [Selected Publications](#) on Stanley's website, or his recent [AMA on Reddit](#).

If you encounter any confusing or incorrect information in this documentation, please open an issue in the [GitHub project](#).

Contents:

- [NEAT Overview](#)

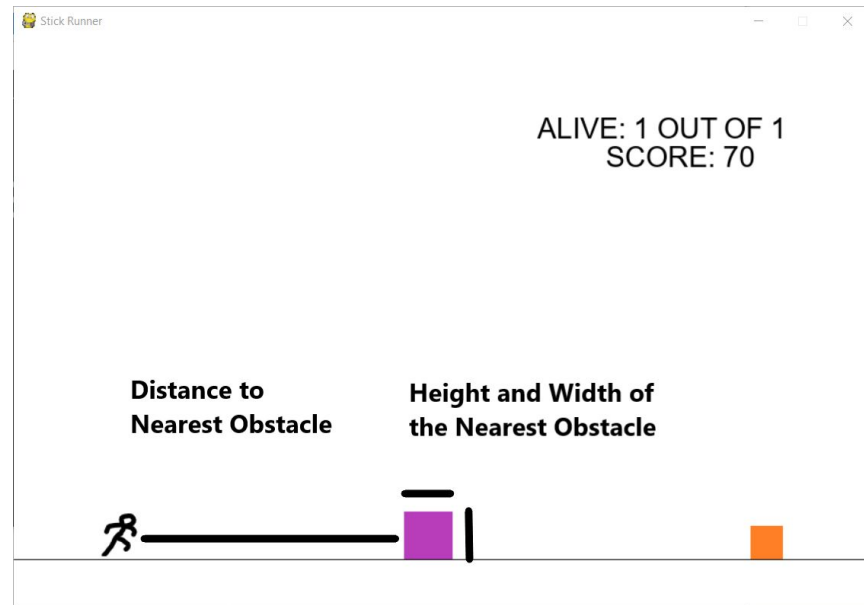
Stick Running - Game Player

Inputs

- player's distance to the nearest obstacle
- nearest obstacle's width
- nearest obstacle's height

Output

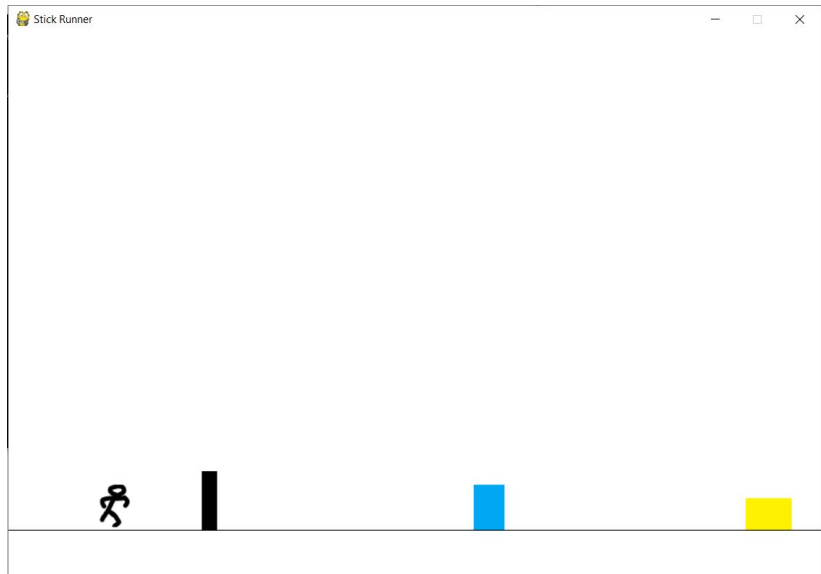
- To jump? Or not to jump



Stick Running - Fitness Function

- **Traveled Distance (TD)**: a counter that increases in every interaction of the agent to the environment
- **Score**: the number of obstacles already transposed

$$\text{FitnessFunction} = \text{Score} + (\text{TD} * 0.001)$$



Stick Running - Game Environment

These seven obstacles then are randomly generated with a minimum gap of 300 pixels to one another.



(a)



(b)



(c)



(d)



(e)



(f)



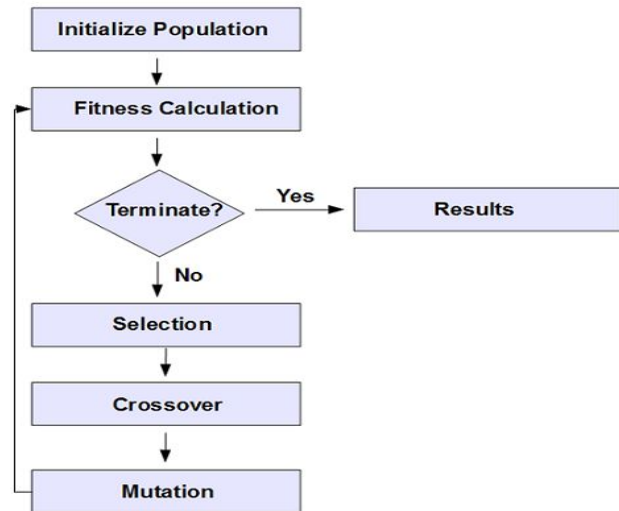
(g)

Fig. 5: Seven Different Obstacles in the Game






NEAT - Overview

- Stops when an individual reaches the Maximum fitness value or the allowed number of generation is reached.



NEAT - Fitness Calculation

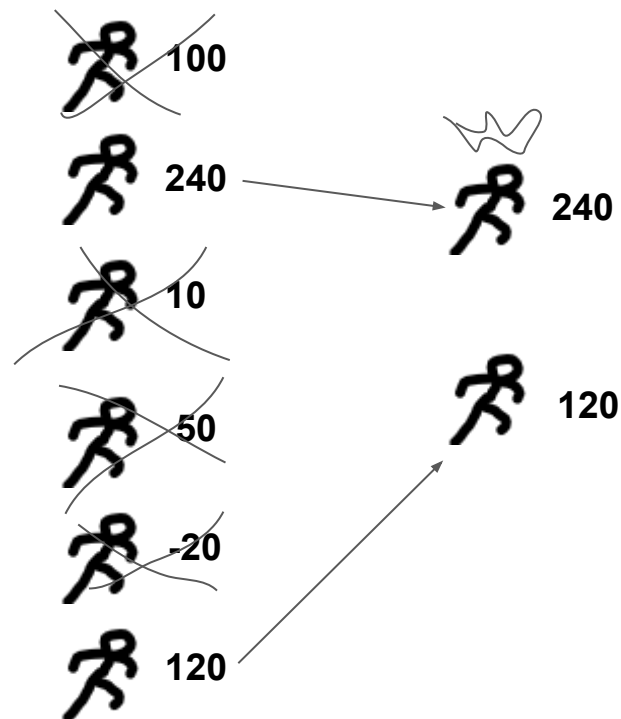
Calculates the fitness value of each individual on the population using the fitness function

 100 240 10 50 -20 120

NEAT - Selection

Elitism = 2 genomes.

Survival_threshold = 20%

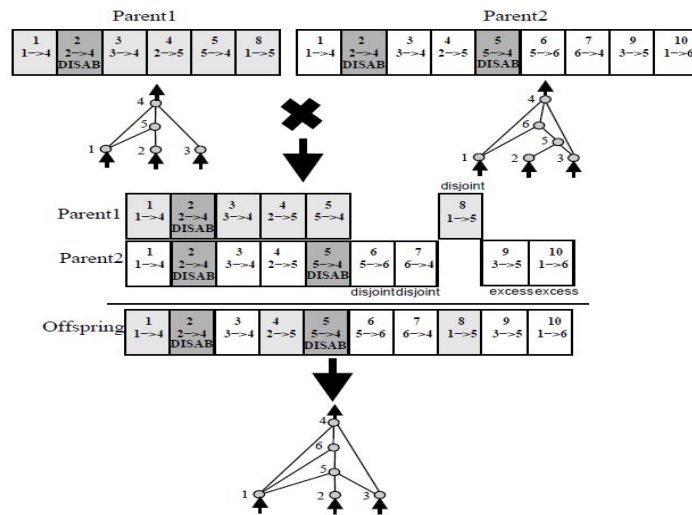


NEAT - Crossover

Innovation Number

Disjoint

Excess



NEAT - Mutation

Weight and Bias Mutate

Weight Adjust (80%)

Weight Replace (10%)

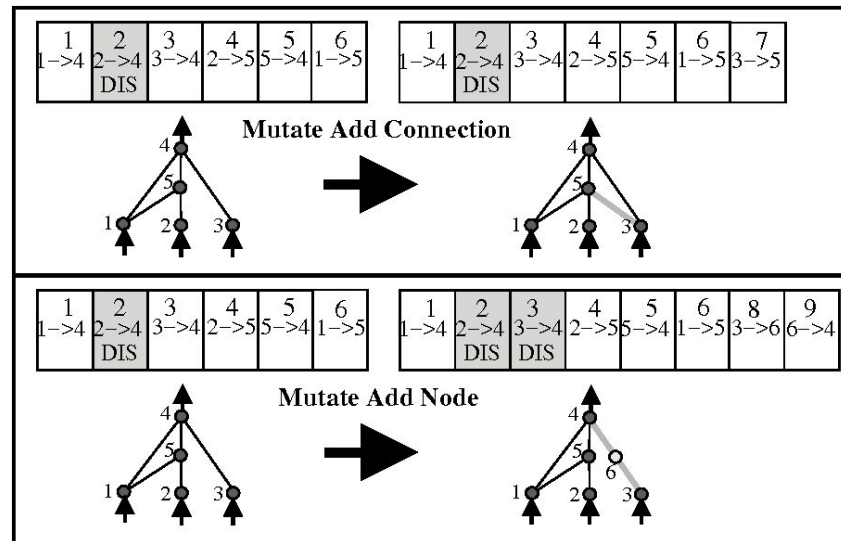
Bias Adjust (70%)

Bias Replace (10%)

Structural Mutate

Add/Delete Node (20%)

Add/Delete Connection (20%)



NEAT - Speciation

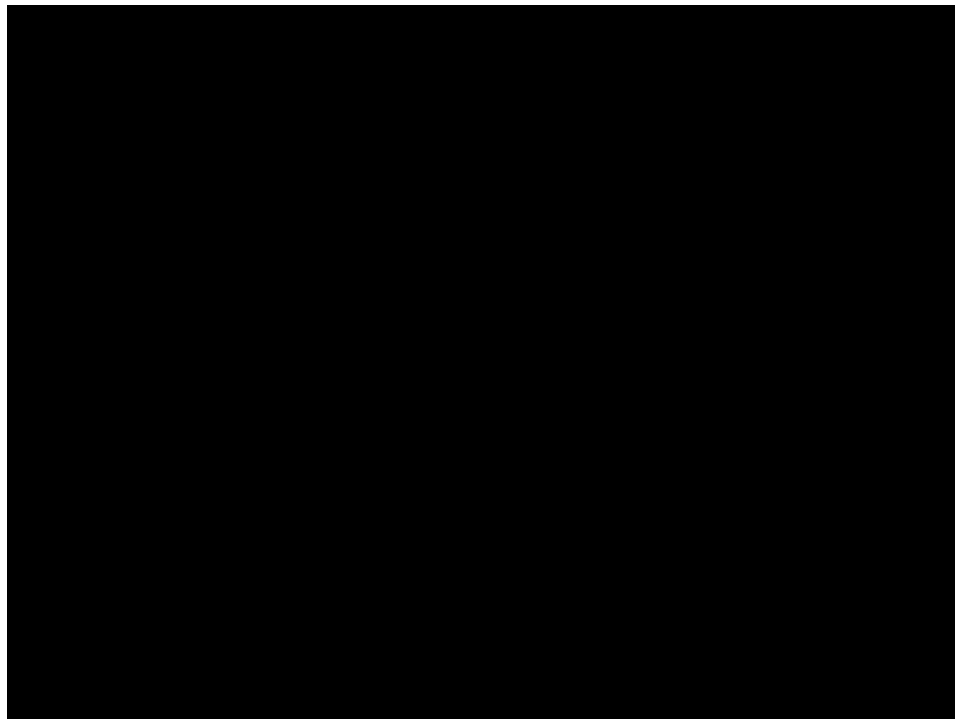
- Preserving Innovation
- Compatibility distance with the basis of Disjoint, excess and average weight of the matching

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \overline{W}$$



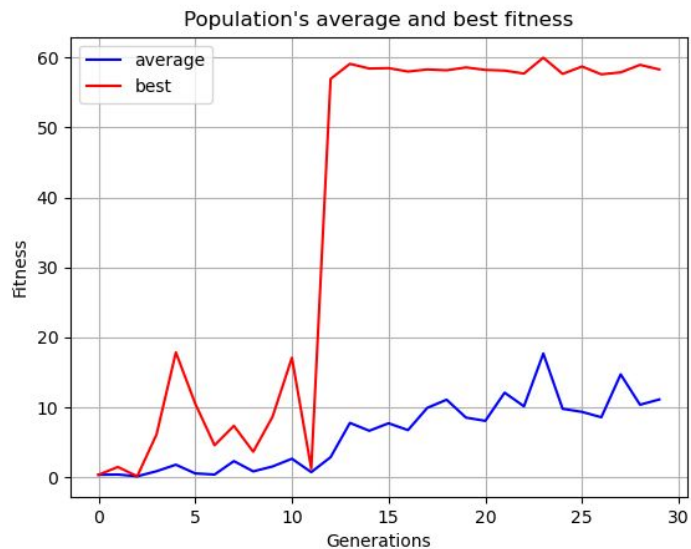
NEAT - Experiment

The NEAT algorithm using sigmoid as an activation function was run for 30 generations with the maximum fitness cap of about 50 scores or 50 obstacles to be transposed. The resulting agent from the training was run on the endless version of the Modified Stick Running.



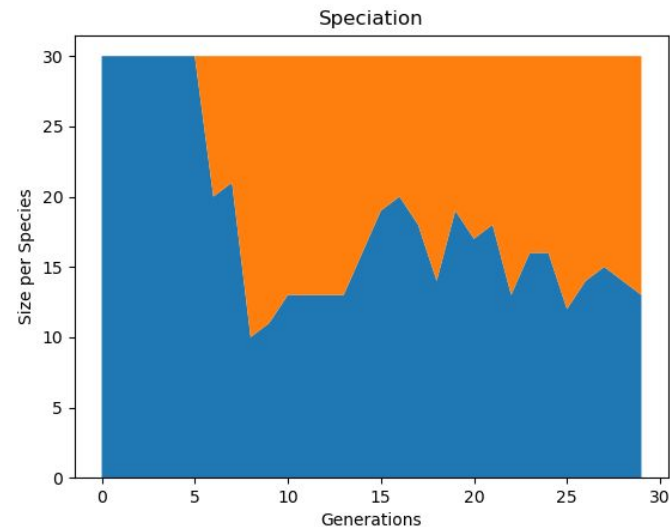
NEAT - Result (Fitness Chart)

Game agents struggled at the first ten generations but managed to make a breakthrough at around 12th generation and stabilizes until the 30th generation



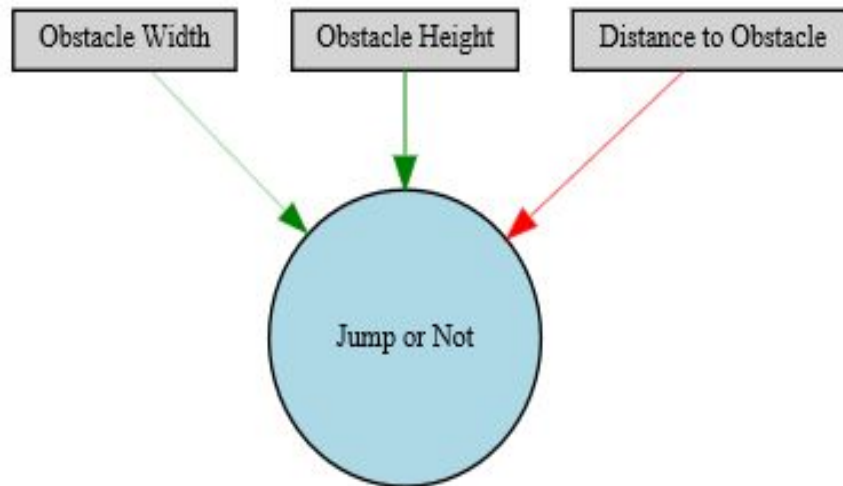
NEAT - Result (Speciation Chart)

Only one species was formed from the first five generations until a second species has surfaced and both coexisted until the 30th generation



NEAT - Result (Best Genome)

The distance to the nearest obstacle is inversely proportional to the probability of jumping as indicated by the negative weight on its connection gene.



NEAT - Result (Best Genome Run)

The best genome extracted from the experiment was run on the endless version and managed to score 5345 points and is still alive

Stick Runner — □ ×

ALIVE: 1 OUT OF 1
SCORE: 5345

