Deep Learning on Atari Games using Deep Q-Learning Algorithm

Mukul Kadel, Dr. Rekha Jain

Student, Professor

Computer Engineering,

Poornima Institute of Engineering & Technology, Jaipur, India

Abstract: Deep reinforcement learning is giving us state-of-the-art algorithms and one of such algorithms from this field is deep q-learning. It uses CNN's and works directly on raw input images. This algorithm was applied on Atari games and evaluated under different environments.

Index Terms **- Reinforcement learning, Deep Q-Learning, Machine learning, Artificial Intelligence**

I. INTRODUCTION

Reinforcement learning is an emerging subfield of machine learning in which algorithms learn by using trail-and-error techniques. In these algorithms, there's an agent which interacts with the environment, observes the result of these interactions and receives a reward based on these interactions. These rewards are later used to evaluate which actions are good or bad under particular environment conditions. This action and reward-based learning process comes from observing human learning process.

Humans learn by observing the environment through their senses and over time become expert on how to use them properly. Unlike traditional RL algorithms where the input environment is highly preprocessed and features are precisely handcrafted, in humans, these observations are done directly on high-dimensional sensory inputs like vision and language. This means there's still a lot of research needed to be done in the area of reinforcement learning.

Recent advancements in deep learning where neural networks are used as a function approximators, has also opened a new window in the field of reinforcement learning. By combining deep learning with RL, a new algorithm known as Deep Q-Learning was developed. It takes a sequence of raw images as an input for the environment, just like humans, and outputs Q-values for each action and action with the highest Q-value is selected. Convolutional neural networks are used to understand complex environments. Each sequence and actions are stored in storage known as experience relay with their future rewards. Rewards may be obtained instantly or after some time. To train the DQN agent, stochastic gradient descended algorithm is used. Random samples are taken from the experience relay and batch trained on the agent.

Further studies are required for this algorithm in order to identify the scenarios where it works well or where it works badly. We'll apply this algorithm on Atari games using OpenAI Gym simulator in python language in order to test it. It'll give us raw frames, rewards, scores, etc. as inputs and also whether the game is over or not.

II. BACKGROUND

The field of machine learning is a subset of artificial intelligence, which gives the computer systems ability to automatically learn a specific task without any human explicitly defining it for them. Computer systems create a model from machine learning algorithms and train them on some data. This data is known as training data. It can be labeled or unlabeled, or structured or unstructured. During the training process, the system finds patterns in the dataset and uses these patterns during prediction tasks.

Formally machine learning algorithms can be stated as "A computer program is said to learn from experience ***E*** with respect to some class of tasks ***T*** and performance measure ***P*** if its performance at task in ***T***, as measured by ***P***, improves with experience ***E***". This field of machine learning is further classified into 3 categories:

1. Supervised learning: In supervised learning, models are created on the data which has both inputs as well as corresponding output labels. These models can be considered as mapping functions. They learn how to map a particular set of input values to an output label. These models are fast to train but require the extra overhead of labeling the dataset. These algorithms are used for classification and regression tasks.
2. Unsupervised learning: In unsupervised learning, models are created using data which only has input part and no output labels. This type of learning is used to discover hidden patterns in the dataset. It, as a result, helps in the clustering of data, and in reducing the number of features in the input dataset.
3. Reinforcement learning: In reinforcement learning, models are known as agents which interact with the environment and take feedback from it. On positive feedback, agents will repeat those actions and on negative feedback, agents will avoid those actions. Our main focus will be on reinforcement learning in this paper.

III. RELATED WORK

A variety of work has been done in the area of deep reinforcement learning after the first publication of Deep Q-Learning algorithm [2]. This algorithm opened a new window of opportunity to make computer systems smarter. Many researchers have either proposed an improved version of this algorithm or their own implementation of deep reinforcement learning.

DQN algorithm sometimes during training tend to overfit on a certain sequence of states. An effort was made by the same team to reduce it and, a new double Q-learning algorithm [3] was made. In this algorithm, two different Q-networks were used.

Some researchers have also added a recurrent neural network in the algorithm [4] making it better in perceiving partial environments.

Furthermore, it was also shown that some states are more important than others during the training process. To maintain their importance, prioritized experience relay was created [5]. It increased the training efficiency of the algorithm.

Also, a new architecture was proposed in "Dueling Network Architecture" [6] paper, which resulted in faster executions.

IV. REINFORCEMENT LEARNING

In reinforcement learning, every algorithm has an agent, a set of states **S**, and a set **A** of actions per state. Agent transitions from one state to another by performing an action $a \in A$. Each action gives a reward to the agent. This reward can be positive, negative or neutral and based on these rewards, the agent will optimize its future actions. The main goal of the agent in reinforcement learning is to maximize the total reward. Negative reward is termed as a bad action and positive as a good action. In our experiments, the environment for the agents will be the Atari games, states will be the sequence of frames, the reward will be whether the current move resulted in score or death. In RL, we have a trade-off between whether to explore or exploit the environment i.e. whether to randomly choose the actions or choose them based on the current best policy.

One of the good algorithms in RL is Q-learning. For any finite Markov decision process (MDP), Q-learning finds an optimal policy such that it maximizes the expected value of the total reward over any and all future steps, starting from the current step. "Q" can be considered as a quality of the action taken on a given state.

$$Q : S \times A \rightarrow \mathbb{R} \qquad 1$$

Before learning begins, Q is initialized to a randomly selected value. Then, at each time t, the agent performs an action a_t either randomly or based on the policy, observes a reward r_t, enters a new state s_{t+1}, and Q is updated.

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}} \qquad 2$$

where α the learning rate ($0 < \alpha \leq 1$) and γ is a discount factor. If discount factor γ is 0, the agent will only look at the current step's reward, whereas, if γ if 1, the agent will endeavor for a long-term high reward.

A Q function is considered to be an optimal action-value function $Q^*(s', a')$ if it gives maximum expected return by following any approach. This function follows an identity known as Bellman equation.

Adding deep learning to the Q-learning algorithm created a new algorithm known as deep q-learning, which is the main focus of our research in this paper.

V. DEEP REINFORCEMENT LEARNING

Neural networks caused a breakthrough in the field of computer vision. Adding NN with reinforcement learning resulted in AlphaGo system which beat the world champion, Lee Sedol in the Go game. This accomplishment was earlier considered to be impossible and would have taken years as per expert's opinion until AlphaGo proved them wrong. This belief was there because it's impossible to brute force all the combination of Go game's board configuration.

Deep Q-learning algorithm trains directly on raw inputs, using small updates based on stochastic gradient descent. With adequate training, it works better than handcrafted feature algorithms as it itself identifies all the important features in the training data. It stores training data in a buffer known as experience replay. In experience replay, at each row e_t we have (s_t, a_t, r_t, s_{t+1}) where s_t is current state, a_t is current action, r_t is reward received after performing action a_t and s_{t+1} is the next state. The size of experience replay is taken to be one million in our experiments and it works as a queue.

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$
  end for
end for

Neural network Q is initialized with random weights. At each episode, a sequence of frames is taken. This sequence is converted into a grayscale resized image and passed through the network, to find Q-values of each action. Selected action is $a_t = max_a Q^*(\phi(s_t), a; \theta)$ i.e. action with maximum Q-value. This action is executed on emulator and new frame and reward is recorded. This experience is added to the buffer and the network is trained from randomly selected minibatch of transitions.

VI. EXPERIMENTAL RESULTS

Until now experiments are performed on 5 Atari games: Assault, Breakout, Crazy Climber, Space Invaders and Pinball. Each game has a different dynamic and requires different strategies to win or score high in the game. For all the games, we've used the same architecture and parameters. All the rewards are normalized as 1, -1 or 0. If the score is increased or the game is won, the reward will be 1. If health is lost or the game is over, the reward will be -1 and if nothing happens, the reward is 0. Size of each minibatch is 32 and experience replay buffer size is 1 million. The rate of exploration and exploitation is controlled by parameter ε. It decants from 1 to 0.1 till 1 million and remains constant at 0.1 thereafter.

Each game is played in 3 different ways during the evaluation process: Random, Human and DQN. In random plays, actions were selected at random and scores of all random plays were averaged. In human plays, 4-5 different humans have played the games over many times and their scores were averaged. In DQN plays, the trained algorithm played on the emulator and its scores were averaged.

During playtime of deep q-learning algorithm, ε was fixed at 0.05 for each step.

Table 1 Results of the experiments

GAMES	RANDOM	HUMAN	DQN
Assault	201.4	841.1	3580.4
Breakout	2.6	22.5	285.5
Crazy Climber	8465.5	46484.1	103763.1
Space Invaders	138.0	1668.7	1578.1
Pinball	13256.2	18767.9	199452.1

VII. CONCLUSION

In this paper, Deep Q-learning algorithm has been further evaluated on various Atari games and it's tested on different environments. This variety in different environments also tested its capabilities and limits. Experiments have also verified the earlier results that were published with the algorithm.

In the future, the comparison between this and other new algorithms will give more insight into deep learning architectures.

REFERENCES

[1] An introduction to deep reinforcement learning. arXiv:1811.12560v2, 2018.
[2] Hausknecht, M., and Stone, 2015. Deep Recurrent Q-learning for partially observable MDPs. arXiv preprint arXiv:1507.06527.
[3] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin. Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.
[4] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 47:253–279, 2013.
[5] Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In ICLR, 2016.
[6] van Hasselt, Guez, Hado, Arthur, and Silver, David. Deep Reinforcement Learning with Double Q-Learning. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016. URL http://arxiv.org/abs/1509.06461.
[7] Watkins, C. J. C. H., and Dayan, P. 1992. Q-learning. Machine Learning 8(3-4):279-292.
[8] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Mare Lanctot and Nando de Freitas. Dueling Network Architectures for Deep Reinforcement Learning. arXiv: 1511.06581v3, 2016.