International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 12 | December - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

FLAPPY BIRD AUTOMATION USING DEEP REINFORCEMENT ALGORITHM

Arya Sharma, Naman Malhotra, Sameer, Sabahul, Abhinav
Department of Information Science and Engineering, Chandigarh university,
Mohali, Punjab – 140413

ABSTRACT

It is fascinating to see the increasing interest in game learning within the field of Artificial Intelligence.

One noteworthy game that has been developed in this context is Flappy Bird. In this game, the agent learns how to manoeuvre through obstacles and aims to maximise its score by receiving rewards and punishments. Notably, the agent is not provided with any prior knowledge regarding the game environment. In contrast to using raw pixels as input, the agent is trained using domain-specific features. These features include parameters such as the bird's speed, the distance between pipes, and the height of the pipes. The paper discusses the application of the reinforcement deep learning algorithm and the use of domain-specific feature extraction for training an agent to play Flappy Bird. Through this approach, the agent can autonomously learn how to navigate the game and maximise its score without prior knowledge about the game environment. This research sheds light on the exciting advancements in game learning and the potential of domain-specific approaches in training AI agents.

KEYWORDS-featureextraction, pygame,reward,fitness,reinforcement learning.

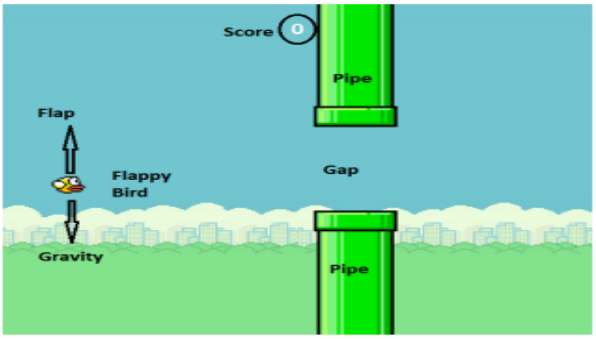
I. INTRODUCTION

The field of study related to video games has witnessed significant growth and enrichment in the past decade. This expansion has allowed researchers to explore various important issues related to commerce, society, economy, and science. With the increasing popularity of mobile devices, there has been a rise in the number of people playing mobile games, including Flappy Bird.

Flappy Bird, created by Dong Nguyen, a Vietnamese video game programmer and designer, was introduced in May 2013. The game gained attention for its addictive nature and high level of difficulty, receiving both praise and criticism. Although the game appears simple, its fast-paced dynamics, diverse environments, and vast search space often result in low scores. This paper focuses on automating the playing of Flappy Bird using reinforcement learning. Reinforcement learning is a method used when there is no predetermined "right" way to carry out a task. It enables an agent to learn and improve its performance through trial and error, receiving rewards or punishments based on its actions. To overcome these challenges, researchers often employ techniques such as deep reinforcement learning. This approach allows the agent to learn directly from pixel data, using deep neural networks to extract relevant features and make decisions based on the learned representations. Deep reinforcement learning has shown promising results in training agents to play complex and visually rich games.

You are correct that relying solely on pixel data and the agent's score makes playing the game effectively highly challenging. Without specific information about the bird's state, the width of the gap between pipes, or the bird's and gap's positions, the agent must learn from the dynamic representations of the game and make decisions based on the pixel data it receives.

Fig: Flappy Bird Game - Schematics



II.RELATED WORK

The related work in this area, primarily conducted by Google DeepMind, has been influential in advancing the field of deep reinforcement learning. Mnih et al., in their papers and, demonstrated the success of training agents to play Atari 2600 games using deep Q-networks (DQN), surpassing human expert-level performance on multiple games.

These works served as inspiration for the project at hand, as they heavily influenced the approach used. Google DeepMind's approach of using DQNs to evaluate the Q-function for Q-learning and integrating experience replay to decorrelate experiences has become the state-of-the-art in deep reinforcement learning.

One of the main strengths of their approach is its ability to train agents despite the high-dimensional input (pixels) and the absence of explicit knowledge about the game's intrinsic parameters. In fact, their agents were able to outperform human experts on three out of seven Atari 2600 games.

However, further improvements have been proposed in subsequent papers. One area of focus is prioritised experience replay, where more informative experiences are given higher priority during training. This can improve sample efficiency and accelerate learning. Another aspect is exploring more efficient training methods to reduce the computational requirements of training deep reinforcement learning agents. Additionally, efforts have been made to enhance the stability of training by introducing techniques such as clipping the loss function to limit its magnitude and updating the target network less frequently.

In summary, Google DeepMind's work on using DQNs and experience replay for deep reinforcement learning has been influential in the field and has inspired subsequent research. While their approach has achieved remarkable results, ongoing efforts are focused on improving different aspects of the training process, such as prioritised experience replay, more efficient training methods, and enhanced stability.

III. METHODOLOGY

proposed work: This proposed system aims to recreate the Flappy Bird game on an FPGA board, providing a smooth playing experience. Players will be able to control the bird's movement by interacting with the hardware or tapping the screen, simulating the original gameplay. By implementing Flappy Bird on an FPGA board, the game can be played on an Android platform with specified hardware configurations.

The computational and hardware aspects of this proposed system will allow for an interactive and enjoyable experience playing Flappy Bird on an FPGA board within the Android platform.

Algorithm: Algorithm to display pictures correctly on the monitor: To display pictures correctly on the monitor, you can follow these steps:

1.Load the image file into memory.

2.Determine the desired coordinate on the frame buffer where the image should be displayed.

3.Iterate over the pixels of the image and map each pixel to the corresponding coordinate on the frame buffer.

4.Write the pixel values to the frame buffer at the mapped coordinates.

5.Repeat steps 2-4 for each image you want to display.

b) Algorithm for image moving: To implement image moving in the game, you can use the following algorithm:

1.Determine the speed and direction of the image movement.

2. Start a game loop that repeatedly updates the position of the image based on the speed and direction.

3.During each iteration of the game loop, calculate the new position of the image by applying the speed and direction to the current position.

4.Update the image's position on the frame buffer accordingly.

5.Continue the game loop until the desired condition for ending the image movement is met.

c)Game over condition setting: To detect the game over condition, follow these steps:

1.Define the conditions under which the game should end, such as collision with an object or reaching a certain score.

2.Check for these conditions during each iteration of the game loop.

3.If the conditions are met, trigger the game over state and perform the necessary actions, such as displaying a game over message or ending the game loop.

d) Adding and fixing some functions: To add and fix functions in the game, follow these steps:

1.Identify the areas of the game that need improvement or additional features.

2.Plan and implement the necessary changes or additions to the code.

3.Test the modified or added functions to ensure they work correctly and do not introduce any issues.

4.Iterate and refine the changes based on feedback and testing results, addressing any bugs or usability concerns.

Set up the architecture of the game, including both hardware and software, and specify the modules for different functionalities:

1.Define the overall structure of the game

2.Develop basic modules for different functionalities

3.Integrate all the modules

IV. MODELLING AND ANALYSIS

To assign fitness to birds in each generation, a reward policy is needed. One possible reward policy could be:

Survival Time: The fitness of a bird can be based on how long it survives in the game. The longer the bird survives, the higher its fitness score.

Distance Covered: The fitness score can also be determined by the distance covered by the bird in the game. The farther the bird flies before colliding with obstacles, the higher its fitness score.

Number of Gaps Passed: The fitness score can be based on the number of gaps the bird successfully passes through without collision. Each successfully crossed gap can contribute to an increase in the fitness score.

To assign fitness to birds in each generation, a reward policy is needed. One possible reward policy could be:

Survival Time: The fitness of a bird can be based on how long it survives in the game. The longer the bird survives, the higher its fitness score.

Distance Covered: The fitness score can also be determined by the distance covered by the bird in the game. The farther the bird flies before colliding with obstacles, the higher its fitness score.

Number of Gaps Passed: The fitness score can be based on the number of gaps the bird successfully passes through without collision. Each successfully crossed gap can contribute to an increase in the fitness score.

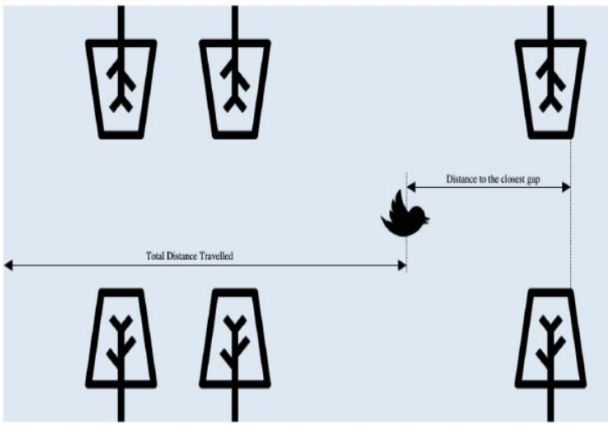


Table I: Reward policy

Description	Reward
alive for a frame.	0.1
passes through pipe	-10
collides with a pip	+10
Bird flies off screen	+20

In a deep Q-network, the input consists of 80x80-pixel sized frames of length 4 stacked together. The raw pixel values are used as input, and the DQN learns to make decisions based on the situation and a learned policy. It makes use of both CPU and GPU. Py game is an open-source library used for developing games. It includes all the necessary tools to build and deploy video games. Scikit-Image is an open-source library used for image processing. It is written in python language and is developed by the SciPy community. OpenCV is an image processing library first developed by intel. It supports a wide range of languages from low level like C to high level like python.

Fig: - flappy bird representation

Architecture

The architecture of the deep Q-network (DQN) used to train the agent to learn how to dodge pipes can be seen in Figure 5. The image below illustrates the neural network architecture:

During the training process, the agent received rewards based on its performance, such as successfully dodging pipes. The DQN learned to update its Q-values using the rewards obtained and the Q-learning algorithm. By updating the network weights iteratively, the agent gradually improved its ability to make informed decisions and navigate through the game environment effectively.

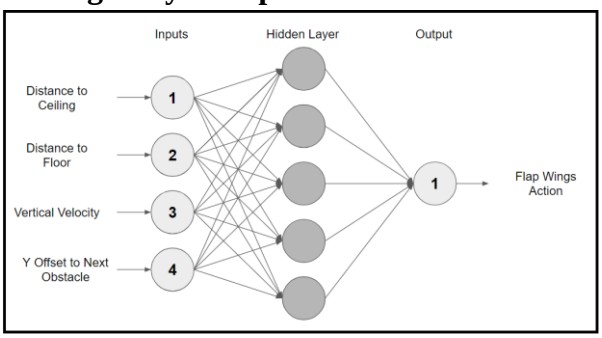
By using this neural network architecture and training methodology, the agent learned to play the game by understanding the current situation and selecting actions based on its learned policy.

•Input Layer. An input layer two neurons shows the agent's perspective and represents what it sees: Horizontal Distance to the closest gap Height difference to the closest gap.

•Hidden Layer. Number of neurons in the hidden layer varied from 1, 5, 10 and 20 and the performance of each was studied.

•Output Layer. An output later with one neuron which provides an action of flap if output >0.5 else does nothing.

Fig: - layers representation



V. EXPERIMENTS AND RESULTS

Based on the provided information, the parameters for the Flappy Bird game are as follows:

Frame Rate: The game is played at 10 frames per second, meaning that the agent receives 10 frames of visual input per second.

State Representation: The agent processes the 3 most recent frames to generate a state. This means that it considers the current frame and the previous two frames to make decisions.

Discount Factor: The discount factor γ is set to 0.9. The discount factor determines the importance of future rewards compared to immediate rewards. A

value of 0.9 indicates that future rewards are valued relatively highly.

Rewards:

r ew ar dPass = +1.0: The agent receives a reward of +1.0 when it successfully passes through a gap between pipes. This positive reward encourages the agent to navigate through the gaps and avoid collisions.

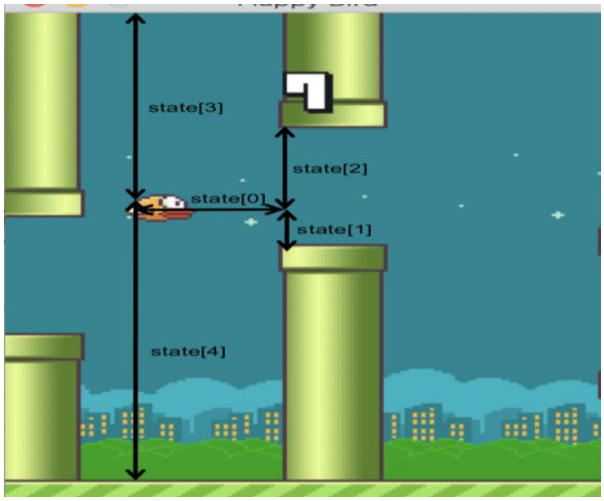
r ew ar dD i e = −1.0: The agent receives a reward of -1.0 when it collides with a pipe or the ground and dies. This negative reward penalises the agent for making unsuccessful moves that lead to its demise.

These reward values help the agent learn to maximise its total reward by incentivizing passing through gaps while minimising collisions.

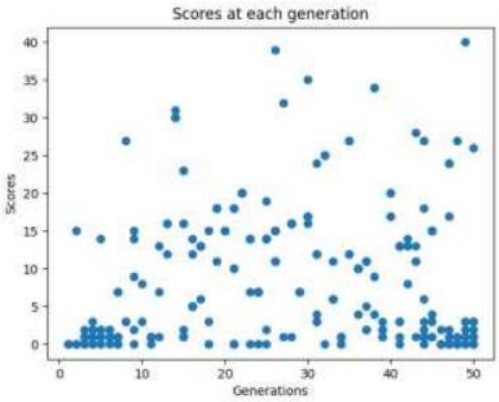
By adjusting these parameters, the game can be configured to encourage desired behaviours and penalise undesired behaviours, allowing the agent to learn effective strategies for playing Flappy Bird.

shows some examples snap and their corresponding scores which make perfect sense.

Fig: - showing rewards



The bird reached a score of 40 after training the agent for 50 generations, which is shown in figure below: -



From the following training data that we have used and after generating training data more than 50 times.

VI. CONCLUSION AND FUTURE WORK

From observing the way, the bird plays, it seems to attempt to reach for gaps but unfortunately crashes at the corners. One possible fix could be to train the model to fit more accurately to the experience data near the pipes. Currently, the experience replays samples uniformly to obtain a minibatch for model updates. Finding a way to sample more experience points close to the dangerous areas could help address this issue, improve the training rate, and enhance convergence.

To implement a deep reinforcement learning framework for playing Flappy Bird close to human level! Even though the results didn't reach super-human performance, it is indeed a step in the right direction.

It is noticeable that the bird tries to reach for gaps but often crashes at the corners. One potential solution could be training the model to fit the experience data more accurately near the pipes. Currently, the experience replays design samples uniformly to obtain a minibatch and update the model. Introducing a way to selectively sample more experience points near the danger areas could help address this issue, improve training rate, and enhance convergence.

In terms of model architecture, it is suggested to consider using a Recurrent Neural Network (RNN)

setup. As Flappy Bird involves decision-making based on informed knowledge of previous states, an RNN can effectively capture the temporal correlations in the game and inform the bird's actions accordingly.

Mentioned that in implementation, removed the background and score to reduce clutter and increase the likelihood of successful training. It would be interesting to investigate how restoring the background affects the agent's performance, as reintroducing the background could potentially provide additional visual cues that impact the decision-making process.

Overall, results demonstrate the capacity of deep neural networks and how a generic reinforcement learning setup like this can learn and play a game with minimal domain knowledge. This achievement opens numerous possibilities for potential applications.

VII. REFERENCES

1. "What is Flappy Bird? The game taking the App Store by storm" by Rhiannon W. This article from The Daily Telegraph discusses Flappy Bird's popularity in the App Store.
2. "ASK for Game Task" website tutorial on machine learning algorithms for Flappy Bird. Accessed on February 10, 2019.
3. "Flappy Bird Game AI using Neural Networks" by A. Kumar, S. Garg, S. Garg, R. Chaudhry (2016). This paper explores the use of neural networks to build an AI for playing Flappy Bird.
4. "Deep Reinforcement Learning for Flappy Bird" by K. Chen (2015). This article presents the application of deep reinforcement learning for playing Flappy Bird.
5. "Type-2 fuzzified Flappy Bird control system" by A. Sahin, E. Atici, T. Kumbasar (2016). This paper discusses a control system for Flappy Bird based on type-2 fuzzy logic.
6. "Evolutionary computation and games" by S.M. Lucas and G. Kendall (2006). This article explores the application of evolutionary computation techniques in games.
7. "Introduction to special section on evolutionary computation in games" by M. Sipper and M. Giacobini (2008). This article introduces a special section on using evolutionary computation in games.
8. "How does a Flappy Bird Learn to Fly Itself" by Y. Nie (2017). This resource discusses the process of training a Flappy Bird AI to learn how to play the game.

These references cover various aspects of Flappy Bird, including its popularity, machine learning algorithms, neural networks, fuzzy logic, evolutionary computation, and the training process for an AI in playing the game.