Universitat Politècnica de Catalunya
Facultat d'Informàtica de Barcelona

Bachelors Degree in Informatics Engineering

Implementation of the Environment of a Practical Work in a Computational Intelligence Course

Bachelor's Thesis of Prashanth Sridhar

Director: RENÉ ALQUÉZAR MANCHO
Co-Director: ENRIQUE ROMERO MERINO

20th June 2019

Abstract

Machine Learning & Artificial Intelligence are currently the cutting edge technologies that can help solve the various problems present in the current world we live in. There is a need for AI/ML engineers/researchers who can learn, code and deploy technologies powered by AI to solve such problems. Colleges like UPC-FIB provide courses in the same field to help students gain the required knowledge to solve problems with ML/AI. A course curriculum isn't complete without practical knowledge. One needs hands on experience with coding to truly understand & master the concepts.

My project aims to provide a practical environment for the "*Computational Intelligence*"[4] course taught at UPC-FIB. The aim of the practical work is to train different Computational Intelligence models that are able to play different simple games. The practical work environment should allow the students to code and test various intelligence models that they've learnt in class.

The idea is to make the practical course as intuitive and interesting as possible. Hence, this project will involve the use of Video Games as a method to generate interest. The student will have to develop Intelligence models to train the game agent to play a video game with human-level performance. All the intelligence models that are to be implemented are in accordance with the course curriculum.

The environment should allow the student to play the game manually. It should have the implementation of the function *generate_data,* which should allow the student to choose the game and select the amount of data to generate for training. The main interest of the environment is the automatic game simulator functions. These are the functions that are used to control the game using a computational intelligence model. These functions are to be completed by the students as the practical work for the course. The student can choose from various intelligence models *(Multilayer Perceptrons, Evolutionary Algorithms, Fuzzy Inference Systems etc)* to control the game. These models are in line with the syllabus of the theory course. The environment should also contain ideal good performing models that can be used by the students to compare and evaluate their models.

Acknowledgement

First and foremost, I would like to thank the Almighty for giving me the opportunity to work and complete my bachelor’s thesis at a prestigious institution like Universitat Politècnica de Catalunya.

The success and final outcome of this project, required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project.

I would like to express my sincere thanks and gratitude to my project directors
Prof. René Alquézar Mancho and Prof. Enrique Romero Merino who took keen interest in my project work and guided me all along, till the completion of my project work by providing all the necessary information for developing a good application.

I would also like to thank Universitat Politècnica de Catalunya for letting me use their resources which played a huge role in the completion of the project.

Lastly, I would like to thank my parents for their eternal love, support & confidence.

Table of Contents

Index of Figures

Index of Tables

Chapter 1

Introduction

1.1 Motivation

We live in a generation full of technological advancements. Gone are the days when almost everything was done manually, and now we live in a world where most of the work is automated by machines which are powered by Artificial Intelligence.

Artificial intelligence or AI is basically giving a machine the ability to think and act like a human, giving the machine the power to do a job which only humans can do. Thanks to this technology, people have been able to build AI powered machines which have solved many problems around the world & have bettered the lives of so many people . So it is considered important for a computer science engineer to be well versed with the concepts involved in the field of Artificial Intelligence. The following chart illustrates the growth of AI in the research community in the recent years [1] [2]. *Figure 1* highlights the growth & importance of the field.

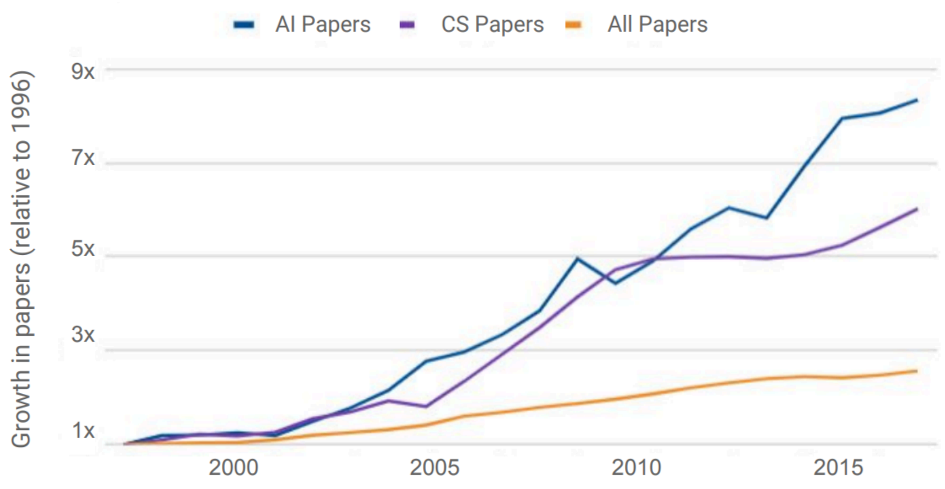


Figure 1: Growth of annually published papers (1996–2017) Source: Scopus [2]

Many universities (including UPC) have courses in the field of Machine Learning & Computational Intelligence so as to prepared students & make them ready to solve problems with AI & ML and contribute to the industry. Courses in artificial intelligence and Machine Learning have become more popular in the recent times. *Figure 2* highlights the increase in popularity among the university students [2].

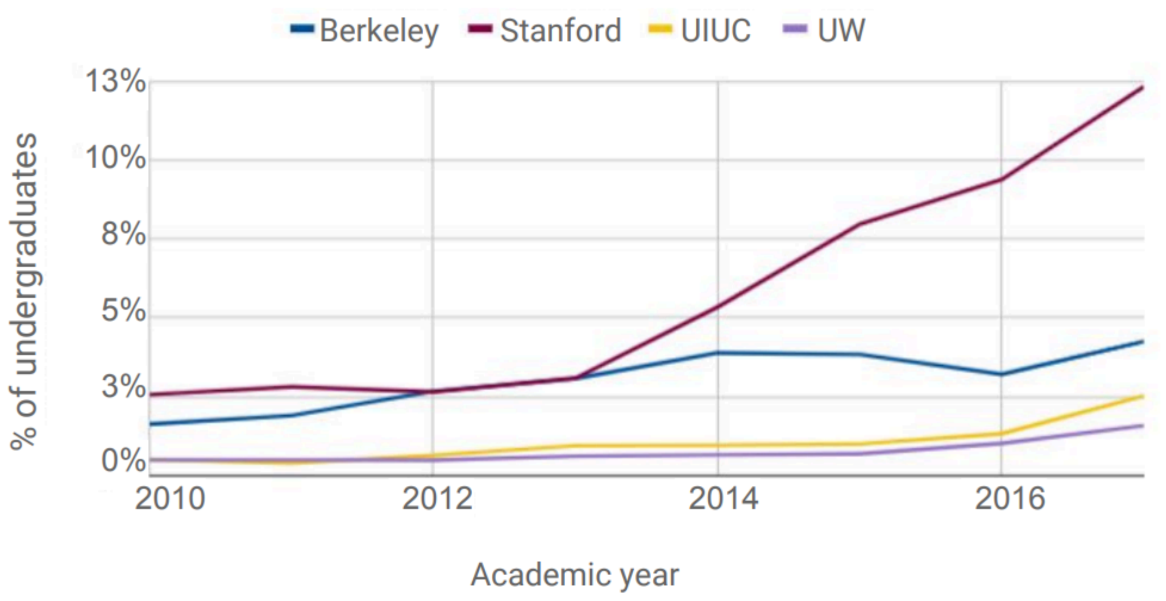


Figure 2: Percent of undergraduates enrolled in Intro to AI (2010—2017) Source: University provided data

Apart from theory, practical hands-on programming knowledge is considered important for effective learning of the subject. Hence, Universities have practical classes along with the theory classes. In UPC-FIB, the course “Computational Intelligence” requires a practical component.[4]

The aim of this project is to build a practical work environment for the course where students can code and test intelligence models. At the end of the day, the project will help students better understand the concepts taught in the theory classes by implementing the ideas themselves in the practical environment.

The idea is to make the practical course as intuitive and interesting as possible. Hence, this project will involve the use of Video Games as a method to generate interest. The student will have to develop Intelligence models to make an AI play a video game with human-level performance.

1.2 The Computational Intelligence Course at FIB, UPC

The aim of this course is to provide the students with the knowledge and skills required to design and implement effective and efficient Computational Intelligence solutions to problems for which a direct solution is impractical or unknown. Specifically, students will acquire the basic concepts of fuzzy, evolutionary and neural computation. The student will also apply this knowledge to solve some real case studies.[4]

This is a mandatory course for all students enrolled in the Masters Program in Artificial Intelligence. This is shared by UPC, UB and URV

Keeping this in mind, the practical environment is implemented.

1.3 Objectives

"***To Implement the Environment of a Practical Work in a Computational Intelligence Course***"

The objective can be elaborated as follows:

- The aim of this practical work is to train different Computational Intelligence models that are able to play different simple games.(snake, etc).
- The environment should allow the student to play the game manually.
- The environment should provide a testing interface for the student to test their CI models which will power the agent in the game.
- The student can choose from various intelligence models (Multilayer Perceptrons, Evolutionary Algorithms, ...) to control the game.
- These models are to be in line with the syllabus of the theory course.
- Reliable references at evaluation time.

1.4 Methodology

These are the steps that need to be carried to ensure a successful completion of the project within the deadline

- Choose & build suitable games that are intuitive for learning.
- Collect sufficient good quality game data for training of the models.
- Clean data if necessary such that it is suitable for training the decision making agent.
- Experiment on more models to see which works best for the game type etc.
- Integrate into interface.
- Add necessary features on the interface to make the application user friendly.
- Deploy & Test on the practical computer labs.

Chapter 2

Scope, Stakeholders & Challenges

2.1 Scope

The original idea of the project is to generate an environment for a practical work in the master course “Computational Intelligence” at FIB-UPC [4], The aim of this practical work is to train different Computational Intelligence models that are able to play different simple games (snake, etc).

The environment should allow the student to play the game manually. It should have the implementation of the function *generate_data*, which should allow the student to choose the game and select the amount of data to generate for training. The main interest of the environment is the automatic game simulator functions. These are the functions that are used to control the game using a computational intelligence model. These functions are to be completed by the students as the practical work for the course. The student can choose from various intelligence models (Multilayer Perceptrons, Evolutionary Algorithms, Fuzzy Inference Systems) to control the game. These models are in line with the syllabus of the theory course. Additionally, I should put myself in the student’s shoes and perform the practical work so that we can have reliable references at evaluation time.

In Summary,

The project should generate the environment of the practical work described above. Specifically, it should give implementation of:

- The program generate_data
- The “manual” game simulators (play_snake_manual)
- The “automatic” game simulators (play_snake_CI)
- An interface for the students to test & evaluate their models.

2.2 Stakeholders

The course “Computational Intelligence” at FIB, UPC is handled by Professors Reńe Alquézar, Enrique Romero & Angela Nebot. Therefore, the stakeholders are the teachers of the course who also happen to be my mentors for the project.

2.3 Challenges

- **Bad Interface Design**

A poor practical environment design could easily produce an application that might not do the job it is intended to do. Having a bad interface might not be user friendly for the students. So the purpose is not served as the student doesn’t learn. This problem was overcome by having frequent meetings with the teachers of the course to get a better understanding of the feature requirements.

- **Bad Performing Models/Problems with Result Interpretation**

Selecting wrong intelligence models or improper features might result in poor models. As a result, the environment won’t have a good reference for the students to compare & test the performance of their models. Sometimes, anomalies might occur which might in turn cause results which might not be easy to interpret.
This was tackled by having weekly meetings with my project mentors, who are specialists in the field of computational intelligence. They were able to guide me and supervise the project.

- **Incompatibility Issues**

Majority of the project is developed on a MacOS machine. Since most of the elements in the application use graphical libraries (To build the games, etc), there is a chance we might run into incompatibility issues. Different operating systems support only certain graphical libraries. This leads to incompatibility issues in operating systems which do not support these libraries.

This can be solved by either building an application that's free of dependencies that change depending on the OS

- **Choice of Programming Language**

Different students might be comfortable with programming in their language of choice. The application is built completely with Python. This means, students who prefer coding on MATLAB might not be able to run their models with the application.

This was solved by adding support for MATLAB as well as Python.

Chapter 3

Project Management

3.1 Overview

3.1.1 Estimated project duration

The estimated duration of the project is 5 months. The work starts on 12th February. The work is to be completed by June 30th

3.1.2 Considerations

The requirements for the project are fully specified, but more requirements and features may be added to the process (eg. a new game or new computational models) later depending on the amount of time left. The idea is to complete the current requirements as soon as possible to have enough time to implemented other requirements, if any.

Furthermore, during April 2019, the development of the project will be paused for about 10 days due to the spring holidays.

3.1.3 Planning & Feasibility

Most of the planning and feasibility analysis was conducted thanks to the Project Management Course. This course first required me to define and throughly understand the problem I was trying to solve. Next section of the course required me to device out an effective schedule plan to complete the project on time. This meant that i had to breakdown the development lifecycle into sections and assign time accordingly. Next, I had to do a financial analysis to calculate the cost of completing the project. Finally, I had to take into consideration all the socio-economic factors that might come into play.

3.2 Planning

3.2.1 Project Iterations

As mentioned in the scope definition. The project is going to be divided into many iterations where different modules are developed separately. As as when the modules are completed, they are integrated.

1. **Learning & Setting up.**
   The aim of this iteration is to learn the required languages, frameworks etc to be able to develop and build the application. This involves learning about MATLAB, Pygame, the python application framework & the framework to link python with MATLAB to use the MATLAB engine within python scripts. This iteration is also to prepare the environment and install the necessary frameworks to develop the application. Once all frameworks are installed, it will be necessary to configure them in order to start the next iteration.

2. **Game Development**
   In this iteration, the game (eg. Snake) is developed ground up using pygame. The game should be built such that all the features required to train the CIs are easy to access. The agent should be able to observe the environment completely such that the necessary features can be extracted and used for the Computational Models.

3. **Data Collection.**
   The aim of this iteration is to collect sufficient data for training. Based on the features that can be selected, the data is to be collected. As a result, training can take place leading to well performing models.

4. **Experimentation**
   The iteration focusses on trying different architectures etc, like switching up different hyperparameters so as to make the CI model perform as efficiently and effectively as possible. Using the link between Python and MATLAB,

Neural Nets are to be built using MATLAB. All computation and decision making features are to be implemented on MATLAB. At the end of this phase, I hope to have well performing Computational Intelligence models

5. **Integration & Application Development**
   All the models which power the agent's decision making, the game are integrated into the application & all the necessary interface and menus are added to build the practical environment.

6. **Final Stage**
   The final stage consists on closing the project development definitively. A final report will be generated for the work that has been done. A user manual for the students will be made to make the learning easier when the students use the platform to learn. A final presentation will be delivered.

3.2.2 Estimated Time

Stage		Estimated dedication (hours)
Planning and feasibility		30
Analysis and design		30
I0	– Learning & Setting up	40
I1	– Game Development	60
I2	– Data Collection	60
I3	– Experimentation	150
I4	– Integration & application development	150
Final stage		50
Total		570 hours

Table 1: Estimated Time

3.2.3 Gantt Chart

The Gantt Chart is the most important aspect of project management. It provides a graphical illustration of a schedule that helps to plan, coordinate, and track specific tasks in a project.

The Gantt chart is as follows :

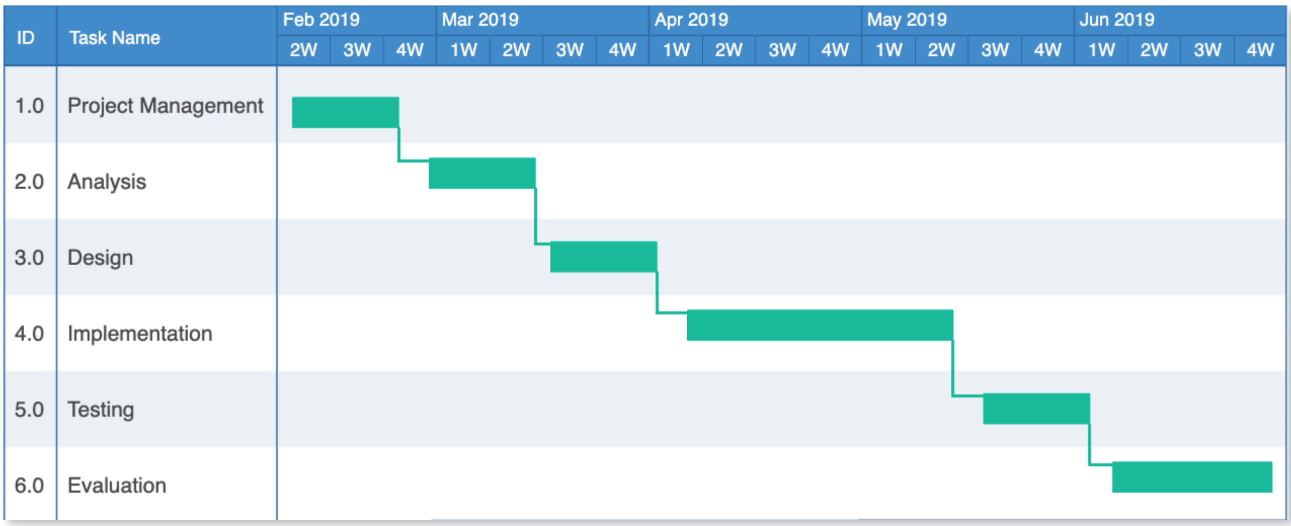


Figure 3: Gantt Chart

3.3 Resources

The following resources were used to complete the project

Hardware:
Apple Laptop: MacBook Pro 2015

Software:
Python
MATLAB
Sublime Text
Pages (Like word for MacOS) Numbers (Like Excel for MacOS)

Most of the coding is done using Python. The application, the interface, the games, are all developed using Python. The computational intelligence models alone are coded and run in MATLAB as the programming language used in the course is MATLAB. By using an interface package, a connection is established between Python and MATLAB. All the computation is done using the MATLAB engine and the results are returned to Python. Since I am the only person working on the project, there is no need to co-ordinate with a third person to develop the project.

With respect to writing reports & making presentations for both the GEP course as well as the final project defence, Pages & Numbers have come in handy to help write neat and clear reports.

3.4 Project Budget

3.4.1 Financial Planning

Considerations

An estimation of the cost of this project is presented by dividing the expenses int their respective categories. They are hardware, software & human resources amortizations.

Since there is only one individual involved in the project, I will take up the various necessary roles required for the completion of the project. (eg Project Manager, Software Dev, Designer etc.)

Budget Monitoring

This is only a rough estimate. As and when the phases of the project gets completed, the budgets are revised. This way of monitoring the amount of spending will make sure there is no excess money spent anywhere.

3.4.2 Budget Estimations

*Amortized Cost = Cost * (No of Yrs in Use/Useful Years)*

The rough duration of the project is 5 months. So "*No of Yrs in Use*" is the duration of the project, which is 0.416.

A. Hardware

Except for a laptop and a mouse, no other hardware equipment/entity is required as this project is entirely software based. There are no other hardware components required.

Hardware Cost Estimate

Product	Price (Euros)	Useful Years	Amortization (Estimate)
Apple MacBook Pro 2015 (8GB RAM)	1000	5	€ 83.3
Mouse	15	2	€ 3.12
Hard drive	50	5	€ 4.16
Total Estimate	1065		€ 90.58

Table 2: Hardware Cost Estimate

B. Software

Software Cost Estimate

Product	Price (Euros)	Useful Years	Amortization (Estimate)
Python 3	Free	NA	€ 0
MATLAB 2018 License	800 € /Year	1	€ 332.8
Sublime Text	Free	NA	€ 0
Xcode	(Included under Hardware Cost)	NA	0
Google Chrome	Free	NA	0
Jupyter	Free	NA	0
MacOS & Other OS Software	(Included under H/W costs)	NA	0
Total Estimate			€ 332.8

Table 3: Software Cost Estimate

C. Human Resources

As mentioned earlier there is only one individual involved in the project, I will take up the various roles required for the completion of the project. (eg Project Manager, Software Dev, Designer etc.)

Human Resource Cost Estimate

Role	Estimated Hours	Price per hour (Estimate) €	Total Estimate €
Project Manager	120	50	6000
Software Developer	200	35	7000
Software Designer	100	30	3000
QA & Test Engineer	100	20	2000
Total Estimate	520 Hours		€ 18000

Table 4: HR Cost Estimate

Total Estimated Budget

Category	Amortization (Estimate)
Hardware	€ 90.58
Software	€ 332.8
Human Resources	€ 18000
Total Estimate	€ 18423.38

Table 5: Total Estimated Budget

3.5 Sustainability

We analyse the sustainability of the project under three main categories: Economic, Social & Environmental Sustainability. They have been described in detail, respectively in the following sections

3.5.1 Economic

This project is economically, a very sustainable incentive, as apparent from the budgets that have been estimated. All aspects of the budget have been calculated and taken into consideration. There might be some indirect and unforeseen costs, that may have escaped our estimate.

Most of the software used in this project is free and open sourced. MATLAB is the only software that requires a working license. This project is completely software based and does not require any hardware to function expect for the system it runs in. This makes it very economical as there is no money spent on extra hardware. So there won't be future expenses to replace any hardware parts.

Overall, this project is very cost effective and is one of the cheapest solutions to the problem in hand. Since the project is to be deployed in university labs, there is no need to take the extra step to get a working license as MATLAB comes installed with an activated license for every system in the lab. Since this is a perpetual license, there is no extra cost incurred after the initial setup. Considering all the factors, this project deserves a 8 on 10, in the Economic Sustainability Analysis

3.5.2 Social

Our project has the biggest impact in the social front. It is a huge value add to the students taking the Computational Intelligence Course at UPC.

The primary idea of this application is to serve as a learning & testing tool. Students will find the features very intuitive and will overall have an enriching learning experience. The application is to be used with the course. The application serves as a testing tool to practically implement the concepts taught in class.

Also, this implementation is more user friendly as compared to the other existing solutions. Plus there is flexibility to further add support for the project later in the future. This means, there is going to be support for more algorithms etc. This has a huge potential of becoming a tool that can make a huge impact in a student's learning process. Considering the usefulness and positive learning impact, this deserves a 9 on 10, in the Social Sustainability Analysis.

3.5.3 Environmental

During the development of the project, the only environmental concern is the consumption of electricity. Electricity is what powers the laptop that enables the development. But, you cannot look at it as an environmental concern as laptops and other electronic devices have become a part of our lifestyle. Our lives depend on electronic devices. Plus the amount of power consumed by the system to run the application is bare minimum as the application isn't heavy. The application is friendly on the processors, so there is no need for extra power to run this application. So we can safely ignore the electricity factor.
This project is entirely software based. There are no hardware components that interact with the surroundings. So there is no e-waste generated. In that aspect, this project is very environment friendly as there is no component that could possibly be of harm to the environment.

Keeping all this in mind, we can reward this project a score of 9 on 10, in the Environmental Analysis

Aspect	Score
Economic	8
Social	9
Environmental	9

Table 6: Sustainability Scores

Chapter 4

Understanding Snake

4.1 Overview

The main purpose of the project is to use Video Games as a learning tool for the students.[6] As mentioned before, the aim of the practical work is to build various computational intelligence models to train agents to play video games with human level performance.[7]

Specifically, In this project, the idea is to use the classic video game 'Snake' as the problem.
The practical work would require the student to understand the problem, collect/generate data, select relevant features, choose an intelligence model, train the model & finally test the model on the game to see how well it performs (plays the game).

So before we get into the details of solving the problem with AI, we need to first understand the problem we are trying to solve.

4.2 Rules of Snake

Snake has simple rules:

- The world is a grid.
- The snake can only travel orthogonally along this grid.
- This world has a border that kills the snake on contact.
- The snake cannot stop moving.
- If the snake runs into itself, it dies.
- Every time the snake eats, it grows longer.
- The goal is to grow as long as possible.

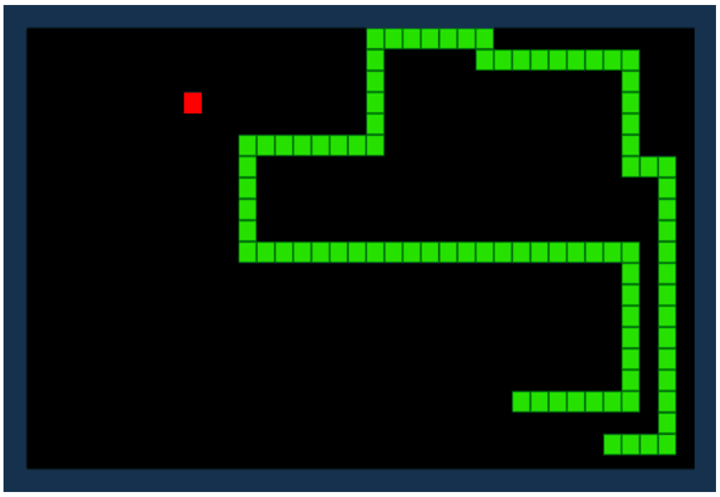


Figure 4: Snake Example

When playing the game, there is a decision to make each time the snake takes a step forward: continue straight, turn left, or turn right.
Our goal is to create an AI to learn how to make this same decision. First assessing the state of the world that the snake lives in, then choosing the move that will keep it alive and continue to grow longer.

4.3 Creating the Game

The game has been built entirely from scratch on Python with the help of a graphics package called PyGame.
The game has been built entirely from Scratch to make sure all the game environment variables are accessible. Some of the environment variables include the positional coordinates of the candy, the information about the spaces that are free/occupied etc. This way, depending on the feature that is selected, the required data can be fetched from the game environment using an API.

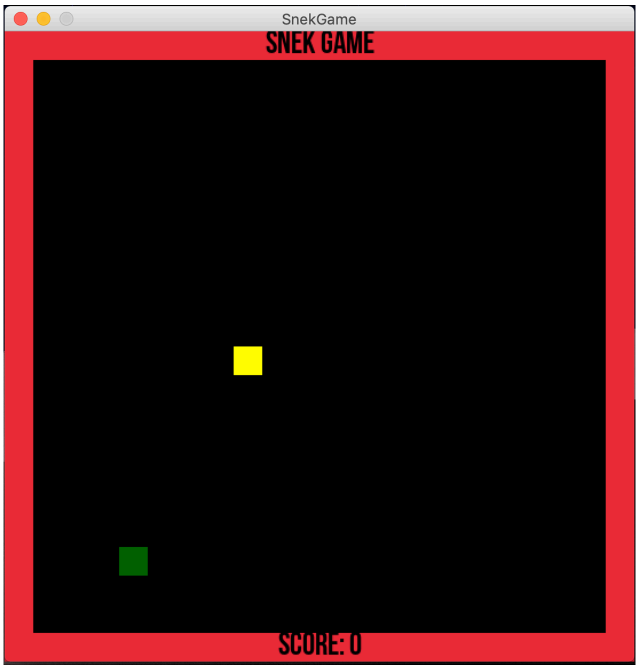


Figure 5: Snake Game

4.4 Brining AI in the Mix

The agent should

- Observe the environment
- Decide which action is the most optimal
- Make the decision (AI)
- Receive rewards based on the action made
- Repeat

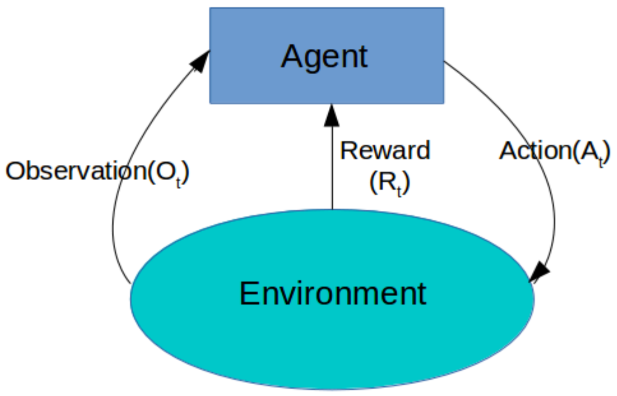


Figure 6: AI Agent Structure

Chapter 5

About the Data

5.1 Overview

In this section, we shall discuss the nature of data required and the various methods that are used to create the data.

5.2 Understanding the Requirement

Before we proceed to collect data, we need to first understand the difference between good and bad data.
Since the project is fundamentally about trying to build a really good AI that can play as '*good*' as a human, we need to collect data that corresponds to a really good player. So the definition of good data is game data generated from a good player. This means having a really good player play the game or coding a really good heuristic that can mimic and play as good as a human player.
So keeping this in mind let us get into the specifics.

5.3 Obtaining the Data

Whenever you think of solving a problem with Machine Learning, the first thing problem we face is finding the right data. We face the same problem here.
If we need to build AIs to play 'Snake', we need a game that can give us the data (features) we need. This means the game should have an API that should let us access the the required information from the environment. This basically forces us to build the entire game of Snake from scratch.
All the data we need to collect has to be obtained from the game environment using the game API. Thankfully, since I built the game from scratch, all the required environment variables are accessible and can be fetched using an API.

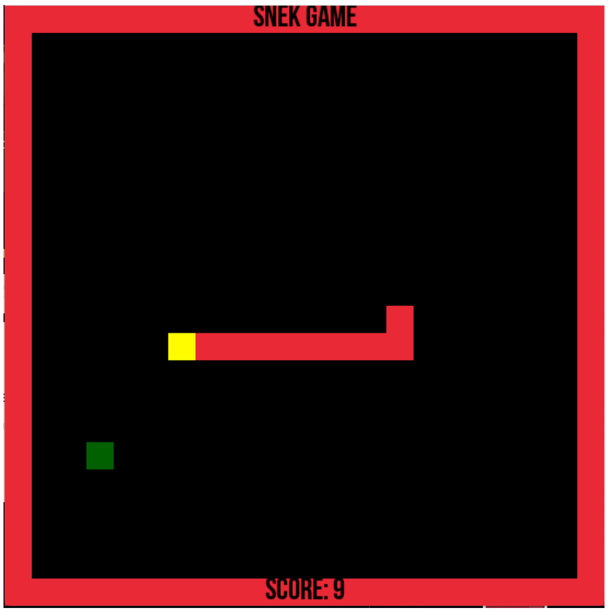
Therefore, the data can be obtained realtime from the game environment. At every instant in time (Game Time) the values need to be collected from the game environment.

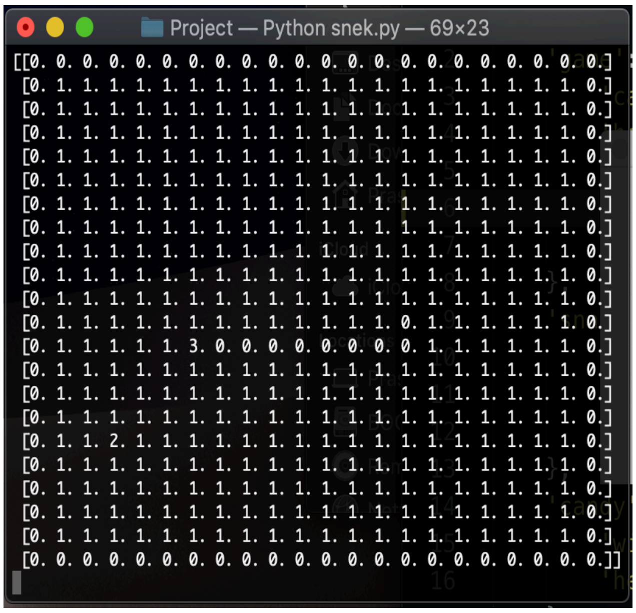
So depending on the features that are selected, the data can be obtained accordingly. This brings us to the next section, data representation.

5.4 Data Representation

At every instance in time, the positional information of the game elements can be obtained. This information tells us the pixel coordinates of the candy, the snake etc. Now this information needs to be used to interpret the surroundings of the Snake (eg Is the front space free ? How far is the candy from the head of the snake?).

To achieve that I generated a 2D array grid that represents the game grid at every instant in time. The 2D array has the same dimensions of the game grid. Each element is filled with numbers to represent free/empty spaces. This way, the game state is represented by the 2D grid. The values of the grid are updated as and when the values change.





```
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 3. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Figure 7: Game State Representation

States

0 - Walls and Spaces occcupied by the body of the Snake

1 - Free Spaces

2 - Candy

3 - Head of the snake

This representation makes it very easy to get the relevant data needed from the environment. The reason behind using numbers to present positions is simply because it makes the learning process easy. Neural Nets love training on simple whole numbers.

5.5 Feature Selection

In this section, I will explain in detail what features are good features. Any feature that helps the Snake understand its surroundings makes it a good feature. So we need to be smart & choose minimum features that best conveys the information the snake will need to play the game.

We need to look at this problem from the point of the snake. At a given point in time, the snake's vision is limited. Let's assume that the Snake can only look at the spaces immediately next to it. This means, it can check to see if the immediate spaces are empty/occupied. This information is really useful as it gives the snake a rough idea of its surroundings.

This can be a set of 3 features. Each feature representing one direction. The snake can look only orthogonally, therefore the snake can check front, left & right. We have successfully chosen 3 features that tell the snake about the spaces available for it to move into.

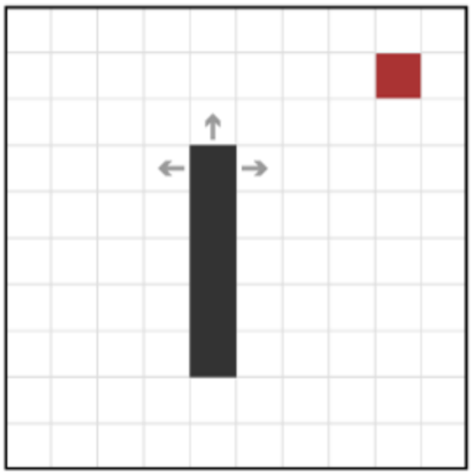


Figure 8: Features for spatial awareness

Next, it needs information about the candy. After all, the idea of the game is for the snake to get to the candy and that is how the score increases.

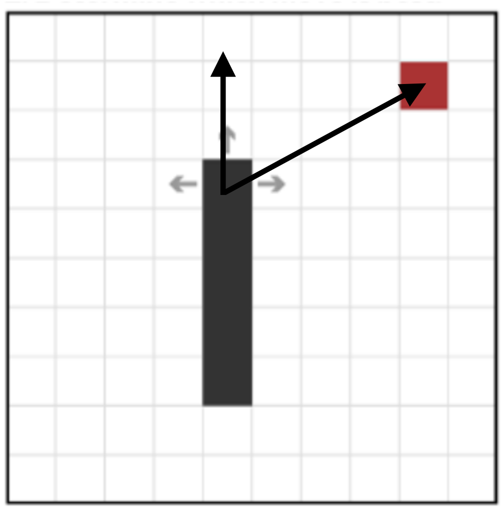
So the next feature we need to add should somehow effectively convey the positional information of the candy.

The best way to do this would be to calculate the angle made by the direction vector of the head of the snake with the direction vector of the candy relative to the head.

So these are the possible scenarios

1. The candy is to the right of the snake. Angle is positive & takes a value between 0 & 180
2. The candy is the the left of the snake. Angle is negative & takes a value between 0 & -180
3. The candy is directly in front of the snake. In that case the angle is 0
4. The candy is directly behind the snale. In that case the angle is 180.

These values can be discretised if needed to simply the data & training. I chose to scale these values to a range [-1, 1]



The Angle between the direction vectors is used to represent the relative position of the candy in the game environment.

Figure 9: Feature for identifying position of candy

In Summary, with effectively 4 features, we are able to give the snake knowledge about the free spaces next to it & give it information about the position of the candy. Features: [Angle, Front, Left, Right]. Example: [50,1,1,1]

5.6 Representing Output/Decision

The working of every intelligence model is to observe the surroundings & decide the best direction to move in. Since the snake can only move orthogonally, the snake is restricted to move along 3 directions at a given point in time.
The Snake can move **Straight**, take a **left** or take a **right.**
Each of these directions are represented by numbers.
0 - Straight
1 - Right
2 - Left

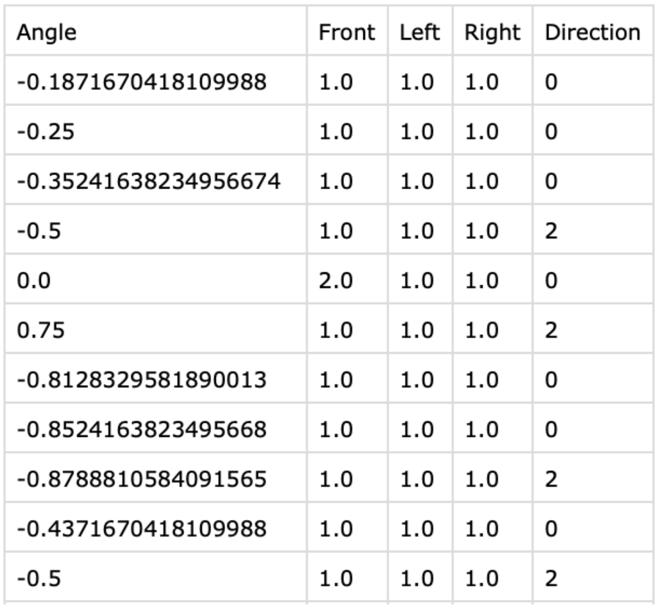
So at every instance in the game, the player needs to decide what direction to take next. If the game is played manually, the input is given from the keyboard with the help of the arrow keys.

When the agent is powered by a trained intelligence model, the model is given the features as input. The model needs to use those inputs and output a value. This value should effectively be a 0, 1 or 2 which represents the direction the snake needs to take for its next move.

5.7 Putting it all together

To create a good data set, we need to collect game data of a snake that plays really well. A snake that scores high corresponds to the quality of data that can be obtained.
How do we build this?

1. Play the game manually. Be an expert. Try not to make wrong moves. You will have good data with correct labels for a set of input.
2. Use a Genetic Algorithm to evolve a snake that can play the game well. Collect game data of that Snake.
3. Code a heuristic. Collect the data.



Angle	Front	Left	Right	Direction
-0.1871670418109988	1.0	1.0	1.0	0
-0.25	1.0	1.0	1.0	0
-0.35241638234956674	1.0	1.0	1.0	0
-0.5	1.0	1.0	1.0	2
0.0	2.0	1.0	1.0	0
0.75	1.0	1.0	1.0	2
-0.8128329581890013	1.0	1.0	1.0	0
-0.8524163823495668	1.0	1.0	1.0	0
-0.8788810584091565	1.0	1.0	1.0	2
-0.4371670418109988	1.0	1.0	1.0	0
-0.5	1.0	1.0	1.0	2

Essentially, this is what a typical game dataset will look like. The direction column represents the label. In other words, it is the decision made for that specific game state. The quality of the data determines the performance of the AI trained on the data.

Table 7: Sample Dataset

Note: The features selected to solve this problem is one way of the many ways of solving the problem. There are obviously other features that can be used. These features were chosen as they best represent the game environment and has given promising results.

Chapter 6

Models & Architecture

6.1 Overview

In this section, we shall discuss the various intelligence models used to train the game agents, explain the architecture & its working.

As mentioned earlier, the models used in this project are corresponding to the course curriculum.
The course 'Computational Intelligence' taught at UPC contains the following concepts.

1. Simple Neural Networks [8]
2. Evolutionary Algorithms: Genetic Algorithms - (Rule Based & Neural Network Based) [9,10,11]
3. Fuzzy Inference Systems (FIS Mamdani & FIS Sugeno) [12,13]

In this project, I solve the problem using all the above mentioned algorithms.
As for as the student is concerned, his practical work will require him/her to do the same as a practical work for the course with the help of the practical environment that is developed as a result of the project.

Structure

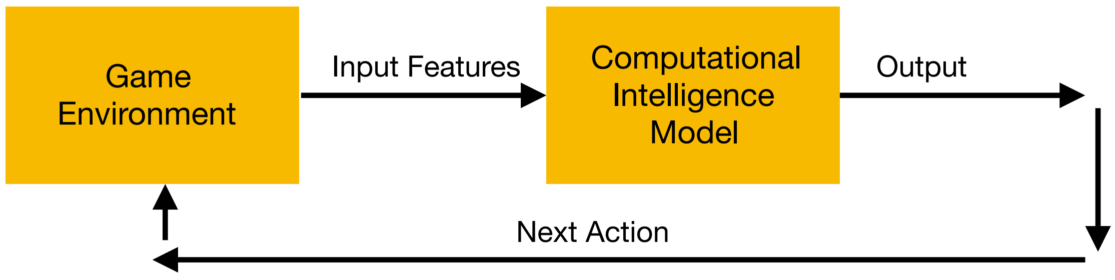
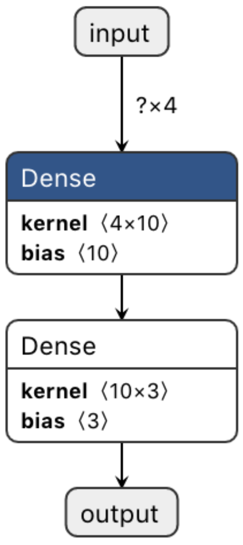


Figure 10: Working Structure

6.2 Simple Neural Network

The problem at hand was trying to predict the correct direction for the snake to pick at every given point in time. For a specific input that describes the current state of the game, the neural network should output the most optimal and safe direction for the snake to take.

6.2.1 Architecture



The neural network architecture that we used to train our snake is as follows:

Input Layer: 4 nodes.

1 node to represent each feature. [Angle, Front, Left, Right]

Hidden Layer: One fully connected layer of size 10 nodes. Activation Function : ReLU

Output Layer: A fully connected layer of size 3 nodes representing the 3 output classes (Straight, Left & Right) Activation Function: Softmax

Figure 11: NN Architecture

6.2.2 Working

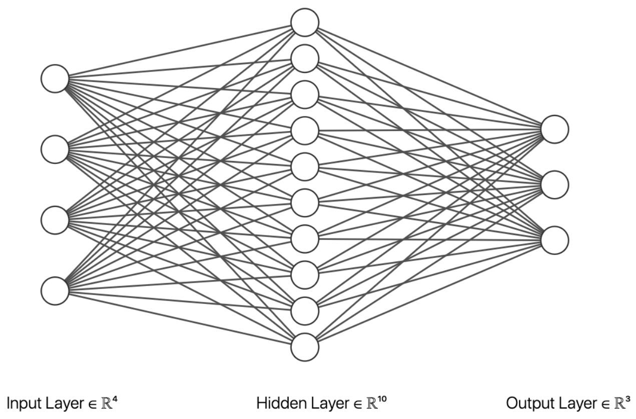


Figure 12: NN Structure

The neural network receives an input of size 1x4, a vector representing the current state of the game. These values are passed to the next layer where the edge weights are multiplied with the corresponding input values. The summed values that are entering each of the 10 nodes in the hidden layer are added to the bias value of the respective node in the hidden layer. Now these 10 values individually are sent to an activation function (ReLU). The outputs of the activation function (10x1) now progress to the final layer where

they are once again multiplied with the edge weights & summed with the bias values respectively. Now the three values that are present, one at each node are passed to a soft max activation function. Softmax converts these values into probabilities. It is used in classification problems to get probability values for each possible output class. The class with the highest probability is picked as the final output. In this problem, the output classes refer to the possible directions the snake can take. So the output of the neural network decides the next direction of the snake.
The weights & bias values of the neural network are learned through back-propagation. After learning, the weights and bias values that are learnt are good enough to power the decision making of the Snake.

6.3 Genetic Algorithms

In this section, we are going to talk about the working of genetic algorithms & how they can be used to evolve a snake that can play the game very well.
Each genetic algorithm requires two basic components.

1. A genetic representation of the solution domain
2. A fitness function to evaluate the solution domain

In this project, we apply the concept of genetic algorithms in two different ways. Both the methods have different genetic representations which make them unique. They are,

A. **Rule Based System**: Using the genetic algorithm, a set of rules are evolved and learnt over time.
B. **Neural Networks**: Using the genetic algorithm, the correct weights & bias values of the neural network are evolved and learnt over time.

Irrespective of the type of representation, the working of a genetic algorithm is same and has the following phases.

1. **Initial population**
2. **Fitness function**
3. **Selection**
4. **Crossover**
5. **Mutation**
6. **Repeat or Terminate**

Initial Population

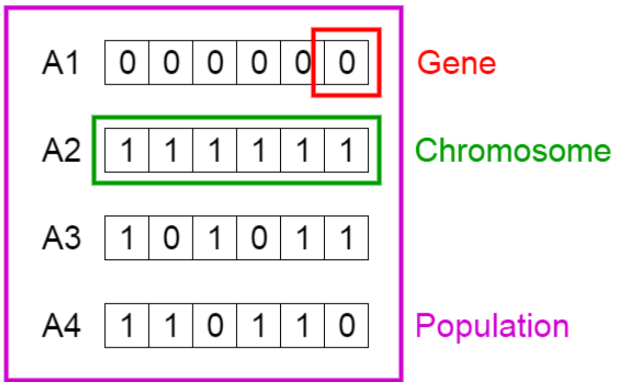


Figure 13: Genetic Representation

The process begins with a set of individuals which is called a Population. Each individual is a solution to the problem you want to solve. An individual is characterised by a set of parameters (variables) known as Genes. Genes are joined into an array to form a Chromosome (solution). This array of bits is what we call a chromosome. We encode the genes in a chromosome. The meaning of a gene in a chromosome changes depending on the type of representation.

Fitness Function

The fitness function determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a fitness score to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

In the case of Snake, the fitness values for each individual are calculated as follows:

+ **1000** for every candy the snake eats

- **150** for colliding with a wall or itself

The final fitness score at the end of the life of each snake are stored & used for selection.

Selection

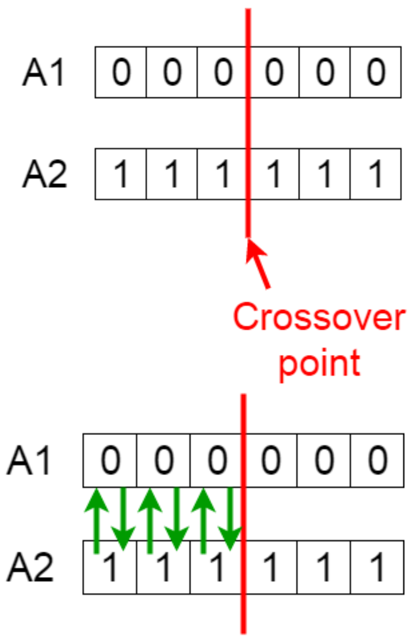
The idea of selection phase is to select the fittest individuals and let them pass their genes to the next generation.

Two pairs of individuals (parents) are selected based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction.

Crossover

Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes. There are different ways to perform crossover. This varies depending on how the genetic information is represented.

Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached. The new offspring are added to the population.



Mutation

In certain new offspring formed, some of their genes can be subjected to a mutation with a low random probability. This implies that some of the bits in the array can be flipped.

Mutation occurs to maintain diversity within the population and prevent premature convergence.

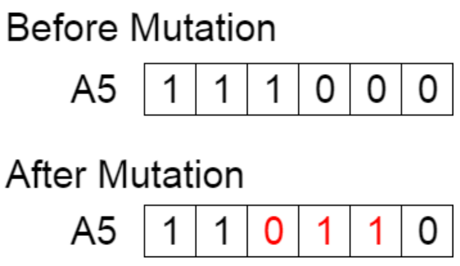


Figure 14: Crossover & Mutation

Repeat or Termination

After a new set of children are generated after crossover and mutation, these children are called the next generation of children. They once again undergo steps 1 through 5.

The algorithm terminates if the population has converged (does not produce offspring which are significantly different from the previous generation). Then it is said that the genetic algorithm has provided a set of solutions to our problem.

6.3.1 Rule Based System

In this section, we will look at how to use a genetic algorithm to train a set of rules. We will look at how to represent the genetic information and understand the working of the genetic algorithm to learn a set of rules.

6.3.1.1 Representation

In a rule-based system, the genetic information is represented as a set of rules. Each and every possible value in the solution domain is represented as a rule.

In this case, the input values are discretised so as to reduce the number of rules. So the angle values were reduced to a total of 4 possible discrete classes.

The 4 classes are as follows:

1. **Positive Angles** (0,1)   (Candy present to the right of the Snake)
Any angle falling between 0 and 1 is put under one class
2. **Negative Angles** (0,-1) (Candy present to the left of the Snake)
Any negative angle falling between 0 and -1 is put under another class
3. **0**   (Candy present directly in the line of sight of the Snake)
4. **-1/1**   (Candy present directly behind line of sight of the Snake)

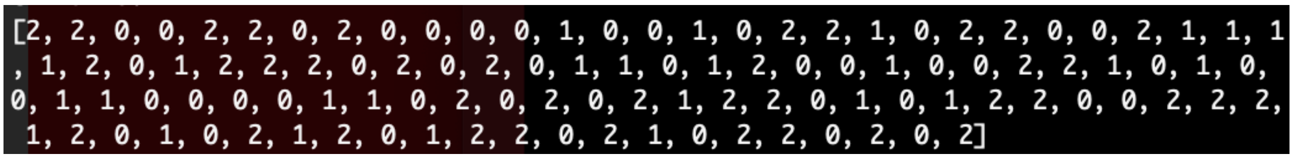
Now that the input features are discretised, we now have a finite number of combination of inputs.

Features:

Angle: 4 possible values
Front: 3 possible values
Left: 3 possible values
Right: 3 possible values

So the snake can encounter a total of 4x3x3x3 possible scenarios in the game. That is **108** different scenarios which together represent the domain.

So we need to create an array of size 108 to represent a unique individual in the population. The value present in each element of this array is the decision it needs to take for that corresponding game state/scenario. This array/genetic information contains the rule/decision the snake needs to take for every scenario it faces in the game. Good snakes are snakes which have the right set of rules. So for a game with 108 possible game states, each snake will have an array of size 108 where each element contains the decision the snake needs to take for that corresponding game state. This is the chromosome or the genetic information of a snake.



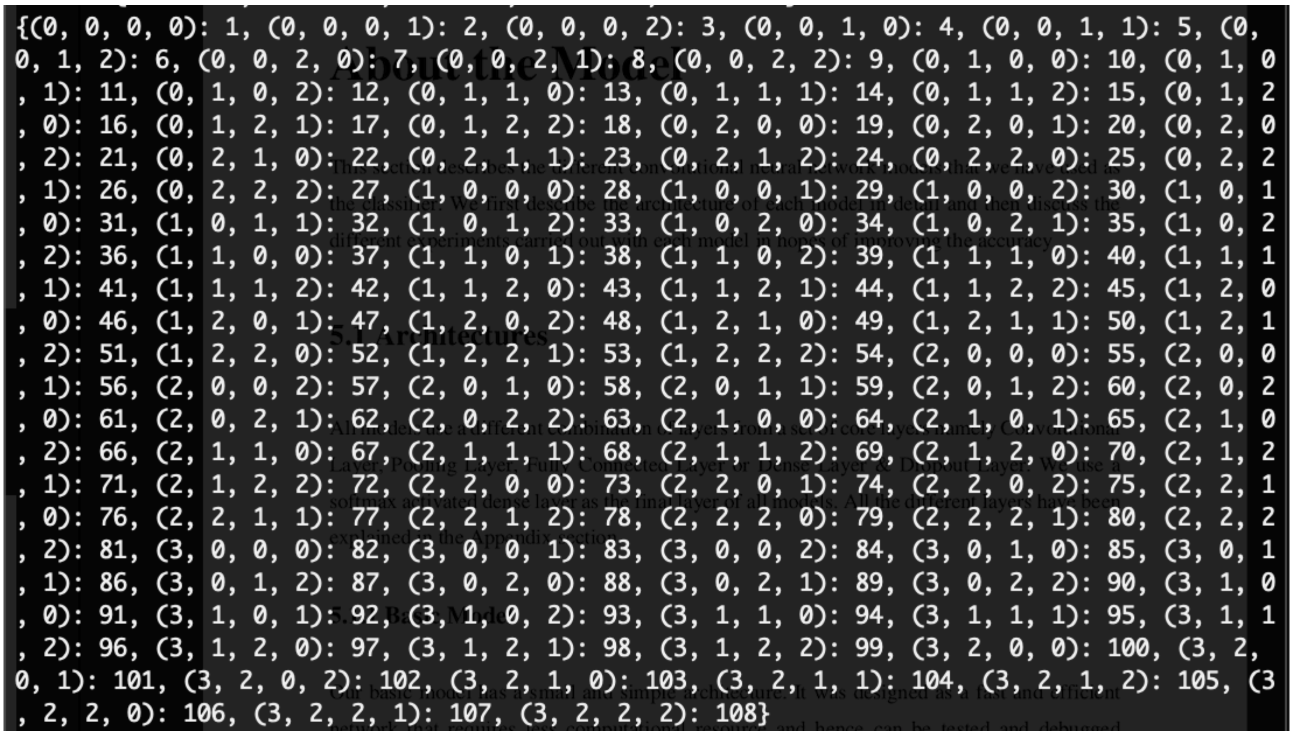
```
[2, 2, 0, 0, 2, 2, 0, 2, 0, 0, 0, 0, 1, 0, 0, 1, 0, 2, 2, 1, 0, 2, 2, 0, 0, 2, 1, 1, 1
, 1, 2, 0, 1, 2, 2, 2, 0, 2, 0, 2, 0, 1, 1, 0, 1, 2, 0, 0, 1, 0, 0, 2, 2, 1, 0, 1, 0,
0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 2, 0, 2, 0, 2, 1, 2, 2, 0, 1, 0, 1, 2, 2, 0, 0, 2, 2, 2,
 1, 2, 0, 1, 0, 2, 1, 2, 0, 1, 2, 2, 0, 2, 1, 0, 2, 2, 0, 2, 0, 2]
```

Figure 15: Example Chromosome

0,1,2 represent the directions the snake needs to take.

0 - Straight
1 - Right
2 - Left

Look Up Dictionary to translate a game state into the respective position in the chromosome. To find the gene that is responsible for a specific game state.



```
{(0, 0, 0, 0): 1, (0, 0, 0, 1): 2, (0, 0, 0, 2): 3, (0, 0, 1, 0): 4, (0, 0, 1, 1): 5, (0,
0, 1, 2): 6, (0, 0, 2, 0): 7, (0, 0, 2, 1): 8, (0, 0, 2, 2): 9, (0, 1, 0, 0): 10, (0, 1, 0
, 1): 11, (0, 1, 0, 2): 12, (0, 1, 1, 0): 13, (0, 1, 1, 1): 14, (0, 1, 1, 2): 15, (0, 1, 2
, 0): 16, (0, 1, 2, 1): 17, (0, 1, 2, 2): 18, (0, 2, 0, 0): 19, (0, 2, 0, 1): 20, (0, 2, 0
, 2): 21, (0, 2, 1, 0): 22, (0, 2, 1, 1): 23, (0, 2, 1, 2): 24, (0, 2, 2, 0): 25, (0, 2, 2
, 1): 26, (0, 2, 2, 2): 27, (1, 0, 0, 0): 28, (1, 0, 0, 1): 29, (1, 0, 0, 2): 30, (1, 0, 1
, 0): 31, (1, 0, 1, 1): 32, (1, 0, 1, 2): 33, (1, 0, 2, 0): 34, (1, 0, 2, 1): 35, (1, 0, 2
, 2): 36, (1, 1, 0, 0): 37, (1, 1, 0, 1): 38, (1, 1, 0, 2): 39, (1, 1, 1, 0): 40, (1, 1, 1
, 1): 41, (1, 1, 1, 2): 42, (1, 1, 2, 0): 43, (1, 1, 2, 1): 44, (1, 1, 2, 2): 45, (1, 2, 0
, 0): 46, (1, 2, 0, 1): 47, (1, 2, 0, 2): 48, (1, 2, 1, 0): 49, (1, 2, 1, 1): 50, (1, 2, 1
, 2): 51, (1, 2, 2, 0): 52, (1, 2, 2, 1): 53, (1, 2, 2, 2): 54, (2, 0, 0, 0): 55, (2, 0, 0
, 1): 56, (2, 0, 0, 2): 57, (2, 0, 1, 0): 58, (2, 0, 1, 1): 59, (2, 0, 1, 2): 60, (2, 0, 2
, 0): 61, (2, 0, 2, 1): 62, (2, 0, 2, 2): 63, (2, 1, 0, 0): 64, (2, 1, 0, 1): 65, (2, 1, 0
, 2): 66, (2, 1, 1, 0): 67, (2, 1, 1, 1): 68, (2, 1, 1, 2): 69, (2, 1, 2, 0): 70, (2, 1, 2
, 1): 71, (2, 1, 2, 2): 72, (2, 2, 0, 0): 73, (2, 2, 0, 1): 74, (2, 2, 0, 2): 75, (2, 2, 1
, 0): 76, (2, 2, 1, 1): 77, (2, 2, 1, 2): 78, (2, 2, 2, 0): 79, (2, 2, 2, 1): 80, (2, 2, 2
, 2): 81, (3, 0, 0, 0): 82, (3, 0, 0, 1): 83, (3, 0, 0, 2): 84, (3, 0, 1, 0): 85, (3, 0, 1
, 1): 86, (3, 0, 1, 2): 87, (3, 0, 2, 0): 88, (3, 0, 2, 1): 89, (3, 0, 2, 2): 90, (3, 1, 0
, 0): 91, (3, 1, 0, 1): 92, (3, 1, 0, 2): 93, (3, 1, 1, 0): 94, (3, 1, 1, 1): 95, (3, 1, 1
, 2): 96, (3, 1, 2, 0): 97, (3, 1, 2, 1): 98, (3, 1, 2, 2): 99, (3, 2, 0, 0): 100, (3, 2,
0, 1): 101, (3, 2, 0, 2): 102, (3, 2, 1, 0): 103, (3, 2, 1, 1): 104, (3, 2, 1, 2): 105, (3
, 2, 2, 0): 106, (3, 2, 2, 1): 107, (3, 2, 2, 2): 108}
```

Figure 16: Space Representation

6.3.1.2 Working

The aim of the genetic algorithm is to evolve the set of individuals over time & breed a snake that has the right set of rules (Genes) that allows the snake to play the game very well.

The working of a genetic algorithm is as follows:

1. Initialisation : Create a set of individuals initialised with random values. They represent the initial population. Since the values are random, the snakes don't perform great.
2. For every generation:
    A. Calculate the fitness values for each individual in the population.
    B. Preserve the best performing individual in that generation.
    C. Select 2 of the best performing individuals.

D. Uniform Crossover (33% Chance of Swap) & Create two new children
E. Mutate (1%)
F. Repeat B through D till you have a new set of individuals for next generation.
G. Terminate if converged, else move to next generation

6.3.2 Training Neural Networks

In this section, we will look at how to use a genetic algorithm to evolve the parameters of a neural network. We will look at how to represent the genetic information and understand the working of the genetic algorithm to train neural networks

6.3.2.1 Representation

In this method, the neural network functions as the brain of the snake. The brain is given information about the surroundings in the form of an input. The neural network performs operations on the input based on the weight and bias values. The output of probabilities is used to determine the decision the snake needs to take next. Here, the genetic information is represented using a neural network. The weights & bias parameter values act as the genetic information. They together function as the chromosome of the same. Refer *Figure 17.*

The idea is to evolve the weight and bias values using a genetic algorithm and eventually produce a neural network that has learnt the correct parameter values. This evolved neural network is used to power the decision making of the snake. So the chromosome is a one dimensional array. The size is dependent on the number of parameters that need to be trained. For this project, the structure of the neural network is as follows.

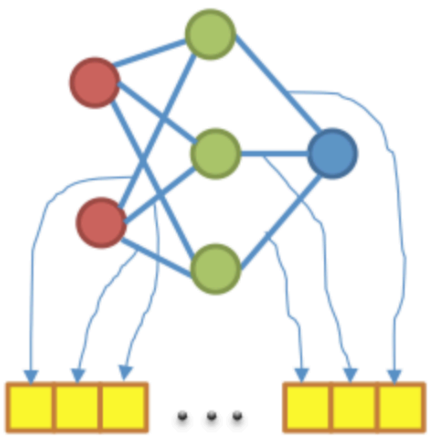
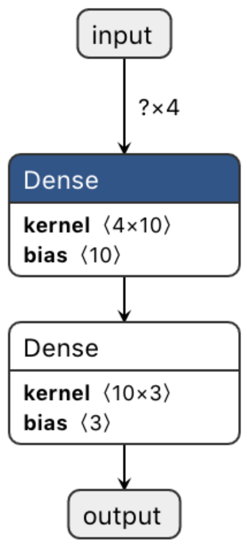
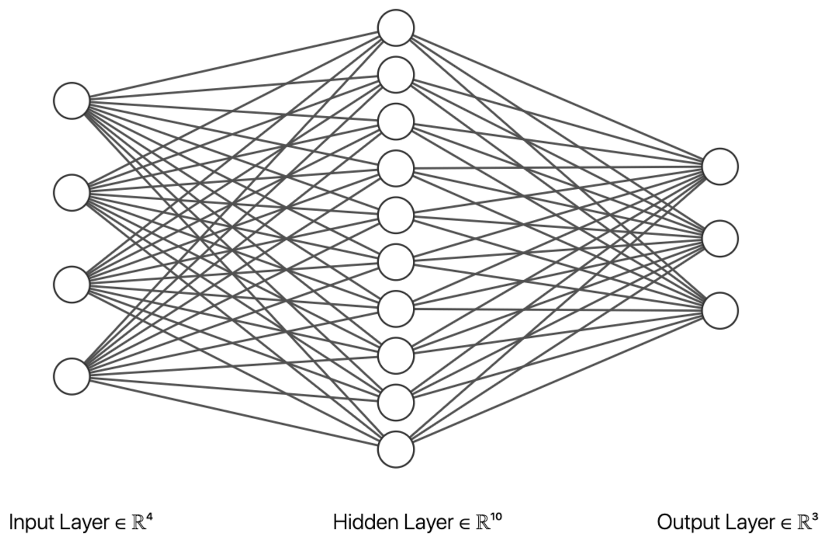


Figure 17: How to represent a NN as a chromosome





Here the structure of the neural network is fixed. The genetic algorithm is used to evolve (& learn) the parameter values.
There are neuro-evolutionary algorithms that can be used to change the structure of the Neural Network as well. These algorithms are used to evolve & learn the ideal structure for the problem. But that is beyond the scope of the project.

6.3.2.2 Working

To evolve the ultimate neural network, we need to first create a group of neural networks initialised with random weights. Each neural network is used to represent an individual snake in the population. The parameter arrays from each network represent the chromosome. Each parameter in the chromosome is a gene.

The working of a genetic algorithm is as follows:

1. Initialisation : Create a set of N Neural Networks initialised with random weight & bias values. They represent the initial population. Since the values are random, the snakes don't perform great.
2. For every generation:
   A. Calculate the fitness values for each individual in the population
   B. Preserve the weights of the performing individual in that generation.
   C. Select 2 of the best performing individuals.

D. Crossover & Create two new children
E. Mutate (1%)
F. Repeat B through D till you have a new set of individuals for next generation.
G. Update the new weight values to all the neural networks.
H. Terminate if converged, else move to next generation

Crossover Function Modified

The crossover function used here is a little different from how normal crossover works. Crossover in this case is applied in parts. The entire chromosome is split into sub chromosomes such that each sub chromosome represents the parameters of a given layer.
For example, When crossover is applied, the weight values of the hidden layer of parent 1 are crossed with the weight values of the hidden layer of parent 2. Similarly, weight values of the final layer are only crossed with the weight values of the final layer. Crossover does not take place across layers. Crossover happens individually for each layer.

6.4 Fuzzy Inference Systems

In this section, we shall discuss the final intelligence model that is applied to solve the problem.

6.4.1 Overview

Fuzzy logic is basically a multi-valued logic that allows intermediate values to be defined between conventional evaluations like yes/no, true/false, black/white, etc. Notions like rather warm or pretty cold can be formulated mathematically and algorithmically processed. In this way an attempt is made to apply a more human-like way of thinking in the programming of computers ("soft" computing).

Fuzzy logic systems address the imprecision of the input and output variables by defining fuzzy numbers and fuzzy sets that can be expressed in linguistic variables (e.g. small, medium and large). Fuzzy rule-based approach to modelling is based on verbally formulated rules overlapped throughout the parameter space. They use numerical interpolation to handle complex non-linear relationships.

There are two types of Fuzzy Inference Systems

1. **Mamdani**
2. **Sugeno**

Though there are two types, in this project, we have only worked with Sugeno Fuzzy inference System to solve the problem. In specific, an artificial neuro-fuzzy inference system (ANFIS) is used to solve the problem.

6.4.2 Background Theory

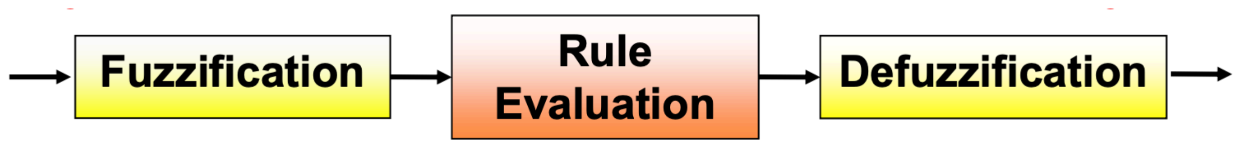


Figure 18: Fuzzy Steps

1. **Fuzzification**: Translate input into truth values

The purpose of fuzzification is to map the inputs from a set of features to values from 0 to 1 using a set of input membership functions.

Based on the value from 0 to 1, you'll be able to calculate the degree.

2. **Rule Evaluation**: Compute output truth values

Inputs are applied to a set of if/then control rules. These rules are to be written manually. The results of various rules are summed together to generate a set of "fuzzy outputs".

3. **Defuzzification**: Transfer truth values into output
Fuzzy outputs are combined into discrete values needed to drive the control mechanism

In summary, these are the steps to compute the FIS output given the inputs

1. determining a set of fuzzy rules
2. fuzzifying the inputs using the input membership functions,
3. combining the fuzzified inputs according to the fuzzy rules to establish a rule strength,
4. finding the consequence of the rule by combining the rule strength and the output membership function (if it's a mamdani FIS),
5. combining the consequences to get an output distribution, and
6. defuzzifying the output distribution (this step applies only if a crisp output (class) is needed).

Mamdani FIS

Mamdani-type inference, expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification.

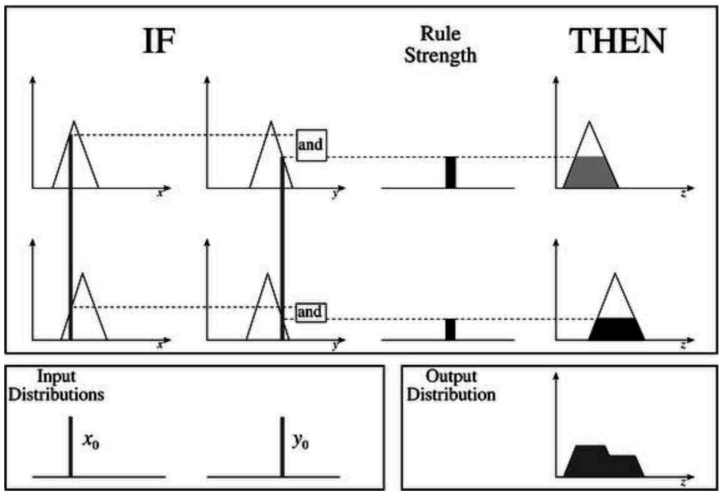


Figure 19: Mamdani FIS

Sugeno FIS

Sugeno FIS is similar to the Mamdani method in many respects. The first two parts of the fuzzy inference process, fuzzifying the inputs and applying the fuzzy operator, are exactly the same. The main difference between Mamdani and Sugeno is that the Sugeno output membership functions are either linear or constant.

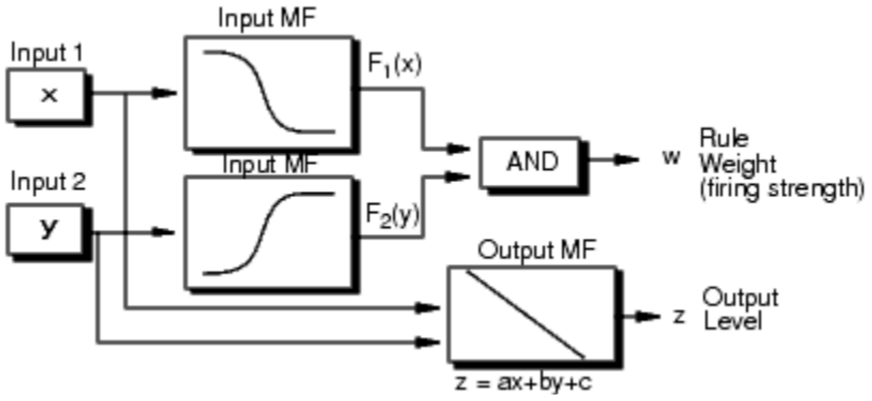


Figure 20: Sugeno FIS

A typical rule in a Sugeno fuzzy model has the form: If Input
1 = x and Input 2 = y, then Output is z = ax + by + c

6.4.3 Artificial Neuro Fuzzy Inference System (ANFIS)

An adaptive neuro-fuzzy inference system or adaptive network-based fuzzy inference system (ANFIS) is a kind of artificial neural network that is based on Takagi–Sugeno fuzzy inference system. Since it integrates both neural networks and fuzzy logic principles, it has potential to capture the benefits of both in a single framework. Its inference system corresponds to a set of fuzzy IF–THEN rules that have learning capability to approximate nonlinear functions.
In this project, using the fuzzy toolbox in MATLAB, the function ANFIS is called and is used to train on the snake data this is collected.

Chapter 7

Experimental Results

7.1 Neural Networks

When training a neural network, the most basic requirement is good quality labelled data. Unlike most problems, there aren't any readily available datasets right off the bat. This meant, we had to generate all the data from scratch. The results depend entirely on the quality of data collected. As mentioned in the previous sections, we decided to choose a set of 4 features that best help describe the game state at a given point in time. With the features in place, we created multiple datasets each unique in its own way.
Most of the experimentation involved trying to find a dataset that works best with the problem.
Here are some of the many datasets that we collected & the model's respective accuracies in test phase. For testing, all the models are tested on a test set with a set of game states with correct labels assigned manually. The performance of the model is measure by finding the average score the snake gets in 50 games.

Dataset	Size (Samples)	Accuracy of Trained Model %	Average Score (50 Games)
Manual Gameplay (Poor)	1200	35%	3
Manual Gameplay (Good)	1200	69%	38.5
Genetic Algorithm Snake Data	3000	73%	44
Heuristic	1500	68%	36

Table 8: How Quality of Data Affects Performance

That quality of data you generate depends on how you collect it & is evident from *Table 8*

In the case of generating data manually by playing, there are two things that can happen

1. You play badly and the data you generate is bad
2. You play really well and generate data that contains mostly the right labels.

As you can see from *Table 8*, the model trained on the poor data performs badly as compared to the model trained on the good data. The accuracy value almost doubles when you have good data.
This brings us to the next observation. Using Genetic Algorithms to generate quality data. As you can see, the model trained on the data collected from a snake evolved using a genetic algorithm seems to perform the best out of the lot. The model trained on GA data seems to pick the right direction 73% of the time.

There wasn't much experimentation done with respect to trying different hyper parameters. The structure used to train was almost same for all the datasets. We could afford to do that as the dataset isn't huge. There are only 4 features. Since the dataset is small, there is no need for complicated network structures to fit the data. The number of samples is pretty low making the training process simple.

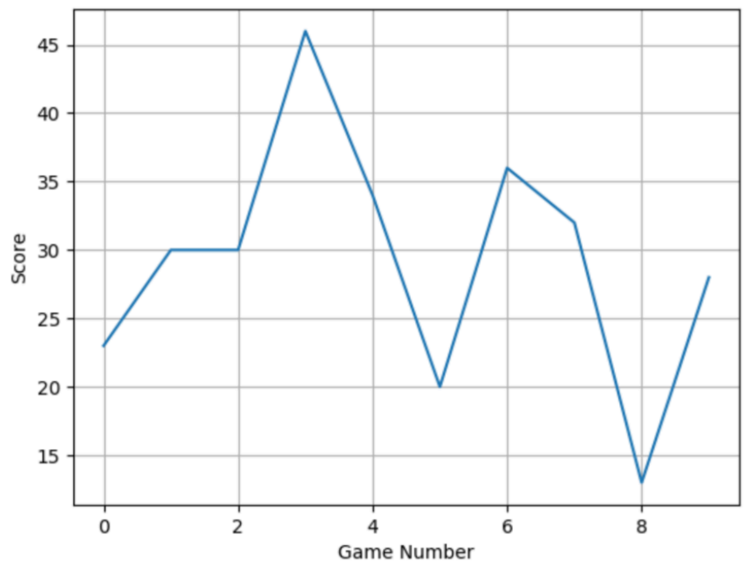


Figure 21: Model trained on Good Manual Data

As you can see in *Figure 21,* this is the performance graph of the Snake trained on Good Manual Data. This is how the Snake has scored while playing 10 games.

In conclusion, the Snake that averaged 38.5 only died in situations where there was no space for it to move. This solution is the most optimal for the snake problem. The Snake is able to survive, avoid the walls, the body and manage to navigate to the candy efficiently almost all the time.The game gets progressively harder as the score increases. The snake is still able to avoid itself and get to the candies. The only time it dies, is if does not have any space to move and is forced to hit a wall or a body.

7.2 Genetic Algorithms

The biggest advantage of Genetic Algorithms is that there is no need for a dataset to train the model. The algorithm tries to explore the entire solution space efficiently and produce a Snake that is able to play the game well.
In our project, we chose two different ways to work with genetic algorithms.
We used the GA to:

1. Learn a set of rules that will help the snake make decisions
2. Train a neural network. Using GA to learn weight values.

Rule Based System

Here are the results:

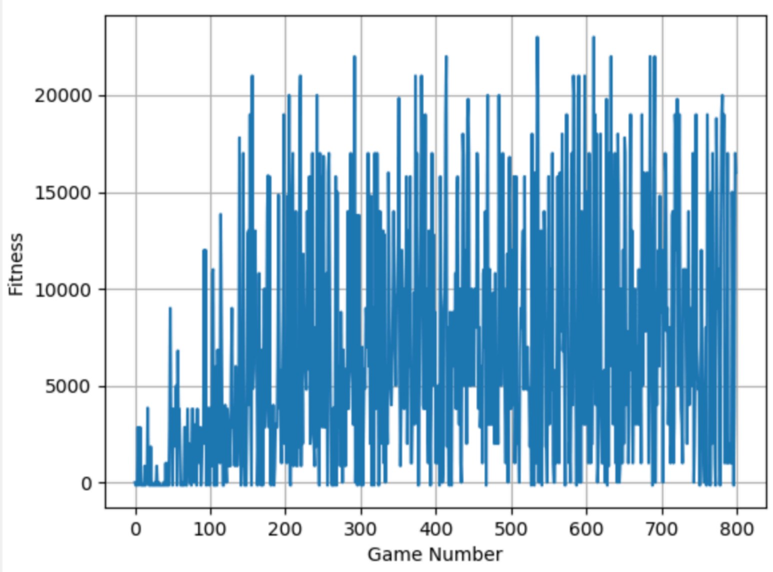


Figure 22: Fitness Values over time

As you can see from *Figure 22*, we created a population of 40 individuals and ran the genetic algorithm for 20 generations. At each generation, the best individuals are saved and they are used as parents to create the next generation by crossover and mutation. As a result, every generation, the snakes keep evolving and keep getting better till they reach a saturation point. The traits of the best performing Snake are passed to the next generation.

As you can see, the fitness values of the snakes are low in the beginning. As the snakes start evolving, you can see a gradual increase in the fitness values.

All the sudden spikes are the snakes that have performed the best. In generation 13, the algorithm produces the best performing Snake. It has the highest fitness out of all the other individuals. It took 13 generations for the genetic algorithm to learn all the rules.

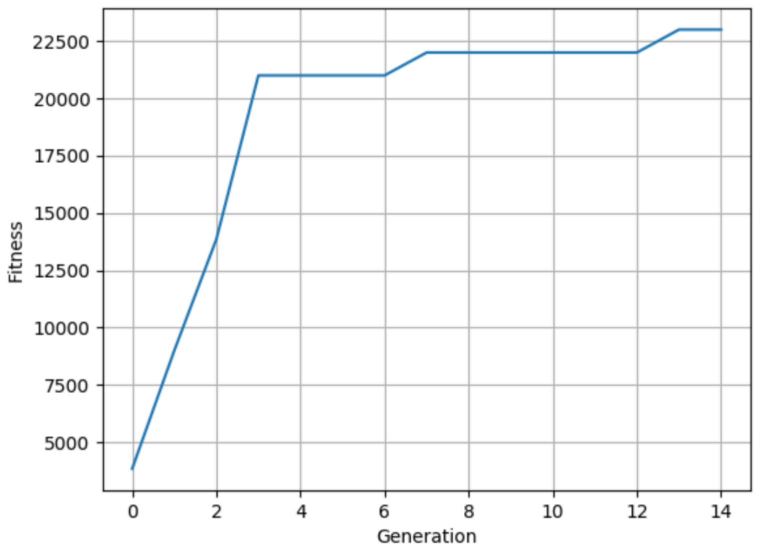


Figure 23: max(Fitness) vs Generation

In *Figure 23*, you can get a better view of the fitness values over time. This graph represents the fitness value of the best performing individual till that generation. As you can see, the fitness values are really low for the first few generations. You can see a steep learning curve during generation 3. After that you can see it rise once more at generation 7. Finally, you can see the Algorithm produce the best performing snake at generation 13. After that, the subsequent generations did not produce a better snake. The algorithm converged at generation 13. The best individual when run for 50 games, averaged a score of **42.**

Training a Neural Network with a Genetic Algorithm

These were the results we observed when we used GA to train the weights of the neural network.

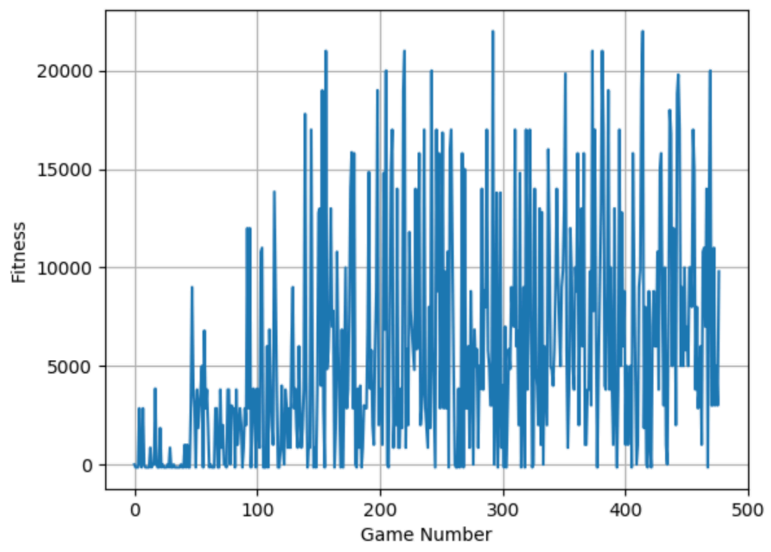


Figure 24: Fitness over time (Neural Network Evolution)

We created a starting population of size 50. We had to create 50 neural networks with random parameter values assigned to them. We let them play and evolved the Neural Networks over time using GA. Like the previous experiment, we decided to run the program for 10 generations. As you can see from *Figure 24*, these are the resulting fitness values of the individuals over time.

Initial generations have low fitness values. The first steep increase is in generation 5. Immediately after that, in generation 6, the Genetic Algorithm peaks. This is the best individual the algorithm has evolved. After that, you can that the values are very similar. There is no increase in fitness after generation 6. So it is safe to assume the algorithm has converged.

The best individual when run for 50 games, averaged a score of **37**.

7.3 Fuzzy Inference Systems

The project didn't require me to design rules or train fuzzy models to play snake. As a part of the project, I have added features in the application that give the student the ability to open MATLAB's fuzzy toolbox from the application. They can then use the toolbox to build both mamdani and sugeno .fis models which they can later open from the application & test on the Snake. The application has support to run MATLAB scripts during runtime within python as well.

On a general note, Fuzzy Inference Systems are used when you experience uncertainty with your input. If your values are not discrete, you need a way to be able to represent intermediate values. If you need to address vagueness in your features, you will find fuzzy useful.

In the case of snake, there is no use for fuzzy inference system for the following reason:
All the features are discrete. There is no vagueness to represent. The spaces are either occupied or free. There are no intermediate value the features take. So writing rules for snake using fuzzy would mean that you write all the possible rules as IF-THEN statements. So this would be same as writing rules for all the possible scenarios by hand. This beats the point of fuzzy.

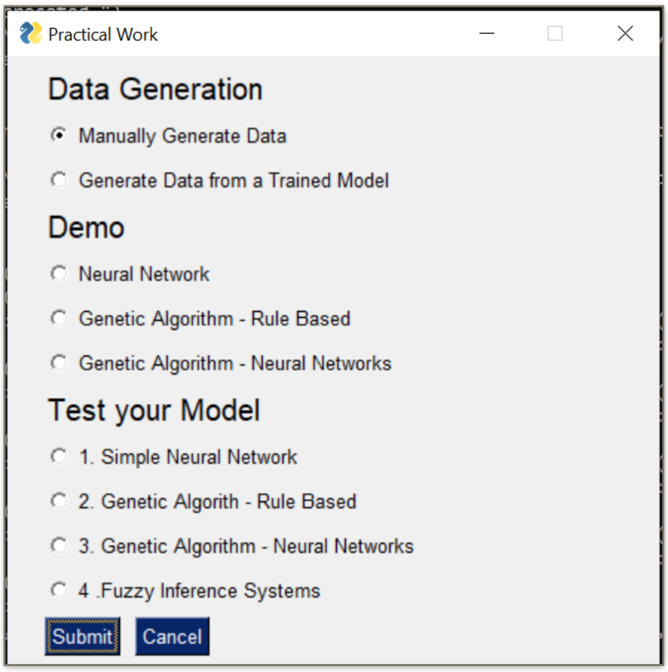
Future Work: There is later going to be support for another game in the application. This game can have features that can make a fuzzy inference system relevant to use. An example for the game could be a car learning to steer, break & accelerate depending on its surroundings. In this example, there is a need to define the amount of break / acceleration /steer that needs to be applied. This is the perfect use of Fuzzy as you can use fuzzy to represent intermediate values.

Chapter 8

Building the Application

8.1 Overview

As mentioned in the previous sections the main aim of this project is not just to use computational intelligence models to solve the problem but also to build a tool that the students can use in order to test their theoretical skills practically on a hands-on environment.
In this section, we aim to discuss and highlight the various features that are added to the application to enable the student to further understand the concepts taught in class.
The entire application is built using Python. The GUI is coded with the help of a package called PySimpleGUI. Though the application is in python, it can run commands and scripts on MATLAB. This is done by calling the MATLAB engine realtime from python with the help of an API. This API establishes a pipeline between Python & MATLAB.


Practical Work

Data Generation

(●) Manually Generate Data

( ) Generate Data from a Trained Model

Demo

( ) Neural Network

( ) Genetic Algorithm - Rule Based

( ) Genetic Algorithm - Neural Networks

Test your Model

( ) 1. Simple Neural Network

( ) 2. Genetic Algorith - Rule Based

( ) 3. Genetic Algorithm - Neural Networks

( ) 4 .Fuzzy Inference Systems

Figure 25: Startup Screen

When the application is launched, this is the splash screen (*Figure 24*) the student is greeted with. The student can choose to do the following things:

1. Generate Data
2. Use the Demo feature to look that benchmark models that was developed and trained during this project.
3. Test the models he has built on the application.

In the next section, the above mentioned features are mentioned in detail.

8.2 Key Features

In this subsection, we will be discussing all the major features that are integrated into the application which can be made use of by the student.
The application's major features can be broken down into three main categories :

1. Demo

Using the Demo Feature the students can run pre-existing trained models just to get an idea of the benchmark that they need to surpass. These are the models that we have trained as a part of the project. These models work really well and is used as benchmark for the students to evaluate their models.

2. Test

Using the Test Feature the student can choose a model that they have trained to see how well their model performs in comparison to the ideal model (which is provided). They can choose which model they want to run with the application. They have the freedom of building a model with Python or MATLAB. Then can also use the model to generate data. The students also get a real time plot of their model's performance. The graph prints real-time as the program runs.

3. Data Generation

This is the most important feature of the application. It enables the student to generate data by one of the many ways mentioned below:

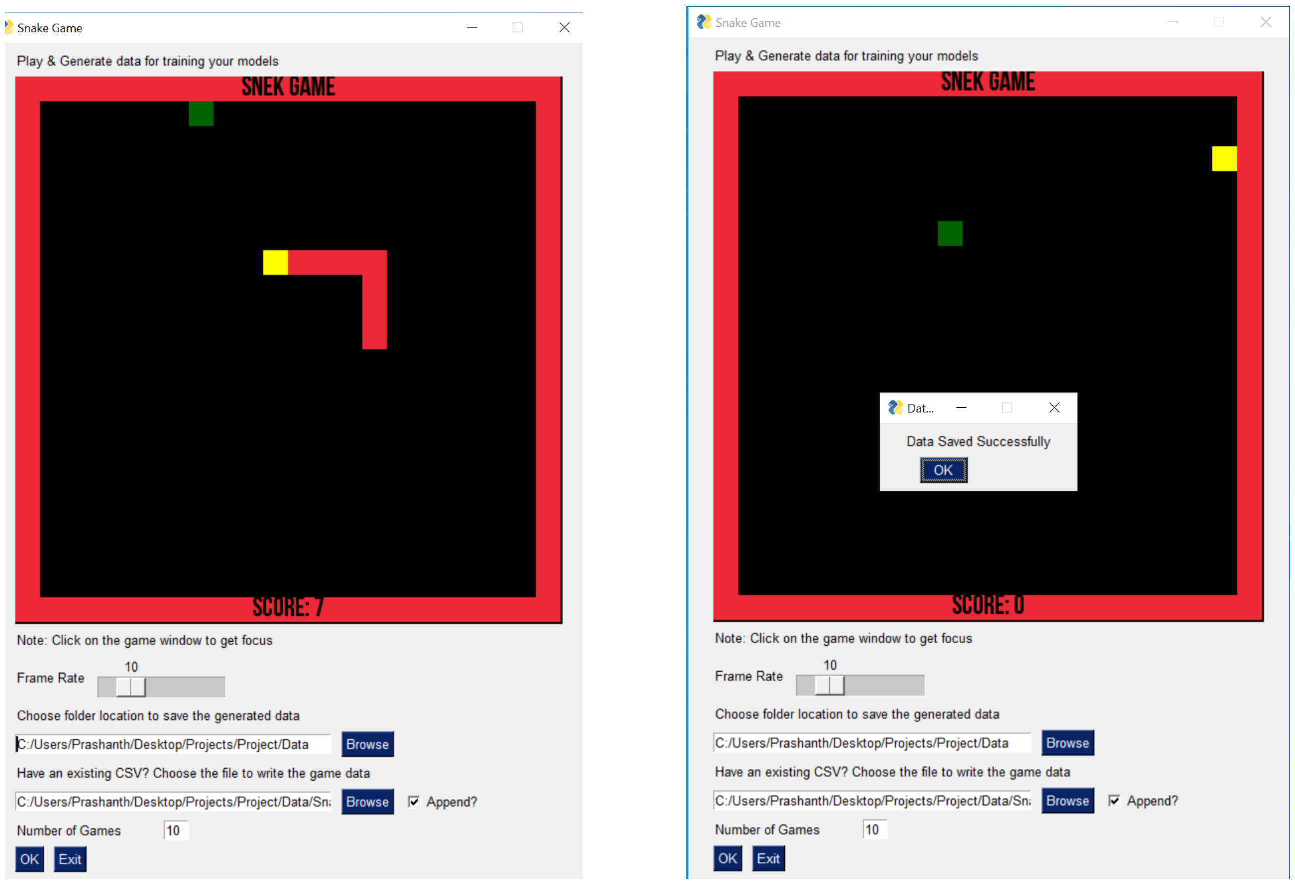
a. Manual
b. From a pre-existing model
c. Using a genetic algorithm to evolve a snake that plays well. This snake is then used to generate quality data

Other Notable Features

1. You can choose the number of games to play/to test/to generate data.
2. You can control the frame rate
3. You can choose which file you want to store data and in which mode (read, append, etc)
4. You can plot your snake's performance in real time
5. This application supports MATLAB as well as python models
6. There is an inbuilt manual that can be accessed at any point to help the student if runs into a problem.
7. Support to open MATLAB from Python. This is used to open the Fuzzy Toolbox if they student wants to build .fis models.
8. There is an inbuilt manual to help resolve issues, if any.

8.3 Snapshots

Here are a few snapshots just to give a glimpse of what this application can offer



Snake Game
Play & Generate data for training your models
SNEK GAME
SCORE: 7
Browse
OK
Exit
Snake Game
Play & Generate data for training your models
SNEK GAME
SCORE: 0
Dat...
OK
Browse
OK
Exit

Note: Click on the game window to get focus

Frame Rate 10

Choose folder location to save the generated data

C:/Users/Prashanth/Desktop/Projects/Project/Data

Have an existing CSV? Choose the file to write the game data

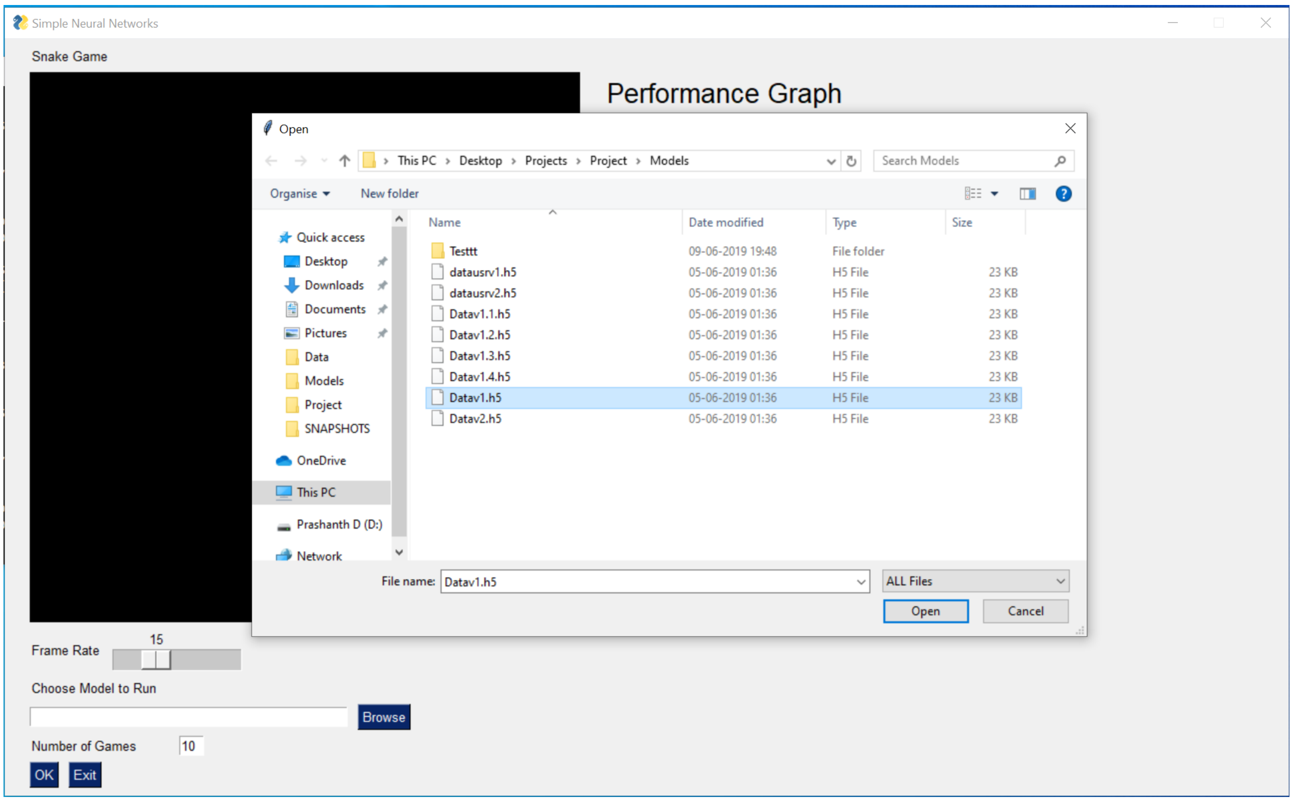
C:/Users/Prashanth/Desktop/Projects/Project/Data/Sn: ☑ Append?

Number of Games 10

Data Saved Successfully

Figure 26: Data Generation Screen

Figure 27: Running & Testing your own models



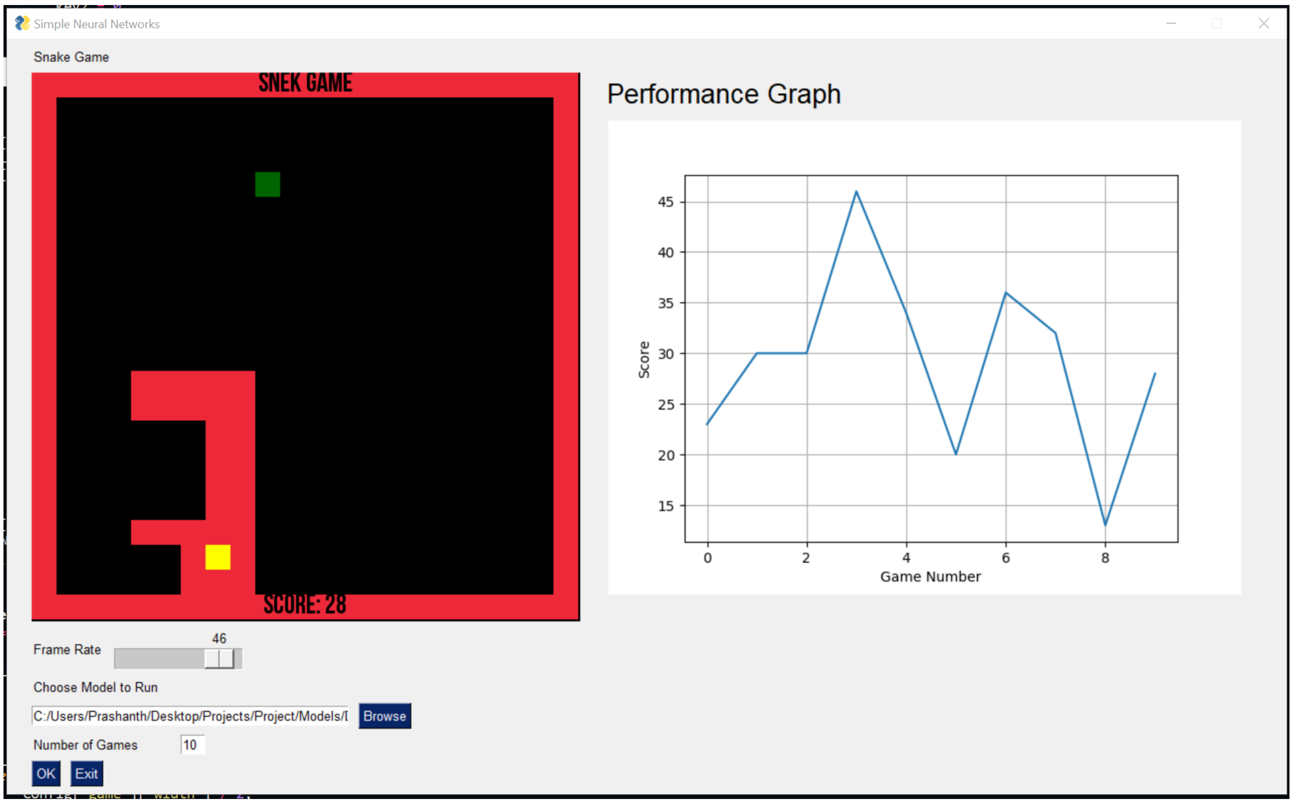
Name	Date modified	Type	Size
Testtt	09-06-2019 19:48	File folder	
datausrv1.h5	05-06-2019 01:36	H5 File	23 KB
datausrv2.h5	05-06-2019 01:36	H5 File	23 KB
Datav1.1.h5	05-06-2019 01:36	H5 File	23 KB
Datav1.2.h5	05-06-2019 01:36	H5 File	23 KB
Datav1.3.h5	05-06-2019 01:36	H5 File	23 KB
Datav1.4.h5	05-06-2019 01:36	H5 File	23 KB
Datav1.h5	05-06-2019 01:36	H5 File	23 KB
Datav2.h5	05-06-2019 01:36	H5 File	23 KB

File name: Datav1.h5

Frame Rate 15

Choose Model to Run

Number of Games 10



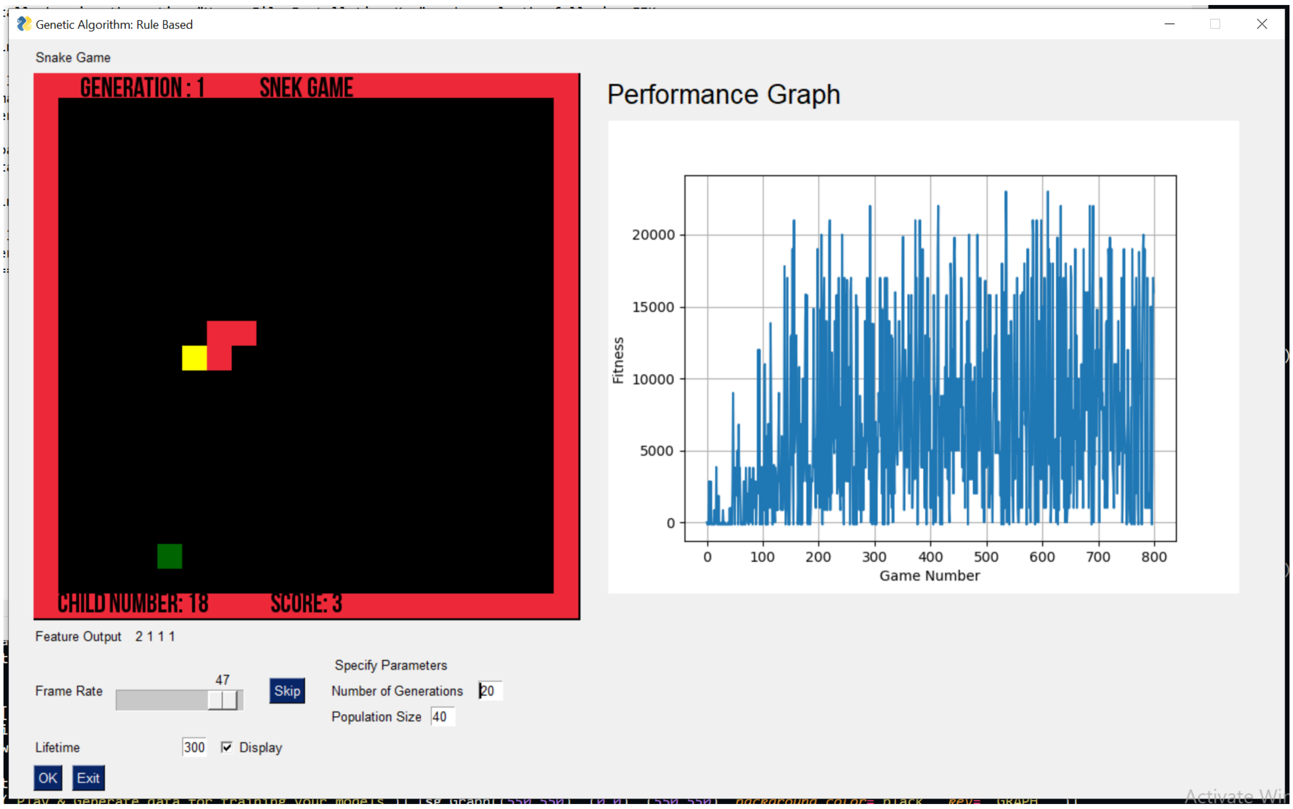
Frame Rate 46

Choose Model to Run

C:/Users/Prashanth/Desktop/Projects/Project/Models/I

Number of Games 10

Figure 28: Performance Plots in Real-time



Feature Output 2 1 1 1

Frame Rate 47

Specify Parameters

Number of Generations 20

Population Size 40

Lifetime 300

☑ Display

Figure 29: Genetic Algorithm Splash Screen

8.4 Limitations

Since the students do not have access to the source code, they do not have the freedom to change the number of features.This means, they will have to work the 4 default features for the time being. Future support will include support for adding and remove features at will.

Also, they do not have the freedom to fetch any value they want from the game environment. They are forced to work with the set of features that are prescribed. The students don't have access to the API can be used to fetch game environment variables. The application works only on windows and linux. There is no support for macOS as macOS has stopped support for SDL1.

Chapter 9

Deployment

9.1 Overview

The project is to be deployed in the computer lab systems at FIB, UPC. The aim of the application is to serve as a testing tool for the students. As and when a theoretical concept is taught, the course requires them to apply the knowledge and solve the problem of trying to design an AI that can play the game well. They need to use the application deployed in the lab systems to test the models that they've trained. All the other notable features have been mentioned in the previous section.
The teachers of the course will be using the application as a tool to test the students' knowledge of the concepts that they have taught by making them train their models and test them via the application.

9.2 Software & Package Requirements

- Python 3.7
- Windows (7 or later) or Linux
- SDL2 driver required (Simple Direct Media Layer)
- MATLAB 2018 (or above) with API engine to communicate with a python script.
- PyGame
- PySimpleGUI
- Tensorflow
- Keras

9.3 Supported Model Formats

- Tensorflow .h5 models (If python is the programming language of choice)
- Keras .h5 models (if Python is the programming language of choice)
- MATLAB .fis models from the fuzzy toolkit
- MATLAB NN models from the Neural Networks Toolkit

9.4 Changes

During the realisation of the project, there were a certain number of changes that were made to the original plan.

i) Initially, there was a plan to implement multiple games into the practical environment, but as of now only Snake Game has been implemented into the practical environment.

ii) At the start, it was decided that the language support for coding the CI models in the application would be in Python. This was later changed as the students are more comfortable using matlab to write code and train their models. Hence, the application now has support for both Matlab and Python. Adding support for Matlab was a problem initially. This was solved by establishing a link between python and the matlab engine using a matlab API. This enables python to call the matlab engine and execute matlab scripts directly from python.

iii) There was no plan to implement Genetic Algorithm for training the weights of a Neural Network. The original plan was to use Genetic Algorithms to train a set of rules which would power the game agent. Since there was more time, Genetlic Algorithm was also used to train Neural Networks to power the Agent's decision making.

iv) Lack of Simple DirectMedia Layer support for MacOS. The entire game interface was built ground up using ‘PyGame’, a graphics library for python. Pygame was built based on SDL1.The latest version of MacOS supports SDL2, <u>not</u> SDL1.This served as a obstacle for development as majority of the development was done on a MacOS system. This works perfectly on windows and linux systems. For the sake of development, the game has to be coded again adding SDL2 support/ or the system for development is to be changed (A windows system instead of a Mac).

Chapter 10

Conclusion and Future Work

We can all agree that Artificial Intelligence has changed our lives for the better. It has completely revolutionised the technology landscape that millions of people reap benefit from. That said, there are so many problems that exist in this world that still need to be solved. It is upto the current & future generation of engineers to use their skills and make a difference. For us to make the difference, we need all the education we can get from the universities we study in. Most colleges teach the theory well, but fail to focus on the practical aspects. In reality, you can't learn with just theory. You need to implement what you learn to truly understand the concepts.

That's why a tool like this would be the perfect learning tool for students learning Machine Learning or Artificial Intelligence for the first time. This tool with its intuitive features will enable the student to take interest in the subject and solve the problem. Video games have been used as a learning tool for a long time now [6]. It is proven that, video games help increase interest in the learning process. After all, learning isn't learning if it is not enjoyed. Hopefully, this tool can make a difference in FIB,UPC by helping the students understand the workings of various Intelligence models really well and enable them to make a huge impact in the industry.

That said, there are still many limitations in the current iteration of the project. In the subsequent versions, these are some of the changes that will be introduced

1. Feature Flexibility
2. Support for macOS systems
3. Support for more programming languages and model formats
4. More features to enhance functionality
5. More intelligence models are to be integrated into the application

References

[1] https://www.weforum.org/agenda/2017/12/charts-artificial-intelligence-ai-index/

[2] http://cdn.aiindex.org/2018/AI%20Index%202018%20Annual%20Report.pdf

[3] https://gym.openai.com/

[4] https://www.fib.upc.edu/en/studies/masters/master-artificial-intelligence/curriculum/syllabus/CI-MAI

[5] Russell, Stuart and Peter Norvig. Artificial Intelligence: A Modern Approach. New Jersey: Pearson Education, 2010.

[6] https://www.forbes.com/sites/forbestechcouncil/2018/10/09/how-video-games-help-students-level-up-stem-learning/#786619de1a78

[7] Galway, Leo & Charles, Darryl & Black, Michaela. (2008). Machine learning in digital games: A survey. Artif. Intell. Rev.. 29. 123-161. 10.1007/s10462-009-9112-y.

[8] Baba, Norio & Kita, Tomio & Oda, Kazuhiro. (1995). Application of artificial neural networks to gaming. Proceedings of SPIE - The International Society for Optical Engineering. 465-476. 10.1117/12.205151.

[9] Thengade, Anita & Dondal, Rucha. (2012). Genetic Algorithm – Survey Paper. IJCA Proc National Conference on Recent Trends in Computing, NCRTC. 5.

[10] Lingaraj, Haldurai. (2016). A Study on Genetic Algorithm and its Applications. International Journal of Computer Sciences and Engineering. 4. 139-143.

[11] Kearney, William T., "Using Genetic Algorithms to Evolve Artificial Neural Networks" (2016). Honors Theses. Paper 818. http://digitalcommons.colby.edu/honorstheses/818

[12] Navneet Walia, Harsukhpreet Singh & Anurag Sharma. ANFIS: Adaptive Neuro-Fuzzy Inference System- A Survey.
International Journal of Computer Applications (0975 – 8887) Volume 123 – No.13, August 2015

[13] Swati R. Chaudhari and Manoj E. Patil. Comparative Analysis of Fuzzy Inference Systems for Air Conditioner.
International Journal of Advanced Computer Research (ISSN (Print): 2249-7277 ISSN (Online): 2277-7970) Volume-4 Number-4 Issue-17 December-2014

Appendix A: Requirements

The following tools are required for the project.

Software

- **Python -** This was chosen as the language of implementation as it is by far the best programming language for Machine Learning project. It being open sourced is also an added benefit. It also comes with a huge library of packages which can be used for completing various tasks. It also has the ability to run MATLAB scripts. So naturally, python was the language of choice.

  Python Packages

  - **Pandas**, **Seaborn**, **Matplotlib** - For all data manipulation & visualisation purposes
  - **h5py** for storing processed data. Models etc.
  - **Keras**. We use Keras for implementing all the neural networks used in the project. It is a high-level neural networks library, capable of fast experimentation
  - **PyGame**. It is an open-source module for the Python programming language specifically intended to help make games and other multimedia applications. Built on top of the highly portable SDL (Simple DirectMedia Layer) development library
  - **PySimpleGUI**. It is an easy to use python GUI library based on tkinter. This is one of the very few GUIs that support PyGame.
  - **Matlab Engine** - A package that helps add MATLAB support for python. It helps establish a connection between python & matlab allowing the user to run MATLAB scripts & commands from python during runtime.
- **Git** - This tool is used for project management & version control. Helps you commit and push changes to your repository where all the project files are stored
- **Sublime Text 3** - A beautiful code editor. Has features that let you use **Git** with the editor. You can handle your branches, make changes etc all from the code editor

- **MATLAB -** MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python**. MATLAB** contains the fuzzy toolbox which is a requirement for the project.