An AI That Uses A combination Of Multiple Learning Techniques, Is Better Than An AI That Uses Only One Learning Technique

By Louie Jones-Strong

E.P.Q

Extended Project Qualification

UTC @ harbourside

2017

Contents

[**Glossary**](#_6gxleqeuiuo2) **2**

[**Introduction**](#_1u2dgttiztff) **3**

[What is self-learning?](#_wt6u4clg6vqs) 3

[The aims of the project](#_n2qqdcj7194o) 3

[Why is this needed](#_bsj62mwa3ykb) 4

[The different learning techniques](#_dg7mhyy8pzpu) 4

[Brute force](#_ygrw83mfbcxh) 5

[General description](#_j7pfi4s5yh74) 5

[Advantages](#_9u3mbeapl0h) 5

[Disadvantages](#_4oof4i1vaird) 5

[In depth description](#_p56yy8srze7v) 5

[Evolution](#_gy40nd76wpek) 6

[General description](#_wdl4xppoy34v) 6

[Advantages](#_2uijgw37bf03) 7

[Disadvantages](#_wechxwmkdi79) 7

[In depth description](#_f633fdlnq2ox) 7

[Neural Network](#_jeawecpatvow) 8

[General description](#_wyrfjnpqstju) 8

[Advantages](#_p6uy6qtmlb5d) 9

[Disadvantages](#_ofadz3ulbnmm) 9

[In depth description](#_4lbrpt7ubvsq) 9

[Combined learning](#_1n0r6zq4tomo) 10

[General description](#_emp7ip7ael5n) 10

[Advantages](#_4x7b5p9owqra) 10

[Disadvantages](#_nac86ojj0pkq) 10

[In depth description](#_9mg0vvp8bhc9) 11

[How to test AI](#_nr0r9g7djaav) 12

[To make fair conditions](#_niby16w8wjag) 12

[How to benchmark AI](#_dfl4i6yv2wsk) 13

[List of environments](#_mqizbubp3huk) 14

[**Results**](#_gv9lwqrviq22) **15**

[Graph no.1](#_yv78zgmhqqk3) 15

[Graph no.2](#_pr7dltfd06yh) 15

[Graph no.3](#_nw8aumf83v8j) 16

[Graph no.4](#_bpzuk1fu2g6) 16

[Graph no.5](#_qbgkpzndbbtp) 17

[Graph no.6](#_polqv2pik4iy) 17

[**Evaluation of results**](#_qgkkzuwjehaq) **18**

[**Conclusion**](#_pno8dpq9a7pr) **20**

[**References**](#_fvxq6bnaaqnq) **21**

[Used to write this paper](#_iwfiyf6j0fcl) 21

[Great for learning how to do this](#_4bic2zn3qid9) 21

# Glossary

AI - Artificial Intelligence

ANN - Artificial Neural Network

Board - inputs to the AI from the environment

CNN - Convolutional Neural Network

DNA - stored list of weights variables to be used for a neural network.

data-set - a collection of inputs and the relating targets

Environment - a task to be learned by the artificial intelligence

Epoch - going through the data-set once normally in a way that helps learning

Evo - Evolution

Fitness - The measurement of the success of a move or a sequence of moves

Iteration - number of epochs completed

ML - Machine Learning

Move - the output from the AI to the next input for the environment

Noughts and crosses - a game also known as tic tac toe

Population - all the organisms of the same group or species

Reinforcement learning - learning a task by only being told how good the output is

Supervised learning - learning a task using being given help from targets

Unsupervised learning - learning a task without being given help from targets

# Introduction

## What is self-learning?

Self-learning is also known as machine learning; This is a subsection of artificial intelligence. Artificial intelligence, as the name suggests, is an “intelligence” that has been made in an artificial way by humans. However, normally this is just hard coded by a programmer to suit one specific task, for example, traffic lights that are normally coded using timers. Machine learning is where a computer learns how best complete a task. In the example given the computer would learn the best timer for each junction to make it more efficient.

## The aims of the project

The goal for this project is to evaluate whether a self-learning program that uses multiple learning techniques in combination with each other learns faster, with more accuracy and more reliably than a self-learning program that only uses a single learning technique.

This hypothesis has been tested on a selection of different testing environments including both artificial and real-world scenarios. These environments were picked so that there is a selection of different environments to test the strengths and weakness of each of the different learning techniques that are being tested.

## Why is this needed

The inspiration for this EPQ project is based the fact that Google’s deepmind team made an AI called AlphaGO[[1]](#footnote-0) that beat the grandmaster at the game of GO. This inspired me to research how difficult it is to code self-learning programs by an individual.

There are different strengths and weaknesses to each of the different learning techniques that are currently used by computers. Combining these different learning techniques together will allow for using the strengths of each to their full extent. This will allow for it to theoretically learn faster with more accuracy than any other learning technique on mid to high complexity learning environments including games. It will also allow for fast speed ups by using human data skill level, this could be collected using big data. This is not available in many other techniques as a form of speed up.

## The different learning techniques

Noughts and crosses is going to be used to explain how each learning technique works. The way the AIs are explained in this section is how they were implemented and coded for this project.

### Brute force

#### General description

The point of brute force is it collects all the data of every play board that is possible by playing repeatedly against itself to see the results. It then works out the best move for each board to make from each of the moves it has stored in its database of the best move to play on each board. It then picks the best one to play based on how many times it won, lost and drawn each time, so that it can pick the best move for each board.

Table no.1 explains the advantages and disadvantages of using brute force as a learning technique.

| Advantages | Disadvantages |
| --- | --- |
| Easy to code | Large amount of data to store once learned |
| Easy to understand | Slow to learn |
| Good to make data-set | Difficult to implement once learned |
|  | Can’t predict outcome if not seen board before |
|  | Must play every board |
|  | Can’t learn big tasks |

#### In depth description

This version of Brute force works by starting with a blank database that is split into the different inputs with a board look up file to make it quicker to find all the different moves. With our example it would make 10 files; one as a lookup file and nine as input files. These are also copied to an array so that they can be read faster and so that there is less file usage.

The next step is playing the first board, to do this we get the current board (starting board for the example it is going to be “000000000” for a blank board). This board is then searched for in the lookup array.

If the board does not exist then it makes a new board in the lookup table and in the data-set, with a blank part for each possible move for that board and plays a random move.

If the board does exist then it looks, finds the move on the board that has been played the least number of times and plays that move.

If the move is invalid then it sets the times played and fitness to “-101” so when it runs this board again it will not play that move again. Once it has finished that game it will get the fitness from the environment and then recalculate the average fitness for each move it played in the game it just played.

### Evolution

#### General description

Evolution learns in the same way that animals evolved over millions of years to perform the best in their environment. However, the learning technique can simulate a generation in a few seconds instead of years, as well as with more accuracy than natural evolution.

A computer does this by starting off with a starting DNA that is then split into multiple populations that have different versions of the DNA. They are then played against each other then ranked on their performance in the task; The half of the population that has the highest rank is picked. This new smaller population breed together to make a fresh population which then continues the process; As it does this over and over it will increase the skill of the overall DNA of the system.

Table no.2 explains the advantages and disadvantages of using evolution as a learning technique.

| Advantages | Disadvantages |
| --- | --- |
| Can be Easy to implement once learned | Can’t learn everything from start |
| Moderate to understand | Slow to learn |
| Can predict outcome if not seen board before | Hard to code |
| Small amount of data to store once learned |  |

#### In depth description

When evolution starts it creates a seed DNA randomly, it then splits this single DNA into between 10 to 40 copies. Each time it makes a new member of the population it copies the DNA then randomly mutates the different elements so that it contains a DNA. Doing this means that each member of the population is different. Each of the population then play the environment this then gives them a fitness for how well they do. Using this fitness each of the population are ranked.

Once the population has been ranked, half are selected at random; there is a higher probability of the higher ranked being selected. Now half has been selected these are then bred together to form the second half of the population. Breeding works by picking two members of the population and going through each of the items of the DNA, picking a number from one of the two parent DNAs to form that item of the new child DNA. Once all of the DNA has been picked it is then mutated to add variation back into the population. Now that the new population has been created the cycle can continue. This works very well for learning many different environments, but it still can’t reliably learn every testing environment if it doesn't have knowledge of the subject before learning.

### Neural Network

#### General description

Neural networks use backpropagation to learn from a data-set.

Firstly, a neural network is a way of processing inputs to outputs in a structure that is very similar to a brain as it is made of nodes and synapses (these are normally called weights when looking at neural networks). This is similar to the structure of the human brain.

The way to train a neural network is called backpropagation as you propagate the error of the network back through each layer. By using derivatives (this is a part of calculus) backpropagation works out how to change the value of each weight to make it closer to the expected output. This process is repeated and it will get closer each time.

Table no.3 explains the advantages and disadvantages of using neural network as a learning technique.

| Advantages | Disadvantages |
| --- | --- |
| Fast to learn | Very hard to code |
| Small amount of data to store once learned | Hard to understand |
| Easy to implement once learned | Can only learn data-sets |
| Can predict outcome if not seen board before |  |

#### In depth description

Neural networks work by having inputs on one side and outputs on the other with layers in between that are multiplied together until they reach the output. Once the outputs are calculated they are compared to the expected outputs to form an error of the network for that input. This is then used to “backpropagate” the error back through the network changing the weights so next time that input is put in the network it will be closer to the target value. This is repeated over multiple inputs so that it learns the pattern across all the different boards.

Backpropagation works by working out how changing the the value of a single weight will change the error of the network. This is expressed as a gradient so the higher the number the further away the weights’ current value is from the value it should be for that input. This is achieved by finding the derivative of how the total error changes in terms of the weight. This is then repeated for each weight. Then it is repeated for all the testing data until it has learnt this data-set. As the error decreases so will the gradient therefore it will slow down as it gets closer to the answer so that it doesn't overshoot the optimum values of each weight.

### **Combined learning**

#### General description

In this project a combination of learning techniques are all used in conjunction so that the AI can benefit from the strengths of each technique. To achieve this brute force is used to gather a data-set of the best move to play on each board. Once enough of this data is collected a neural network is used to learn this data-set. When near 100% of the data has been learned the weights of this are then saved. These weights are then used as a seed in evolution. Evolution then finishes the last part so that it can learn how to do better at the boards it is not as good at. With this combination of the different learning techniques it should, theoretically, learn faster with more accuracy. It will also allow for speed ups as instead of using brute force it can learn straight from human level skill. This is not available to normal learning techniques.

Table no.4 Explains the theoretical advantages and disadvantages of using combined learning techniques.

| Advantages | Disadvantages |
| --- | --- |
| Easy to implement once learned | Very hard to code |
| Very Fast to learn | Hard to understand |
| Can predict outcome if not seen board before |  |
| Small amount of data to store once learned |  |
| Can learn more environments |  |
| Reliably learn each environment |  |

#### In depth description

This combination of learning techniques should theoretically work well together as it has the best parts of the other learning techniques. The structure allows for the best application of each technique. This is done by starting with brute force to create a data-set of the best move to play on each board.

Once the combination feels that enough of the game has been learnt it then moves on to the next step. To know when to move there is a modicum of hardcoded AI to decide for each task when the best time is to change to each step. The plan is to at some point make this a small self learning AI able to regulate when to switch as this will make it faster.

Now that the combination has picked for the next step the data-set from the brute force is passed into the neural network. At this point the neural network learns how to match the data-set that it was given. Once it has achieved over 90% to 99% accuracy it then stores the trained weights that it has learned as well as the metadata of what shape and the activations functions were used for each layer. After this has been saved evolution is then set up using the pre-trained weights from neural network. Then the original testing environment is then loaded so that the network can be tested against the real thing. Using this data the evolution then improves it to the point where it can do better against boards it has never played before.

Other benefits of using a combination of learning techniques and single artificial intelligence is that instead of starting from an unlearned state using brute force this process can be jump started by using a data-set created by humans playing this environment against each other. This means that the neural network can learn from human level intelligence instead of brute force level intelligence; this cannot be achieved on other artificial intelligence that only uses a single learning technique.

One of the other speed ups that can be achieved by using this combination of learning techniques is that while the brute force is trying to collect a data-set the neural network can be learning this changing data-set at the same time. This means that for one iteration it can learn more. This also means it can make more use of the parallel processing nature of current and future generation computers more efficiently.

## How to test AI

These conditions are in place to make it as fair as possible on each learning technique to make the results as conclusive as possible. These rules are needed to make the results conclusive, as self-learning AI are known to be very difficult to benchmark against other AI. This is because different AIs are suited to learning different tasks.

### To make fair conditions

The different AIs’ learning abilities will be tested under these conditions:

1. Must learn to master a selection of different tasks
2. No parameters can be changed between tasks
3. Running on the same hardware
4. Coded in the same language
5. Coded using the same libraries
6. Must have the same information given to them from the task

### How to benchmark AI

These AI where benchmarked with different criteria to make it fair:

1. The time it takes to learn
2. How the skill compares to random inputs
3. How the skill compares to human performance
4. How the skill compares to the other AI
5. How hard it is to code
6. The sort of hardware it is suited for

### 

### 

### List of environments

These environments were picked as they give a selection of different kinds of tasks for the different AI to learn. This should level the playing field between the AIs as each AI will do better at each task.

1. balancing a pendulum (cartpole)
2. Noughts and crosses
3. Learning a simple data-set
4. Learning a complex data-set

More have been tested using this combination of AIs on more resources by other people[[2]](#footnote-1). However only these ones were included in this paper. But you can be reassured that this was not due to cherry picking purposes as the results still supported the conclusion of this paper.

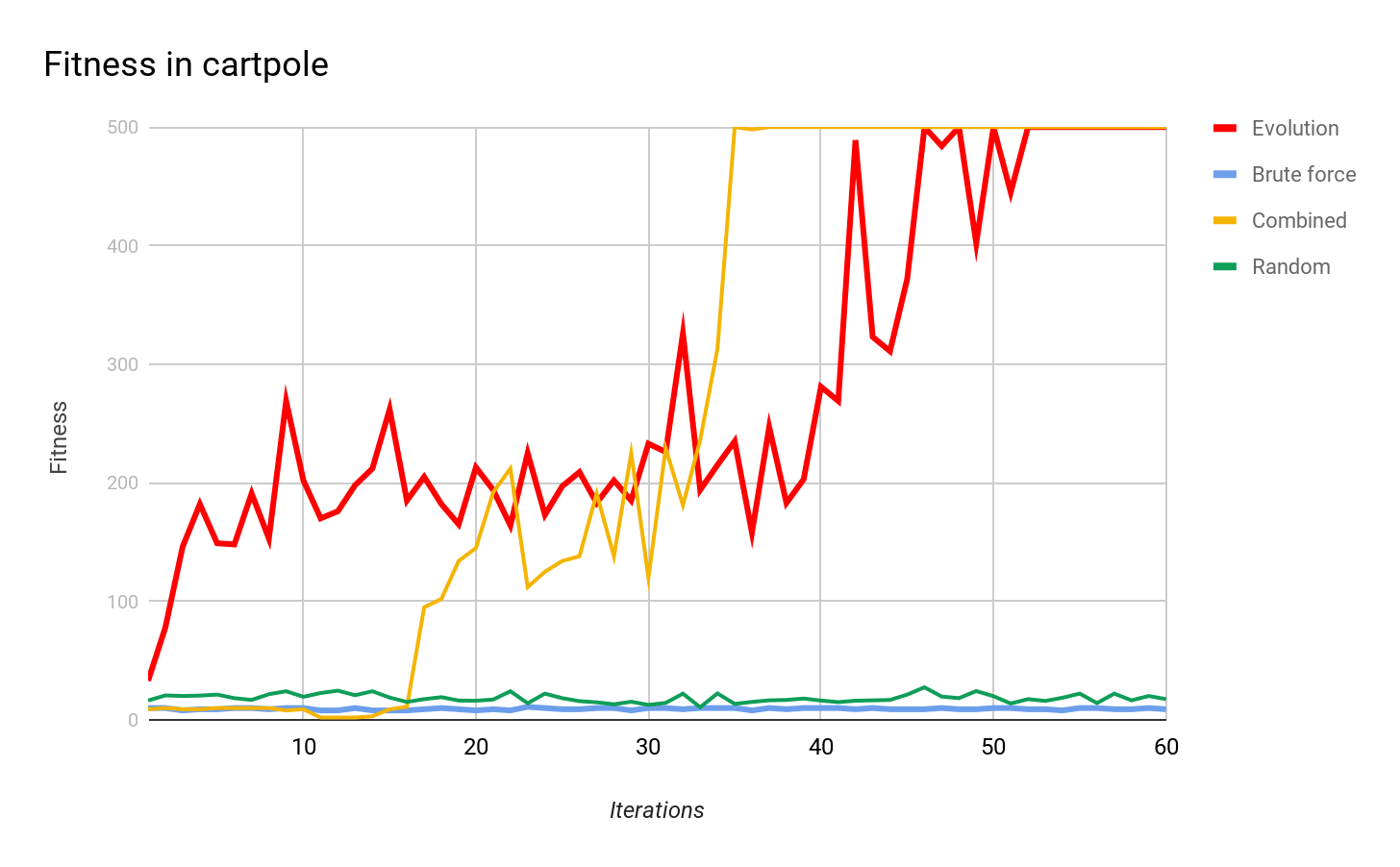
# 

# 

# Results

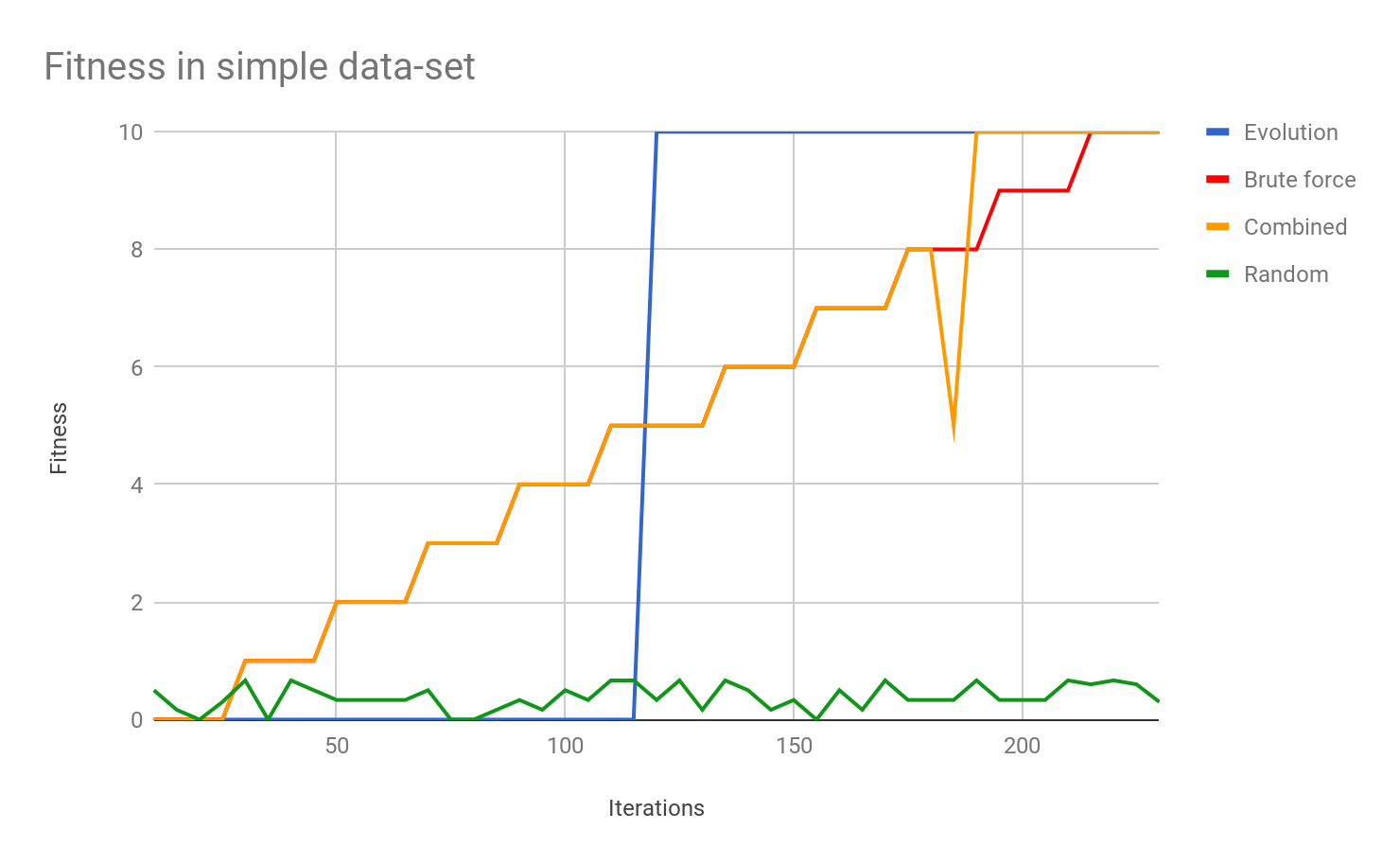
## Graph no.1

Plotted fitness against iterations for each AI at the cartpole environment



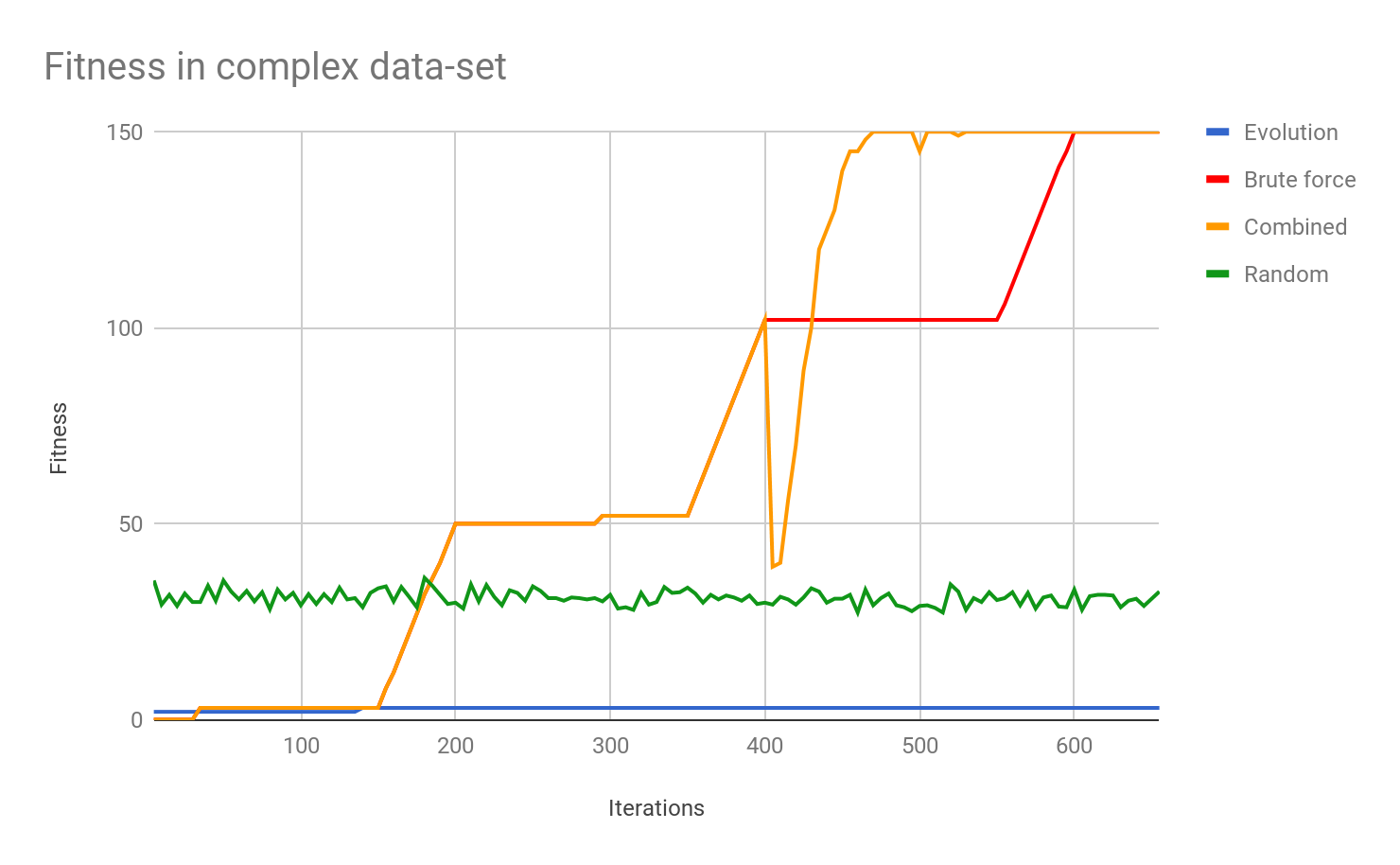
## Graph no.2

Plotted fitness against iterations for each AI at the simple data-set environment



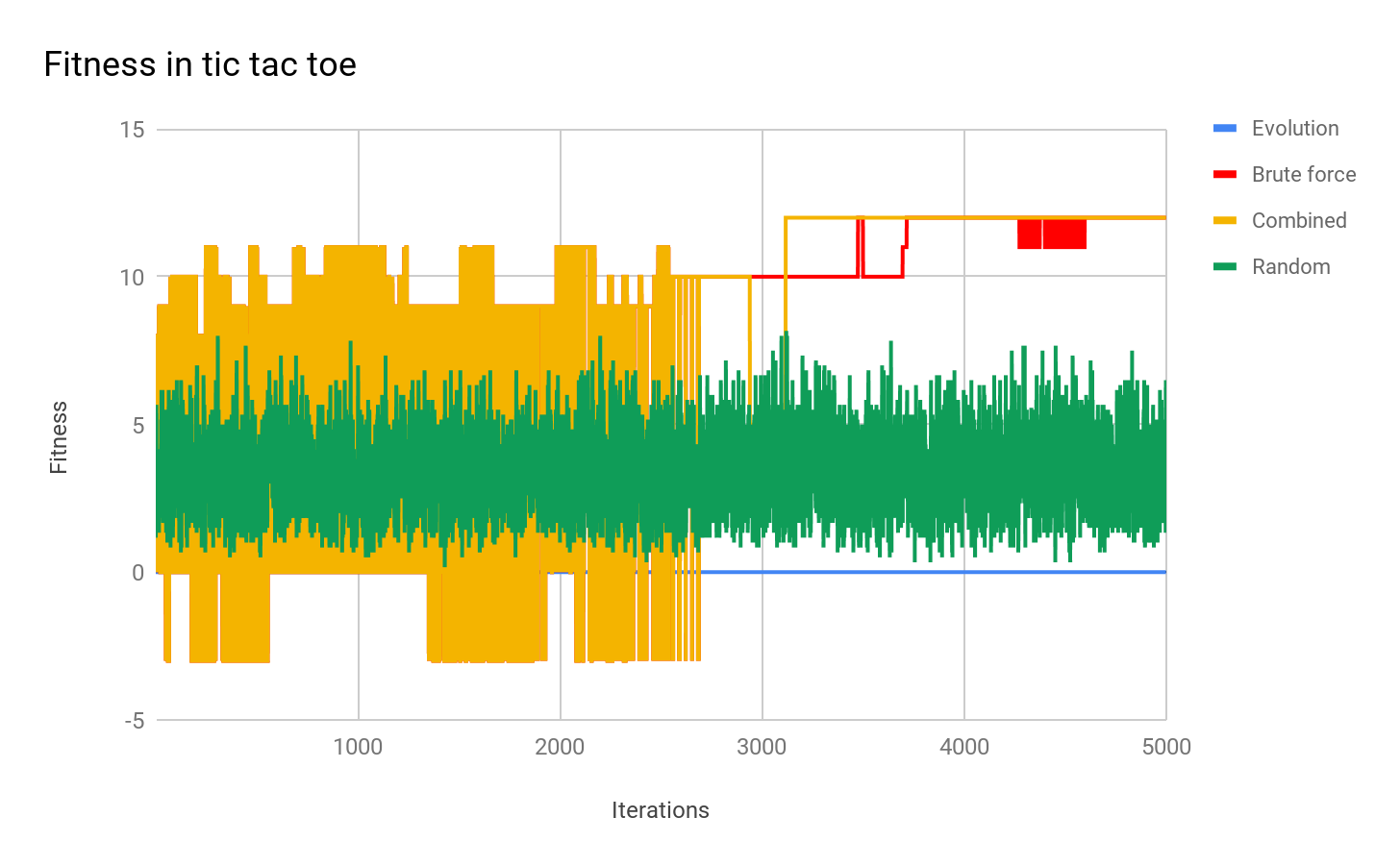
## Graph no.3

Plotted fitness against iterations for each AI at the complex data-set environment



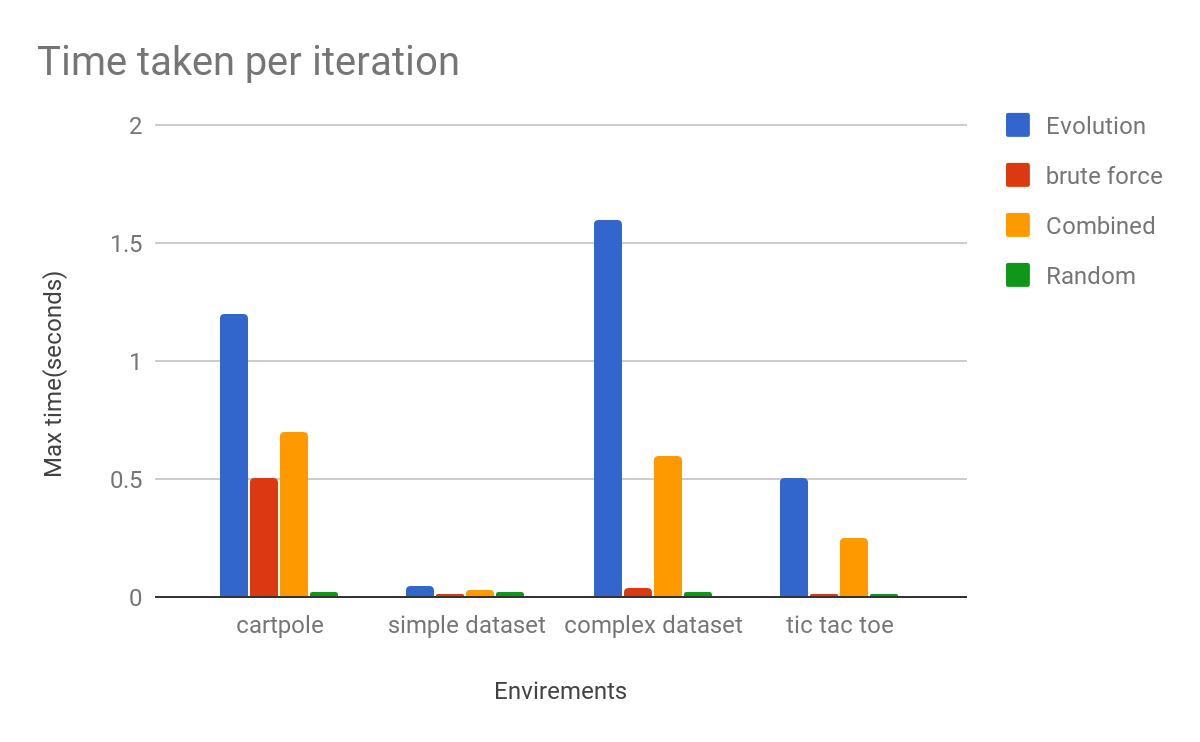
## Graph no.4

Plotted fitness against iterations for each AI at the tic tac toe environment



## Graph no.5

Shows the max time for one iteration by each AI when leaning each environment



## Graph no.6

Shows the time taken to learn each environment by each AI

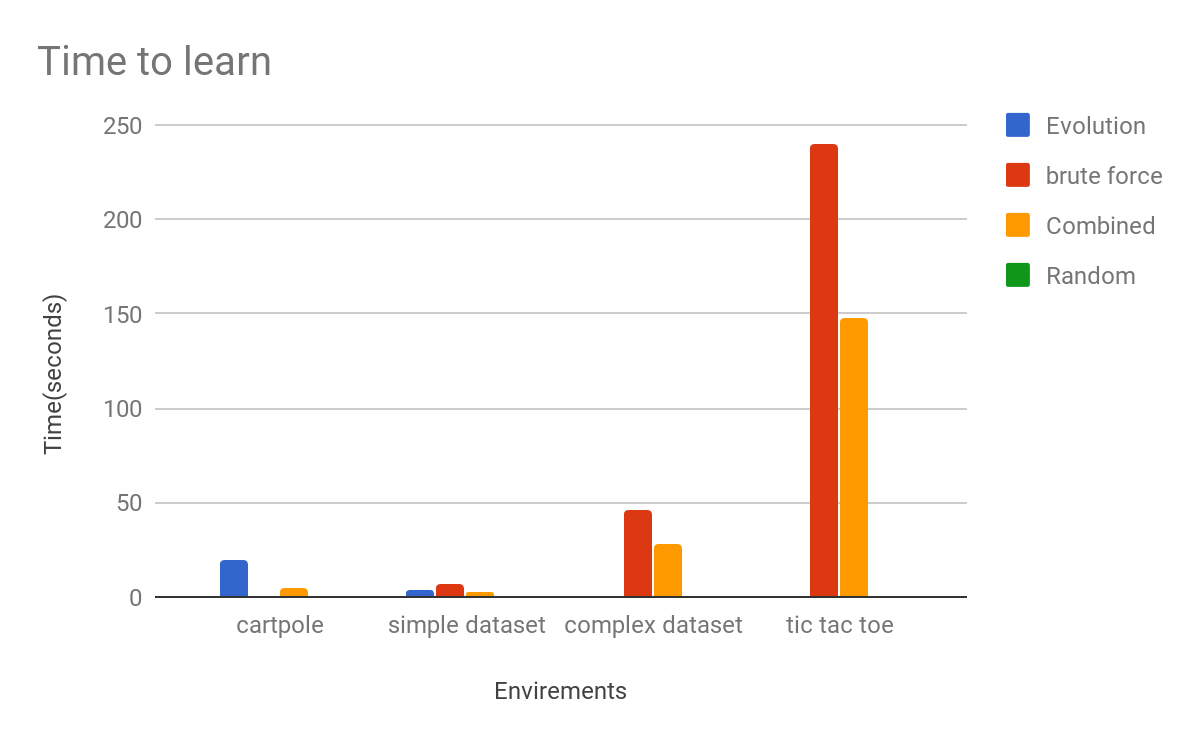


Table no.5 shows the time taken for each AI to learn each environment (in seconds)

|  | Evolution | brute force | Combined | Random |
| --- | --- | --- | --- | --- |
| cartpole | 20 | N/A | 5 | N/A |
| simple data-set | 3.5 | 6.5 | 2.5 | N/A |
| complex data-set | N/A | 46 | 28 | N/A |
| tic tac toe | N/A | 240 | 148 | N/A |

# Evaluation of results

From the results it is evident that the different AIs do perform better in different types of learning environments. This makes sense because each learning technique is designed for a different sort of task.

In graph no.1 it is easy to see how the combined AI has learned this task faster than the other learning techniques. It also took a quarter of the time as the next best AI to learn this task. It is also clear to see the distinct steps taken in learning this task. The reason brute force can't learn this environment is because there are just too many boards for it to learn.

In graph no.2 brute force can be seen to steadily improve as it learns each board. But once the combined AI feels it has learned enough it changes the learning technique it is using and finishes the learning. However, evolution does learn the task faster but this is just luck so it isn't reliable at this task.

In graph no.3 also shows that brute force slowly learns each board in turn and that the combined AI then changes learning technique that it is using and learns it in the fastest time. Evolution can't learn in this environment as there is not enough guidance.

Graph no.4 Tic Tac Toe can't be learnt by evolution as there is no guidance to help it learn. Brute force takes a while but manages to learn it in the end. But again the combined AI manages to learn it faster and to a higher level.

Graph no.5 shows how long it takes for each learning technique to run through one iteration of each environment. From this it is clear to see that evolution is the slowest at every environment. This means that even if it completes it in less iterations it can still take a long time to learn. The combined AI takes the next longest time but it is takes less iterations to learn as show previously.

This also links in with what is shown in the the graph no.6 and table no.5 as the combined AI is the only AI that can learn every environment. It is also the fastest on every task.

# 

# 

# Conclusion

The rigorous benchmarking process proved the hypothesis that; an AI that uses a combination of multiple learning techniques is better than an AI that uses only one learning technique. As all the results show that an AI with combined learning techniques creates a noticeable increase in speed, accuracy, reliably, ease of implementing and level of skill gained in each task. These are all attributes that are very useful for the future of artificial intelligence. I feel that this warrants further research into using a combined AI to better understand just how influential this can become.

This paper has also shown that benchmarking AI in this way can provide very good insights with easy understanding to a non-subject audience. Therefore, I feel that benchmarking AI in a similar way to this should be used across more media to allow self-learning to become more easily accessible to a wider audience.

Using the knowledge gained from writing this paper, this project is going to be continued by being tested on more environments. More importantly, the focus will be concentrated on improving the combined AI. This is so that it can be used in more real-world scenarios with fewer adaptations needing to be made, as well as improving the time taken per iteration and overall time needed to learn.

# References

## Used to write this paper

* <https://gym.openai.com/> - for environments
* <http://www.numpy.org/> - to handle large arrays fast
* <https://www.pygame.org/> - used to make sound when it is completed
* <https://www.python.org/> - used to code the project
* <http://smallbasic.com/> - used to code tests and first versions
* <http://tflearn.org/> - for making the networks
* <https://www.tensorflow.org/> - for making the networks
* <https://deepmind.com/> - some of the inspiration

## Great for learning how to do this

* <http://www.derivative-calculator.net/>
* <http://experiments.mostafa.io/public/ffbpann/>
* <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>
* <http://rednuht.org/genetic_cars_2/>
* <https://www.openprocessing.org/sketch/205807>
* <http://boxcar2d.com/index.html>
* <http://neurovis.mitchcrowe.com/>
* <http://robotics.stanford.edu/~ronnyk/glossary.html>
* <https://medium.com/towards-data-science/tflearn-soving-xor-with-a-2x2x1-feed-forward-neural-network-6c07d88689ed>
* <https://stats.stackexchange.com/questions/277216/tic-tac-toe-neural-network-predicting-a-piece-where-there-is-a-piece>

1. In 15 March 2016 Alpha GO beat Lee Se-dol in GO this was considered to be impossible [↑](#footnote-ref-0)
2. These AI were also tested against Open AI’s gym this included: acrobot-v1, pendulum-v0, mountainCar-v0 and GO9x9-v0. [↑](#footnote-ref-1)