A Major Project Final Report on

**ENNAT**

Submitted in Partial Fulfillment of the Requirements for the **Bachelor of Engineering** in **Software Engineering** under **Pokhara University**

Submitted by:

**Miroz Devkota, 14715**

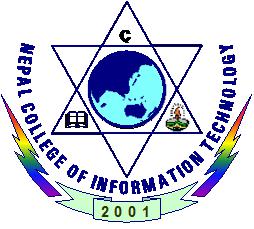
**Ramit Bhari, 14726**

**Under Supervision of**

**Er. Saroj Bista**

Date: December 6, 2018.

**Department of Software Engineering**

****

**NEPAL COLLEGE OF**

**INFORMATION TECHNOLOGY**

**Balkumari, Lalitpur, Nepal**

ACKNOWLEDGEMENT

First and foremost, we would like to thank our supervisor Mr. Saroj Bista for his guidance throughout this project. Without his invaluable advice and assistance, it would have not been possible for us to complete this part of our project on time.

We would like to provide our special thanks to Dr. Roshan Chitrakar, for providing technical support with enormous suggestions which helped us to coordinate our project. We would also like to thank Er. Birendra Bista (Head of Department of Software Engineering), for their valuable guidance throughout the project development period.

Moreover, we would like to owe our special thanks to all subject teachers who provided us with necessary information during the course of studies, which somehow helped us in various ways for the development of project. Similarly, we truly admire the effort of all persons who knowingly or unknowingly provided us with the concept about the project that assisted us for the development of the system.

**ABSTRACT**

ENNAT consists of two phases: Neural Network Design Phase and Neural Network Training Phase. In Design Phase, ENNAT follows Feed-forward Neural Network. A **Feed-forward neural network** is an [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) wherein connections between the nodes do *not* form a cycle. As such, it is different from [recurrent neural networks](https://en.wikipedia.org/wiki/Recurrent_neural_networks). In Training Phase, we use genetic algorithm which is the sub-division of reinforcement learning, based on reward and punishment shown by the fitness score.

The Feed-forward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network, which outperforms a general path following algorithm on a challenging learning task. This process of continual iteration allows finding highly sophisticated and complex neural networks. We test this claim through a series of studies that demonstrate that each generation is necessary to the system as a whole and to each other which results is significantly faster learning. Feed-forward NN is also an important contribution to Genetic Algorithm because it shows how it is possible for evolution to both optimize and make the generation more complex simultaneously.

**Keywords:** Genetic algorithms, Neural Networks, Neuro-Evolution, Feed-forward, Reinforcement learning

**Table of Contents**

**1 INTRODUCTION1**

1.1 Problem Statement2

1.2 Project Objectives2

1.3 Significance of Study3

1.3 Scope and Limitations3

**2 METHODOLOGY6**

**2.1** **SOFWTARE DEVELOPMENT LIFECYCLE**7

2.1.1 Extreme Programming7

**2.2 TECHNICAL DESCRIPTION**7

2.2.1 Tools and Technology Used7

**3 LITERATURE REVIEW5**

**4 USE CASE DIAGRAM8**

**5 WORK DETAILS9**

**6 TESTING11**

**7 PROJECT TASK AND TIME SCHEDULE13**

**8 RESULTS AND CONCLUSION15**

**9 LIMITATION AND FUTURE ENHANCEMENT17**

**BIBLIOGRAPHY**

**List of Figures**

* **Figure 3.1** Agile Software Development**6**
* **Figure 3.2.1** Feed-forward Neural Network**6**
* **Figure 3.2.2** Random Crossover**6**
* **Figure 4.1** A Single Layered Feed-forward Neural Network**6**
* **Figure 4.1** Use Case Diagram of Genetic Algorithm**10**
* **Figure 9.2** Gantt Chart for Overall Project Timeline**10**

**List of Tables**

**Table no. 10.1** Time Scheduling**12**

1. **INTRODUCTION**

Artificial neural networks, or shortly neural networks, have been successfully applied to many diverse fields. Pattern classification/recognition; system modeling and identification; signal processing; image processing; control systems and stock market predictions are some of those main fields of engineering and science. Human are prone to making mistakes due to reasons such as tiredness, vision impairment, sudden change in environment etc. This AI tries to reduce such human errors by constantly evolving.

**1.1 PROBLEM STATEMENT**

In today’s world, humans are being replaced by machines due to its ability to do certain task consistently because the more human beings work, the more mistakes they are likely to make. Such cases can be seen in our country like as we are all familiar with hundreds of road accidents (For example: Truck accidents) that occurs and the cause being mistakes made by humans due to various reasons like fatigue, poor eyesight, distraction etc.

Human beings get tired and are not very motivated to keep on doing tedious work after a certain number of tries. Our simulation will try to show how the AI keeps on trying until it succeeds on reaching its goal by constantly learning and evolving throughout its lifecycle by minimizing any faults as much as possible. Once the AI has perfected itself in a certain environment, it will try to implement that learning to a different environment it has never seen before and will try to improve on it with the knowledge it has gained in the previous environment.

**1.2 PROJECT OBJECTIVES**

Our objective on the surface is to create a computer program that is a mere infant at the early stage of certain tasks assigned to it but will learn how to do the assigned task. Our objective is to create a self-learning AI by emulating evolution to get through certain task. ENNAT follows Feed-forward neural network.

Furthermore, we are creating a simulation, where the AI which is at its infancy in the beginning can learn and constantly evolve in a certain environment, and implement that learning to a different more complex environment it has never seen before and try to improve on it with the knowledge it has gained in the previous environment. The Fitness data that is being obtained from each generation is being used by the AI to move forward to do more complex task.

**1.3 SIGNIFICANCE OF STUDY**

The findings of this study will be redounded to the benefit of society considering that it plays an important role in complex reinforcement learning tasks. This program can contribute various things such as betterment of society, our program has the potential to learn and evolve continuously. This approach to solving complex problems represents an alternative to statistical techniques that attempt to estimate the utility in particular states of the world. Feed-forward neural networks are the most popular and most widely used models in many practical applications. They are known by many different names, such as ‘multilayer perceptrons’ (MLP). It consists of a number of simple neuron-like processing units, organized in layers and every unit in a layer is connected with all the units in the previous layer. These connections are not all equal, as each connection may have a different strength or weight. The weights on these connections encode the knowledge of a network. Often the units in a neural network are called nodes. Data enters at the input and passes through the network, layer by layer, until it arrives at the output.

**1.4 SCOPE AND LIMITATIONS**

The scope of this project is to develop a simulation where the AI will run through a series of self-made courses, constantly learning, making mistakes and evolving progressively through each generation. As the AI learns through different courses, the data from the past courses can be used and applied to different more complex environment with more variables added to it and observe how much further it can get. We can also study how the AI evolves from a simpleton to a more complex structure as it learns through the previous data.

Although, it is a very interesting and exciting project. There are certain limitations of doing it. For instance, it will require a lot of resource, research and study to make this as close to perfect or anywhere around it as possible. It will also take a lot of time for both the maker as well as the program itself to keep on learning, adapting and evolving. Therefore, patience will also play a vital role in order to carry this further in the future.

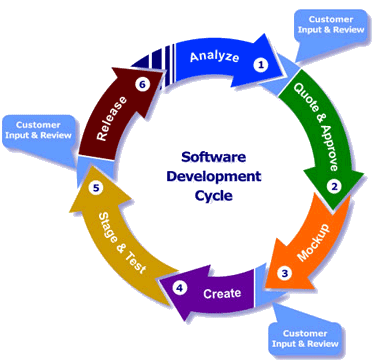
The quality of the AI depends upon its fitness, so the higher the fitness data the more knowledgeable and experienced AI we get. So, in order to get the desired result more time must be spent in training the AI.

2. METHODOLOGY

We have planned to work following these methodologies for the application of knowledge, skills, tools and techniques to a broad range of activities in order to meet the requirements of our project, “ENNAT”.

**2.1 SOFWTARE DEVELOPMENT LIFE CYCLE**

For this project, we have used the Agile Development model. The Agile Development model is an umbrella term for several iterative and incremental software development methodologies used in software development process. While each of the agile methodologies is unique in its specific approach, they all share a common vision and core values. They all fundamentally incorporate iteration and the continuous feedback that it provides to successively refine and deliver a software system. They all involve continuous planning, continuous testing, continuous integration, and other forms of continuous evolution of both the project and the software. They are all lightweight, especially compared to traditional waterfall-style processes, and inherently adaptable. What is more important about agile methods is that they all focus on empowering people to collaborate and make decisions together quickly and effectively.



**Fig 3.1**: Agile Software Development

The Manifesto for Agile Software Development is based on twelve principles:

1. Customer satisfaction by early and continuous delivery of valuable software
2. Welcome changing requirements, even in late development
3. Working software is delivered frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the primary measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from self-organizing teams
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly

Popular agile software development frameworks include:

* [Adaptive software development](https://en.wikipedia.org/wiki/Adaptive_software_development) (ASD)
* Agile modeling
* Agile unified process (AUP)
* Disciplined agile delivery
* [Dynamic systems development method](https://en.wikipedia.org/wiki/Dynamic_systems_development_method) (DSDM)
* [Extreme programming](https://en.wikipedia.org/wiki/Extreme_programming) (XP)
* [Feature-driven development](https://en.wikipedia.org/wiki/Feature-driven_development) (FDD)
* Lean software development
* [Rapid application development](https://en.wikipedia.org/wiki/Rapid_application_development) (RAD)
* Scrum

**2.1.1 Extreme Programming**

Since 70’s widely used approach had linear structure characteristic known as “Waterfall Model”, project scope was planned out at the beginning of the project life-cycle, in which changing requirements in the middle of the development, made it very costly. Frequently changing requirements by customers required for more agile approach.

XP is known as a methodology of agile development. This is used in small, collocated project teams. Development is flexible and lightweight. XP is based on twelve practices and four groups - collaboration, feedback, revision and respect.

* **Collaboration** – active participation of all stakeholders of the project in which customer is a part of the development team. Project starts with creating simple design and ‘hands on’ coding. Test Driven Development approach is applied by adding new functionality to the code. Focus is on today tasks and requirements rather than looking at long-term. Regular meetings are held and frequent verbal communication between programmer team is practiced. Feedback acquired immediately through iterative structure of the development.
* **Feedback** – feedback is effective if it is instantaneous. System feedback is achieved through Unit Testing. Unit Tests are created for the software throughout development phase on which complete software is built. Any changes to the requirements is done through collaboration process with customers who give the feedback to the existing model, changes are taken into account and presented as new requirements.
* **Revision** – XP development focuses on current tasks and problems. This makes written code often unstructured as it should be. Since the structure of the code is not planned out at the beginning of the project, regular refactoring is recommended and even compulsory for further development and maintenance.
* **Respect** – XP practices programming in pairs. Programmers split tasks, one is typing code (person is known as a ‘driver’), while the other programmer is following the code and working on an algorithm to solve the problem. Roles change in time. This is a good practice because every decision of the team matters which leads to the team loyalty.

Code is the most important part of XP; complete documentation is considered as a waste of assets. Simple code and design form the basis; this is understandable to programmers who join the team at the later stages.

There are twelve core practices in XP stated as a standard:

1. **Coding Standards** – mutually agreed standard by the team throughout the development on language used, vendors, rules and format.
2. **Collective Code Ownership** – every programmer in the team has a right on the code. Suggested to use repositories to store the code, every programmer has access rights.
3. **Simple Design** – problem should be tackled and the simplest outcome should be applied.
4. **System Metaphor** – classes and functions should do what they say on the ‘label’, naming should be given in mutual understanding for the functionality.
5. **Pair Programming** - two programmers work on one task, one is typing and following the lead of the other who focuses on the functionality and algorithms to solve the problems, also reviewing written code.
6. **Planning Game** – regular project meetings are held and are divided into release planning in which customers request necessary changes and new functionality, and iteration planning which is done after completing the iteration by presenting functional software.
7. **Test Driven Development** – creation of a test case following by writing a code that satisfies the desired test result.
8. **Whole Team** – all project stakeholders are actively involved and work together.
9. **Continuous Integration** - team members should work on the latest release of the software, some team members who work on the delayed parts should integrate the code into the main release repositories as soon as possible.
10. **Refactoring** – refactoring should be performed on a regular basis.
11. **Small Releases** – functional releases should be built iteratively that form the project. These visible modules encourage customers, also used for evaluation purposes.
12. **Sustainable Pace** – also known as a 40-hour week. Well-being of the team is important, burnt out programmers do not perform at their best. Productivity is achieved by these means.

**2.2 TECHNICAL DESCRIPTION**

Technically to improvise this application practically we have chosen Unity 2017.3.1f1 (64-bit) and reference for neural network is taken from lua script. Using unity, we were able to create a basic front end canvas. The character model, the course builder, the environment etc. were built with the help of unity. The neural network and things related to it were referenced from lua script. We have altogether around nine scripts from main menu to neural network. We have create a basic interface that can be accessed by the user to train and inspect how the AI is progressing. For developing the AI efficiently, prototype approach and unit testing is given the emphasis. Feasibility study, operational feasibility is studied and can be operated in Windows.

**2.2.1** **TOOLS AND TECHNOLOGY USED**

Although various reference was taken.

* Basic tools and technology that were used are unity3d.
* Reference for neural network was taken from lua.
* Minimal amount of Photoshop was used.
* Asset store was used to import character model, hex block.

**Front End** – Unity3d, Photoshop, Asset Store, snipping tool

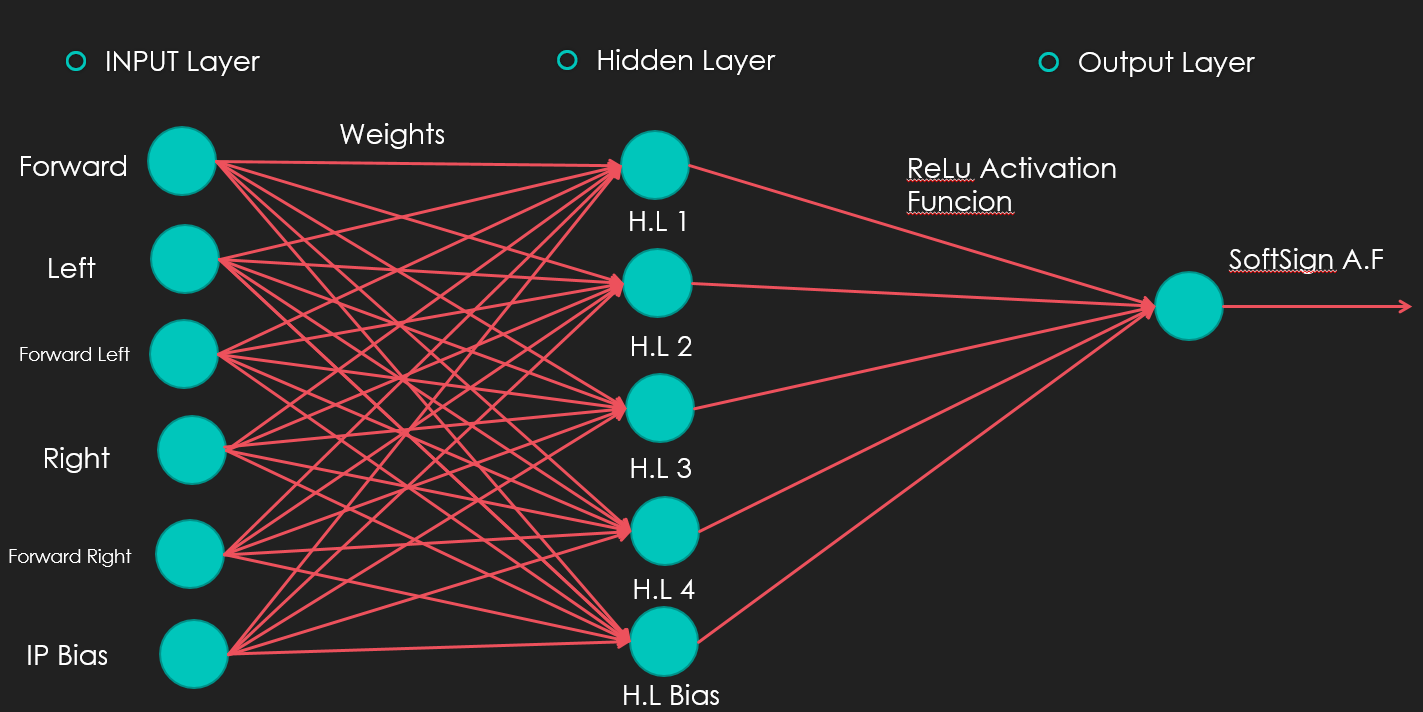
**Unity Features:**

* **Graphics:** Unity has a lot of features for graphics. For lighting, there is now real-time global illumination - that is mobile, desktop & console ready. New improvements for shading include reflection probes and physically based shading. There are also updates to the user interface (UI) tools that create an intuitive design system for 2D and 3D.
* **Audio:** Improvements for audio include real-time mixing and mastering that now allows you to create detailed mixing graphs, tweak your mix in play mode and create snapshots & blend between them. Now you can insert effects anywhere in the mix, create sends + returns between any group and implement ducking quickly & easily.
* **Animation:** Direct blend trees allow you to fully control all animation weights at runtime. The state machine lets you transition between state machines with entry and exit nodes and add behaviors directly to states. The root motion authoring allows you to drive any object via delta animation.
* **All-in-one editor:** It includes a range of [artist-friendly tools](https://unity3d.com/unity/features/editor/art-and-design) for designing immersive experiences and game worlds, as well as a strong suite of developer tools for implementing game logic and high-performance gameplay.
* **2D & 3D:** Unity supports both [2D](https://unity3d.com/solutions/2d) and 3D development with features and functionality for your specific needs across genres.
* **User interfaces:** Built-in UI system allows you to create user interfaces fast and intuitively.
* **Physics engines:** Take advantage of Box2D and NVIDIA PhysX support for highly realistic and high-performance gameplay.
* **Custom tools:** extend the Editor with whatever tools you need to match your team’s workflow. Create and add customized extensions or find what you need on our Asset Store, which features thousands of resources, tools and extensions to speed up your projects.

**Neural Network:**

Consists of two phases:

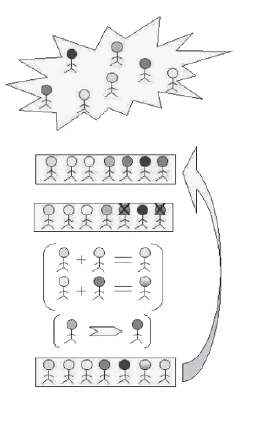
* **Neural Network Design Phase:** In Design Phase, we used Feed-forward Neural Network. A Feed-forward neural network is an [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) wherein connections between the nodes do *not* form a cycle.



**Fig 2.2.1:** Feed-forward Neural Network

Our Feed-forward neural network consists of three layers:

* + 1. **Input Layer:** consists of five input nodes and one input bias. The five input nodes are forward, left, forward left, right, forward right directions. Random values are added to each input nodes which are the inputs for the hidden layer nodes. Input bias is used to add randomness to the pool.
    2. **Hidden Layer:** consists of four hidden layer nodes and one hidden layer bias. Each hidden layer takes data from the input layer and calculation is done using activation functions specifically ReLU (Rectified Linear Unit) which gives the max value between zero and input. This provides input to the output layer. Hidden Bias is used to a add randomness to the pool.
    3. **Output layer:** it takes input from the hidden layer nodes. Activation function called softsign is used to give a value between -1 and 1. This acts as a direction defining value for the AI.
* **Neural Network Training Phase:** In this phase, we use reinforcement learning which is based on the concept of rewards and punishment. Rewards and punishment are given in the form of fitness score. Those thought process which does not perform well are punished via a low fitness score and rewards via high fitness score are given to those which are performing comparatively well. Those with higher fitness score are breed using random crossover for next generation. This process can be continued until one is satisfied.

****

**Fig 2.2.2** Random Crossover

3. LITERATURE REVIEW

Artificial Intelligence is the general category that encapsulates or consists of Machine and Deep Learning. [8]AI is basically any intelligence demonstrated by a machine that leads it to an optimal or suboptimal solution given a problem. **Artificial intelligence (AI)** is the ability of a digital [computer](https://www.britannica.com/technology/computer) to perform tasks commonly associated with intelligent beings. The term is frequently applied to the project of developing systems given with the [intellectual](https://www.merriam-webster.com/dictionary/intellectual) processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience. Since the development of the [digital computer](https://www.britannica.com/technology/digital-computer) in the 1940s, it has been demonstrated that computers can be programmed to carry out very complex tasks—as, for example, discovering proofs for mathematical theorems or playing [chess](https://www.britannica.com/topic/chess)—with great proficiency. Still, despite continuing advances in computer processing speed and memory capacity, there are as yet no programs that can match human flexibility over wider domains or in tasks requiring much everyday knowledge.

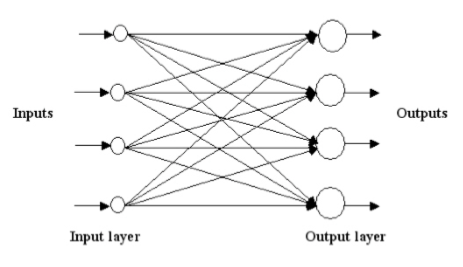
The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. You can apply the genetic algorithm to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly nonlinear. The genetic algorithm can address problems of mixed integer programming, where some components are restricted to be integer-valued.

The genetic algorithm uses three main types of rules at each step to create the next generation from the current population:

* Selection rules select the individuals, called parents, which contribute to the population at the next generation.
* Crossover rules combine two parents to form children for the next generation.
* Mutation rules apply random changes to individual parents to form children.

ENNAT follows the concept of Feed-forward Neural Network.

[9] In Feed-forward Neural Network, Data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called *Feed-forward* neural networks. There are two main categories of network architectures depending on the type of the connections between the neurons, “feed-forward neural networks” and “recurrent neural networks”. If there is no “feedback” from the outputs of the neurons towards the inputs throughout the network, then the network is referred as a “feed-forward neural network”. Otherwise, if there exists such a feedback, i.e. a synaptic connection from the outputs towards the inputs (either their own inputs or the inputs of other neurons), then the network is called a “recurrent neural network”.



**Fig 3.1** A single layer feed-forward neural network

Usually, neural networks are arranged in the form of “layer”s. Feed-forward neural networks fall into two categories depending on the number of the layers, either “single layer” or “multi-layer”.

[11] *Activation Function* is a thing (**node**) that you add to the output end of any neural network. It is also known as **Transfer Function**. It can also be attached in between two Neural Networks. It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function). Activation functions are an extremely important feature of the artificial neural networks. They basically decide whether a neuron should be activated or not.

Whether the information that the neuron is receiving is relevant for the given information or should it be ignored.

The activation function is the nonlinear transformation that we do over the input signal. This transformed output is then sent to the next layer of neurons as input.

Activation function is essential if you are doing or want to progress in something complex. When we do not have the activation function the weights and bias would simply do a linear transformation. A linear equation is simple to solve but is limited in its capacity to solve complex problems. A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks. We would want our neural networks to work on complicated tasks like language translations and image classifications. Linear transformations would never be able to perform such tasks.

Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases. Without the differentiable nonlinear function, this would not be possible.

### [12] Some of the types of Activation functions -

1. Tanh — Hyperbolic tangent
2. ReLu -Rectified linear units
3. Softsign

**ReLu- Rectified Linear units:** It has become very popular in the past couple of years. It was recently proved that it had 6 times improvement in convergence from Tanh function. It’s just R(x) = max (0, x) i.e if x < 0, R(x) = 0 and if x >= 0, R(x) = x. Hence as seeing the mathamatical form of this function we can see that it is very simple and efficinent. A lot of times in Machine learning and computer science we notice that most simple and consistent techniques and methods are only preferred and are best. Almost all deep learning Models use **ReLu** nowadays.

**But its limitation is that it should only be used within Hidden layers of a Neural Network Model**.

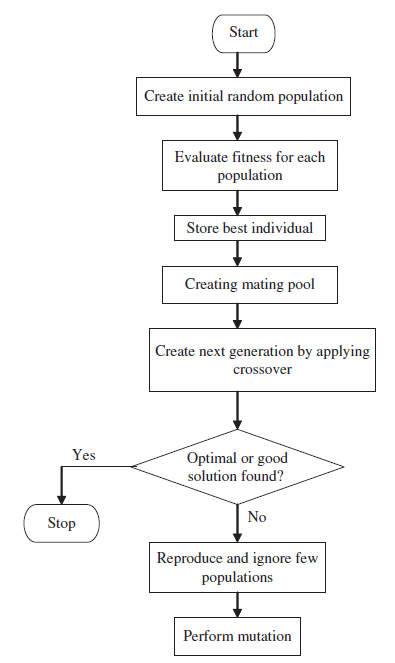
Hence for output layers we should use a **Softsign** function for a Classification problem to compute the probabilites for the classes, and for a regression problem it should simply use a **linear** function.

[13] ***Soft sign function*** is another nonlinear function which can be considered an alternative to tanh.

Softsign function: y = x / (1 + |x|).

Softsign function always gives the output within the range between (-1, 1). Therefore, it can be used to map something between -1 and 1, that something can be an input. Softsign is one of the different types of activation functions. It may not be adopted by professionals and this makes it uncommon. But the fact that, choice of the activation function is state-of-the-art should not be forgotten. It might be the most convenient transfer function for your problem.

**4. USE CASE DIAGRAM**

****

**Fig 4.1** Use case Diagram of Genetic Algorithm

# 7. Testing

We wanted to make sure that all the elements of the developed worked functioned properly. For this, we created a test plan for our work, in which elements such as validation, reliability and user acceptance is tested. The system is tested for normal condition, primarily.

## Testing Methods Used

The different testing methods used are:

### 7.1.1 Static vs. Dynamic testing

The reviews, walkthroughs, or inspections are referred to as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. Static testing is often implicit, as proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete functions or modules. Typical techniques for this are either using stubs/drivers or execution from a debugger environment.

Static testing involves verification, whereas dynamic testing involves validation. Together they help improve software quality. Among the techniques for static analysis, mutation testing can be used to ensure the test-cases will detect errors which are introduced by mutating the source code.

### Testing Levels

Also the testing was carried out on different levels for a proper integration and better functionality of the product. The different testing levels used are discussed below:

#### **Unit Testing**

Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. Developers usually write these types of tests as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other.

#### **Integration testing**

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

#### **Component interface testing**

The practice of component interface testing can be used to check the handling of data passed between various units, or subsystem components, beyond full integration testing between those units. The data being passed and the range or data types can be checked, for data generated from one unit, and tested for validity before being passed into another unit. One option for interface testing is to keep a separate log file of data items being passed, often with a timestamp logged to allow analysis of thousands of cases of data passed between units for days or weeks.

#### **System testing**

This refers to testing for the better support of the web application to the system specified and performance on the different systems and hardware. This helps to get an insight on the hardware specifications that were specified in the requirement analysis of the project. The different testing methods and levels assure that the quality of the product is maintained with a better performance and best level of data security and integrity.

**8. TASK COMPLETED**

* Allowed user to create their own Environment.
* Made the AI run on that specific environment and be good at it.
* Measure the fitness data at the specified environment.
* Evolve the AI through multiple iterative generation at that specific environment.
* Improving the AI to learn more about the environment.
* Making more variable oriented courses to test the AI’s capacity.
* Saving the fitness data so that it can be used to evaluate the AI on other courses.
* Allowing the users to make their own environment and save it.
* Measuring who gets the most fitness data among all users.

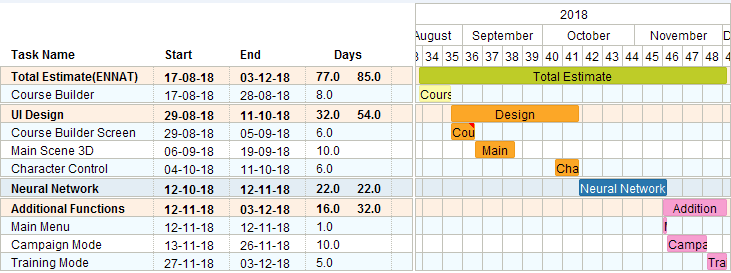
**9. PROJECT TASK AND TIME SCHEDULE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Course Builder | UI Design | Character Control | Neural Network |
| Requirement Analysis | 1d | 2d | 2d | 1w |
| Analysis Of system | 2d | 1d | 1d | 2w |
| Design System | 1d | 1d | 3d | 3w |
| Implementation | 8d | 16d | 6d | 20d |
| Testing and Debugging | 1d | 1d | 2d | 2w |
| Develop Documentation | - | - | - | 4d |

**9.1 TASK AND TIME SCHEDULE**

**Table no.** **9.1** Time Scheduling

**9.2 GANTT CHART**



**Fig no 9.2** Gantt chart for Overall Project Timeline

**9. LIMITATION AND FUTURE ENHANCEMENT**

**Limitation**

There are many kind of advantages of our project but some amounts of limitation are present. These certain limitation is following here:

* It is only a simulation and is not directly applicable to real world without further research and improvements.
* It cannot be directly applied to hardware due to its evolving nature which requires multiple numbers of failures (collision in this case).
* It is limited to an environment that does not change continuously in each iteration.

**Future Expansion**

* With much more research and improvement, it can be applied to real world environment.
* With more study and resources, our project can be applied to changing environment (For e.g. our project can be applied to self –driving cars).
* With time and assets, more training data can be collected and applied in our AI so that it can learn vastly about different and more complex environment.

**10. RESULTS AND CONCLUSION**

**10.1 DEVELOPMENT OF BASIC AI PROJECT**

In final phase of our project, we successfully created an AI that can run on the course that is assigned to it. After a certain course is built and assigned to the AI, at the beginning it will fail but constantly continue to learn to run on the course and after certain tries will be able to run on the given course without any collision. The AI’s progress is measured by the fitness score, the more the AI progresses, the more the score increases.

Now the remaining work to do which was to make a main, more difficult course that the AI has to complete and also, enable the user to make their own course and train the AI and allow them to test their AI on the main course, determining how much more the AI has to be trained if it is unable to complete the main course, has been completed.

**10.2 CONCLUSION**

After the development of the AI we have come to the following conclusion:

* Allowed user to create their own Environment.
* Made the AI run on that specific environment and be good at it.
* Measure the fitness data at the specified environment.
* Evolve the AI through multiple iterative generation at that specific environment.

We believe that any project no matter how or where is done we must fulfill the condition or reach the goal of the system. Also the project must solve the real world problem and also help to upgrade the existing system. Also, our team has learn about the importance of teamwork, work division and time management skill while doing our project.

Finally, one must focus on what they want to create or innovate in whatever he/she is involved as we know that creating the masterpiece takes time and real hard work.

**BIBLIOGRAPHY**

1. <https://www.lua.org/start.html>
2. <https://www.lua.org/pil/1.html>
3. <https://www.slideshare.net/dloiacono/evolving-neural-networks-through-augmenting-topologies>
4. <http://bugra.github.io/work/notes/2015-05-17/introduction-to-deep-learning-with-lua-gradients/>
5. [https://www.sciencedirect.com/topics/chemical-engineering/Feed-forward-neural-networks](https://www.sciencedirect.com/topics/chemical-engineering/feedforward-neural-networks)
6. [https://brilliant.org/wiki/Feed-forward-neural-networks/](https://brilliant.org/wiki/feedforward-neural-networks/)
7. https://en.wikipedia.org/wiki/Reinforcement\_learning
8. https://www.britannica.com/technology/artificial-intelligence
9. https://www.researchgate.net/publication/228394623\_A\_brief\_review\_of\_feed-forward\_neural\_networks
10. <https://medium.com/@dim.wael/how-to-build-a-genetic-algorithm-in-unity3d-77d336da14de>
11. https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f
12. <https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning->activation-functions-when-to-use-them/
13. https://sefiks.com/2017/11/10/softsign-as-a-neural-networks-activation-function/