Revue d'Intelligence Artificielle
Vol. 37, No. 2, April, 2023, pp. 249-256
Journal homepage: http://iieta.org/journals/ria

Analogical Study of Activation Concept in Neural Networks with Neat- Python Module

Nishit Kaul[1], Sameer Kaul[2], Majid Zaman[3], Waseem Jeelani Bakshi[4], Sheikh Amir Fayaz[2]*

[1] Jhonson Space Center, NASA Pkwy, Texas 77058, US
[2] Department of Computer Sciences, University of Kashmir, Srinagar 190006, J&K, India
[3] Directorate of IT & SS, University of Kashmir, Srinagar 190006, J&K, India
[4] Department of Computer Sciences and Engineering, University of Kashmir, Srinagar 190006, J&K, India

Corresponding Author Email: skh.amir88@gmail.com

https://doi.org/10.18280/ria.370201

Received: 15 February 2023
Accepted: 31 March 2023

Keywords:
activation function (ω), trigonometric ratios, mathematical detection, neural networks, NEAT, PyGame, CsV

ABSTRACT

Determining real-time machine simulation and functionalities of complex AI Engines is difficult to comprehend and is rarely discussed. We present a technique to analyze the workflow of one such engine, the NEAT engine, one of the fundamental and robust training engines in the current machine learning scenario. Computer Vision also presents a great approach towards working in real-time speedy functioning virtual simulators and visual platforms, whereas NEAT is not specialized in the same, but can perform to the best of the extent in the present day Machine Learning aspect. Technologies like Python, PyGame, and CsV were used to facilitate the research. So far, we have tested both the frameworks on real time cases, and it is safe to say that the NEAT module has presented an accurate trajectory, besides greater time complexity. Thus, we not only evaluate the accuracy but the other key factors as well. This study demonstrates that NEAT has the ability to address other difficult issues in the future and can produce excellent outcomes with a relatively small population. Robotics, artificial intelligence for video games, natural language processing, and healthcare are some of the potential future applications for NEAT.

1. INTRODUCTION

Flappy Bird is a popular mobile game that was released in 2013. The objective of the game is to guide a bird through a series of pipes without touching them. The bird can be controlled by tapping the screen, causing it to flap its wings and gain altitude.

One approach to designing an AI agent that can play Flappy Bird is to use the NEAT (NeuroEvolution of Augmenting Topologies) algorithm. NEAT is an evolutionary algorithm that evolves neural networks with complex topologies by adding and removing nodes and connections.

To apply NEAT to Flappy Bird, we can represent the game state as inputs to a neural network. These inputs could include the bird's height, the distance to the nearest pipe, and the height of the opening in the pipe. The neural network would then output a single value indicating whether the bird should flap or not.

During each generation, a population of neural networks is created and evaluated by playing Flappy Bird. The networks that perform the best are selected to reproduce and create a new generation of networks. Through this process of selection, reproduction, and mutation, the neural networks evolve over time to become better at playing Flappy Bird.

The NEAT algorithm has been successfully applied to Flappy Bird, resulting in AI agents that can achieve high scores and outperform human players. This demonstrates the power of evolutionary algorithms and their ability to solve complex problems.

The machine learning approach known as NEAT (Neuroevolution of Augmenting Topologies), developed by Kenneth O. Stanley, involves training a neural network using evolutionary techniques [1]. Using an evolutionary algorithm, the best-performing networks are chosen from a population of neural networks with varied topologies (i.e., different numbers and configurations of neurons and connections) to breed and generate the following generation of networks.

In the well-known smartphone game Flappy Bird, the user must lead a bird between the spaces between ordinarily placed pipes. It's a popular game that was first developed for mobile platforms [2]. Its only goal is to maintain the player or bird's life, extending as far as possible by avoiding contact with the other pair while navigating a space between two pairs. Figure 1 below depicts the various game phases.

The game's gameplay is extremely straightforward: at every moment, the player has a choice between two options: touching the screen to cause the bird to fly higher or doing nothing to cause it to drop [3]. By modifying the connections and weights inside the network through evolution, NEAT might be used to train a neural network to play the Flappy Bird game.

The network would receive information from the game (such as the position of the bird, how far away the next barrier is, etc.), and it would output bird actions (such as jumping or not jumping). The evolutionary algorithm would rate how well the network performed in the game and use that score to pick the top-performing networks to breed and produce the subsequent generation. With the intention of eventually creating a neural network that can play the Flappy Bird game at a high level, this procedure would be repeated over multiple generations [4]. The major contribution of this study is to understanding the concepts of Entity Size, Generative Index, Reaction Time, and activation functions. To develop an understanding of essential artificial intelligence basic principles and the right neural network process.

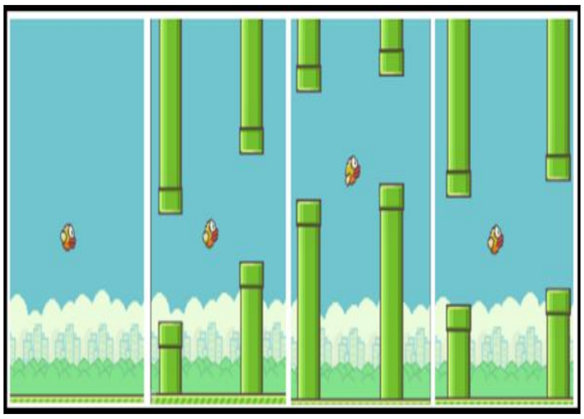
Figure 1. General phases of the Flappy Bird game

2. LITERATURE SURVEY

Numerous researchers used a variety of methodologies, including neural networks (Tensor flow), reinforcement learning, neuro-evolutionary methods, and others to contribute to Flappy Bird. This paper describes a few of the brief investigations by several researchers.

In order to create autonomous virtual players utilizing the NEAT neuro-evolutionary method to generate an agent that can play the Flappy Bird game, this research [5] suggests a minimal training technique. The simplest neural network architecture capable of playing the game flawlessly was discovered using NEAT. In order to achieve proper representation of the issue in comparison to the actual game, the fitness function and scenario modelling were set. A weighted average based on several circumstances and scenario-specific components makes up the fitness function. The method has a short convergence time, a low complexity network, and ideal game behaviour by combining the minimal training approach, a representative fitness, and NEAT. The authors of this study [6] apply a Reinforcement Learning model to the smartphone game Flappy Bird that develops control strategies based on visual observations and feedback from when the bird hurts an object. The authors use a general approach that trains the agent to make the proper choice at any time during the game based on the raw pixel values of the game frames. In this study, authors use a kind of Q-Learning to train a Flappy Bird agent to fly and navigate obstacles while they also look at the effects of image preprocessing and other strategies to shorten training time and increase the rewards the agent receives.

This paper [7, 8] illustrates how to generate AI using GA and DNN to build an AI that can win the Flappy Bird game. The neural network is a simple dense neural network with one layer of concealing. The application of GA, especially the mutation of the weight vector, shows its importance for attaining the goals of the AI. In this paper, each bird starts out with an established network. The genetic algorithm, which also oversees the process of mutation, regulates the generation of the subsequent population. After a predetermined number of repetitions, the network finally develops the mutational strategies necessary to win the game.

This paper [9] presents the development and evaluation of an artificial intelligence game-playing agent using genetic algorithms and neural networks. The agent is designed to learn optimal gameplay strategies for the popular "Flappy Bird" game by safely avoiding obstacles and progressing through the levels. The study investigates the impact of varying parameters such as the number of neurons on the hidden layer, gravity, speed, and gap between obstacles on the learning process. The gameplay is divided into two difficulty levels to enable a detailed analysis of the learning process. The Phaser Framework is utilized for HTML5 programming to introduce real-life factors such as gravity and collision. The Synaptic Neural Network library is adopted to implement neural networks and avoid the need to create a neural network from scratch. The machine learning algorithm employed in the study is based on the concept of Neuro-Evolution, which uses algorithms like genetic algorithms to train artificial neural networks. Overall, the study provides insights into the potential of using genetic algorithms and neural networks to develop effective game-playing agents.

This paper [10] examines autonomous controllers for the mobile game Flappy Bird. Three controllers were created and tested: a manually-tuned controller, an optimization-based controller, and a model-based predictive controller (MPC). The optimization-based controller scored the highest, followed closely by the MPC, while the manually-tuned controller scored the least. The choice of planning horizon was critical for achieving high scores. The MPC provided the best compromise between performance and computation speed without requiring elaborate tuning.

This study [11] proposes the use of deep Q-learning to train a model to pass obstacles in the game "Flappy bird". An Image Translation Process and Deep Q-Learning Algorithm were developed to improve model performance. The game background was removed to speed up convergence. After 50 thousand experiments and data recording, the model was able to pass through the third pipe. A total of 2 million attempts were made, which took about 30 hours of experimentation. The study suggests that the proposed model could be adapted for other reinforcement learning tasks in the future.

Authors in this paper [12] proposed using artificial environments and games to test pathfinding and search space optimization algorithms. They chose "Flappy Bird" as an environment and implemented two algorithms: NeuroEvolution of Augmenting Topologies (NEAT) and Reinforcement Learning (RL). NEAT algorithm showed improved performance with larger initial populations. RL algorithm used a Deep Q-learning Network and remembered the state, action, and reward. The goal was to compare the performance of the algorithms based on how quickly the agent could differentiate between rewarding and hostile actions. The study demonstrates the usefulness of game environments for testing AI algorithms.

Overall, these papers demonstrate that the NEAT algorithm is an effective approach for training AI agents for Flappy Bird. It can produce agents that achieve high scores and outperform human players. However, there is ongoing research to improve the performance of evolutionary algorithms for this game and explore their potential in other games and applications.

Since accuracy was not taken into account in any one of these experiments, this conclusion can be drawn. This encourages us to use the NEAT algorithm to focus on the accuracy parameter without first examining how accuracy would affect other parameters like time complexity and other factors.

3. METHODOLOGY

The methodology of Flappy Bird is relatively simple (Figure 2). The player controls a bird, which is constantly moving forward, by tapping the screen or pressing a button to make the bird flap its wings and fly upwards. The objective of the game is to navigate the bird through a series of pipes without hitting them. Each time the bird successfully passes through a pipe, the player earns a point. The game ends when the bird collides with a pipe or the ground. Being Precise, we have used a particular configuration locker file, trained weights and configured the NEAT-Algorithm to our needs. This is very simple as all other developers do, but the main structure and framework, aim rather is to prove the NEAT may look like an ancestor to new robust models, but it's not obsolete [13-16].

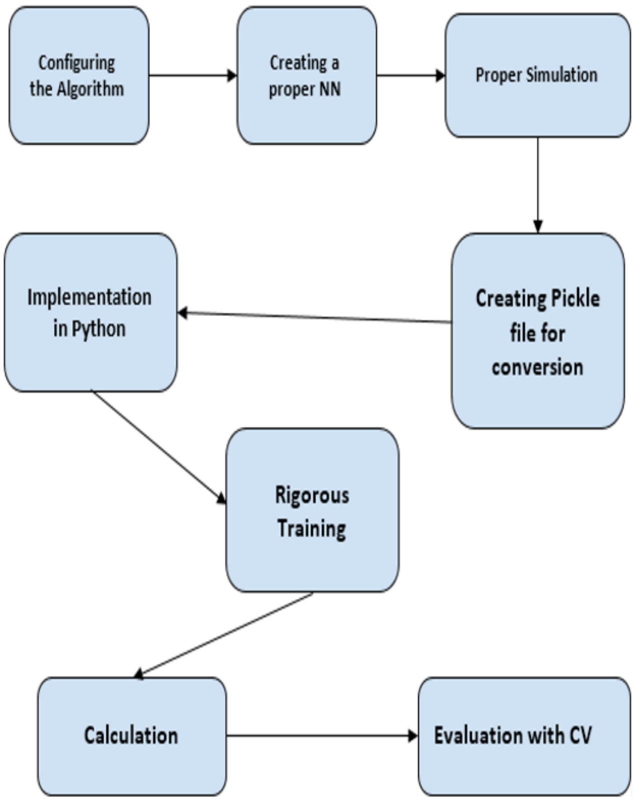

Figure 2. General flowchart of the implemented methodology

The game mechanics of Flappy Bird are designed to be challenging and addictive, with the player needing to have quick reflexes and good timing to navigate the bird through the gaps in the pipes. The game has simple graphics and sound effects, but the difficulty level and the addictive gameplay make it a popular choice among casual gamers.

The methodology of Flappy Bird can be summarized as follows:

Simple game mechanics: The player controls a bird by tapping the screen to make it fly, with the objective of avoiding pipes and earning points.

Challenging gameplay: The game is designed to be difficult, with the player needing to have quick reflexes and good timing to succeed.

Addictive qualities: The game is designed to be addictive, with players wanting to keep playing to improve their score and beat their high score.

Simple graphics and sound effects: The game has basic graphics and sound effects, but they are effective in creating an immersive gaming experience.

Without explicitly constructing the network architecture or manually adjusting the connections and weights, NEAT enables the training of neural networks to play games. Instead, the evolutionary algorithm takes care of these tasks automatically, letting the network experiment with the game and learn how to play it. Following are the steps we take to initialize and set up the environment.

3.1 Initialization

The popular Flappy Bird game was developed; it contained a few obstacles that the user's taps were supposed to help the flying bird navigate through. By teaching an AI Engine to perform a classical masterwork, we give a thorough analytical analysis of AI process. In this instance, we have an initial entity to comprehend the methodology and procedure. Let's say that thing is a Bird_Y (Figure 3) in an Environment (Ĵ).

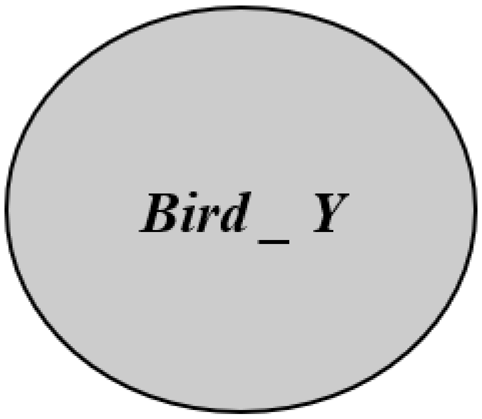

Figure 3. Flappy Bird representation

We used the classical method of giving coordinates to a particular engine X, in this case cored upon NEAT. Keep in mind that these heights are not always constant. In order for the engine to relate to each coordinate of the impediments, we train it to reproduce itself with every data coordinate.

The engine employs the function of $Tan\ H$, where H is the obstruction's height, and uses it as the activation function. It then calculates the generative index (k) (obstruction generation rate), utilizing both size estimation and randomizing the function of $Tan\ H$. The formula can be represented as follows (1):

$$\Delta\, k \;=\; Tan\ H\,.\,S\,[H \,+\, \omega(S)] \tag{1}$$

where k is the generative index, $Tan\ H$ is the function of height and S is the size estimation on past records. ω is the omega of S i.e. factorial value of the size and height. This was the case of height estimation. Now, to detect the obstruction, the engine uses Bird_Y by sending the above activation function to every coordinate and detecting the perfect $Tan\ H$ value. In this way, the engine is able to detect the obstruction. Below table (Table 1), shows the ratio of the activation functions with different bird generations. In the classical Flappy Bird game, we have seen that there is no particular end to the game, even anyone may get old playing that [17-19]. To revise here, Flappy Bird was a classical icon, known for it's 2-

D physics and very-minute mechanical_features. The game is very constant as you progress to the never-ending end. During research, we observed that the speed_mechanics remain very common in every progressing stage, but the pole-height dynamics keep on changing, which are very much defined in the algorithms.

Table 1. Ratio Functions with different bird generations

Gen_ID	Rate Index	Ratio to Activation Function	Pole_Height_Reaction	Bird_Height(+ve for above pole) (-ve for below pole)
Bird_Y: 1	1	1:1	9.0(0.80s)	+3.0, -2.3
Bird_Y: 2	3	1:3	9.3(0.60s)	+2.2, -0.1
Bird_Y: 3	5	1:5	9.88(0.46s)	+1.9, -2.2

Now, we can see the proof of this being a reinforcement learning method, the results get better with each new generation, but it is crucial to note that the Bird Y:2 has the most challenging environment, which show the gameplay remains balanced at the start and end, with a flux in- the middle, the part where most of the players get eliminated. The flux, too, in pole heights, which creates a very short space for the bird to fly, evident from the last column of the Table 1 above [20].

3.2 Environmental setup: Flappy Bird

The steps that must be taken in order to set up the environment are as follows:

1. Create a development environment: We set up Python and the tools required, such as Pygame and Pyglet, to create the Flappy Bird game and execute the NEAT algorithm.
2. Build the Flappy Bird game: We build up the user input management, visuals, and game logic for the Flappy Bird game. A game creation library like Pygame or Pyglet was used for this.
3. Define the neural network's inputs and outputs: The input and output parameters for the neural network that manages the Flappy Bird game must be chosen. The position of the bird, the distance between obstacles, and other information are inputs. The output was a binary value that indicated whether or not the bird should jump.
4. Implement the NEAT algorithm: The NEAT algorithm, which entails generating a population of neural networks with various topologies, assessing their performance, choosing the best-performing networks to breed and generate the following generation, and repeating this process over a number of generations, implemented in Python.
5. Train the neural network: After implementing the NEAT method, we must train the neural network by playing the Flappy Bird game and modifying the connections and weights within the network using the NEAT algorithm. This can be accomplished by repeatedly running the game, feeding the neural network data from the game, and then assessing and enhancing the network's performance using the NEAT algorithm.

Once the neural network has been trained, we can test its effectiveness by running the Flappy Bird game and observing how well it functions. To enhance the performance of the network, we might have to tweak the NEAT settings or other factors.

3.3 Neural network: Flappy Bird

These general procedures must be followed in order to create a neural network for NEAT-Flappy Bird:

1. Define the network's inputs and outputs: The input and output parameters for the neural network that will operate the Flappy Bird game needed to be chosen. The position of the bird, the separation between obstacles, and other information are inputs.
2. Create a population of neural networks: Using the specified inputs and outputs as the network's input and output layers, we had to build a population of neural networks with various topologies. The number of hidden layers and the number of neurons in each layer could be included in the network structure.
3. Initialize the weights and biases of the networks: Because these values are altered throughout training, we had to initialize the weights and biases of the networks randomly.
4. Training the neural networks: To train the neural networks, we had to play the Flappy Bird game while modifying the connections and weights in the network using the NEAT algorithm. This can be accomplished by repeatedly running the game, feeding the neural network data from the game, and then assessing and enhancing the network's performance using the NEAT algorithm.

The NEAT method was used to pick the top-performing networks after the neural networks had been trained in order to breed and create the following generation of networks. Using the NEAT method, we must repeatedly train the neural networks across a number of generations in order to assess and enhance their performance. Since, this is a Reinforcement-Learning based method, repetitive analysis and feedforward_configurations are to be made. Furthermore, we did not extend the cycle of 3-repetitive conjunctions in the network_modification process. Yes, it is a hectic process during the initial initialization process, and weight collection, but an expert would know that no generic-process under the umbrella of Reinforcement-Learning is straight-forward and uni-processed [21]. Also, it is essentially a problematic situation when you deal with both NEAT, and NN hand-in-hand, as there are certain weight-cycles to go through in the process. Furthermore, the newer process of gradient-boosting has a reduced number of cycles to go through. Hereafter, NEAT and NN would have got a very less number of retries, if the quality of data was greater. Further, **CV** has got a complete lead here due to its procession on a certain caffe_model, which helps it to go from the second or even very_first try of the program [22-24].

4. EXPERIMENTAL ANALYSIS AND RESULTS

The purpose of the analytical experiment was to demonstrate the continued relevance and effectiveness of the

NEAT algorithm in its original and robust form. The NEAT algorithm, which stands for NeuroEvolution of Augmenting Topologies, is a machine learning technique that is used to evolve artificial neural networks. The goal of this experiment was to show that NEAT is still a viable option for real-time applications, and to compare its performance to that of a Generic CV model.

To carry out the experiment, we tested both the NEAT and Generic CV models around 15 times with different input rates on serial_baud. Then we compared the results and included the two best results in both cases. The NEAT algorithm was found to work repeatedly on evolution, which increases the time complexity compared to the Generic CV model.

One of the key findings of the experiment was that NEAT is still effective in real-time applications, despite the increasing availability of modern styled visual forms of services in the industry. The results showed that the NEAT algorithm was able to adapt and evolve to different input rates, and that its performance was comparable to that of the Generic CV model.

However, we did note that the NEAT algorithm has a higher time complexity compared to the Generic CV model. This means that the NEAT algorithm may take longer to process data and produce results, which could be a potential drawback in certain applications.

Overall, the experiment demonstrated the continued effectiveness of the NEAT algorithm in its original and robust form, and showed that it is still a viable option for real-time applications. We suggested that future studies could focus on optimizing the time complexity of the NEAT algorithm, and on exploring its potential applications in other fields. The results of this experiment could be useful for developers and researchers who are interested in using machine learning techniques for real-time applications.

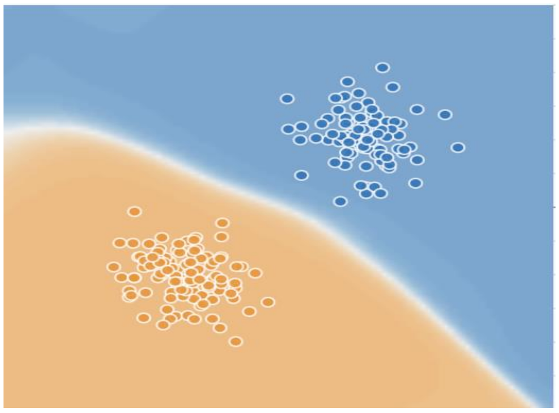

Figure 4. Mutations in network

The figure displayed as Figure 4 represents the mutations that are associated with discrimination in the proper schematic neural network. The schematic neural network is a visual representation of a computational neural network, showing how different parts of the network are connected to each other.

In this case, the figure shows how specific mutations within the neural network can impact its ability to discriminate between different inputs. These discriminatory mutations are highlighted in the figure, showing how they alter the connections within the network.

To better understand the impact of these mutations on the behavior of the network, a demo of the trajectory of a flappy bird is shown in Figure 5. This demo shows how the bird's trajectory can vary based on its position within the game. For example, when the bird is at the bottom of the screen, it may need to flap its wings more frequently to avoid hitting obstacles, while at the top of the screen, it may need to flap less frequently.

The trajectory of the bird is influenced by the neural network's ability to process inputs and generate outputs. Specifically, the network takes in information about the bird's position and speed, as well as the location of obstacles within the game. It then uses this information to generate an output that controls the bird's movement, such as whether it should flap its wings or not.

The discriminatory mutations highlighted in Fig 4 can impact the network's ability to process this information and generate accurate outputs. For example, if a mutation alters the connection between two parts of the network, it may cause the network to misinterpret certain inputs or generate incorrect outputs. This, in turn, can impact the bird's trajectory within the game.

Overall, these figures demonstrate the importance of understanding the underlying mechanisms that influence the behavior of neural networks. By identifying specific mutations that impact the network's ability to discriminate between inputs, researchers can gain a better understanding of how the network functions and develop strategies to optimize its performance.

Additionally, the demo of the bird's trajectory highlights the importance of considering context when analyzing neural network behavior. The bird's movement is influenced by a variety of factors, including its position within the game and the location of obstacles. By taking these factors into account, researchers can develop more nuanced understandings of how neural networks function in complex, real-world environments.

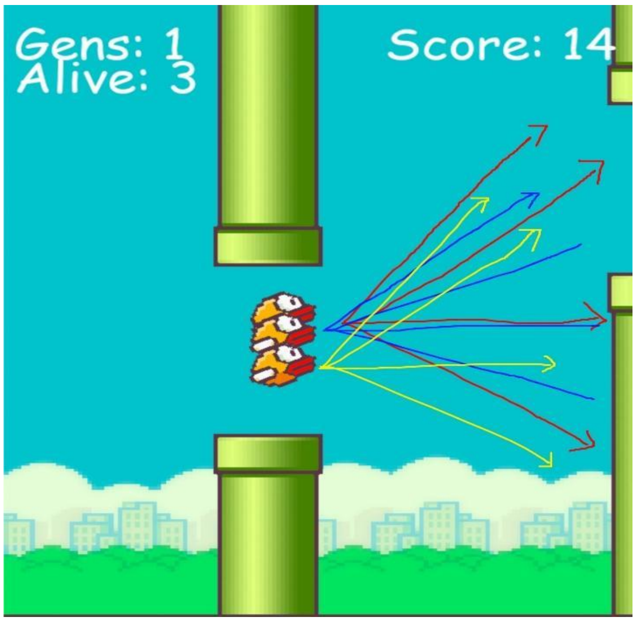

Figure 5. Demo of the trajectory mutation

For the feedforward neural network we have setup the various fitness criteria, threshold, node activation options like mutation rate and other aggregation options and so on, The parameters, setted here, are a fusion of the optimal game mechanics and our demands, for a proper simulation, for example the pop_size is the max generation rate of the birds, so that the module does not harm the game environment. the reset_on_extinction part is the part which suggests the game should be restarted automatically once all Gens die, as shown in Figure 5, the amount of Gens alive is 3. The other options are necessary to configure the model, like the declaration of the activation function, we have the following data:

```
[NEAT]
fitness_criterion     = max
fitness_threshold     = 100
pop_size              = 50
reset_on_extinction   = False

[DefaultGenome]
# node activation options
activation_default      = tanh
activation_mutate_rate  = 0.0
activation_options      = tanh

# node aggregation options
aggregation_default     = sum
aggregation_mutate_rate = 0.0
aggregation_options     = sum

# node bias options
bias_init_mean          = 0.0
bias_init_stdev         = 1.0
bias_max_value          = 30.0
bias_min_value          = -30.0
bias_mutate_power       = 0.5
bias_mutate_rate        = 0.7
bias_replace_rate       = 0.1
```

5. DISCUSSION

NEAT, although being an outmoded NN method, provided a 98% accurate and equivalent outcome when compared to CV-AI. The output tables (Table 2 and Table 3) below demonstrate the same.

Table 2. Neat performance

Input rate	Baud_loss	Time Complexity $O(n)$	Correct trajectory ($O(TanH)$)
1.03333(TanH)	-0.33%	4.333 sec	92.33%
1.09999(TanH)	-0.333377%	3.864 sec	94.09%

Table 3. Generic CV model Performance

Input rate	Baud_loss	Time Complexity $O(n)$	Correct trajectory ($O(TanH)$)
1.03333(TanH)	0.238%	2.23sec	89.02%
1.09999(TanH)	0.11199%	2.48 sec	88.74%

The figure displayed as Figure 6 represents a graphical representation of the performance of NEAT and Generic models. While CV-based models operate more quickly and efficiently in real-time, NEAT-based models offer greater accuracy and stability in the flow of operations. However, it is also possible to train NEAT models in real-time and achieve good performance, as long as the appropriate constraints are put in place.

The processing and use functions of the model can also have a significant impact on its performance and the level of difficulty in managing the project. For example, the mathematical activation function used in the model can influence its performance. In computer vision applications, the Tan H activation function can be replaced by Sigmoid to improve performance.

There are several significant differences between NEAT-based and CV-based Flappy Bird AI models, including differences in complexity of development. NEAT-based models rely on the evolution of neural networks to improve their performance over time, while CV-based models typically use pre-trained models and algorithms to recognize patterns and make decisions.

A genuine note for the readers, we are not trying to completely defenestrate the **Generic-CV** model, instead we are trying to show our panglossian attitude towards the **NEAT-module.** There are a lot of key factors that have an edge over the traditional RL-based NEAT module. **Baud_loss** while structuring is one, and many others have been mentioned further. We have had common arguments on the time taken by NEAT-module portray the data-based results, which is often considered illogical by many experienced ML Engineers, but it is grave to note, that the **NGS- Framework (UNIX)** is still a main project operating on the NEAT- module. When it comes to Pole-balancing, nothing can beat **NEAT-module.** Automated Gas-Pipelines were a sharp architecture built with the oldest of oldest versions of these algorithms. Yes, there are methods like Gradient-Boosting, which have a handsome edge over this algorithm, but NEAT is the obvious frontline on minute-details based projects. The other most challenging part is the slow convergence to optimal results, especially in highly confusing environments, like space-tech, or even generic astronomy, but the competence of this module with (algorithms like **A3C and DDPG**) has proved its worth. Furthermore, the modules that today are compared with NEAT, are somewhat built with the methods that pre-existed in NEAT. Like, generic_convergence and diverse_framworks. The CV model has inherited the Caffe-byte conversion from the NEAT Architecture.

The graphical representation of the performance of NEAT and Generic models, shown in Figure 5, provides a visual representation of how these two models compare in terms of their performance. The figure shows how the two models perform across different input rates on serial_baud, with the two best results for each model highlighted.

Overall, this statement highlights the importance of understanding the differences between different types of AI models and the factors that can impact their performance. It suggests that while NEAT models may require more complex development and training, they can offer greater accuracy and stability in certain applications. However, the choice of model and processing functions should be carefully considered based on the specific requirements and constraints of the project at hand. Now, NEAT is a module that was introduced very early in this rapidly over-expanding field of machine learning, and is often considered as the initial-zone for simulative engines and complex-mathematical procedures. It is grave to consider other overall factors that are a crucial part of this evaluation of NEAT's legibility and true for real time scenarios in today's world. We'll be now, taking a brief look at some other key factors that stay in the defense of NEAT's compatibility, and some others that oppose the modus-operandi of this veteran simulation module. It is very obvious that Generic-CV has some beneficial improvements, and will definitely be able to tackle some factors against NEAT. The factors that we'll consider in this part of the evaluation are: {**Time Complexity o(*n*), Time-to-Time coordination with the bird *(w), and Generic Tracing (R)***}. The main difference where the efficiency factor comes into play is the use of **.pickle** file extension which is very crucial while operating on NEAT. The basic purpose for this, is to serialize Python Object Structures which refers to the process of converting an object into a byte stream, thus each and every trajectory has to be converted into a stream and transferred to the memory. The CV model has got an advantage, as it operates on real-time graphic-byte-streams. We have discussed the **Time Complexity (*0(n)*)** before, and the Computer Vision model has got a clear advantage there. Hereafter, if we check on the **Generic Tracing (R) ,** We can conclude that NEAT-module has to rest down the debate, because there is no model better than **CV or OpenCV** at capturing real time streams of images (whether in presence of Cudart64_dll or not). The **CV** model has got a knack of tracing the object to the best of it's potential. These factors create a fast-paced environment for the **CV-model** which is Agathokakological for it, because it can increase it's efficiency, but compromise the accuracy a bit. Sometimes, the CV model has an edge, where the NEAT-module is not able to compute in a certain heap, or benchmark.

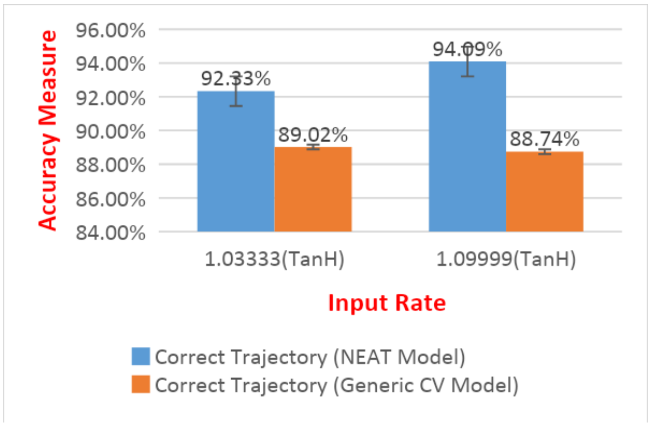

Figure 6. Performance of NEAT model and Generic CV model

The way the AI analyses the game data is one notable difference. A neural network used by an NEAT-based AI to interpret game data receives input from the game and generates output actions based on this input. Contrarily, a CV-based AI processes game data using computer vision methods, which entails examining the game screen and extracting pertinent data from it. The manner the AI is trained is another distinction. By building a population of neural networks with various topologies and using an assessment function to choose the best-performing networks to breed and produce the following generation, an evolutionary algorithm is used to train an NEAT-based AI. Contrarily, supervised learning is often used to train CV-based AI. This method entails giving the AI labelled training data and using an optimization algorithm to modify the network weights and biases in order to reduce the error between the expected and actual outputs. The degree of interpretability of the AI's decision-making process is a third distinction. As an NEAT-based AI is trained via an evolutionary algorithm that modifies the connections and weights inside the network rather than providing explicit rules or guidelines for decision-making, it is often less interpretable than a CV-based AI. Contrarily, because CV-based AI is trained via supervised learning and can be made to extract particular traits or patterns from the game screen that are important to the decision-making process, it is frequently easier to interpret.

In summation, the processing of game data, training methods, and interpretability of decision-making processes differ between NEAT-based and CV-based Flappy Bird AIs. The project's particular objectives and limits will determine the optimal method.

6. CONCLUSION

The conclusion of the research states that we were able to examine the model's operation in this study, including the neural network's mathematical processing and visual computation. We also learned how to build an AI engine using the NEAT algorithm and observed how to improve its statistical performance in situations like Flappy Bird.

We also note that despite having a relatively small population, NEAT was able to master Flappy Bird in just 8 generations, demonstrating its excellent method and high level of effectiveness. The simplicity of the solutions generated by NEAT makes it incredibly effective, and future games can adopt this method to discover the ideal solution in a relatively small number of generations.

Additionally, we were able to predict the time and expected number of generations required for the program to learn the same thing due to the rigorous learning process. This information can be used to optimize the training process for future AI agents for Flappy Bird and other games.

Overall, the conclusion emphasizes the effectiveness and potential of the NEAT algorithm for training AI agents for games like Flappy Bird. The study suggests that NEAT (NeuroEvolution of Augmenting Topologies) can achieve impressive results with a relatively small population, indicating that it could be a useful tool for solving other complex problems in the future.

This study also implies that NEAT has been tested on a complex problem and has achieved impressive results, indicating that it could be applied to other complex problems in the future. The findings of the study may encourage researchers to explore the potential of NEAT for solving other complex problems.

In conclusion, this research suggests that NEAT can achieve impressive results with a relatively small population and has potential for solving other complex problems in the future. The future scopes for NEAT include robotics, artificial intelligence for gaming, natural language processing, and healthcare. By further exploring the potential of NEAT, researchers may be able to develop new and innovative solutions to complex problems in various fields.

REFERENCES

[1] Stanley, K.O., Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. Evolutionary Computation, 10(2), 99-127. https://doi.org/10.1162/106365602320169811

[2] Heier, C. (2015). Free to play: mobile gaming and the precipitous rise of freemium. The Review: A Journal of Undergraduate Student Research, 16(1), 5-11.

[3] Appiah, N., Vare, S. (2018). Playing flappybird with deep reinforcement learning. Junthbasnet, pp. 1-6.

[4] Gu, C., Chen, J., Lin, J., Lin, S., Wu, W., Jiang, Q., Yang, C., Wei, W. (2022). The impact of eye-tracking games as a training case on students' learning interest and continuous learning intention in game design courses: Taking Flappy Bird as an example. Learning and Motivation, 78: 101808. https://doi.org/10.1016/j.lmot.2022.101808

[5] Cordeiro, M.G., Serafim, P.B.S., Nogueira, Y.L.B., Vidal, C.A., Neto, J.B.C. (2019). A minimal training strategy to play flappy bird indefinitely with NEAT. In 2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), Janeiro, Brazil, pp. 21-28. https://doi.org/10.1109/SBGames.2019.00014

[6] Pilcer, L.S., Hoorelbeke, A., Andigne, A.D. (2015). Playing flappy bird with deep reinforcement learning [C]. IEEE Transactions on Neural Networks, 16(1): 285-286. http://dx.doi.org/10.13140/RG.2.2.13159.96165

[7] Deepkumar, N., Vasudevan, V. (2022). Neuro-Evolution in Flappy Bird Game. In 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), Mysuru, India, pp. 1-5. https://doi.org/10.1109/MysuruCon55714.2022.9972637

[8] Nayak, D., Butt, M.A., Zaman, M., Themazi, D.A. (2013). Empowering cloud security through sla. Journal of Global Research in Computer Science, 4(1): 30-33.

[9] Mishra, Y., Kumawat, V., Selvakumar, K. (2019). Performance analysis of flappy bird playing agent using neural network and genetic algorithm. In Information, Communication and Computing Technology: 4th International Conference, ICICCT 2019, New Delhi, India, pp. 253-265. http://dx.doi.org/10.1007/978-981-15-1384-8_21

[10] Piper, M., Bhounsule, P., Castillo-Villar, K.K. (2017, October). How to beat Flappy Bird: A mixed-integer model predictive control approach. In Dynamic Systems and Control Conference, vol. 58288: V002T07A003. https://doi.org/10.1115/DSCC2017-5285

[11] Gu, J., Guo, Y., Lam, Y., Pu, Z.B. (2023). Flappy bird game based on the deep Q learning neural network. Highlights in Science, Engineering and Technology, 34, 191-195. http://dx.doi.org/10.54097/hset.v34i.5448

[12] Selvan, J. P., Game, P.S. (2022). Playing a 2D game indefinitely using NEAT and reinforcement learning. arXiv preprint arXiv:2207.14140. https://doi.org/10.48550/arXiv.2207.14140

[13] Kaul, S., Fayaz, S.A., Zaman, M., Butt, M.A. (2022). Is decision tree obsolete in its original form? A burning debate. Revue d'Intelligence Artificielle, 36(1), 105-113. https://doi.org/10.18280/ria.360112

[14] Rehman, A., Butt, M.A., Zaman, M. (2021, April). A survey of medical image analysis using deep learning approaches. In 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, pp. 1334-1342. https://doi.org/10.1109/ICCMC51019.2021.9418385

[15] Amir, S., Zaman, M., Ahmed, M. (2022). Numerical and experimental investigation of meteorological data using adaptive linear M5 model tree for the prediction of rainfall. Review of Computer Engineering Research. http://dx.doi.org/10.18488/76.v9i1.2961

[16] Fayaz, S.A., Zaman, M., Butt, M.A. (2021). An application of logistic model tree (LMT) algorithm to ameliorate prediction accuracy of meteorological data. International Journal of Advanced Technology and Engineering Exploration, 8(84), 1424. http://dx.doi.org/10.19101/IJATEE.2021.874586

[17] Altaf, I., Butt, M.A., Zaman, M. (2021, September). A pragmatic comparison of supervised machine learning classifiers for disease diagnosis. In 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, pp. 1515-1520. https://doi.org/10.1109/ICIRCA51532.2021.9544582

[18] Mir, N.M., Khan, S., Butt, M.A., Zaman, M. (2016). An experimental evaluation of bayesian classifiers applied to intrusion detection. Indian Journal of Science and Technology, 9(12), 1-7. http://dx.doi.org/10.17485/ijst/2016/v9i12/86291

[19] Ashraf, M., Zaman, M., Ahmed, M. (2018, January). Performance analysis and different subject combinations: an empirical and analytical discourse of educational data mining. In 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, pp. 287-292. https://doi.org/10.1109/CONFLUENCE.2018.8442633

[20] Vu, T., Tran, L. (2020). FlapAI bird: training an agent to play flappy bird using reinforcement learning techniques. arXiv preprint arXiv:2003.09579. https://doi.org/10.48550/arXiv.2003.09579

[21] Ebeling-Rump, M., Kao, M., Hervieux-Moore, Z. (2016). Applying q-learning to flappy bird. Department of Mathematics And Statistics, Queen's University. http://kilyos.ee.bilkent.edu.tr/~eee546/FlappyQ.pdf, accessed on Jan. 5, 2023.

[22] Brandão, A., Pires, P., Georgieva, P. (2019, September). Reinforcement learning and neuroevolution in flappy bird game. In Pattern Recognition and Image Analysis: 9th Iberian Conference, IbPRIA 2019, Madrid, Spain, pp. 225-236. https://doi.org/10.1007/978-3-030-31332-6_20

[23] Fayaz, S. A., Kaul, S., Zaman, M., Butt, M.A. (2022). An adaptive gradient boosting model for the prediction of rainfall using ID3 as a base estimator. Revue d'Intelligence Artificielle, 36(2), 241-250. http://dx.doi.org/10.18280/ria.360208

[24] Zaman, M., Butt, M.A. (2012, October). Information translation: a practitioners approach. In World congress on engineering and computer science (WCECS). USA: San Francisco.