Solving Atari Game (Ms. Pac-Man) Using Deep Reinforcement Learning Algorithms

Sabbir Ahmed
Electrical and Computer Engineering
University of California, Riverside
Riverside, USA
sahme094@ucr.edu

***Abstract*—Reinforcement learning (RL) algorithms facilitates an agent to optimize its behavior through interaction with the environment. Although RL algorithms have shown remarkable performance in solving large number of Atari games, performance of the algorithms suffer in complex game environments. And in this project, Deep RL algorithms have been used to solve the popular Atari video game Ms. Pac-Man, which is one of the games where performance of simpler RL algorithms like DDQN has remain quite inferior to the human benchmark. To improve performance of the simpler algorithms, two modification has been proposed. First, reward function modification is done to provide better feedback to the RL agent regarding how well it is performing in the game. This modification has been shown to improve performance of the DDQN algorithm. Second, training objective has been modified to estimate the optimal Q-function in conjunction with estimating the condition of the ghost in the game. The proposed training objective allows the Convolutional Neural Network (CNN) used in the algorithms to extract better informative features that are relevant for learning the optimal policy. This modification has been shown to be promising for improving performance in both the DDQN as well as Dueling DDQN algorithm.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Algorithms based on reinforcement learning (RL) [1]–[4] are attractive for learning how to control an agent according to its environment. RL has been successfully applied to play large number of Atari games, but it still remains a challenge to deal effectively with large state spaces in order to achieve excellent results after a relatively short training period. In this project, we present two modification to the standard baseline algorithms in order to train Ms. Pac-Man effectively at playing the game.

Since its release as an arcade video game in 1982, Ms. Pac-Man has become one of the most popular games of all time. In addition to being popular, due to the simple rules of the game, paired with the complex strategies needed to achieve a high score, Ms. Pac-Man is an intriguing topic for research in Artificial Intelligence [5]. Furthermore, the game of Ms. Pac-Man fits into the frameworks of RL algorithms as well [6].

It is difficult to predict the environment, because of the stochastic ghost behavior. A well-defined reward function can be defined as an incentive for completing in-game activities, like collecting pills. Moreover, the the environment has a small action space i.e., the way Pac-Man may move is restricted. Nevertheless, because the agent has to move constantly and pills can be placed in a variety of places, there is a huge state space for which a large number of values are required to describe a single game state. And such complexity makes the game a suitable one to solve using RL algorithms.

There has been a number of research efforts to alleviate the aforementioned problem of large state space. For example, there exists simplified game environments for this particular Atari game Ms. Pac-Man where the position of the agents are restricted to discrete squares in the maze [7], [8]. In addition, literature reveals that the size of the maze has been lowered as well to deal with such large state space [6]. However, in this project, we take the pixel based screen of the game as observation for training the RL agent.

It can be hypothesized that, a compact state which has all the information regarding position of pills, power pills, the ghosts including the condition of the ghost (whether powerful/vulnerable) would be an ideal state represntation. However, the game environment is such that we don't have such information and all that is available as state for training the RL agents are the pixel based screen of the game. The aforementioned state would not only be the ideal one for training the RL agent but also would be a state representation requiring lower memory and computational complexity. Fewer dimensional states have been used in the literature for characterizing the game-state in Ms. Pac-Man. However, in this project, we modify the training objective so that the CNN extracts informative features from the pixel domain in an end-to-end manner.

Statement of contributions. In this project, two modification has been proposed to standard baseline DDQN and Dueling DDQN algorithms to learn better policy quickly. The first modification involves modification done to the reward function, where in contrast to the baseline algorithms which considers equal reward for no score gain and death of Ms-PacMan, the proposed modification considers small negative reward for no pellet/power pill consumption and large negative reward for death of Ms. Pac-Man resulting from clash between Ms. Pac-Man and the ghosts when they are powerful. Interestingly, this reward function modification improves the training of standard DDQN algorithm. This shows that, the standard DDQN baseline which gives equal reward for no pellet/power

pill consumption and death of the agent is suboptimal and has room for improvement. The second modification involves the modification of training objective that encourages the CNN to extract informative features corresponding to the state that helps in differentiating the states properly. The modified training objective involves a new task of ghost state prediction in addition to the standard RL training objectives that enables the network to extract useful features. This modification has been shown to improve the training of both DDQN algorithm as well as Dueling DDQN algorithm. This suggests that additional training objective with existing RL training objective may improve training and the performance of trained agents i.e., learn better policy quickly if the objective encourages informative feature extraction relevant to the RL training objective.

II. Environment

A. Game environment

OpenAI gym enivornment [9] is used for simulating the game and training the agent. The gym environment for Ms. Pac-Man is an environment just like the original game except that it provides the option to observe the game screen, take action and transition to next game screen and also returns the score as reward. Figure 1 shows a screen in the game environment. Ms. Pac-Man moves through the maze filled with pellets. The main motive of Ms. Pac-Man is to collect all of this pellets and the power pills to successfully complete the level. However, Ms. Pac-Man is attacked by four ghosts that move around the maze to makes the job as tough as possible. Collision with one of the ghost causes Ms. Pac-Man to lose a life. Therefore, the the main objective of collecting all the pellets is interlinked with another objective i.e., to avoid collision with these ghosts. Interestingly, four power pills are placed in four corners of the maze. And collection of one of the power pills makes the ghost scared for a period of time. In this scared condition, collision with ghost does not result in life loss for Ms. Pac-Man, rather the ghost is killed and a bonus score is obtained. Furthermore, there are tunnels placed in the maze which allows Ms. Pac-Man to move from one side of the maze to the other. The speed of motion of Ms. Pac-Man depends on a couple of factors, such as, collection of pellets and moving from one side to another side of maze via tunnel. And the speed increases with collecting pellets and level completion. To decrease the difficulty of the game, Ms. Pac-Man is given three lives. The game terminates after all of the lives are lost.

B. Ms. Pac-Man game as Markov Decision Process

Through reinforcement learning, an agent is taught to interact with an unknown environment in order to perform a task. The environment provides the agent with feedback through rewards whenever it performs actions. The goal is to gradually improve the agent's behavior and to adjust its policy such that the total long-term expected reward is maximized.

The use of Markov Decision Processes is a great way to describe RL tasks. A Markov Decision Process (MDP) [10]

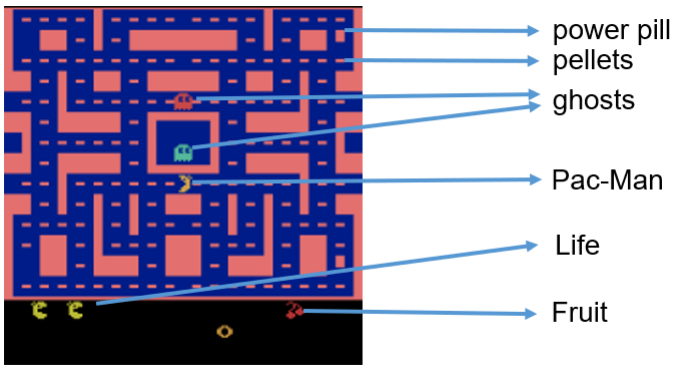


Fig. 1. Screenshot of game screen.

can be represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. Here, $\mathcal{S}$ is the set of all states in the environment; $\mathcal{A}$ is set of all actions that can be taken in the environment; $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the Markovian transition model that provides transition probability $P(s, a, s')$ that specifies the probabability of transitioning to state s' by taking the action a given the current state s; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the function mapping all state-action space to rewards and $\gamma \in (0, 1)$ is the discount factor used for calculating the return. An agent makes sequential decision of choosing actions while interacting with the environment until an episode terminates. After termination of an episode, the environment resets and a new episode starting from initial state follows.

In RL, a fundamental assumption is that all relevant information is available in the current state that provides adequate information for decision making. In other words, history or travel paths are irrelevant in deciding how to proceed. This assumption is widely known as the Markov property [11]. The Markov property states that the probability of the next state and reward calculated from the complete history of a system is equal to that calculated from the present information alone. The assumption holds fairly well in Ms. Pac-Man. This is because, it is only the ghosts who are involved in preventing Ms. Pac-Man from accomplishing her goal in the game which are the only sources of uncertainty about the future.

For solving Ms. Pac-Man using Deep RL algorithms, pixel-based game screen i.e., image is used to construct the states. To reduce computational complexity, each image is converted from RGB to gray scale and resized to 84×84. A sequence of four frames are stacked i.e., a tensor of $84 \times 84 \times 4$ is constructed (see Figure 3) which is the state for the problem. Ms. Pac-Man observes the state and then takes action, then the game environment returns the score (reward) as feedback and the environment transitions to next state. A schematic diagram for Ms. Pac-Man as an MDP is shown in Figure 2.

Reward function plays one of the most important role in RL by providing proper feedback as to how well the agent is performing in terms of choosing optimal actions. The reward function used in the baseline algorithms are listed in Table I.

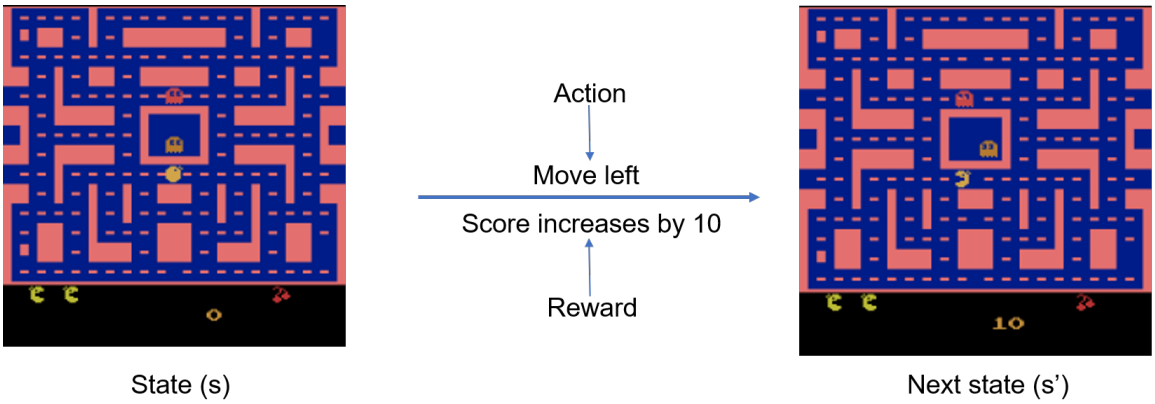


Fig. 2. Schematic diagram of MDP for Ms. Pac-Man.

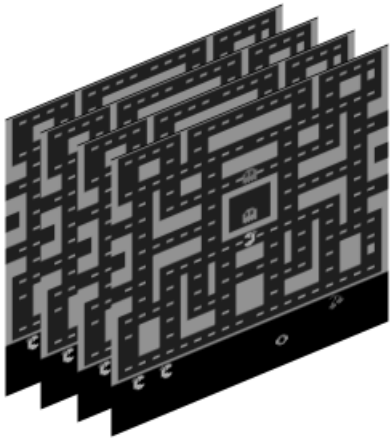


Fig. 3. Four frames stacked that represents the state of the game environment.

Certain rewards are offered by the gym environment when an event takes place due to the actions taken by the agent, such as, collecting pellets, collection of power pills or collision with a ghost. Whenever an action causes multiple game event to occur, the rewards for each of the events are added and returned as a single reward.

TABLE I
LIST OF GAME EVENTS AND THE CORRESPONDING REWARDS, AS USED IN THE BASELINE DDQN AND DUELING DDQN ALGORITHMS.

Event	Reward	Description
Stationary	0	Ms. Pac-Man stays at previous position
Pellet	10	Ms. Pac-Man collected a pellet
Power pill	50	Ms. Pac-Man collected a power pill
Fruit	100	Ms. Pac-Man has collected fruit
Scared ghost	200	Ms. Pac-Man and a scared ghost has collided
Powerful ghost	0	Ms. Pac-Man and a powerful ghost has collided and loses one life

III. Technical Methods

In RL, the idea of value function is of central interest. The value function $V^{\pi}(s)$ of a state s under a certain policy π, is the total expected return that is obtained starting from the state s until the episode terminates. It is defined as the following,

$$V^{\pi}(s) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t) | s_0 = s, \pi]$$

The value function must also obey the Bellman's optimality equation:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[R(st) + \gamma V^{\pi}(s_{t+1}) | s_t = s]$$

The above represents the relationship between consecutive states in an episode. Similarly, the state-action value function $Q(s, a)$ expresses the the total expected return received after taking action a from the state s and thereby following the policy π. The state-action value function for state s and action a following the policy π can be written as,

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t) | s_0 = s, a_0 = a]$$

DDQN and Dueling DDQN algorithms are adopted for solving Ms. Pac-Man. These algorithms are chosen because they are simple and at the same time applicable for the discrete action space involved in the game. Both DDQN and Dueling DDQN are considered the baseline algorithms and modification on them are performed to improve performance.

A. Double DQN

Double Q-learning was first proposed by Van Hasselt *et al.* in [1] which is a modification to DQN. In Double DQN, a copy of the online network known as the target network is kept and updated after τ iteration steps. And the greedy policy is evaluated according to the online network that uses the target

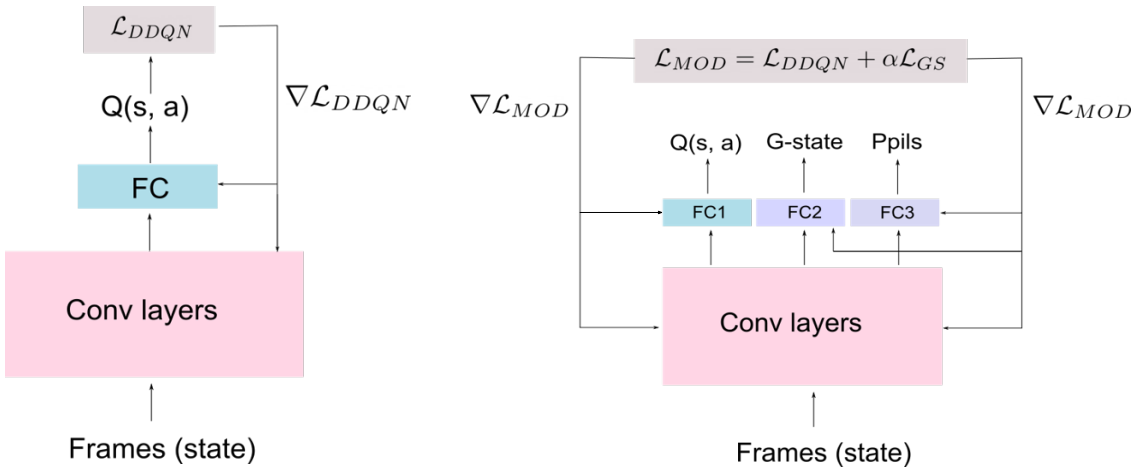


Fig. 4. Training pipeline of baseline algorithms (on left) and algorithms with modification 2 (on right).

network to estimate its value. In Double DQN, target values are generated according to the following:

$$Y_t^{DoubleDQN} = R_{t+1} + \gamma Q(s_{t+1}, \text{argmax}_a\ Q(s_{t+1}, a; \theta_t), \theta_t^-)$$

B. Dueling DQN

The Q-function in RL theory can actually be split up into two independent terms: the value function $V(s)$ that represents the value of being in the current state, and the advantage function $A(s, a)$, which reflects the relative importance of each action for the agent. This can be described by the following:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

A dueling network architecture was proposed in [4] by Wang *et al.*, which consists of splitting the neural network into two streams one for estimating each function, that share a convolutional base.

C. Modification 1

The first modification done over the baseline algorithms is the use of modified reward function. From Table I we observe all the rewards for each events which are used for training agents using the baseline algorithms. One of the problem associated with reward formulated in Table I clearly looks to be suboptimal is the rewards associated with the events *Stationary* and *Powerful ghost* both of which provides reward of 0. This is suboptimal, because the action that results in Ms. Pac-Man remain in the previous state are problematic for reaching the ultimate goal since these set of actions does not lead to collection of pellets or other objective for solving the game. However, collision with a powerful ghost is a more problematic event and biggest hindrance in reaching the ultimate goal of the agent. And, the problem seems to be that both the events provide same feedback when collision with a powerful ghost should be more negatively rewarded than when Ms. Pac-Man remains stationary in the game. In light of this observation, a modified reward function is designed that is given in Table II which considers collision with a powerful ghost more problematic event than the ghost remaining stationary and therefore, offers larger negative reward (-200) due to collision with powerful ghosts leading to loss of lives than the reward (-1) when the ghost remains stationary.

TABLE II
LIST OF GAME EVENTS AND THE CORRESPONDING REWARDS, AS USED IN MODIFICATION 1.

Event	Reward	Description
Stationary	-1	Ms. Pac-Man stays at previous position
Pellet	10	Ms. Pac-Man collected a pellet
Power pill	50	Ms. Pac-Man collected a power pill
Fruit	100	Ms. Pac-Man has collected fruit
Scared ghost	200	Ms. Pac-Man and a scared ghost has collided
Powerful ghost	-200	Ms. Pac-Man and a powerful ghost has collided and loses one life

D. Modification 2

The second modification done over the baseline algorithms is the use of an additional objective for updating the CNN in conjunction with the RL training objective of the baseline algorithms. From Table II, we observe that the collision with the ghost is not always problematic i.e., collision with scared ghost is suitable for reaching the objective. However, when the agent is observed to play the game during the episodes, it is seen that it makes moves closer towards the ghost when the ghost is powerful resulting in loss of life and moves away from the ghost when it is scared/vulnerable losing valuable points that could be added to the scoreboard. This is completely suboptimal decision making as compared to how a human would decide for solving the problem.

It can be hypothesized by observing the state as can be seen in Figure 3, that, a valuable information of the ghost state is not available in the states which is the color information that

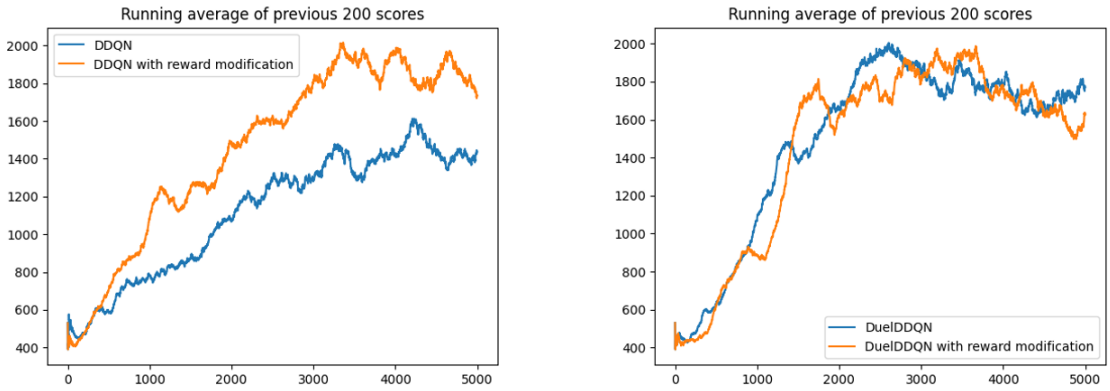


Fig. 5. Episodic return over the course of training using modification 1.

is a clear indicator of the condition of the ghost i.e., whether the ghost is scared or is powerful. However, to incorporate color in the state will not only increase memory requirement of the problem but also will increase the computational burden leading to huge training time. To alleviate this problem, we modify the deep CNN architecture as shown in Figure 4 and force the network to classify the ghost condition present in the state i.e., whether the ghost is powerful or scared. In addition to this, number of power pills is closely related towards the ghost state since, the ghost becomes scared after collecting any of the power pills for a certain period of time. Therefore, the network is also forced to predict the number of power pills in the state. Both the classification of ghost condition and number of power pills is termed as ghost state prediction and the loss calculated for the task in conjunction with the RL training objective can be written as follows:

$$\mathcal{L}_{MOD} = \mathcal{L}_{DDQN} + \alpha \mathcal{L}_{GS}$$

where $\mathcal{L}_{DDQN}$ is the baseline RL training objective, $\mathcal{L}_{GS}$ is the loss for the task of ghost state prediction and α is the weighting term emphasizing the prediction of ghost state in conjunction with the RL training objective. Furthermore, $\mathcal{L}_{GS}$ is defined as,

$$\mathcal{L}_{GS} = \mathcal{L}_{CE}(\hat{y}_{GS}, y_{GS}) + \mathcal{L}_{CE}(\hat{y}_{Ppills}, y_{Ppills})$$

where $\mathcal{L}_{CE}(\hat{y}_{GS}, y_{GS})$ represents the Cross-Entropy loss calculated across predicted ghost state and extracted ground truth ghost state. And, $\mathcal{L}_{CE}(\hat{y}_{Ppills}, y_{Ppills})$ represents the Cross-Entropy loss calculated across predicted number of power pills and extracted ground truth number of power pills. The modification to the neural network and the training procedure is outlined in Figure 4.

E. Hyper-parameters

Same set of hyper-parameters are used for both the baseline algorithms and also for training the agents with modified algorithms. The set of hyper-parameters used in the experiments are listed in Table III

TABLE III
LIST OF HYPERPARAMETERS AND THEIR VALUES USED IN ALL THE EXPERIMENTS.

Hyper-parameter	Value
Batch size	64
learning rate (α)	1e-4
ϵ_{max}	1
ϵ_{min}	0.1
ϵ_{decay}	5e-6
episodes	5000

IV. Experimental Results

In this section we discuss the results obtained in training the baseline algorithms and the efficacy of each of the proposed modification in improving performance of the baseline algorithms. All of the experiments were performed in a hardware environment that includes a Intel(R) Core(TM) i7-8700 CPU, 32 GB RAM and Nvidia RTX 2080 (8 GB Memory) GPU. At the very last stage of the project, access to a server with AMD Ryzen Threadripper PRO 3975WX 32-Cores CPU, 512.0 GB RAM and NVIDIA RTX 2080 GPU (8 GB Memory) was granted which was also used for running the experiments. All of the necessary codes are written in Python and Pytorch DL library was used to implement the neural networks.

A. Modification 1

In this subsection, we discuss the efficacy of the proposed reward modification as described in Section III-C. We observe the episodic return over the course of training for both DDQN and Dueling DDQN before the reward modification and after reward modification. The episodic return curve for the comparison is shown in Figure 5. As can be observed in Figure 5, the reward modification improves training using DDQN algorithm by a large margin. This clearly shows the benefit of differentiating between reward when the agent is stationary and when it is colliding with the powerful ghost. However, the reward modification fails to result in any improvement for Dueling DDQN algorithm. This may be due to the proposed modification not being numerically optimal

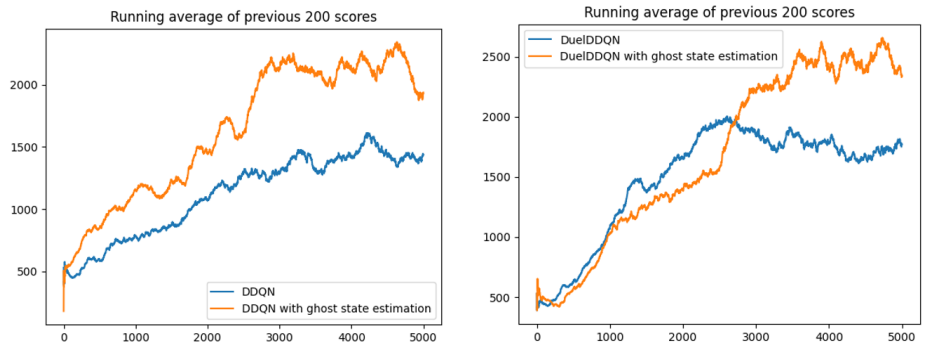


Fig. 6. Episodic return over the course of training using modification 2.

for Dueling DDQN algorithm and there might be some other numerical modification of the reward which might improve performance of Dueling DDQN algorithm as well. However, due to huge computational time and time constraint we only show a single modification to the reward function that has improved performance over baseline DDQN algorithm.

B. Modification 2

In this subsection, we discuss the efficacy of the proposed modification to the training objective as described in Section III-D. Similarly as done in Section IV-A, we observe the episodic return over the course of training for both DDQN and Dueling DDQN algorithm before adding the additional training objective and after adding the additional training objective of ghost state prediction. The episodic return curve for the comparison is shown in Figure 6. As can be observed in Figure 6, the modification where the network also predicts condition of the ghost improves training in both DDQN and Dueling DDQN algorithm over the baselines. This clearly shows that the ability of the network for predicting ghost state also benefits it in learning more accurate Q-function. This might be due to the accurate prediction of ghost state being informative to the agent for optimal action sequence. However, we see that the Dueling DDQN algorithm curve gradually improves over the baseline Dueling DDQN. One of the reasons for initial lag of performance might be the noise present in the labels for the ghost state and number of power pills.

C. Performance of algorithms before and after modifications

In this subsection, we evaluate the trained agents after training with DDQN and Dueling DDQN algorithm before and after performing different modifications. The value of ϵ is set to 0.05 and mean of raw scores of 100 episodes played by the trained agents are used for evaluation as reported in Table IV. From Table IV, we observe that, baseline Dueling DDQN offers better performance than that of the baseline DDQN algorithm. In addition to this, we observe that, reward modification as proposed in Section III-C offers improvement only in DDQN over the baseline DDQN algorithm and results in performance degradation in Dueling DDQN over the baseline Dueling DDQN. Furthermore, we see that the additional training objective as proposed in Section III-D offers improvement in both DDQN alogrithm and Dueling DDQN algorithms over their respective baseline.

TABLE IV
MEAN RAW SCORES OF 100 EPISODES PLAYED BY THE AGENTS TRAINED USING DIFFERENT ALGORITHMS.

Algorithm	α	**Mean Score**
DDQN (baseline)	N/A	1869.2
DDQN (modification 1)	N/A	2871.2
DDQN (modification 2)	0.5	2437.5
Dueling DDQN (baseline)	N/A	2431.4
Dueling DDQN (modification 1)	N/A	1916.2
Dueling DDQN (modification 2)	0.25	3155.9

V. Conclusion

In this project, two new approaches for improving performance of the baseline DDQN and Dueling DDQN algorithms are proposed. The first proposed modification is done to the reward function over the baseline algorithms in light of the obvious discrepancy of reward used in the baseline algorithms. To this end, this simple modification of the reward function offers improvement in DDQN algorithm. Furthermore, a second modification is proposed which adds an additional training objective with the existing RL training objective that is relevant to the condition of the ghost. In addition, number of power pills are also predicted in conjunction with the aforementioned training objective. This has shown to improve training of both DDQN and Dueling DDQN and also their evalulation performance respectively. The results obtained in the project suggests that in addition to learning the value functions, some additional training objective relevant to the extraction of features from the pixel states that are informative for optimal value functions might be helpful. Moreover, Autoencoder based approach was attempted to compress the states from pixel to high level feature domain. However, to have decent reconstruction performance the number of states needs to be significantly large which does not offer much benefits in terms of computational cost or memory and increases the overall training time.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] M. Wiering and M. van Otterlo, "Reinforcement learning: State-ofthe-art (vol. 12)," 2012.

[3] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[4] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[5] M. Gallagher and M. Ledwich, "Evolving pac-man players: Can we learn from raw input?" in *2007 IEEE Symposium on Computational Intelligence and Games*. IEEE, 2007, pp. 282–287.

[6] I. Szita and A. Lorincz, "Learning to play using low-complexity rule-based policies: Illustrations through ms. pac-man," *Journal of Artificial Intelligence Research*, vol. 30, pp. 659–684, 2007.

[7] M. Emilio, M. Moises, R. Gustavo, and S. Yago, "Pac-mant: Optimization based on ant colonies applied to developing an agent for ms. pac-man," in *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*. IEEE, 2010, pp. 458–464.

[8] D. Shepherd, "An agent that learns to play pacman," Ph.D. dissertation, Bachelor's Thesis, Department of Information Technology and Electrical ..., 2003.

[9] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[10] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[11] A. Markov and N. Nagornyj, "Theory of algorithms, ser," *Works of the Mathematical Institute im. VA Steklov. Israel Program for Scientific Translations*, 1971.