PLAYING A VIDEO GAME BY DEEP LEARNING

GRADUATION PROJECT PRESENTATION

EMRE YENIAY – 150110013

ADVISOR: DR. DAMIEN JADE DUFF

INTRODUCTION

- **Artificial Intelligence:** "Science and engineering of making intelligent machines, especially intelligent computer programs" - John McCarthy [1]
- **Machine Learning:** "The ability to learn without being explicitly programmed" - Arthur Samuel [2]
- **Deep Learning:** Using deep artificial neural networks for learning
- The main objective of this project is developing a software using deep learning which works on personal computers and can learn how to play a video game by itself.

REINFORCEMENT LEARNING

- After agent takes an action, environment returns new state and reward
- The designer is expected to provide only reward
- The agent tries to find an optimal policy which maximizes the reward

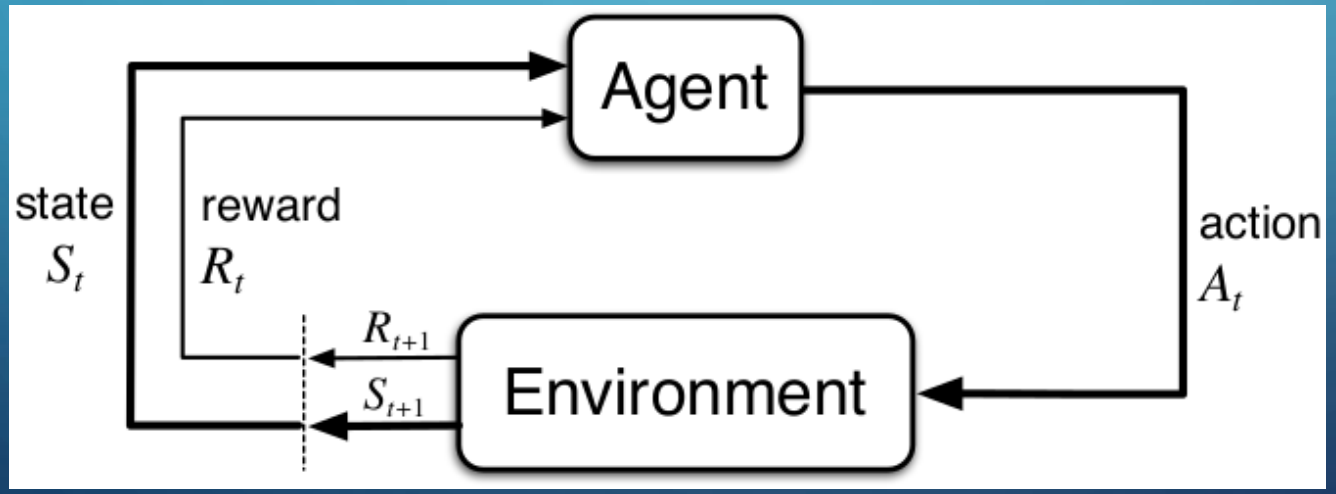

Figure 1: Relationship between agent and environment in reinforcement learning [3]

Q-LEARNING

- Model-free reinforcement learning technique
- Can be implemented using a table to keep Q-values
- But what if our state size is huge? (2^{416} different states in this project)
- Function approximation instead of storing a table can be realized using deep learning
- Input layer will be state information, meanwhile output of the neural network will be actions
- Highest Q-value at output layer will be selected at action

SUPER MARIO BROS

- Well-known 2D platform game published in 1985 on NES
- Video games make a good environment due to their nature
- Training is done on the first level of the game

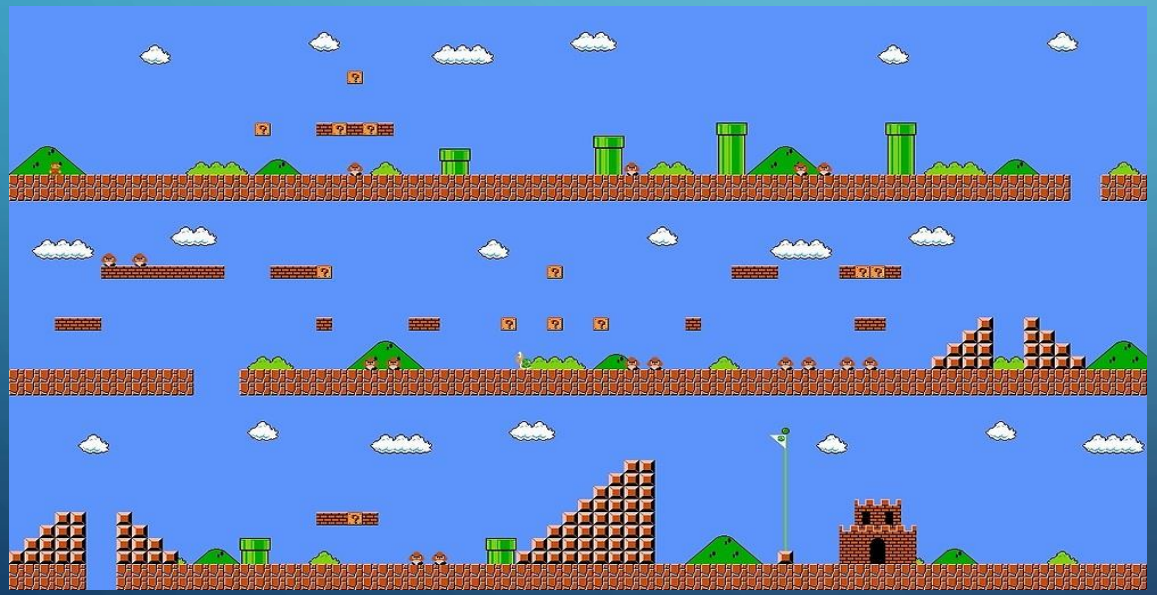

Figure 2: The map of World 1-1 of Super Mario Bros. [4]

STATE DATA

- 16x13 sized tile-based map working with 15 fps
- Each square in tile can have 4 different integer values:
  - 0: Nothing
  - 1: Platform
  - 2: Mario
  - 3: Enemy
- One Hot Encoding is used to convert categorical data to binary data
- 16 x 13 x 4 = 832 input of neural network

REWARD FUNCTION

- -10 to +10 points depending on amount of change of position
- -0.5 points if Mario is standing still
- -50 points if Mario is dead by enemy or pit
- +100 points if Mario finished the level

ACTION SPACE

- NES controller has total of 8 buttons, 6 buttons for gameplay
- However, agent need to press more than one button at same time
- Combinations of 6 buttons turned into 16 meaningful actions
- Selected actions are repeated for one more frame

Figure 3: A picture of NES controller [5]

NETWORK STRUCTURE

- 1 input layer with 854 nodes
- 3 hidden layers with 256 nodes in each
- 1 output layer with 16 nodes

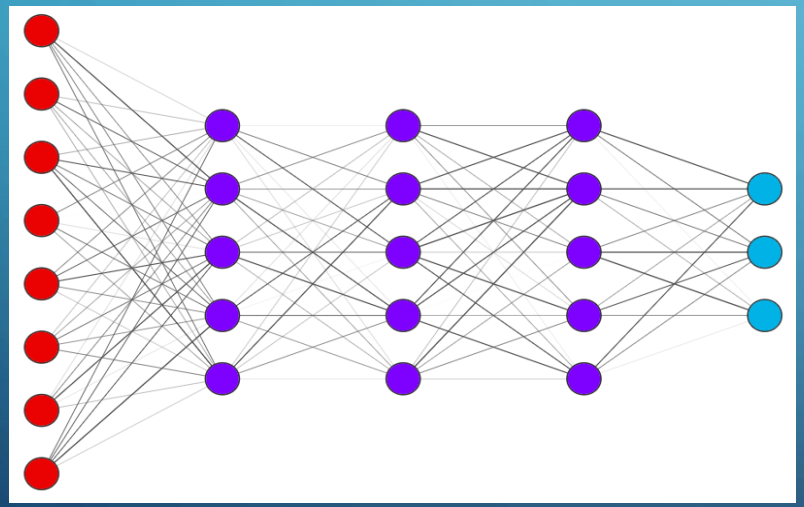

Figure 4: Illustration of network structure

MODELS

- Three training models are developed for the first experiment
- Batch size, learning rate and when the training phase occurs are different in each model
- The best one overall will be selected for the next experiment
- The number of trained frames is kept nearly same
- Each one prints stats of an episode after each episode
- A csv file is created at the end of every program including all stats

MODEL A

- Memoryless approach – Online learning
- Training based on a single experience each time
- Keeps training while playing
- Changes weights frequently while playing
- Used with a low learning rate

```
for episode number N to 250 do
    while episode is not finished do
        Get state s_t from environment;
        if random value is smaller than ε then
            Take a random action a_t;
        else
            Take the action a_t which maximizes predicted Q(s_t,a_t) value;
        end
        Get the new state s_{t+1} and reward r_{t+1};
        Train the model using experience E(s_t,a_t,r_{t+1},s_{t+1});
    end
end
```

Figure 5: The pseudocode of Model A

MODEL B

- Utilizes memory to save old experiences – Offline learning
- Training based on large batches
- Training and playing phases are separated
- Changes weights only at the end of the episode
- Used with a high learning rate

```
Initialize the experience memory M with size 20000;
for episode number N to 250 do
    while episode is not finished do
        Get state s_t from environment;
        if random value is smaller than ε then
            Take a random action a_t;
        else
            Take the action a_t which maximizes predicted Q(s_t,a_t) value;
        end
        Get the new state s_{t+1} and reward r_{t+1};
        Store experience E(s_t,a_t,r_{t+1},s_{t+1}) in memory M;
    end
    Get a random sample S with size of 1024 from memory M;
    Train the model using sample S as a batch;
end
```

Figure 6: The pseudocode of Model B

MODEL C

- Midpoint between Model A and Model B
- Training based on small batches
- Keeps training while playing
- Changes weights seldom while playing
- Used with a medium learning rate

```
Initialize the experience memory M with size 20000;
for episode number N to 250 do
    while episode is not finished do
        Get state s_t from environment;
        if random value is smaller than ε then
            Take a random action a_t;
        else
            Take the action a_t which maximizes predicted Q(s_t,a_t) value;
        end
        Get the new state s_{t+1} and reward r_{t+1};
        Store experience E(s_t,a_t,r_{t+1},s_{t+1}) in memory M;
        if frame number is multiple of 32 then
            Get a random sample S with size of 32 from memory M;
            Train the model using sample S as a batch;
        end
    end
end
```

Figure 7: The pseudocode of Model C

UTILIZED SOFTWARE

- Primary neural network API: Keras 2.1.5
- Backend machine learning API: TensorFlow 1.6.0
- Primary environment: Gym Super Mario 0.0.7
- Backend environment: OpenAI Gym 0.92
- NES Emulator: FCEUX 2.2.2
- Python IDE: PyCharm 2017.3
- Operation System: Ubuntu 16.04.4 LTS

HARDWARE SPECIFICATIONS

- CPU: Intel Core i7 2600k
- RAM: Kingston 16GB DDR3
- Storage: Samsung 850 EVO 256GB SSD
- GPU: Not utilized (CUDA can be used only on Nvidia GPUs)

Figure 8: A screenshot of the work environment

MODEL TEST

- Comparing Model A, Model B and Model C
- Each model played and trained on first level for 250 episodes
- Approximately equal to 100,000 frames for each model
- Trying to determine the best model to proceed

MODEL TEST RESULT

- Error is converged nearly same on all models with a nice curve
- Error in Model A remained high, whereas Model B has the error closest to zero

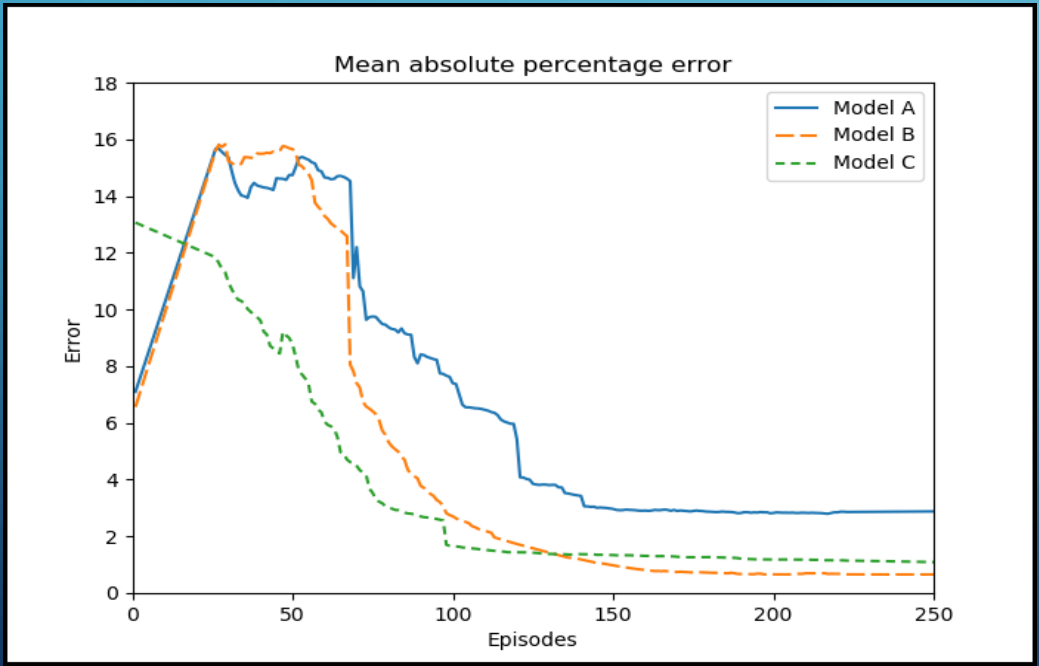

Figure 9: Mean absolute percentage error in episodes

MODEL TEST RESULT

- Model A learned faster to determine a way to gain rewards
- Model B became the second in this criteria
- Model C was learning the slowest

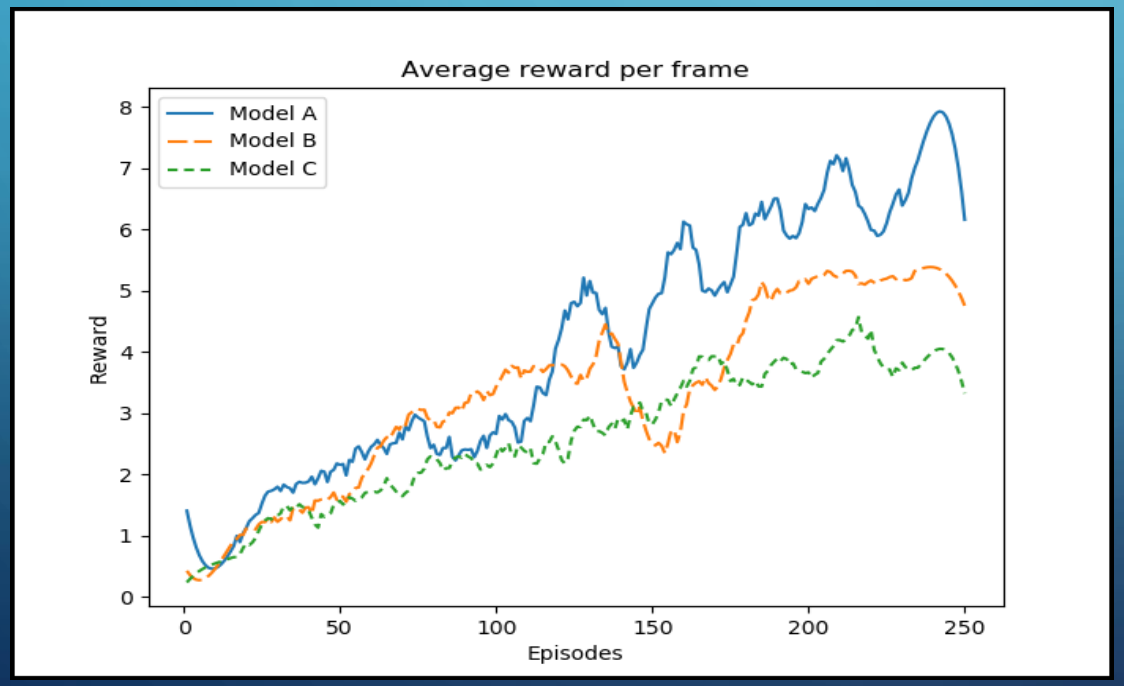

Figure 10: Earned average reward per frame in each episode

MODEL TEST RESULT

- Model B is the most efficient in time by far
- Model A was the worst, meanwhile Model C was the close second

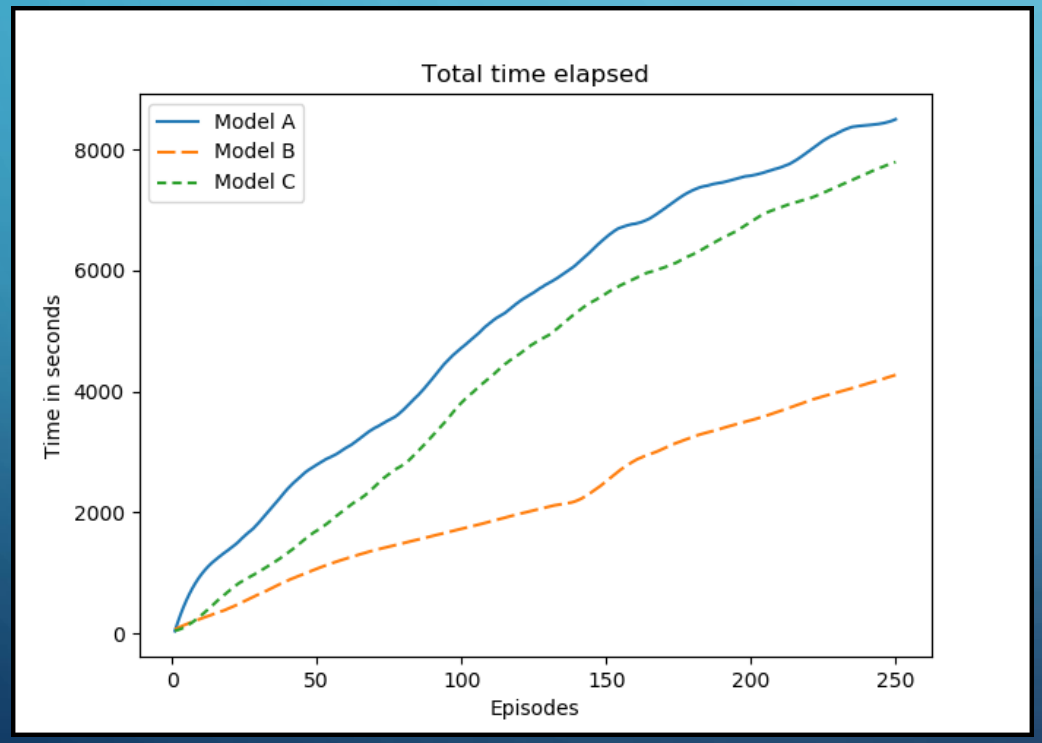

Figure 11: Total time elapsed during training

TRAINING TEST

- Model B is selected as the best model in previous experiment
- The goal is finishing the first level of the game at least once in this test
- Episode count is increased from 250 to 500
- The weights of the trained model are saved into a file at the end of the experiment

TRAINING TEST RESULT

- Able to finish the game 3 times in 500 episodes
- The first time is occurred in episode number 370
- Training is done here to prevent overfitting

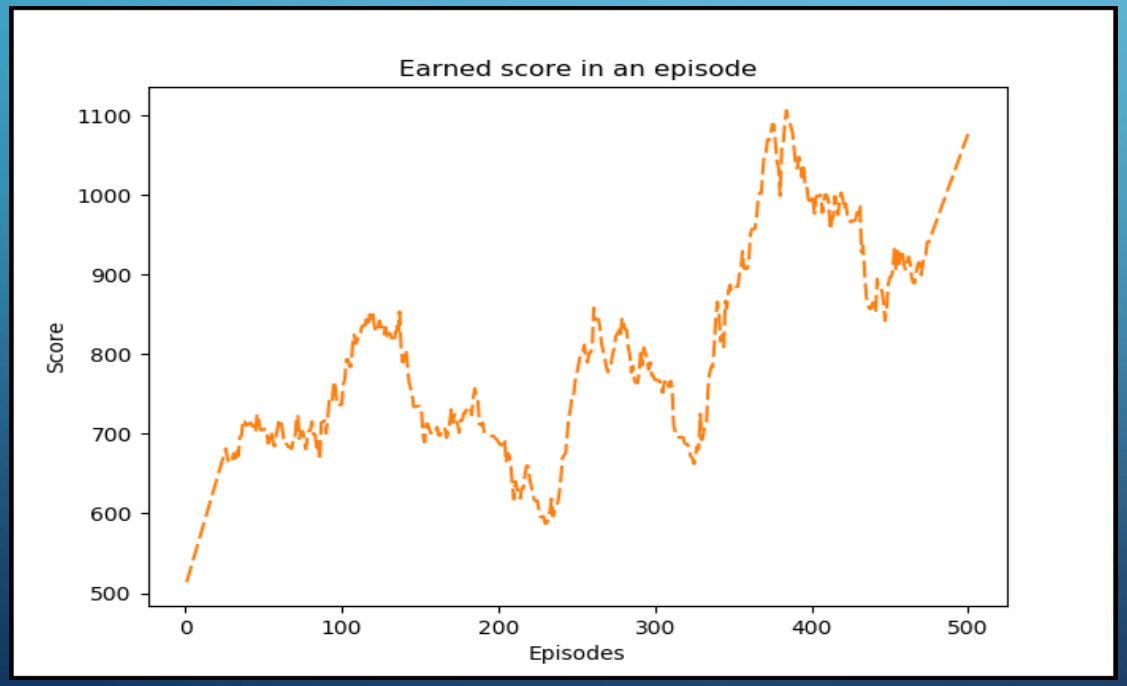

Figure 12: Earned score in an episode for Model B

GENERALIZATION TEST

- Can trained model show gains on a level which it never trained on?
- Tested on another level of the game versus a random model
- The weights are loaded from previous experiment

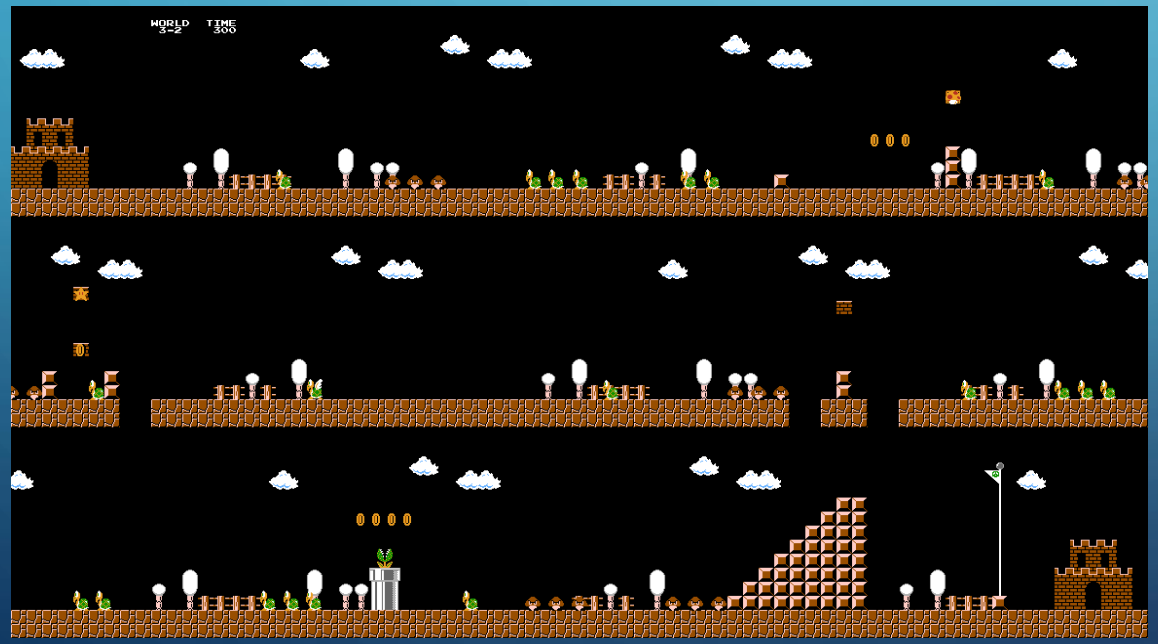

Figure 13: The map of World 3-2 of Super Mario Bros. [6]

GENERALIZATION TEST RESULT

- This level played for 100 episodes
- Trained model showed improvement over random model
- Much better gained reward
- Average and the highest score are nearly doubled
- Can be said that the model generalizes well

Table 1: Comparative results of models

	Average Reward	Average Score	Highest Score
Random Model	0.304	253.76	1177
Trained Model	6.269	561.6	2017

CONCLUSION

- Observed affect of batch size and training timing
- Successfully won the first level of Super Mario Bros.
- Trained model also showed good results on another level
- It is possible to train deep network with a limited computational power
- Results can be observed with low number of trained frames compared to other projects in this area

FUTURE WORK

- Changing tile based state data into RGB data of pixels
- Comparison between different learning algorithms:
  - Gradient Temporal Difference
  - SARSA
  - Double Q-learning
  - Some other learning algorithms

DEMOS

QUESTIONS

REFERENCES

- [1] John McCarty, “What is artificial intelligence?”, Stanford University, Nov. 2007. Available: http://www-formal.stanford.edu/jmc/whatisai.pdf
- [2] A. Munoz, “Machine Learning and Optimization”, Courant Institute of Mathematical Sciences, n.d. Available: https://cims.nyu.edu/~munoz/files/ml_optimization.pdf
- [3] R. S. Sutton and A. G. Barto, Reinforcement learning: an introduction, Second edition. Cambridge, MA: The MIT Press, 2017.
- [4] Redbubble, Super Mario Bros World 1-1. [online] n.d. Available: https://www.redbubble.com/people/stowball/works/16243810-super-mario-bros-world-1-1 [Accessed May 21, 2018]
- [5] Wikipedia, List of Nintendo controllers. [online] n.d. Available: https://en.wikipedia.org/wiki/List_of_Nintendo_controllers [Accessed May 18, 2018].
- [6] Super Mario Wiki, World 3-2 (Super Mario Bros.). [online] n.d. Available: https://www.mariowiki.com/World_3-2_(Super_Mario_Bros.) [Accessed July 14, 2018].