RESEARCH ARTICLE OPEN ACCESS

A Machine Learning-Driven Framework for Autonomous Gameplay in 2D Arcade Systems

Sangeeta Soni*, Sambit Kumar Panda**, Shivang Agrawal**, Sachin Singh Shekhawat**, Sahil**

*Associate Professor, Department of CSE, Global Institute of Technology, Jaipur, Rajasthan, India
**B.Tech Student, Department of CSE, Global Institute of Technology, Jaipur, Rajasthan, India

ABSTRACT

The Computer games are being used more and more as experimental platforms for machine learning and artificial intelligence research because of their deterministic environments, repeatability, and clearly defined evaluation metrics. Arcade, style games are especially good for reinforcement learning experiments since they have simple mechanics but still, present the player with decision, making challenges that require fast and correct actions as well as adaptation of the behavior. Out of these, Flappy Bird has been recognized as a benchmark environment owing to its feature of having very few rewards and timing, sensitive dynamics which makes it a good test for learning stability and policy optimization. We are proposing a single, comprehensive machine, learning Flappy Bird framework in this work that can run autonomous AI control, allow human gameplay and facilitate multiplayer interaction in real, time. No need for manual rules with reinforcement learning here agents can figure out control policies by themselves, and we use classic Q, Learning as a reference to show tabular learning behavior in limited environments. Moreover, to deal with the issues of scalability and generalization, a Deep Q, Network (DQN) agent has been deployed to estimate value functions over continuous state spaces. Various metrics such as average score, survival time, and learning convergence are used to compare the performances of human players and learning agents]. Also, the multiplayer setup allows us to study how agents behave when there are synchronization and network, induced latency issues.

Keywords **—** Flappy Bird, Machine Learning, Reinforcement Learning, Q Learning, Deep Q Network, Multiplayer Game.

1. Introduction

Over the last few years, machine learning has greatly depended on controlled experimental setups for the evaluation of learning behavior, convergence properties, and decision, making quality. Games have been identified as justifiable testing environments with the opportunity for fast deterministic dynamics, repeatable experiments, and quantitative metrics such as score and survival time [1], [3]. Games allow agents to experiment with sub, optimal actions and failure states without the risk of real, world consequences, which is necessary for reinforcement learning, based methods, which rely on trial, and, error interaction [4].

Arcade games, in particular, are very interesting for reinforcement learning research because of their limited action spaces and fast execution cycles. These features, while reducing computational overhead, still present a challenging control problem that requires precisely timed and adaptive behavior [5], [6]. Therefore, arcade games have been commonly used as benchmarking platforms for reinforcement learning algorithms as well as game AI systems [7].

Flappy Bird is an ideal case study because it effortlessly and elegantly incorporates a simple game with very sensitive control dynamics. Even though the player is limited to just one binary action, the underlying control problem entails continuous state variables such as position and velocity. Therefore, minor timing mistakes can result in immediate failure, making rewards sparse and the length of learning episodes very short [8],.[9] These factors have made Flappy Bird a popular testbed for investigating the stability of learning, reward shaping, and policy generalization in reinforcement learning agents [10], [11].

In fact, Flappy Bird is a well, suited environment for machine learning studies, however, it still raises a couple of issues that are smartly addressed by learning agents.

The first issue is that a manual gameplay is limited by a human nature. Data shows that humans perform accurately only in a limited number of tasks and their performance fluctuates due to tiredness, delay in processing stimuli, lapses of attention, etc. Hence, humans are not precise in performing any motor task [12].

Secondly, non, learning AI, driven solutions to a problem cannot change over time by themselves. A developer who builds a rule, based or a heuristic, based controller needs to manually tune parameters of the system and the solution may not be efficient for the cases of the initial environment. However, the methods perform well only under some limited scenarios and their performance drops significantly once the environment changes [13], [14].

Third, the majority of existing Flappy Bird agents have very little or no capabilities for smart multiplayer interaction at all. Most of the research works focus on controlling a single agent only, resulting in neglecting multi, agent or human interactive scenarios. Besides that, a multiplayer game setting brings in challenges such as synchronization, latency and non, stationary environment caused by other players[15], [16]. These challenges are very relevant when one needs to evaluate agents capable of real, world interactive systems.

The following are the contributions of this work in an attempt to address the issues raised in the previous discussion:

1). Machine LearningBased AI Agent: Without the use of handcrafted rules, a reinforcement learning, based autonomous agent is created to regulate Flappy Bird gameplay. The agent finds the best control policies by learning reward, driven through the direct interaction with the environment [4], [17].

2). Human vs AI Performance Comparison: A thorough evaluation framework is set up for comparing human players and machine learning agents under the same conditions. The performance is checked by means of measurable metrics such as average score, survival time, reaction consistency, and learning convergence [12], [18].

3). Real, Time Multiplayer Integration: A multiplayer game architecture is set up to allow HumanHuman, HumanAI, and AIAI gameplay scenarios. This setup makes it possible to study intelligent agent behavior under real, time networking constraints such as synchronization and latency [15], [19]. As a result, these additions take Flappy Bird beyond a solo, agent learning test to a real device for investigating reinforcement learning, humanAI collaboration, and multiplayer smart systems.

2. Literature Review

Gaming has now become a leading tool for artificial intelligence (AI) and machine learning (ML) research, especially through digitally simulated environments. Games offer determinism in their internal working, repeatability of experiments, and clear performance measurements, which make them especially attractive as platforms for our field of study. Due to these features, games are perfect for experiments on the behaviour of learners, characteristics of convergence, and decision, making strategies within highly controlled settings [10], [12]. Furthermore, arcade, style games are great as benchmarks for reinforcement learning (RL) algorithms as they combine the simplicity of the game mechanics with the control complexity [6], [11].

2.1 Reinforcement Learning in Game Environments

Reinforcement learning is a technique that allows an agent to figure out the best course of action practically by itself through trial and error in an environment, where the only feedback from the environment is the rewards the agent gets. This way, the agent doesn't have to depend on labeled data and predefined rules. Q, Learning and SARSA, which are two classical RL algorithms, were initially game, playing agents' methods considered mainly because of their theoretical convergence guarantees in finite Markov Decision Processes (MDPs) [1], [2]. Sutton and Barto illustrated a scenario where Q, Learning is able to find and stick to the best policy when it is given a set of conditions, thus making it an appropriate method for the first AI game research efforts [1].

On the other hand, tabular RL methods cannot handle continuous or very high, dimensional state spaces effectively, hence the major drawback of poor scalability. That is, discretizing the continuous variables typically results in losing some information and thus weakening the ability to generalize, which, in turn, has a negative impact on performance, especially in games where timing is crucial [2], [8]. Such mode of functioning has led to the pondering on and experimentation with function approximation methods in the game of reinforcement learning.

2.2 Deep Reinforcement Learning for Arcade, Style Games

Deep neural networks as function approximators greatly propelled the reinforcement learning field. Deep Q, Network (DQN) by Mnih et al. branched Q, Learning with deep convolutional neural networks, achieving human, level performance on many Atari 2600 games [3], [6]. Important methods like experience replay and target networks were proposed to stabilize training and avoid divergence problems [3].

Later refinements such as Double Q, Learning and dueling network architectures have additionally contributed to the

stability of the learning process as well as decreased value overestimation [4]. Through these improvements, deep reinforcement learning has become the primary method for arcade, style games with sparse rewards and continuous state dynamics [7], [19].

2.3 Flappy Bird as a Benchmark for Reinforcement Learning

Flappy Bird is a popular game for testing reinforcement learning methods as it offers a simple action scheme along with a control mechanism that is very sensitive. Even though the agent only has two actions to choose from, the state space is composed of continuous variables like the vertical position, velocity, and distance to the obstacles. This means that the agent must make very accurate decisions in terms of both timing and choice of action [8], [9].

Deep reinforcement learning agents, trained in Flappy Bird scenarios, have been demonstrated to produce stable and high, accuracy performances over time, at times even outperforming humans in terms of consistency and reaction speed [8], [9]. On the other hand, heuristic, based and rule, controlled systems rely to a great extent on the manual setting of parameters and are not very adaptable to changes in the environment [13]. Reinforcement learning agents, however, learn directly from interaction, thus, they become more robust and are able to generalize better.

2.4 Human versus AI Performance Evaluation

Human players' gameplay performance naturally varies as it is influenced by factors like reaction delays, tiredness, and concentration changes. Therefore, the performance of humans is typically quite inconsistent over different sessions [12], [16]. In contrast, reinforcement learning agents make decisions at fixed intervals, which are in sync with the simulation clock, thus allowing them to perform control actions consistently and accurately [7].

Literature, based empirical comparisons show that deep RL agents not only beat human players in average scores but also in survival time and performance stability, especially in settings that necessitate very precise motor control [5], [9]. These results emphasize the appropriateness of arcade, style games as a good medium for benchmarking AI performance against human, level scores.

2.5 Multiplayer and Networked Game AIs

Single, agent learning has been the main focus of Flappy Bird and similar arcade game AI research. On the other hand, real, time multiplayer games not only The use of neural networks as function approximators gave learning, based agents the ability to cope with more complex environments. Neural networkbased require synchronization of game state and handling network latency but also have to cope with ever, changing game dynamics due to player interactions [15], [16]. The work in distributed systems demonstrated that delays in the network connection degrade fairness and speed of reaction that can be felt in first, person shooting games, for example [15], [23].

Multi, agent reinforcement learning research indicates that the complex interactions among several agents cause non, stationarity of the environment, thus violating the stationarity assumptions of traditional RL algorithms and agents being unable to converge [17], [18]. The use of RL agents in real, time multiplayer arcade, style games still lacks extensive studies compared to single, agent ones and it is a gap that needs to be filled.

2.6 Research Gap Summary

It is perceived from the literature review that, on one hand, researchers have used reinforcement learning, especially deep reinforcement learning, to solve arcade, style games and Flappy Bird game variants, but most of the experiments have been done in a single, agent setting. On the other hand, human, AI comparisons are not in, depth, and even fewer studies address the implementation of real, time multiplayer with learning agents [12], [16]. Besides, hardly any of the proposed systems have made attempts to bring together classical RL, deep RL, and multiplayer networking in one experimental platform.

This gap in research is a challenge to the development of integrated systems for autonomous learning agents, human benchmarking, and multiplayer interaction in real time, thus opening up reinforcement learning in interactive and networked game environments for further exploration.

3. Related work

3.1 Playing Game Agents with Machine Learning

Game, playing agents have been a major focus of machine learning research for a long time. The reason being games are structured, yet at the same time, they present challenges. The initial methods were based on classical reinforcement learning algorithms, i.e. Q, Learning and SARSA that showed the ability to learn in environments with small and discrete state spaces [4], [20]. Such methods resulted in understandable learning processes but were not very scalable when the state dimension grew.

reinforcement learning agents have become very successful in different game areas by learning value functions or policies directly from state features [7],

[21]. These methods brought a big jump in generalization and performance consistency in comparison with tabular methods.

3.2 Reinforcement Learning in 2D Games

Among the various domains, reinforcement learning has been extensively used in 2D games because these games strike a good middle ground between the simplicity of computational demands and the complexity of decision, making processes. Basically, in these kinds of environments, the agents figure out the right actions by getting rewards as feedback instead of being told directly what to do [10], [22].

Nevertheless, problems arise when rewards are very few, feedback is delayed, and the state spaces are continuous. These problems, on the other hand, necessitate the very thorough work of designing the representations of states as well as the functions of reward [11], [23].

Researchers have suggested state abstraction and state, action space reduction methods that keep the necessary information and at the same time lower the complexity of the computations to solve these problems [24]. Using feature, based representations has been demonstrated to facilitate learning and attain better stability than using raw pixel data in a number of 2D gaming environments [9], [25].

3.3 Multiplayer and AI, Driven Games

Multiplayer games dramatically increase the complexity of the scenario by letting several learners or humans interact in the same environment. In such cases, the environment becomes non, stationary since the state transitions depend on other players' actions [26]. Hence, this breaks the traditional reinforcement learning framework and makes the learning policy more difficult. Research for multiplayer and AI, driven games has focused on synchronization mechanisms, latency compensation, and adaptive behavior to ensure fairness and responsiveness [15], [27]. Hybrid HumanAI gameplay has been used to understand cooperation, competition, and adaptive intelligence in interactive systems [16], [28].

However, there are very few papers that introduce reinforcement learning agents into real, time multiplayer arcade, style games. This is the reason the proposed framework is designed to combine reinforcement learning and multiplayer system design in a single Flappy Bird environment.

4. System Architecture

4.1 Overall Architecture

The Flappy Bird framework designed here, adopts a modular and layered system architecture which is capable of supporting single, player gameplay, machine learningbased autonomous control, and real, time multiplayer interaction. Modular configuration is crucial for reinforcement learning systems as it enables the independent development, testing, and changing of learning algorithms, game logic, and networking components without compromising the overall system stability [1], [2]. The architecture can be thought of as a three, layered structure:

1). Game Engine Layer: The core game mechanics such as physics simulation, obstacle generation, collision detection, scoring, and rendering come under the purview of this layer. To ensure deterministic behavior, which is critical for reproducible reinforcement learning experiments [3].

2). Machine Learning Decision Layer: The decision layer houses the reinforcement learning agent that is in charge of selecting actions based on the current state of the game.

3). This layer is pretty much algorithm, independent so that classical Q, Learning and Deep Q, Network models can be integrated. The decision, making is separated from the simulation to ensure that the human and AI agents will be compared under the same conditions [4], [5].

4). **Networking Layer:** The networking layer facilitates real, time multiplayer gameplay. It oversees the communication between clients and the server, game state synchronization, and player agreement.

In order to keep the game state under authoritative control and avoid desynchronization or unfair advantage, a clientserver model is implemented [6], [7]. The presented layered architecture is a very smart move in terms of AI, driven game systems as it enhances scalability, maintainability, and experimental flexibility, which are the major needs of such systems [8].

4.2 Data Flow Diagram

The interaction between system components is based on a data flow pipeline that works at every simulation step. This pipeline guarantees the synchronisation of decision, making, learning feedback, and multiplayer consistency. The data flows through the following steps:

1. The game engine layer changes the environment state by getting the bird position, velocity, and obstacle locations.
2. The new state is sent to the decision layer where

it is changed into a structured state representation that is the best fit for learning algorithms.

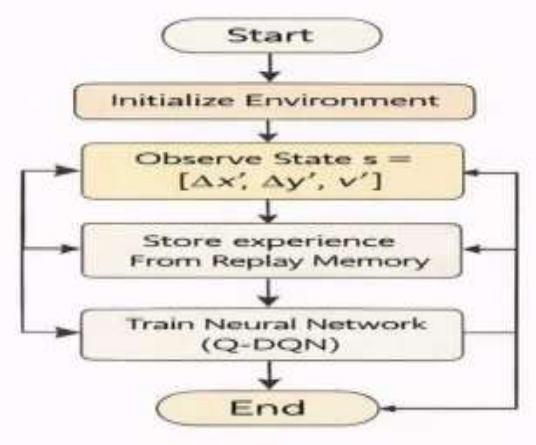

Figure 1: Data Flow Diagram

3. The controller (human input or AI agent) picks an action based on the state that has been seen.
4. The action picked is sent back to the game engine layer and implemented in the simulation.
5. The game engine calculates the reward and next state resulting from the action taken.
6. The reward and next state are given back to the learning agent for policy update (while training).
7. For multiplayer mode, the networking layer will synchronise the updated game state to all connected clients.

This closed, loop feedback mechanism is at the core of reinforcement learning and guarantees that agents not only link actions to long, term outcomes but also keep their real, time responsiveness [9], [10].

5. Machine Learning Model Design

5.1 Problem Formulation as Reinforcement Learning

The Flappy Bird control task has been formulated as a reinforcement learning problem, where an agent interacting with the game environment learns the best control policy by itself. Reinforcement learning is particularly suitable for this problem since there are no explicit labels or optimal trajectories, and the correct behavior must be found through interaction [11], [12]. The problem is represented as a Markov Decision Process (MDP), which is characterised by the tuple S, A, R, P, . At each discrete time step, the agent perceives the state of the environment s S, selects an action a A, and obtains a reward r R from the environment. The agent is to maximize the sum of rewards that are discounted over time and averaged over possible outcomes [13]. This formulation unifies the theoretical basis for both tabular methods and deep reinforcement learning methods to be implemented in the same environment [14].

5.2 State Representation

Choosing the right state representation is a key factor in the success of the learning algorithm. The state vector should be representative of the environment needs to have enough information, on the other hand, it should not contain too much unnecessary information that could make the algorithm slower to converge or even result in over-fitting [15], [16]. The state representation of the proposed system is shown in Fig. 2 where relative pipe distance, vertical offset, and velocity are first normalised and then given to the learning agent.

The state representation in the present system is composed of the following feat**ures:**

1. Vertical position of a bird, which represents its current height relative to the screen.
2. Distance to the next pipe, which is a measure of the time left for action planning.
3. Vertical velocity, which shows the effect of gravity and flapping.
4. Gap position, which is the vertical distance between the bird and the center of the next pipe opening.

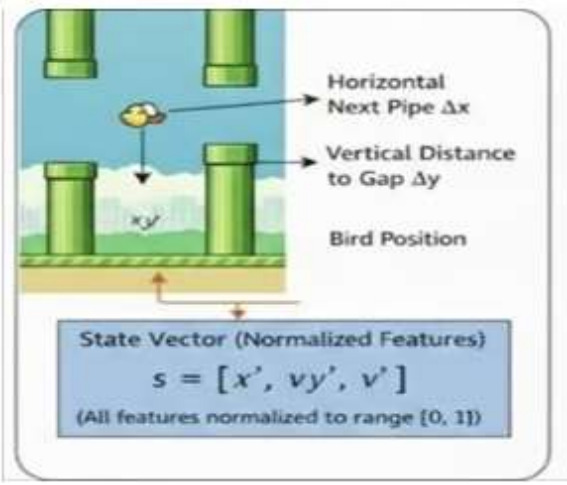

Figure 2: State Representation in Flappy Bird

All these continuous features are normalized to fixed intervals to keep the numerical stability during the training process. It has been demonstrated that the training time can be greatly reduced when using feature, based representations like this one instead of raw pixel data in 2D games [17], [18].

5.3 Action Space

In Flappy Bird, the action space has only two discrete actions which are:

- Flap: Give an upward impulse bird.
- No Flap: By providing no action, the gravity will control the downward motion.

Besides, the binary choice makes the control problem to be complicated because these actions are immediate

and irreversible.

Because each action has an immediate and irreversible impact on future states, the binary action space (no flap, flap) creates a challenging control problem. If the action is ill, timed, the result is almost always an immediate termination, which makes policy learning more difficult.

Binary action spaces are often used in reinforcement learning benchmarks to limit the influence of mechanical complexity when judging decision, making quality [21].

5.4 Reward Function Design

The design of reward has a direct impact on the behavior of the agent and how efficiently it learns. In scenarios where terminal rewards are sparse, like the game of Flappy Bird, inadequate reward functions may cause the agent to converge very slowly or behave unstably [22], [23].

The reward function in this work comprises three parts:

Positive reward for going through a pipe successfully, thus motivating to keep going. A large negative reward for colliding with a pipe or the ground, thus penalizing a fall. A small survival, point reward for each time step the agent stays alive.

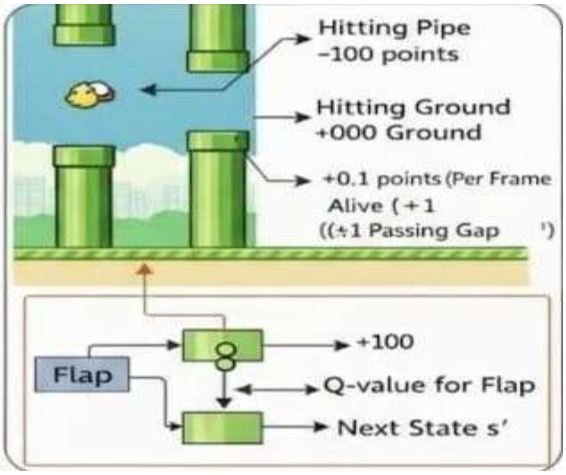

Figure 3: Q-Learning Rewards in Flappy Bird

The way reward shaping works, as illustrated in Fig. 3, uses a combination of sparse and dense rewards to keep policy learning stable.

This reward conditioning method mixes sparse and dense signals from the environment, thus helping the agent learn the goal of stable flight while not giving it the incentive for too aggressive behavior. Reward schemes similar to this one have been demonstrated to increase the rate of convergence and the robustness of policies in arcade, style games [24], [25].

6. Machine Learning Algorithm Implementation

6.1 Q-Learning Algorithm

Q, Learning is used as a baseline reinforcement learning algorithm primarily because of its simplicity, transparency, and the fact that it has been historically significant in game, playing AI research [1], [2]. Technically, it is a model, free, off, policy method of learning that calculates the value of the expected total reward for each state, action combination, thus making it possible for the agent to figure out the best behavior even when not given the environment model [3]. In Q, Learning, the agent keeps a Q, table with entries for each pair of state & action, i.e. each entry Q(s, a) is the predicted value of taking action a in state s. The values are updated during learning by iteratively incorporating the effect of each new reward and state transition. The traditional Q, Learning update formula is:

$$Q(s, a)Q(s, a)+[r+maxaQ(s, a)Q(s, a)]$$

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a'), Q(s, a) \right]$$

$$Q(s, a)Q(s, a)+[r+amaxQ(s, a)Q(s, a)]$$

Where \alpha is the learning rate that controls the size of the update, and \gamma is the discount factor that determines how much future rewards are taken into account [4].

As the state space of Flappy Bird is continuous, the Q, Learning with tables demands a discretization of the state variables to be able to work. Although it allows the learning in a much simpler way, discretization carries the risk of losing some detail and the agent becomes less capable of generalizing to new but similar states [5], [6]. So, in essence, Q, Learning is mainly a baseline model that advanced techniques are tested against.

6.2 Deep Q, Network (DQN)

To address the problems of scalability and generalization of tabular methods, a Deep Q, Network (DQN) is used. Instead of a Q, table, DQN employs a neural network to estimate the action, value function Q(s, a;)Q(s, a;)Q(s, a;), where stands for the network parameters [7], [8]. In the same way that Fig. 4 illustrates, the DQN gets the normalized state vector and produces Q, values for each action that could be taken.

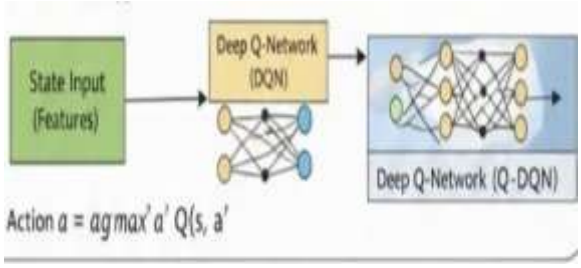

Figure 4: Deep Q-Network Decision Process in Flappy Bird

The structure of the neural network is as follows:

An input layer corresponding to the normalized state

features, Several fully connected hidden layers with nonlinear activation functions.

An output layer producing Q, values for each possible action. By using function approximation, the DQN agent is able to generalize to new states which have not been seen and learn control policies that are more smooth, which is very important when you have continuous variables in the environment like in Flappy Bird [9], [10].

Deep Q, Network, based agents have shown excellent performances in various game environments and have even been able to outperform humans in terms of consistency and long, term stability [7], [11].

However, training deep reinforcement learning models is a difficult task that inevitably leads to instabilities and divergence problems, and, therefore, these problems need to be carefully taken into account during the algorithm design phase.

6.3 Training Strategy and Policy Learning

Training of reinforcement learning agents, if done efficiently, is really about balancing exploration with exploitation. In the system set up by the authors, both Q, Learning and DQN agents run an epsilon, greedy policy, a method in which the agent takes a random action with probability \psilomelane otherwise picks the action with the highest estimated Q, value [12]. The training mechanism depicted in Fig. 5 is consistent with the usual DQN workflow, incorporating replay memory and the target network update at regular intervals.

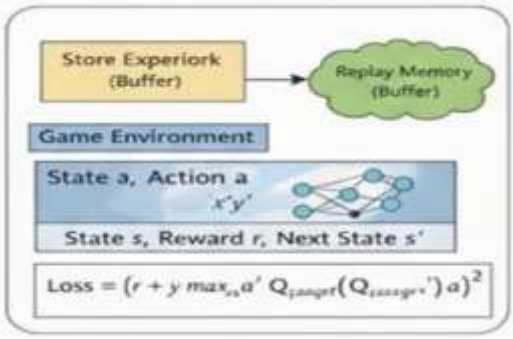

Figure 5: DQN Training Process

Generally, early training phases with a high exploration rate help the agent to experience diverse state, action pairs while the exploration rate is gradually suppressed in favor of learning exploitation. This technique has been proved to aid in convergence and to avoid early trapping at suboptimal behavior [13].

DQN training also makes use of a couple of additional stabilisation methods such as:

- Experience replay, i.e., storing the past transitions and sampling them randomly to decorrelate the samples in the time domain [7].
- Target networks, i.e., using a separate network to generate fixed Q, value targets that are regularly updated, thus providing the right targets completely [14].

Training episodes are multiples, each episode is a trophy of a game in a session. Some metrics of performance which are average reward, episode length, and score variance are checked in order to assess the learning progress and the optimization behavior [15], [16]. In order to make the training more stable, the framework uses experience replay and a target network that is updated periodically, depicted in Fig. 6

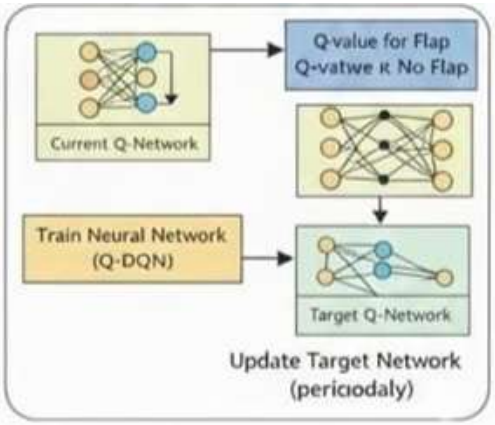

Figure 6: Experience Relay in Reinforcement Learning

6.4 Analysis of Q-Learning and DQN

While Q, Learning is simple and clear, cut, it has a big drawback of depending on discretized state spaces that limits its scalability and performance in changing environments. DQN, on the other hand, with the help of function approximation, can generalize and also achieves great learning stability and consistency in Flappy Bird gameplay [6], [9]. But, DQN comes with increased computational complexity and also demands careful tuning of hyperparameters like learning rate, network depth, and replay buffer size. This compromise emphasises the significance of learning algorithms selection according to the problem complexity and system constraints [17], [18].

7. Game Modes Implementation

7.1 Single Player Mode (Human-Controlled Gameplay)

The single, user mode is essentially the original core of the Flappy Bird game in which the bird's movement is totally dependent on the player's control input. This mode has two main functions in the designed system.[1], [2]

Firstly, it allows for a proper standard to be set against the evaluation of the learning, type artificial intelligence (AI) systems. Secondly, it acts as a human performance reference under the same game conditions, thereby helping to understand reaction patterns and performance variability.[1], [2]

Actions of the player in this mode are directly translated into the binary action ling used by the reinforcement learning agents, thus maintaining the methodological consistency across all the modes.

Every time the user inputs a command, the character will perform the "flap" action as well as the gravity, based falling if the player doesn't do any input. Such a combined control system allows comparison of humans and AI agents on equal terms without changing the original game mechanics.[3]

The performance of a single player in the game is judged mostly based on scores, the player survival time, and the steadiness of their clicking behavior. Earlier research works demonstrated that playing arcade games of a type that requires stopped time human responses is greatly impacted by start delay and human fatigue, which leads to large variation in one's gameplays[4], [5].

Such human shortcomings call for the application of automated agents that can perform at a consistently high level during long runs.

7.2 AI Mode (Machine Learning–Based Gameplay)

The AI mode makes it possible for a game to be played autonomously by reinforcement learning agents that have been trained within the proposed framework. This mode is running in two separate phases: the training phase and the inference phase, each one serving a different experimental purpose [6].

During the training phase, the agent communicates with the environment using an exploration, driven policy, usually epsilon, greedy. An agent selects actions either randomly or according to the current value estimates which allows the agent to discover various stateaction pairs and control patterns through reward feedback [7], [8]. Updates for learning are done online for Q, Learning agents or through mini, batch optimization for DQN agents.

In the inference phase, the agent is no longer exploring or is exploring to a very small extent, and the agent's decisions are purely based on the policy it has learned. The final performance of the agent is shown in this phase thus, it can be compared directly to human players under the same gameplay conditions [9].

AI conserves the decision, making process at each simulation step. Being the game engine deterministic and the state representation very compact, the latency of inference steadily remains negligible, ensuring smooth and responsive gameplay [10]. Learning curves are analysed by tracking performance metrics such as average reward and score improvement across episodes, offering a glimpse into the convergence behavior and policy stability [11].

7.3 Multiplayer Mode Implementation

The multiplayer mode extends the game framework to enable interaction among multiple players or agents in the same environment. It supports three multiplayer configurations:

Human vs Human: Under the Human vs Human mode, several human players either compete or coexist through synchronized game states. A network communication layer and synchronization accuracy are the main focus of this configuration whereas learning behavior is usually ignored [12].

Human vs AI: In Human vs AI mode, a human player fights against an AI, controlled agent. This arrangement is a very good tool for judging the perceived intelligence, quickness of response, and fairness of the learning agent. It has been demonstrated that the key to an enjoyable humanAI game interaction is the AI behaviour being at the same time predictable and adaptable [13], [14].

AI vs AI: When we put multiple AI agents against each other simultaneously they have a multi, agent autonomous interaction. Such a set, up allows one to look at how agents keep their traits, get tougher, and get used to changing situations caused by other agents which are also learning. Interaction among multi, agent adds more thread as the unfolding of the environment depends on which move other agents make and thus the traditional single, agent reinforcement learning assumptions are violated [15], [16].

Throughout all multiplayer variations, the game play logic stays the same and only prevails differences from where the decision actions come. This setting makes sure that the performance differences that can be observed are due to control strategies and not due to environmental variations.

8. Multiplayer Networking Model

8.1 Networking Architecture

In order to enable real, time multiplayer gameplay with a mix of human players and AI, controlled agents, the system described uses a clientserver networking architecture. This model is the most common one in online games because it allows the game to keep authoritative control over the game state and at the same time ensures fairness among players [1], [2].

Thus, the server is the entity that keeps track of the global game state including player (bird) positions, obstacle (pipe) layouts, scoring and terminal conditions (i.e., when the game ends) in the proposed method. The clients, whether they are human players or AI agents, do not change the game state directly. Rather, they send actions (e.g., flap or no flap) to the server at every

frame. This means that the server receives these actions, verifies and executes them and, finally, sends the new updated game state to all the connected clients [3].

The system architecture guards against state inconsistencies and significantly lowers the potential of cheating, while also making it possible to deterministically replay multiplayer sessions for evaluation and debugging purposes [4], [5]. Additionally, for AI agents, the server, side execution provides that the learning and inference happen with the same conditions as human players.

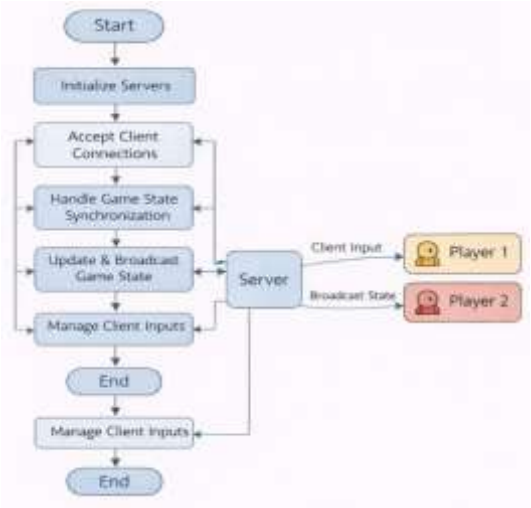

Figure 7: Client-Server Multiplayer Networking Model

Maintaining interactive gameplay requires instant communication between clients and the server.

In this system, small data packets are sent/messages exchanged at regular time intervals that correspond to the game's simulation step.

Each packet carries only necessary player actions and synchronization timestamps to minimize bandwidth and transmission delay[6], [7].

The server collects the player actions from different clients and runs the game logic for them one at a time within the same simulation frame, thereby creating the new game state. Then all players receive this update to ensure they have the same game experience[8].

Human players reach moderate average scores with a large variance. Q-learning agents can attain human, level or even slightly higher average scores with better consistency. DQN agents deliver better performances than both humans and Q, Learning agents in terms of average score, survival time, and performance stability. Poll time analysis shows that AI agents make decisions at fixed time intervals synchronized with the simulation step, thus, they do not have human input delay.

9. Result and Analysis

The learning effectiveness of reinforcement learning agents is gauged by observing how their scores, survival time, and rewards change over the course of training episodes. Initially, both Q, Learning and Deep Q, Network (DQN) agents show Q, Learning agent measures its learning comprehensively through advancement in score, survival duration and reward in the training sessions. Q, learning agent is initially quick in learning due to the fact that it has discretized states and explored all its possible actions of these states. However, after a certain number of episodes, the learning of the agent saturates. This results in the agent having an inability to generalize to the different states which are similar and to adapt to fine control variations due to state discretization [3], [4]. Conversely, the DQN agent exhibits a gradual and continual increase in performance level. The average score and survival duration become higher with the increase in training sessions and the variance diminishes in the later stages. Function approximation through a neural network allows the DQN agent to learn control patterns directly from continuous signals, which leads to stable flight trajectories and hence fewer failures [5], [6].

Concerning convergence, DQN agents were found to require more training episodes than traditional Q, Learning but significantly outperform the latter in terms of the asymptotic level of performance. This compromise between final achievement and training cost matches the earlier deep reinforcement learning works [7].

9.1 Human vs AI Performance Comparison

A direct side by side comparison between human players and trained reinforcement learning agents shows distinct differences in performance features. Typically, human players can perform well in short bursts, however, with high variability over different runs, which results from reaction delays, fatigue, and inconsistent timing [8]. Quantitative findings indicate that:

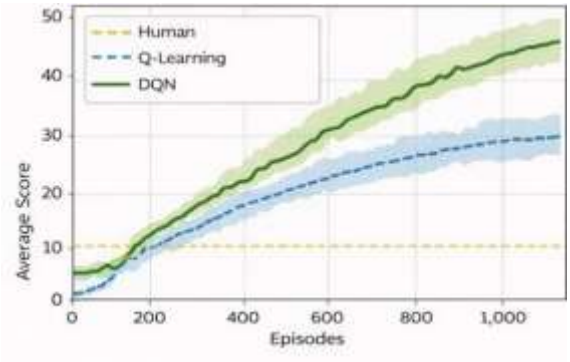

Figure 8: Performance Comparison of Human, Q-

Learning, and DQN Agent in Flappy Bird

Consequently, AI agents are able to perform perfectly timed moves, especially in very narrow gaps where the human error level seeps [9], [10].

These results are in line with what has been seen before, i.e., reinforcement learning agents have capabilities to outperform human players in situations where highly precise and repetitive controls are required even the action spaces are quite limited [5], [11].

9.2 Multiplayer Performance Evaluation

Multiplayer experiments shed light on the performance of learning, based agents when subjected to real, time networking constraints. In HumanAI scenarios, it has been found that AI agents can keep their performance level steady even with network, induced latency; however, human players tend to be more affected by delay and synchronization issues [12].

In AIAI multiplayer scenarios, trained DQN agents are reported to show strong and consistent behavior even when the environment becomes non, stationary due to the influence of other agents' actions. While one of the main assumptions of reinforcement learning algorithms is that they work in stationary environments, the studied agents have exhibited a level of policy robustness that is sufficient for them to be able to carry out their tasks stably [13], [14].

As regards the analyzing of the impact of latency, it has been concluded that the slightest delays affect AI agents to a very small extent if these agents determine when to act and the actions are executed on the server side. On the other hand, if the latency is too high, it can be a factor that leads to inferior performance not only from humans but also from AI agents. It is, therefore, very important to have synchronization and delay compensation mechanisms which are next discussed in Section 10 [15].

In a nutshell, multiplayer experiments have demonstrated that reinforcement learning agents that were trained in single, agent environments are capable of generalizing effectively to interactive multiplayer settings, provided that appropriate system, level constraints are in place.

10. Conclusion and Future Scope

This extensive study introduced a comprehensive Flappy Bird game framework based on machine learning, which combines single, player gameplay, autonomous AI control, and real, time multiplayer interaction in one system. Treating the game as a reinforcement learning task, the paper illustrated that agents are capable of learning good control policies simply by getting feedback from the environment, thus they do not have to resort to pre, programmed rules or domain, specific heuristics [1], [2].

Two popular Q, Learning and Deep Q, Network (DQN) techniques were carried out and tested under the same conditions. The test results indicated that although Q, Learning is a very simple and straightforward method, its effectiveness is constrained by the way the state is discretised and also by scale problems. Conversely, agents using DQN showed greater learning stability, increased average scores, and a major enhancement in consistency, especially when being controlled in time, sensitive and continuous environments [3], [4].

A key contribution of this study is the deployment of reinforcement learning agents in a real, time multiplayer environment. The communication model proposed, facilitated HumanHuman, HumanAI, and AIAI gameplay while still keeping synchronization, fairness, and state consistency. Results from multiplayer experiments showed that AI agents trained still be robust to network, induces latency and non, stationary interaction, thus emphasizing the possibility of releasing learning, based agents in interactive and distributed systems [5], [6].

In a nutshell, the results confirm that even simple 2D arcade games are capable of providing effective experimental platforms for reinforcement learning research, comparative AI evaluation, and humanAI interaction studies. The framework offered effectively closes the gap between single, agent learning benchmarks and real, world interactive systems, thus opening a viable route for the future research of game AI, multi, agent learning, and intelligent multiplayer environments.

Reference

[1] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[2] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming. Hoboken, NJ, USA: Wiley, 2019.

[3] V. Mnih et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, pp. 529–533, 2018.

[4] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in Proc. AAAI, 2019.

[5] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2019.

[6] A. Bellemare et al., “The arcade learning environment: An evaluation platform for

general agents," IEEE Trans. Games, vol. 12, no. 3, pp. 279–290, 2020.

[7] J. Schulman et al., "A survey of reinforcement learning algorithms for control," IEEE Robot. Autom. Mag., vol. 28, no. 2, pp. 62–78, 2021.

[8] L.-S. Pilcer et al., "Playing Flappy Bird with deep reinforcement learning," in Proc. IEEE Conf. Games, 2019.

[9] I. Saputra et al., "Optimizing Flappy Bird performance using deep reinforcement learning," in Proc. IEEE ETECOM, 2025.

[10] G. Yannakakis and J. Togelius, Artificial Intelligence and Games. Cham, Switzerland: Springer, 2018.

[11] J. Togelius et al., "Search-based procedural content generation: A taxonomy and survey," IEEE Trans. Comput. Intell. AI Games, vol. 10, no. 3, pp. 172–186, 2018.

[12] D. Perez-Liebana et al., "General video game AI: A multi-track framework for evaluating agents," IEEE Trans. Games, vol. 11, no. 3, pp. 195–207, 2019.

[13] R. Lopes and R. Bidarra, "Adaptivity challenges in games and simulations," IEEE Trans. Games, vol. 12, no. 1, pp. 1–15, 2020.

[14] S. M. Lucas and G. Kendall, "Evolutionary computation and games," IEEE Comput. Intell. Mag., vol. 13, no. 3, pp. 10–18, 2018.

[15] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, Distributed Systems: Concepts and Design, 6th ed. Pearson, 2021.

[16] S. Claypool and K. Claypool, "Latency and fairness in online multiplayer games," IEEE Internet Computing, vol. 24, no. 1, pp. 68–76, 2020.

[17] Y. Shoham and K. Leyton-Brown, Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge Univ. Press, 2020.

[18] L. Busoniu et al., Multi-Agent Reinforcement Learning. Springer, 2019.

[19] D. Silver et al., "A general reinforcement learning algorithm that masters complex decision-making," Science, vol. 362, no. 6419, pp. 1140–1144, 2018.

[20] OpenAI, "Spinning Up in Deep Reinforcement Learning," 2019–2024.

[21] S. Zander, G. Armitage, and P. Branch, "A survey of techniques for reducing latency in online games," IEEE Commun. Surveys Tuts., 2018.

[22] K. Chen et al., "Synchronization techniques for online multiplayer games," IEEE Trans. Games, 2020.

[23] L. Pantel and L. C. Wolf, "On the impact of delay on real-time multiplayer games," in Proc. ACM NetGames, 2018.

[24] S. Foerster et al., "Counterfactual multi-agent policy gradients," in Proc. AAAI, 2018.

[25] J. Schulman et al., "Proximal policy optimization algorithms," arXiv preprint, 2019.

[26] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in Proc. ICML, 2018.

[27] S. Thapar, G. K. Soni, H. Kaushik, R. Singh, S. Bisht, and S. K. Bansal, "A Comparative Machine Learning Framework for Detecting Fake Accounts on Facebook," in Proceedings of the 4th International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), pp. 1567–1571, 2025.

[28] A. Johari, R. Sharma, A. Meena, and V. Tiwari, "Advancements in Pre-Trained Language Models and Their Impact on Various NLP Tasks," International Journal of Engineering Trends and Applications, vol. 11, no. 3, pp. 201–209, 2024.

[29] K. Gautam, G. K. Soni, R. Ajmera, N. Hemrajani, J. Ahuja, and M. K. Jha, "Deep Reinforcement Learning for Stock Market Portfolio Optimization," in Proceedings of the 5th International Conference on Communication, Computing and Electronics Systems (ICCCES), pp. 1835–1839, 2026.

[30] S. Soni, M. K. Jha, and D. Jangid, "A Comprehensive Review of Blockchain and Machine Learning Convergence," International Journal of Global Research in Science and Technology, vol. 10, pp. 242–249, 2025.

[31] M. K. Jha, G. K. Soni, G. Jain, S. Tiwari, K. Gupta, and B. Singhal, "Comparative Analysis of Classical Machine Learning Models for Twitter Sentiment Classification," in Proceedings of the 5th International Conference on Communication, Computing and Electronics Systems (ICCCES), pp. 1949–1954, 2026.

[32] R. Ajmera, A. Johari, A. Goyal, A. Purohit, A. Kumar, and J. A. Ashok, "Multilingual Sentiment Analysis Based on Fine-Tuned Transformer Architectures," in Proceedings of the 5th International Conference on Communication, Computing and Electronics Systems (ICCCES), pp. 1589–1592, 2026.

[33] D. Saxena, J. Sharma, G. K. Soni, Y. Rao, S.

Sharma, and S. Lavania, "Sentimental Analysis and Forecasting using Machine Learning Algorithms," in Proceedings of the 4th International Conference on Automation, Computing and Renewable Systems (ICACRS), pp. 917–921, 2025.

[34] I. Yadav, V. Shekhawat, K. Gautam, G. K. Soni, and R. Yadav, "Artificial Intelligence for Cybersecurity: Emerging Techniques, Challenges, and Future Trends," in Proceedings of the 3rd International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), pp. 1176–1180, 2025.

[35] S. Soni, K. Paliwal, and G. K. Jain, "Reinforcement Learning in Autonomous Systems: Advancing Intelligent Decision-Making," International Journal of Global Research in Science and Technology, vol. 10, pp. 321–325, 2025.

[36] M. K. Jha, K. Kumar, N. Hemrajani, D. S. Rao, A. Goyal, and R. Ajmera, "AI Powered Student Performance Prediction using Explainable ML," in Proceedings of the 4th International Conference on Automation, Computing and Renewable Systems (ICACRS), pp. 1140–1144, 2025.

[37] V. Sharma and S. Soni, "Data Mining Techniques and Applications in Modern Information Systems," International Journal of Global Research in Science and Technology, vol. 9, pp. 277–281, 2024.

[38] K. Paliwal and S. Soni, "Artificial Intelligence in Healthcare: Transforming Modern Medical Systems," International Journal of Global Research in Science and Technology, vol. 9, pp. 194–197, 2024.