International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-8, June 2020

Reinforcement Learning using Convolutional Neural Network for Game Prediction

Suvain Goyal, Vaibhav Somani, S.Sharanya

Abstract*— The paper presents a Deep learning model for playing computer games with elevated level information utilizing Reinforcement learning learning. The games are activity restricted (like snakes, catcher, air-bandit and so on.). The implementation is progressive in three parts. The first part deals with a simple neural network, the second one with Deep Q network and further to increase the accuracy and speed of the algorithm, the third part consists of a model consisting of convolution neural network for image processing and giving outputs from the fully connected layers so as to estimate the probability of an action being taken based on information extracted from inputs where we apply Q-learning to determine the best possible move. The results are further analysed and compared to provide an overview of the improvements in each methods.

Keywords-Deep Q Network, Convolutional Neural Networks , Q-Learning

I. INTRODUCTION

Solving games is considered interesting and also the basis of problem solving structure because of the hostile, competitive challenges and agents involving the setting. Thus, developing a game AI opens the trail of implementing the problem-solving ideas in reality of the world. Besides that, deep reinforcement learning has achieved many big-profile successes in tough decision-making problems. But these algorithms generally need a large data-set before they give acceptable result [1]. In reinforcement learning (RL) a straight forwards answer does not exist, however your support learning operator despite everything needs to choose the manner in which it needs to act to play out an undertaking. In the absence of existing preparing information, the specialist gets information and gains as a matter of fact. The training data is collected (this activity was ok, that activity was terrible) via experimentation as it attempts to carry out its responsibility. It's goal is to maximize the long-term reward. Thus, the implementation given within this paper involves coaching a snake bot in the game to make decisions at every step to collect as much reward as possible.In snakes game the player tries to avoid the walls and realize the best path for the reward (an apple). A bot is made to understand and play the game but by not telling it about the environment or the rules. Like what the wall is or apple is. The algorithm just gives them the feedback of what it is doing. This is the way for solving the self-learning feature, using the deep Q learning [2].

Revised Manuscript Received on June 30, 2020.
***** Correspondence Author
Suvain Goyal*, Dept. of CSE SRM KTR Chennai, India
Vaibhav Somani, Dept. of CSE SRM KTR Chennai, India
Mrs S.Sharanya, Dept. of CSE SRM KTR Chennai, India

One challenge is to form that huge dataset before the learning process actually began. To solve that issue, several approaches can be used like watching the replay of professional players for that game or feed in some random information or someone can play for actual and record the actions taken for each instance of gameplay. However, all of these are tedious as well as require past information which is being avoided because of the feature i.e. self-learning.

Another challenge is to make a learning model for games, where each game is extraordinary, the situations are unique and so are the actions. Thus, there is no such fixed model for fitting all the games.

Finally, designing the reward strategy is quite challenging. Getting the rewards for each game and choosing the appropriate rewards can have a large influence on the game play of the bot. The objective of the agent is to make its total (future) reward maximum. It does this by adding the reward for achieving its current state to the maximum reward attainable from future states, effectively impacting the present action by the potential later reward. This potential reward is a weighted aggregate of the expected values of the rewards of all future steps beginning from the current state.

II. LITERATURE REVIEW

A. Deep Reinforcement Learning

A Reinforcement Learning (RL) specialist teaches itself via experimentation ,interactions with a dynamic domain helps it balance the reward trade-off between short-term and long-term planning. RL techniques have been broadly studied in multiple disciplines, such as operational studies, simulation-based optimization, evolutionary computation and multi-agent system, including games. The cooperation between the RL methods and Deep Learning (DL) has led to fruitful applications in games [3] [4]. Training an agent to beat human players, and to adjust its score, can teach us how to streamline various processes in a wide range of different and exciting subfields. Its what Googles DeepMind did with its popular AlphaGo, beating the strongest Go player in the world and scoring a goal considered impossible at the time [5]. Reinforcement Learning is an methodology based on Markov Decision Process to make decisions. Traditional ML algorithms need to be trained with an input and a correct answer called target. The framework shall then try to teach itself how to predict the target according to a new input. In this example, the best move to take at each state of the game isn't known, so a traditional approach would not be viable.
In Reinforcement Learning a reward is acceptable, positive or negative reliant on the move the system took,

and the network needs to learn what actions can make the reward maximum, and which actions will impact negatively. To understand how the agent takes decisions is important and also to know what a Q-Table is. A Q-table is a matrix, which correlates the state of the agent with the possible actions that the system can adopt. The values in the table are the actions probability of success, based on the rewards it got during the training.Deep Q-Learning increases the potentiality of Q-Learning, continuously updating the Q-table based on the prediction of future states

B. Deep Q Network

To copy the impacts of responsive fields the Deep-Mind framework utilized a profound convolutional neural network in which layers have convolutional filter . A non-linear approximate of a function can easily make Reinforcement Learning, for example, a neural system is utilized to speak to Q table. This flimsiness originates from the relationships present in the arrangement of perceptions, the fact that little changes to Q-table can maybe fundamentally change the strategy and information dispersion, and the connections among Q and the objective values. The procedure utilized experience replay, an organically motivated system that utilizes an irregular example of earlier activities rather than the latest activity to continue. This evacuates connections in the perception grouping and helps in smoothing changes in the data dissemination. Iterative update modifies Q-tables towards target values that are just intermittently refreshed, further diminishing connections with the objective.

C. Convolution Neural Network

A type of deep artificial neural networks that are utilized fundamentally to classify pictures (for example name things they see), group them by closeness (photograph finding), and object recognition with the help of Convolutional Neural Networks. These are calculations that can recognize people, faces-features, cancer tumors, road signs, and numerous different parts of visual information. Optical-Character-Recognition (OCR) made possible with CNNs to make text available in digital format and make NLP conceivable on analogous and manually ascribed records, where the pictures are images to be translated. A spectrogram which is nothing but an visual representation of an audio can used by CNNs to process autido and are able to create models with multiple functionality. Text Analysis and analysis of graph data are made possible by GCN which is graph type of CNN . [7]. The world and the Science community where only made aware of the absolute potential of CNNs after they showed feats in image recognition . They are controlling significant breakthroughs in PC vision (CV), which has clear uses for mechanical technology, rambles, autonomous vehicles, security, clinical judgments, and medications for the outwardly weakened.

S no	Objective	Algorithm	Datasets	Relevance	Performance Measured
1	To create a deep learning model with level of input	Open AI GVGAI	Different algorithms to achieve the result and	Measure the relative difficulty of games	Max score achieved
			different games		
2	Introduction to convoluted neural networks	Machine Learning, Pattern recognition, Image Analysis	Database of Handwritten digits	Better understanding of CNN.	1. Accuracy
3	To use TensorFlow to make a machine learning System	TensorFlow	Predefined Datasets	Usefulness of Tensor Flow.	1. Accuracy 2. Sensitivity 3. Precision
4	To make a simple and lightweight framework for Deep RL	N-step Q Learning	Arcade Enviornme--nt Lab	Better method for implementing Reinforced Learning	
5	To introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks.	Monte Carlo tree search	Initial positions of checkers.	Advancement in Monte Carlo Tree search, thus achieving 99.8% win rate.	Player with the greatest score wins
6	To outline a sound event classification framework that compares auditory image front end	Support Vector Machine. DNN classifiers	Noise free sample;High Noise Condition	To build a robust sound classification system	SVM and DNN classifiers
7	To directly test agents against each other in more complexand dynamic environments.	Open Loop Monte Carlo Search	Competitive and cooprative real time stochastic two player games	Expansion of the GVGAI competition to a two player track.	Comparison with currently available sample controllers in the framework.

III. PROPOSED METHODOLOGY

Several games couldn't be tried with the different methodology proposed because of less GPU power available. Because of this reason, a 2D snake game was picked Snake game is a straightforward game where the player controls the snake and endeavors to to make the snake grow to the extent that is possible by eating an apple that spawns randomly around the play space. The snake dies in 3 instances, if it hits any obstacles, if it hits any boundaries or even if it hits its own body. This simplified the whole network structure and the entire algorithm for improvement clarification and correlation.

A. Simple Neural Network

For the first method, a simple neural network is proposed with 2 parts in mind. The first part focuses only on the survival of the snake i.e., it avoids the boundary of the environment and the second part focuses on the reward system by eating the apple.

The first part consists of 4 inputs given in an array of 4 numbers:

Is the movement of snake blocked from left (1,0) (yes, no)
Is the movement of snake blocked from front (1,0) (yes, no)
Is the movement of snake blocked from right (1,0) (yes, no)
Calculated Suggestion to move (-1,0,1) (left, straight, right)

If the output =1 Go to the selected direction, if 0 Choose another direction. Since the input and the output are simply in binary, linear activation function is used. The result of the first part will cause the snake to travel in circles as it only tried to avoid the rectangular boundary.

For the second part, a new feature is added in the network with one extra input:

Normalized angle through location of the reward of an apple and snake's movement (from 1 to -1)

With the new system, the snake now focuses on if the turn is effective to near the distance to the apple or not instead of just endurance. The new output will be:

-1 : Death of snake
0 : Wrong direction but the snake survives
1 : snake survived and the direction is right

With this technique, the snake can effectively eat an apple on its own. This method is quite basic and isn't relevant for dynamic game condition. It likewise disregards the fact of finding the snake the shortest path to get the apple. Thus the network also lacks AI optimization.

The network architecture for this implementation is as follows:

B. Q Learning

In the second methodology Q Learning has been used, which is a part of Reinforcement Learning.

The following hyperparameter are as follows:

Learning Rate: At what pace the AI will learn to play (close to 0 will be slow, while close to 1 will cause it to overshoot). Preferably we try and keep a lower learning rate
Discount Factor: How much importance is given to future rewards and immediate rewards
Randomness: Number of times a random action will be chosen rather than the effective one. This is done to prevent overfitting

We use a Markov Decision Process to manage and make decisions. MDP constantly updates the Q-table which is nothing but a 2D Data Structure that can store different states and the chance of the next actions happening. Every time the agent performs an action the table is updated. The action with the largest Q-value is deemed the most efficient action to perform next.

The functions are:

Function	Explanation
s = game.state()	Get state s
	Execute best action act given state
act = best action(Q(s))	s
rew = game.reward( )	Receive reward rew
s' = game.state()	Get next state s'
Q(s, act) = update-qTable( )	Update Q-Table value q of state s and action act

1) Actions We can choose from a set of actions which are Left, Right, Up, Down. These simulate user inputs to the snake. The action whose value in the Q-table is determined to

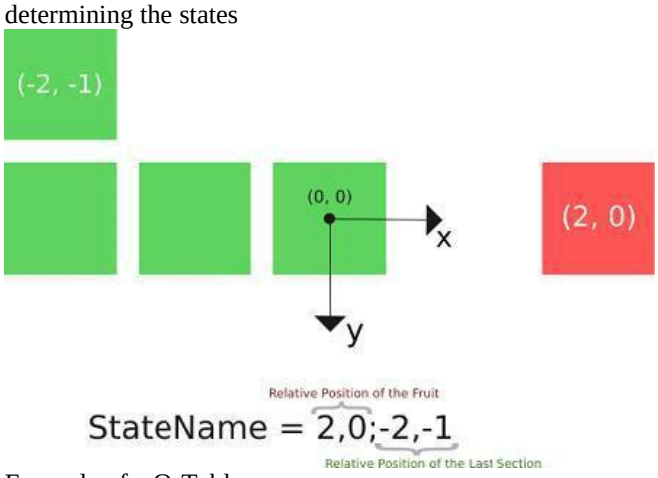
↑	↓	←	→
UP	DOWN	LEFT	RIGHT

Be max is chosen to go into the next state. But a random action is done if the zero is the maximum value;

Reward: When the snake reaches the food a reward is given (+1). Also +0.1 when it was able to explore the environment without dying. The penalty is given when the game starts-over (-1), that is, when the snake hits a wall or its own snake body. A negative reward(-0.1) is awarded for any other scenario. This is the best way for the path minimization which is chosen to reach the reward and makes the process of training quicker.

Action	Reward
Gotten the rewrds	+1
Dies by hitting its own body	-1
Dies by hitting wall	-1
Other	-0.1

3) States: The positons of the snakes head to the position of the reward and the position of its own tail helps in determining the states



Example of a Q-Table :

```
{ '2,0,-3,0':   { up:0.43,  down:0.25, left:0.12, right:0.8   } },
{ '-2,0,3,0':   { up:-0.5,  down:-0.3, left:0.38, right:-0.15 } },
{ '0,-1,-3,-1': { up:0.6,   down:0.18, left:0,    right:0.53  } },
{ '0,1,3,1':    { up:-0.32, down:0.75, left:0.23, right:0.49  } }
```

Fig. 1. Example of States and Q values for each action

The Q-Table is updated by the following equation:

$$Q^{\pi}(s_t, a_t) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ..., s_t, a_t]$$

Q-Values for the state given a particular state

Expected discounted cumulative reward

Given the state and action

It is executed after the action is taken and the reward is known.

C. CNN with Deep Q Learning

Not many years back, a little organization in London called Deep-Mind transferred their pioneering paper [8].

The same model design, with no change, was utilized to learn seven unique games, and in three of them the calculation performed stunningly better than a human.

The third segment of the paper involves implementation of similar technique, CNN alongside the Deep-Q Network. References were taken from [9] [10].

To train the neural network to learn the snake game, input to the network will be an instance of the play space, and output

would be three activities: straight, right and left. It would bode well to regard it as a classification problem as for every instance the algorithm has to determine, whether it should move straight left or right. But for this a large number of training examples are required. Games played by humans can be used as examples, but it is a not a feasible approach here where multiple players have to make millions of conceivable moves. Only a periodic feedback of a move is required and the rest can made sense of. This is the problem to which reinforcement learning tries to provide a solution to. In reinforcement learning, one has sparse and time-delayed labels for the rewards. Only with these rewards the agent needs to figure out how to carry on in the environment.

The architecture of the Network is as follows representing the activations and filters used at each hidden layer.

Layer	Input	Filter size	Stride	Num filters	Activation	Output
conv1	84x84x4	8x8	4	32	ReLU	20x20x32
conv2	20x20x32	4x4	2	64	ReLU	9x9x64
conv3	9x9x64	3x3	1	64	ReLU	7x7x64
fc4	7x7x64			512	ReLU	512
fc5	512			18	Linear	18

Fig. 2. Network Architecture

Here we have a generic CNN with 3 conv1d layers and then 2 connected layers each with ReLU Activation. Without any pooling layers. Pooling layers have translation invariance because of which the network shows insensitivity towards the objects position in an image. Thus, such information can be easily avoided. Input to the network are four 84x84 black-white screen of an instance of the game. The network outputs for every possible action's Q-values [Fig.3]. Simple squared loss can be used for optimization of the regression task because the Q-values can be equivalent to any real values

$$L = \frac{1}{2}[\underbrace{r + max_{a'}Q(s', a')}_{\text{target}} - \underbrace{Q(s, a)}_{\text{prediction}}]^2$$

Fig. 3. Squared Error Loss Function

For every transition we replace the Qtable's update rule with these steps:

1) Fetch the Q-values that have been predicted for all possible actions with a feed-forward pass.
2) Compute the overall maximum of all the network-outputs using function max(a,Q(s, a)).
3) Target the action's Q-value equal to sum(r,max (a,Q(s, a))) (we make use of the computed max from the previous step). Set the values for all the other actions equal to initial values set in the Q-table in Step 1) this leads to a 0% error rate.
4) Weight Updation via backpropagation.

At this point we have a how to assess the compromise of individual scenarios utilizing Q-learning and estimated the Q-function using a convolutional neural network. But it turns out that approximation of Q-values using non-linear functions is not exactly stable. There are multiple methods you have to use to actually make it converge. And it takes a large amount of compute, almost 7 days on only one Graphical Processor. The method with the most significance is experience replay. During each playthrough many of the interactions and encounters are put away in a replay memory. When the network is being trained, arbitrary small batches retrieved from storage are utilized rather than the most recent transition. This prevents over-fitting and due the next set of training samples being identical, that can cause the network to descend to a local minima.

Also as usual supervised learning is comparable to using the replays of old experiences, the task of debugging and testing the algorithm are a much simpler process. One could in reality gather every one of those encounters from manual playthroughs and afterward train the network on these. The credit allotment problem is attempted to be solved using Q-learning, it considers the older rewards, until it is able to reach the vital choice that may be the real reason for the obtained reward. In any case, our work does not go into exploration-exploitation dilemma. It is to be noticed that whenever the Q-table is filled with arbitrary values, then the predictions are at first also arbitrary. After choosing a move having the largest value, the action chosen shall remain random and the bot goes for rough exploration. As the function moves towards a single point, it returns more and more steady Q-values while the need of more of space-exploration becomes less. So, basically, we can say that , space-exploring is added as a part of the algorithm via Q-learning . But this type is greedy, it goes for the most compelling method it can find. A possible way to rectify this issue is by using randomness in our greedy exploration, if not feasible we just choose the greedy move having the largest value. In their framework DeepMind really diminishes after some time from 1 to 0.1, initially the framework chooses wholly arbitrary actions to allow maximum space exploration, and afterwards the frameworks reaches at a steady rate of exploration.

This gives the final algorithm the experience replay concept

IV. EXPERIMENTAL RESULTS

The paper presents the results of training the baselines on the Snake Game. This segment is organized in three sections. First, the training of the snake AI utilizing simple neural networks. Second, the Deep Q Network is used on the Snake Game Source Code to assist it with figuring out how to move in the environment (using Actions and States). Third, the reinforcement agent is trained using Convolutional Neural Network (CNN) applied along with Deep Q Learning.

In Fig.4, the whole Q table gets updated when any one of the action is performed by the snake bot, whereas in Fig. 5, a

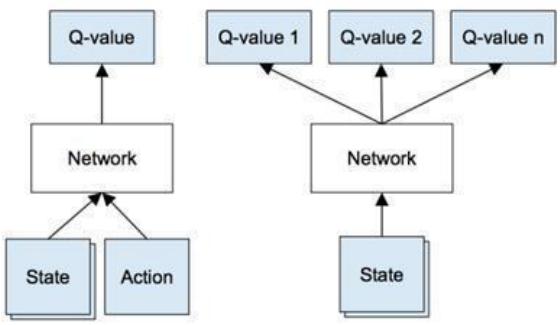


Fig. 4. Fig. 5.

Figure 4. represents the result of training the Snake Game AI using Deep Q Network.

Figure 5 represents the result of training the Snake Game AI when CNN is applied with Deep Q Network

separate Q table is created for each of the actions that can be taken by the bot. For a particular action taken, only one Q table corresponding to that particular action gets updated. Hence, the computation time is reduced in Fig. 5, resulting in high scoring by the Snake AI Bot in this second case. The average score increases from 2-3 occupied from Fig.1 result to 5-6 obtained from the result of Fig. 5.

After calculating the average steps and points taken by the snake with respect to each algorithm, the results are as follows:

1. For Simple Neural Network:
   These are the results achieved after 10000 test games are played:
   – Avg. No. of steps- 167.61
   – Avg. No. of points- 12.142
   These are the results achieved after 100000 test games are played:
   – Avg. No of steps- 398.959
   – Avg. No of points- 25.333
2. For Deep Q Network:
   Average score obtained by applying Deep Q Network Between 2-3
3. For CNN implemented along with Deep Q Network: Average score obtained by applying CNN along with Deep Q Network Between 5-6

V. CONCLUSION

In this research paper, three networks were implemented from simple to complex CNN Deep Q Network. The main reason for this paper was to show the efficiency of teaching the agent/bot in a dynamic environment using Deep Learning. The results so far were of good outcome. However, since this algorithm was used only in one specific type of game, it might show unexpected results for different types of games. But however unexpected it might be, the base method will still be the same and may required only some changes in activation functions, normalising technique and optimizers.

Other implementation for video game AI using this model is to help the game testers and designers in creating a level suitable for the players. AI using Deep Learning is not only useful for games but also for Robotics. A lot of research and work has been dedicated in this field [11]. There is so many opportunities and scope in this matter and Deep Learning is believed to be the gate opener for an ideal start.

REFERENCES

1. 1.Ruben Rodriguez Torrado and Philip Bontrager,"Deep Mrs.S.Sharanya, Assistant Reinforcement Learning for general video game AI" Professor, Department of Comp- (6-jun-2018)
   -uter Science, Kattankulathur, SRM Institute of Science and
2. An Introduction to Convoluted Neural Networks. Keiran O'shea Technology and Ryan Nash.(2015)

3. TensorFlow: A System for Large-Scale Machine Learning by Martín Abadi, Paul Barham (2016)Asynchronous Methods for Deep ReinforcementVolodymyr Mnih1 ,Adrià Puigdomènech
4. Mastering the game of Go with deep neural networks and tree search. David Silver and Aja Huang(2016)
5. Robust Sound Event Classification using Deep Neural Networks. Ian McLoughlin, Haomin Zhang, Zhipeng Xie, Yan Song, and Wei Xiao (2015)
6. General Video Game for 2 Players: Framework and Competition. Raluca D. Gaina, Diego Perez-Li ´ ebana, Simon M. Lucas(2016)
7. Wikipedia.*AlphaGo Zero*[Online]. Available from: https://en.wikipedia.org/wiki/AlphaGoZero.
8. Yen-Chen Lin . *Using deep Q-Network to Learn How To Play Flappy Bird*. GitHub Dev blog.
9. Tambet Matiisen . *Demystifying Deep Reinforcement Learning*. Intel blog
10. Tuomas Haarnoja, Student Researcher and Sergey Levine, Faculty Advisor, *Robotics at Google. Google AI. Soft Actor-Critic: Deep Reinforcement Learning for Robotics*.

AUTHORS PROFILE

Suvain Goyal, Bachelors of Technology Specializing in Computer Science from SRM Institute of Science and Technology , Kattankulathur campus

Vaibhav Somani, Bachelors of Technology Specializing in Computer Science from SRM Institute of Science and Technology , Kattankulathur campus

Mrs.S.Sharanya, Assistant Professor, Department of Comp- -uter Science, Kattankulathur, SRM Institute of Science and Technology