# **Introduction**

Reinforcement learning is a method of learning that determines the required actin for current state fetch from for environment to maximize the reward of AI agent. Algorithms that use this technique must determine behaviors based on previous experience to optimize the future reward. For this, a single move is less significant than a policy, which is the succession of correct activities that leads to a desired outcome. The purpose of reinforcement learning is to learn from previous successful action scenes to develop a policy [1]. Two factors are important for this type of algorithm: the experimentation process and the possibility of a delayed reward.

The Q-learning algorithm [2] can be used to do reinforcement training. Each action in the Q-learning system is rewarded with a prize. Activities that must be done receive rewards, while actions lead towards the failure receive penalties. Every action is stored in a table row.  The table keeps track of the current state of the environment (game), the decision made, the award obtained, and whether the game continues or not.  The data of Q-table is used to make further decisions. In the Q table the number of action and the numbers of states are increases or decreases direct proportionally. In multivariable situations, constructing the table takes a long time. There is a requirement to minimize the table's size. To tackle these issues, the upgraded version of Q-learning labelled as Deep Q-Network algorithm (DQN) has been emerged and with the advances of deep learning, Q-tables has replaced by the deep convolutional networks.

Unlike the major two types of machine learning techniques, supervised learning which required the labeling of data and unsupervised learning in which classify the data on later stage, reinforcement learning requires the definition of several attributes like the penalty and reward criteria, information of the penalty and reward from environment, possible action taken by the agent and information of state space. Literature put a lot of emphasis on the usability of neural network models in reinforcement learning while building algorithms because it looks to be an approach that can produce superior outcomes. The fundamental reason of using the NN with reinforcement learning is to automatically extract the features and adjust the networks weights [3].

In the last few decades, the advances of neural networks models including the layers of neural network made it possible to categorized the objects directly from raw data. It is also observed that the deep neural network showed the promising outcomes in computer vision problems such as classification of objects and detection of objects in a scene. Here is the goal to learn the game specific information base on the pixel information to decide the next most suitable action for maximum reward. By inspiring the study of [4] and [5], we used a reinforcement learning base algorithm for the learning of game and playing game. Here, the goal of this work is to create and test an AI-based optimal solution for the popular "Flappy Bird" game, which will result in a higher game score. We begin by creating an agent that learns how to play the game optimally by safely avoiding all obstacles and flapping its wings through them. The goal of this work is to create and test an AI-based optimal solution for the popular "Flappy Bird" game, which will result in a higher game score. We begin by creating an agent that learns how to play the game optimally by safely avoiding all obstacles and flapping its wings through them. A convolution neural network (CNN) base model is suggested to broaden the possible solution space and generalization capabilities. We conduct extensive tests to demonstrate the utility and efficiency of the proposed CNN algorithm. The quantifiable results suggest that proposed CNN can boost gaming scores dramatically.

In the rest of the report, the next section (Chapter 1) will discuss the detailed comparison of different literatures studies related to game scene learning using reinforcement learning technique. In Chapter 2, the novel approaches for the proposed study related to AI will be discuss. While in Chapter 3, the methodology, design and implementation of proposed study will be mention. Lastly, Chapter 4 will contain the critical evaluation results of proposed model.

# Literature Review

There are two techniques used for the training and play of Flappy Bird game that are Q-table base learning or model base learning. The authors of the article [6] employ a very similar strategy to play the Flappy Bird game that he used to play seven other Atari games [5]. The technique is tested using a Python-based game emulator [7]. The average score of the agent was very small and the maximum score of the agent was a few tens. The usage of deep reinforcement learning technique is an interesting idea to learn the learning and playing procedure of Flappy Bird game. It is also very interesting to compare the results of deep reinforcement learning algorithms with other algorithms. Another study also proposed the deep reinforcement learning algorithm to learn how to play Flappy Bird game. The state of the agent is defined by the raw pixel value of the screen. However, the author also described the three types of scores including the survival reward, passing pipe reward and hitting of pipe penalty [8].

In the proposed study of [9], author proposed the Q-learning algorithm. The study additionally added the information about the bird that is alive or not. But the authors did not used the rounded function to reduce the state space. A study proposed by the author of [10] showed the very presentable results for Flappy Bird game. The also added the velocity of bird as an extra feature to define the state space. The authors also used the SVM machine learning model. In the Ph. D. thesis [11], additional methods, such as heuristic algorithms, were compared. In the conclusion of the research, it's worth noting that, as a game with relatively basic rules, Flappy Bird can be an excellent domain for evaluating the efficacy and characteristics of various learning algorithms.

Mnih et al. [5] created the DQN algorithm in which agent is taught by using on unseen photos. The algorithm that was tested on Atari games generated results that were significantly superior to those of humans. In their work, Appi ah and Vare [6] used DQN to train Flappy Bird. They've discovered significantly superior human results than they had previously. Chen [9] used multiple iterations of the Q-Learning and DQN algorithms to train Flappy Bird in his study. An author [9] used Q-Learning versions to train the flappy bird game. Rosset et al. [12] used multi-agent Reinforcement Learning to train in 2016. In Signh's blog [13], he created an A3C-learned agent for game. In reinforcement training, making a comparison is difficult. Because every article may not be in the same setting. Open AI has created an environment that improves environments and compares them. Gym [14] was the name given to this setting. There are a variety of settings. The applications that have been written can simply be tested in a gym setting. Different AI-based studies for various challenges may also be found in [15]–[18].

# Novel Approach of Proposed work

The proposed worked is based on the training and playing of Flappy Bird game using Q-learning technique. To achieved this, we used the GitHub repository of Flappy Bird game that used the Reinforcement learning technique. The downloaded repo used the Q table technique for the learning of the agent (Bird). Each row of Q-table contained the current state of the bird, the action against this state, reward or penalty score against the decided action and the status of bird, is it alive or not. The data of Q-table is used to make further decisions. In the Q table the number of action and the numbers of states are increases or decreases direct proportionally. In multivariable situations, constructing the table takes a long time. There is a requirement to minimize the table's size. To tackle these issues, the upgraded version of Q-learning labelled as Deep Q-Network algorithm (DQN) has been emerged.

To tackle the problem of the minimizing the Q-table and achieved the maximum score, we proposed a Convolutional Neural Network (CNN). The purpose of the proposed CNN is to take the decision based on its output layer score. It will totally replace the Q-table that ultimately solve the problem of minimizing the Q-table. The proposed model will take the information of screen in raw pixel format and predict the possible action against the current state. The proposed CNN will update their weights based on the reward or penalty score getting from against the proposed action. Furthermore, the proposed CNN will also maximize the game score relative to the score by Q-table.

# Methodology

In this section, we will discuss the methodology of the proposed worked that contain training mechanism of Flappy Bird game. For the training of the AI agent of Flappy Bird game, firstly we used the emulator that is developed in python. A CNN model was proposed for the learning of bird from the environment. The architecture of the proposed CNN is shown in Figure #. The proposed model is based on the three hidden layers. Firstly, the CNN contain two convolutional layers followed by the 2 fully connected layers. Each convolutional layer is followed by the Batch Normalization, and max pooling layer. We used the 3x3 Kernel size for convolutional layers with zero padding and 2x2 kernel size for max pooling layer. The output is obtained from the last fully connected layer of the proposed CNN. The output of the fully connected layer is equal to the number of possible actions. As the Flappy Bird game have the two possible states that are Move-Up, Do-Nothing. Thus, the output layer of proposed CNN will predict the Score for both classes. The binary cross entropy loss function was used on the output layer of model. The purpose of the loss function is to calculate the loss between true action and predicted action and update the weights of the model accordingly. The gradient decent function was used to update the weights by Adam optimizer with the learning rate of 0.0001. The architecture of proposed model is shown in Figure 1.

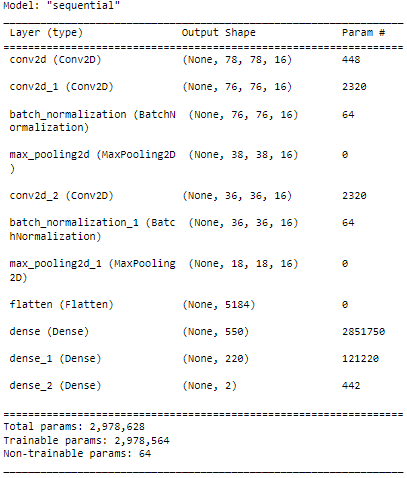


Figure 1: Architecture of proposed DQL (CNN)model.

After the development of the proposed model, the task of the AI agent is to capture the images from game emulator and give the image to CNN model to decide action from feasible actions after preprocessing. In the preprocessing, we downscale the image size to save the memory. The original size of the image in emulator was 284 x 512 while we resize this to 80 x 80. Additionally, the image was in colored form and we covert it into gray scale image. After the removing of noise from gray scale image, the image was ready to pass the CNN model.

The preprocessed image was passed to the CNN model for the decision of best suitable action to maximize the reward score from the feasible actions. It is like the classification problem to predict the category or label of the image. Bu it is unlike the classical classification problem because we don’t have the already labeled images. Instead of the labeled images, DQL model tries to evaluate the proposed action based on the reward score. On the basis of reward score, model will also improve its weights and after achieving the goal of maximum reward score, model will deploy to play game with AI agent. The complete pipeline of proposed study is shown in Figure 2.

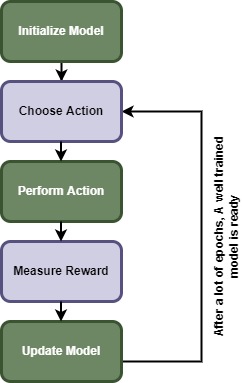


Figure 2: Complete pipe line of our proposed methodology.

# Design

The design section is based on two subsection that are designing of the model and the designing of the user interface. For the architecture design of the CNN model, the Numpy, TensorFlow, Keras and matplotlib libraries were used. The proposed model was also train in the virtual environment of python. The images dataset of the Flappy Bird game was used in proposed work. The images of the game were fetched on runtime during the training of the model.

The emulator of Flappy Bird was used from the GitHub repository that was developed in python using pygmy library. The emulator code was accessed from the given link of GitHub (<https://github.com/TimoWilken/flappy-bird-pygame.git>). The few design snapshots of the emulator are shown in Figure 3.

|  |  |  |
| --- | --- | --- |
| Flappy Bird Think Pieces | The Mysteries of Apps: Flappy Bird Shows That Dumb Luck Matters - Bloomberg |  |

Figure 3: Sample images of pygame emulator.

# Implementation

The implementation section of this report will discuss the complete implementation of model training, user interface development and integration of model with user interface. The architecture of the model was developed by using the python libraries as mentioned in methodology and result section. After the development of the model the model was required the training for the assistance of AI agent in playing game.

The training of the model was initialized with 1.5 million iterations with learning rate of 0.0001. The proposed work will take the image from game emulator and processed it according to the proposed methodology (methodology chapter). The images were fetched from the emulator with 10 frames per second. The fetched preprocessed images were feuded to the CNN model for predicting the suitable action from the two feasible actions to maximize the reward. The CNN will output the score for both actions and the final action will be taken on the basis of highest probability score. On the basis of final action predicted by the CNN model, the model will receive a reward or penalty score. If the bird successfully crosses the pipe, then it will receive the reward or if the bird hit the pipe, then it will receive ethe penalty. We set the reward score is +1 and penalty score is -1. On the basis of reward or penalty, loss function will calculate the loss and model will update its weights accordingly by using gradient descent function with learning rate of 0.0001. The snapshot of model code is shown in Figure 4.

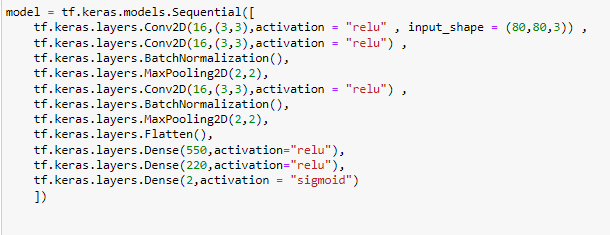


Figure 4: Python code of proposed DQL model.

During the training process, the DQL model showed the highest score of 396 on the iteration level of 17326. The complete reward score during each iteration of training process is shown in Figure #. As the main goal of the proposed work is to play game using AI agent, so completing the training process, model was deployed for the playing of Flappy Bird game. To test the model, we execute the model different time and find the reward score. We find the maximum reward score for the agent in testing environment is 362. The running snapshot of game and highest score is showed in Figure 5.

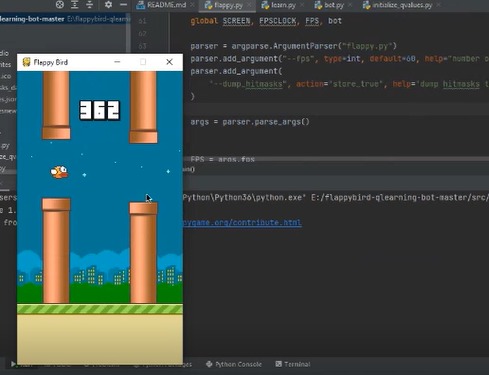


Figure 5: Maximum reward score of proposed model in testing phase.

# Critical Evaluation

In the evaluation section we perform different types of testing to evaluate our AI agent. For the evaluation of model, we did unit testing, integrated testing, black box testing and calculation of reward score.

**Unit Testing:** In unit testing we run our two main modules separately to test the proposed work. For this we run the emulator in human playing mode to test that the emulator is working perfectly. We also collect some images from the emulator and made prediction using our model. Model made the prediction of action on all the images without any error. As there is purpose to execute the modules and debug the errors so there is no need to authenticate the predicted actions.

**Integrated Testing:** After the complete training of the DQL model, the trained model was integrated with the emulator to give the decision about the current state. After the integration of the model, the emulator was run 20 time in AI agent model to play Flappy Bird game via AI agent. In every iteration agent successfully executed and play the game. In every iteration agent show different reward scores.

**Reward Score Calculation:** After the deployment of the model, we compile the results multiple time to compare them. We run the emulator with AI agent twenty time to play the game and AI agent showed the maximum 362 maximum reward score as shown in Figure #. The AI agent also showed the average 347 average reward score in twenty attempts of game. We also compile the results twenty times from our base repo to analyze the fair comparison. The base repo showed the 271 maximum score and 246 average score by using the Q-table technique. Lastly, we paly the Flappy bird game twenty times in human control model and got the 40 maximum score and 26 average score. The human control scores were compiled corresponding to beginner’s level. The comparison of all experiments is also shown in Table 1.

Table 1:Maximum and Average reward scores of different experiments.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Human | Q-Table | Proposed CNN |
| Maximum Score | 40 | 271 | 362 |
| Average Score | 26 | 246 | 347 |

# References

[1] “Introduction to Machine Learning, fourth edition - Ethem Alpaydin - Google Books.” https://books.google.com.pk/books?hl=en&lr=&id=tZnSDwAAQBAJ&oi=fnd&pg=PR7&dq=Ethem+Alpaydin.+Introduction+to+machine+learning.+MIT+press,+2014.&ots=F3WQ6Vdrwi&sig=S1hDlvI0e3aWVc\_hIeRPl-pwgZI#v=onepage&q=Ethem%20Alpaydin.%20Introduction%20to%20machine%20learning.%20MIT%20press%2C%202014.&f=false (accessed Apr. 28, 2022).

[2] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning 1992 8:3*, vol. 8, no. 3, pp. 279–292, May 1992, doi: 10.1007/BF00992698.

[3] Y. Li, “Deep Reinforcement Learning: An Overview,” Jan. 2017, Accessed: Apr. 28, 2022. [Online]. Available: http://arxiv.org/abs/1701.07274

[4] C. Clark and A. Storkey, “Teaching Deep Convolutional Neural Networks to Play Go,” *32nd International Conference on Machine Learning, ICML 2015*, vol. 3, pp. 1766–1774, Dec. 2014, doi: 10.48550/arxiv.1412.3409.

[5] V. Mnih *et al.*, “Playing Atari with Deep Reinforcement Learning,” Dec. 2013, doi: 10.48550/arxiv.1312.5602.

[6] N. Appiah and S. Vare, “Playing FlappyBird with Deep Reinforcement Learning”.

[7] “GitHub - TimoWilken/flappy-bird-pygame: A clone of Flappy Bird, using Pygame.” https://github.com/TimoWilken/flappy-bird-pygame (accessed Apr. 28, 2022).

[8] “[PDF] Deep Reinforcement Learning for Flappy Bird | Semantic Scholar.” https://www.semanticscholar.org/paper/Deep-Reinforcement-Learning-for-Flappy-Bird-Chen/b56c7703337cb9db008422b9b3410c97fff8bb54 (accessed Apr. 28, 2022).

[9] M. Ebeling-Rump, M. Kao, and Z. Hervieux-Moore, “Applying Q-Learning to Flappy Bird”.

[10] Y. Shu, L. Sun, M. Yan, and Z. Zhu, “Obstacles Avoidance with Machine Learning Control Methods in Flappy Birds Setting,” 2014.

[11] M. Piper, P. Bhounsule, K. Castillo, and A. Taha, “HOW TO BEAT FLAPPY BIRD: A MIXED-INTEGER MODEL PREDICTIVE CONTROL APPROACH,” 2017.

[12] “[PDF] Cooperative Multi-agent Reinforcement Learning for Flappy Bird\* | Semantic Scholar.” https://www.semanticscholar.org/paper/Cooperative-Multi-agent-Reinforcement-Learning-for-Rosset-Cevallos/f8ab9d6d1bf0bf45289046392da065fd818bed0c (accessed Apr. 28, 2022).

[13] “Deep Reinforcement Learning to play Flappy Bird using A3C algorithm – Shalabh Singh – I support Vector Machines.” https://shalabhsingh.github.io/Deep-RL-Flappy-Bird/ (accessed Apr. 28, 2022).

[14] G. Brockman *et al.*, “OpenAI Gym,” Jun. 2016, doi: 10.48550/arxiv.1606.01540.

[15] A. M. Karim, M. S. Güzel, M. R. Tolun, H. Kaya, and F. v. Çelebi, “A New Generalized Deep Learning Framework Combining Sparse Autoencoder and Taguchi Method for Novel Data Classification and Processing,” *Mathematical Problems in Engineering*, vol. 2018, 2018, doi: 10.1155/2018/3145947.

[16] A. M. Karim, M. S. Güzel, M. R. Tolun, H. Kaya, and F. v. Çelebi, “A new framework using deep auto-encoder and energy spectral density for medical waveform data classification and processing,” *Biocybernetics and Biomedical Engineering*, vol. Vol. 39, no. 1, pp. 148–159, Jan. 2019, doi: 10.1016/J.BBE.2018.11.004.

[17] H. O. Dalgic, E. Bostanci, and M. S. Guzel, “Genetic Algorithm Based Floor Planning System,” Apr. 2017, doi: 10.48550/arxiv.1704.06016.

[18] M. S. Güzel, M. Kara, and M. S. Beyazkılıç, “An adaptive framework for mobile robot navigation:,” *http://dx.doi.org/10.1177/1059712316685875*, vol. 25, no. 1, pp. 30–39, Jan. 2017, doi: 10.1177/1059712316685875.