



# Evolving a Learning Agent using Neuroevolution in the FightingICE Game Framework

Deliverable 1: Final Year Dissertation
Heriot-Watt University

Robert John Dunn
H00163867
BSc Honours in Computer Science

Supervisor:
Dr Patricia A. Vargas
School of Mathematical & Computer Sciences
Heriot-Watt University

Co-Supervisor:
Dr Fabrício Olivetti de França
Center of Mathematics, Computing and Cognition
Federal University of ABC

Second Reader:
Dr Mohamed Abdelshafy
School of Mathematical & Computer Sciences
Heriot-Watt University

## Declaration

I, Robert Dunn confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.
Signed:
Date:

**Abstract**

Neuroevolution is a popular technique for machine learning in which the topology and/or weights of an artificial neural network are adjusted by an evolutionary algorithm. The technique takes inspiration from the evolution of the biological nervous system and is a popular approach for reinforcement learning problems. One way to demonstrate the effectiveness of neuroevolution is through artificial intelligence in games. This project aims to implement a learning agent in the FightingICE platform, a two-dimensional Java fighting game organised and maintained by Ritsumeikan University, Kyoto. The agent is designed to evolve through neuroevolution to improve its performance in the game, eventually becoming competitive versus a human opponent. By implementing a neuroevolution method in a simplistic environment, we hope to evaluate the effectiveness of neuroevolution as a method of machine learning and explore the potential of our agent's performance.

# Contents

# 1 Introduction

## 1.1 Motivation

Machine learning is a heavily researched field of artificial intelligence in which machines are given the ability to learn without being explicitly programmed. Various methods of machine learning exist which allow the machine to adapt itself in order to process new data. One of these methods is neuroevolution, which involves the weights and/or topology of an artificial neural network being adjusted by an evolutionary algorithm.

The neuroevolution method is loosely based on the way the biological nervous system operates: with neurons communicating through axons, represented by nodes of the neural network with weighted connections. Neuroevolution is a form of reinforcement learning, which means a notion of reward is introduced and the machine attempts to maximise said reward. The wide application of reinforcement has given neuroevolution popularity in the fields of artificial life and evolutionary robotics. This notion of reward also allows the method to easily be applied to computer games since the reward is usually simple to measure, e.g. the hit-points of the player's character.

## 1.2 Objectives

This project aims to implement a neuroevolution algorithm acting on a learning agent in the FightingICE game platform. FightingICE is a two-dimensional fighting game written in Java by Intelligent Computer Entertainment Lab, Japan. The platform allows easy implementation of artificial intelligence by sending delayed game information to the agent. Using neuroevolution we hope to evolve an agent to the point of being competitive versus a human opponent, though not unbeatable. The performance of the agent will be evaluated at various stages of the evolution in order to evaluate the effectiveness of the algorithm.

## 1.3 Professional, Legal, Ethical and Social Issues

# 2 Literature Review

## 2.1 Machine Learning

### 2.1.1 Learning Paradigms

Learning is an essential human function in which we modify our behaviour tendency according to experiences, to become better when a similar situation occurs. In the study of machine learning, algorithms, computer applications, and systems, utilise learning to improve their performance at certain tasks. There are two main entities in the machine learning model, the teacher and the learner. The teacher contains the knowledge to perform a given task while the learner has to learn the knowledge the teacher holds. [9] There are three types of machine learning:

1. *Supervised Learning*
   A teacher provides the learner with a set of input and desired output pairs. The learner can then use these examples to improve its performance at the task.

2. *Unsupervised/Self-organised Learning*
   There is no teacher, so the learner learns based only on the stimuli received. A common application is finding patterns or grouping within data, using methods such as cluster analysis.

3. *Reinforcement Learning*
   The agent uses goal-directed learning where a notion of reward is introduced and the agent attempts to maximise this reward. There is no teacher for the agent and no explicit model of the environment.

## 2.2 Neuroevolution

### 2.2.1 Fundamentals

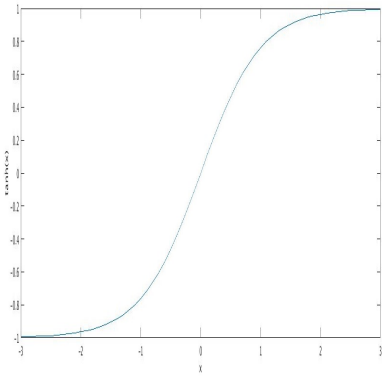
Neuroevolution is a popular biologically-inspired method of machine learning in which an artificial neural network is evolved using an evolutionary algorithm. The method has proved popular, especially in the fields of artificial life, evolutionary robotics and computer games. One reason for its popularity is the fact that neuroevolution is a form of reinforcement learning, which can be applied more generally than its counterpart, supervised learning.

### 2.2.2 Artificial Neural Networks

Artificial neural networks model the way the brain solves problems with collections of neurons communicating through axons, represented with neurons with weighted connections. Neural networks have an associative (content-addressable) memory and can also implement parallel processing. They are often applied to pattern finding within data.

The most basic version of an ANN (artificial neural network) is a perceptron. A perceptron is a simple machine which takes a variable number of input nodes which connect to a single output node. From the perceptron evolved the multi-layer feed forward network, which incorporates extra 'hidden' layers of neurons between the input and output. The more hidden layers there are and the more nodes in these layers, the more complex behaviour a network can experience.

In order to saturate neuron values to a usable range (usually -1 to 1), an activation function is used. The most basic version of this function is ?, where any values below zero will result in an activation of zero, and any values above zero in an activation of one. It is also possible to use more general, non-linear functions such as tanh, shown below.



### 2.2.3 Evolutionary Algorithms

Evolutionary algorithms are methods of optimisation, where potential solutions to the problem are seen as individuals of some population. Each individual in the population is assigned a fitness value which is calculated according to the performance of the provided solution, and the algorithm works to find the best (fittest) of these solutions. Using techniques inspired by biological evolution such as reproduction, mutation, recombination, and selection, new populations of solutions can be generated and assessed. Using evolutionary algorithms allows fine tuning of the search space through constants such as rate of reproduction and mutation rate. [Elitism?GAs?]

### 2.2.4 NEAT (NeuroEvolution of Augmenting Topologies)

NeuroEvolution of Augmenting Topologies (NEAT) is a method of neuroevolution developed by Ken Stanley in 2002, which involves both the weights and the topology of the ANN being altered. The method is a genetic algorithm and involves applying the following three techniques:

1. Track genes with history markers to allow crossover among topologies
2. Apply speciation to preserve innovations
3. Developing topologies incrementally from simple initial structures

A notable extension of NEAT which could potentially be appropriate for the project is HyperNEAT. HyperNEAT is a hypercube-based extension of the method developed by the Evolutionary Complexity Research Group at UCF.

### 2.2.5 Neuroevolution in Games

How neuroevolution can be used to evolve learning agents in games, usually agent is optimising some value e.g. HP. How fitness in games is evaluated, giving some examples. How neuroevolution fares compared to other learning algorithms. Present examples of projects implementing neuroevolution to learn.

### 2.2.6 Incremental Evolution

When an agent is provided with a complex behaviour, it may have trouble evolving to perform at its potential. Using incremental evolution, the behaviour can be learnt incrementally with tasks gradually increasing in difficulty. This form of evolution proved effective in one implementation [5]. The agent's task was a prey capture task: the agent moves through the environment and must catch its prey before the set number of time-steps. With increasingly difficult tasks, the agent was able to rapidly improve its performance, and skip many potential generations of evolution.

## 2.3 FightingICE

### 2.3.1 Game Platform

FightingICE is a Java based game platform organised and maintained by Intelligent Computer Entertainment Lab., Ritsumeikan University. The game is based in an arena where two fighters are competing versus each other, attempting to deplete the other's hit-points while preserving their own. The FightingICE platform was designed to allow easy development and evaluation of artificially agents in the game for research or hobby purposes. Once implemented, an agent receives information about the state of the game from the platform periodically, such as the opponent player's location and current energy levels. A delay is added to this game information in order to simulate the delay a human player would experience from reaction time.


### 2.3.2 Game Agents

There are four characters available in the game: Zen, Garnet, Lud, and Kfm. Each of the characters is capable of moving, performing attacks, and combining these attacks into unique combos. The game starts with each player at 0 hit-points and ? energy. Once a player successfully connects an attack against its opponent, the player's energy is increased and the opponent's hit-points decrease. In order to compete against an opponent, the character must dodge opposing attacks and make effective use of energy to land attacks and reduce the opponent's hit-points.

### 2.3.3 Related Projects

Discuss and reference relevant projects which have been completed using the FightingICE framework. Discuss what has been achieved in said projects and the potential further research that can be undertaken.

# 3 Organisation

## 3.1 Project Task Analysis

The objective of this project is to evolve an agent in the FightingICE game platform with a neuroevolution method. The agent will be evolved to the point of being competitive versus a human opponent.

## 3.2 Requirement Analysis

### 3.2.1 Table of Requirements

| No. | Requirement | Priority | Predecessors |
|---|---|---|---|
| 1 | Implement a neural network to control agent's behaviour | High | |
| 1.1 | Initialise network with random weights, test if sensors can be read and actions output | High | |
| 1.2 | Implement a simple rule-based agent for the agent to compete versus | High | 1.1 |
| 1.3 | Test and execute at least one agent from the FightingICE AI competition | Medium | |
| 2 | Implement an appropriate evolutionary algorithm to evolve the neural network | High | 1 |
| 2.1 | Test whether a simple evolutionary algorithm evolving the weights of the network improves the agent's performance | High | 1 |
| 2.2 | Adapt and use the HyperNEAT method, test whether it improves on the simple algorithm | Medium | 1, 2.1 |
| 3 | Create an incremental evolution environment for the agent | Medium | 1, 2 |
| 3.1 | Limit the capabilities of the opponent and incrementally return them to test whether speed of evolution is improved | Medium | 1, 2 |
| 3.2 | Evolve against a simple agent, replace opponent with best agent after convergence and continue | Medium | 1, 2, 3.1 |
| 4 | Evaluate the agent's performance versus a human opponent at various stages of its evolution | High | 1, 2 |

### 3.2.2 Requirements Textual Descriptions

**1 - Implement a neural network to control agent's behaviour**

**1.1 - Initialise network with random weights, test if sensors can be read and actions output**

**1.2 - Implement a simple rule-based agent for the agent to compete versus**

**1.3 - Test and execute at least one agent from the FightingICE AI competition**

## 3.3 Performance Assessment

### 3.3.1 Table of Project Prototypes

| Objective | Date | Assessment |
|---|---|---|
| Prototype 1: Agent using neural network to sense environment and output simple actions in FightingICE | 06/01/17 | Agent can proficiently perceive its environment and output simple actions |
| Prototype 2: Agent controlled by neural network being evolved by simple evolutionary algorithm, altering only the weights of the network | 20/01/17 | Evaluate agents performance versus simple AI opponent at different stages of its evolution |
| Prototype 3: Agent evolved with HyperNEAT or other appropriate neuroevolution method | 17/02/17 | Evaluate agent's performance versus AI and compare whether method was more effective than simple EA |
| Prototype 4: Agent with appropriate learning method implemented in incremental evolution environment | 10/03/17 | Agent's environment is adapted to utilise incremental evolution. Agent's performance assessed and compared versus learning without incremental environment. |

### 3.4 Risk Assessment

# 4 Appendices

# References


[1] Wilde, P (1997) **Neural Network Models** [book] P4-P14

[2] Risi, S & Togelius, J (2015) **Neuroevolution in Games: State of the Art and Open Challenges** [online] Available at: https://arxiv.org/pdf/1410.7326.pdf [Accessed 03 Nov. 2016]

[3] Pace, A (2014) **Improving AI for simulated cars using Neuroevolution** [online] Available at: *http://commerce3.derby.ac.uk/ojs/index.php/gb/article/view/3/1* [Accessed 07 Nov. 2016]

[4] Lampropoulos, A (2005) **Machine Learning Paradigms** [online] Available at: *http://file.allitebooks.com/20150722/Machine%20Learning%20Paradigms-%20Applications%20in%20Recommender%20Systems.pdf* [Accessed 18 Nov. 2016]

[5] Gomez, F & Miikkulainen, R (1997) **Incremental Evolution of Complex General Behaviour** [online] Available at: *http://nn.cs.utexas.edu/downloads/papers/gomez.adaptive-behavior.pdf* [Accessed 18 Nov. 2016]

[6] Batsford, T (2014) **Calculating Optimal Jungling Routes in DOTA2 Using Neural Networks and Genetic Algorithms** [online] Available at: *http://commerce3.derby.ac.uk/ojs/index.php/gb/article/view/14/12* [Accessed 18 Nov. 2016]

[7] Hausknecht, M & Lehman, J & Stone, P (2014) **A Neuroevolution Approach to General Atari Game Playing** [online] Available at: *https://www.cs.utexas.edu/ mhauskn/papers/atari.pdf* [Accessed 15 Nov. 2016]

[8] Braylan, A & Hollenbeck, M & Meyerson, E & Miikulainen, R (2015) **Reuse of Neural Modules for General Video Game Playing** [online] Available at: *https://arxiv.org/pdf/1512.01537v1.pdf* [Accessed 17 Nov. 2016]

[9] Dehuri, S & Ghosh, S & Cho, S-B (2011) **Integration of Swarm Intelligence and Artificial Neural Network** [book] P1-P23