AN EMPIRICALLY OPTIMAL BATCH SIZE FOR DQN

Hui Wen Goh[*]
Tandon School of Engineering
New York University
New York, NY 10003
huiwen.goh@nyu.edu

Yara Kyrychenko[†]
Courant Institute of Mathematical Science
New York University
New York, NY 10003
yara@nyu.edu

Eion Tyacke[‡]
Tandon School of Engineering
New York University
New York, NY 10003
et1799@nyu.edu

ABSTRACT

Deep Q Learning has proven to be a great algorithm for achieving human-level performance on many tasks. However, training a Deep Q Network (DQN) successfully is tricky since it requires proper hyper-parameters. When hyper-parameters are not properly optimized, similarly performing models can take drastically different times to train. In this paper, we focus on one such hyper-parameter – the batch size. We build on the work done by Choi 2019 to find an empirically optimal batch size for more complicated network architecture and the cart-pole task. Based on metrics such as the average rewards obtained, the training time taken per episode and completion rate, the optimal batch size we obtained in our analysis is around 32 to 64, which confirms Choi's results.

1 Introduction

Many modern developments, like self-driving cars and recommender systems, benefit a lot from reinforcement learning. However, for reinforcement learning tasks with very large (or continuous) state and action spaces, traditional Q learning proves impractical since it requires computing and storing a table of state-action values. Deep Q learning builds atop classical Q learning by incorporating a deep neural network to approximate the action-value function. For performance improvement, deep Q learning employs the two key techniques of experience replay and target networks. The one we are interested in is experience replay, in which we store trajectories of the Markov Decision Process and then sample some number of them to train the neural net. This way we obtain less temporally-correlated transition samples which gives us more accurate updates [2].

While training a DQN, there is a tradeoff between the number of steps required to train the agent, and the training time taken per step. If a small batch size is used, the network parameters update per step is small and more steps would be required to successfully train the model. As we increase the batch size the number of steps needed to train the model would decrease, however each step would be more computationally expensive, hence increasing the training time for each step [1]. As such, we aim to find the optimal batch size to train an agent to do the cart-pole task.

Building on Choi 2019 results on the empirically optimal batch size, we conduct our own experiments with a different network architecture [1]. Seeing if and how the two optima differ between the architectures would help us see how universal Choi's analysis is.

2 Methods

We performed our analysis on OpenAI Gym's Cart Pole environment[4]. The goal of the task is to keep the pole balanced on the cart by moving the cart horizontally. The pole starts upright, and a reward of $+1$ is obtained for every timestep that the pole remains upright. An episode terminates if the pole tilts too far (an angle predetermined by the environment)

[*]Environment setup, DQN code and hyper-parameter tuning, running experiments, report writing.
[†]Literature review, running experiments, results visualization, report writing.
[‡]Literature review, DQN hyper-parameter tuning, running experiments, report writing.
[4]Environment can be accessed at https://gym.openai.com/envs/CartPole-v0/

from its upright position or if it has achieved 200 timesteps. The task is considered completed if the agent has achieved a hundred-episode reward average greater than or equal to 195.

We used a DQN agent with parameters presented in table 1. The neural network is a feed-forward neural net consisting of 2 fully connected layers with 256 nodes each. We chose this architecture because Choi 2019 also used a feed-forward neural net with 2 fully connected layers but the layers had 16 and 2 nodes, which was smaller than the architecture we used. However, this change has increased the time taken to run our experiments, which resulted in slightly different results obtained in our simulations.

We tested nine batch sizes: $4, 8, 16, 32, 64, 128, 256, 512$ and 1024. We trained most of them ten times for 3000 episodes. The exception was 1024 with only five experiments run due to computation time and 32 with 15 experiments. In total we ran 90 experiments, all of them on Google Colab's GPU. For all experiments, the environment was seeded. We didn't seed the network due to technical difficulties.

During our experiments, we kept track of rewards gained at each episode and time spent per episode. We use those in our analysis to empirically determine which batch size of the tested performs optimally, meaning gets the best average rewards in the shortest amount of time. We also consider if the agents had completed the task while training and the mean time required for one episode in our analysis.

Table 1: DQN hyperparameters used besides batch size.

α, learning rate	γ, discount factor	ϵ	ϵ - decay rate	copy steps
0.001	0.9	0.95	0.995	25

3 Results

From the simulations, we recorded the reward obtained by the agent and time taken for each episode. Then, the mean of rewards obtained across the different runs was taken to obtain an average reward per episode for various batch sizes. To obtain a smoother plot, we also took the running average of the average rewards for each consecutive 100 episodes.

Figure 1 shows the 100-episode running average for the average reward obtained per episode for the various batch sizes. We observe that on average, the experiments that were trained with batch sizes of 4 and 8 performed extremely poorly, while the experiments with batch sizes 32 and above had similar performances, all obtaining an average reward of above 150 at episode 1500.

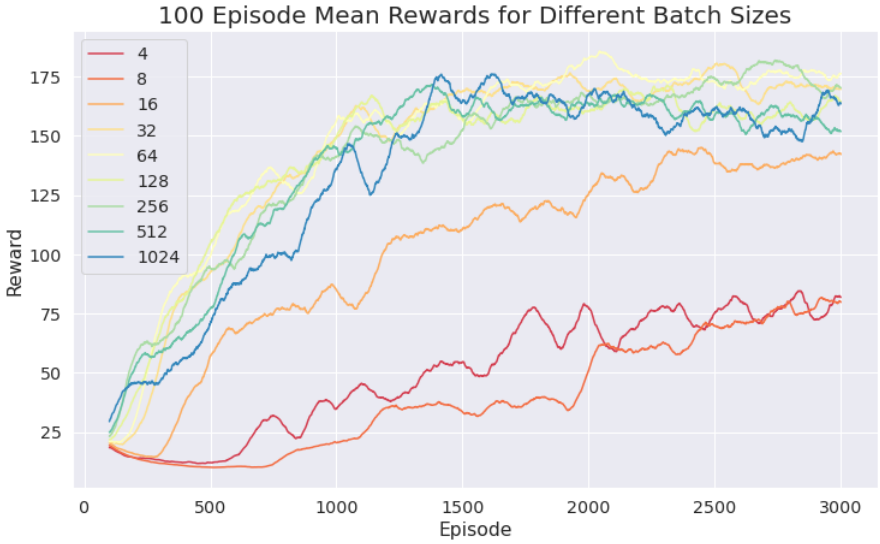


Figure 1: 100-episode running average of the average reward per episode for the nine batch sizes.

Next, we calculated the average reward obtained by the agent across all runs and episodes, and the mean time taken to train the network for each batch size. The relationship between the two variables is shown in figure 2.

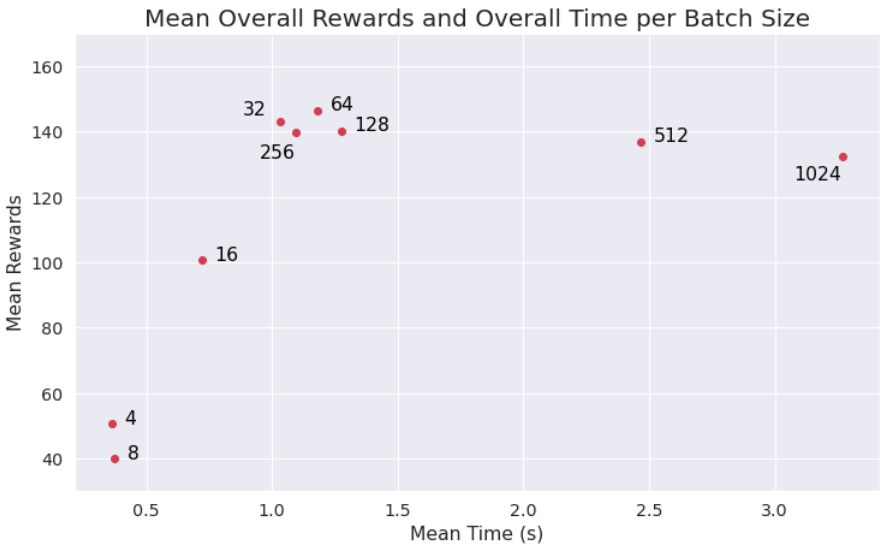
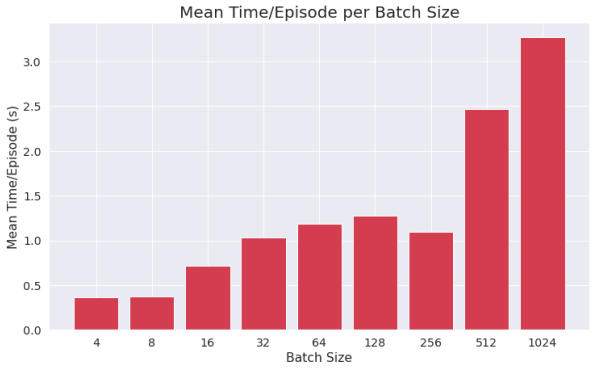


Figure 2: Mean training time versus mean rewards.

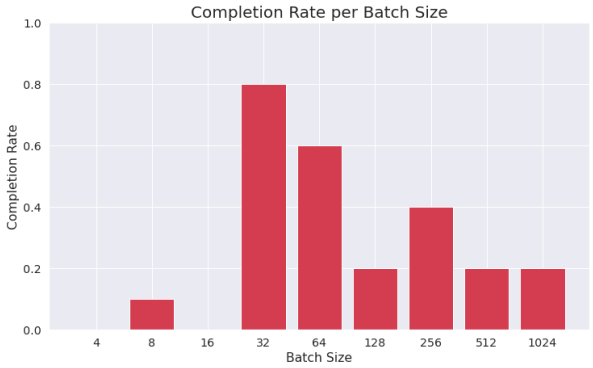
From the scatter plot above, we can see that the mean overall reward for experiments of batch size 32 and above are similar, however the mean training time for batch sizes 512 and 1024 are significantly higher than that of batch sizes 32, 64, 128 and 256. This indicates that increasing the training batch size beyond 256 will not drastically improve the performance of the agent, however the time taken for each episode will increase at large batch sizes. Therefore, a good batch size for this task is between 32 and 256, yet the greatest mean reward was obtained by batch size 64.

Figure 3a shows the mean time taken per episode for the various batch sizes. From the bar plot we observe a general increase in time taken per episode as the batch size increases. At low batch sizes, each episode length would be shorter due to the episode terminating when the pole tilts too far from the center. However, from previous results we know that the average reward, which is synonymous to episode length, of experiments with batch sizes 32 and above are the same, hence we can see the drastic increase in time taken for each timestep as the batch size increases.

Lastly, figure 3b shows the completion rates for experiments of various batch sizes. The experiments with batch size 32 have the highest completion rate at 80%. The experiments with low batch sizes have a very low completion rate, and experiments with high batch sizes also had sub-par performances compared to batch sizes 32 and 64.



(a) Mean time per episode for the nine batch sizes.



(b) Completion rates for the nine batch sizes.

Figure 3: Barplots indicating performance of batch sizes.

4 Discussion

We empirically tested performance and training time of nine batch sizes for a DQN agent with feed forward neural net of two 256-dimensional fully connected layers. We obtained results consistent with the previous literature on empirically optimal batch sizes, namely that the optimal batch size for the cart-pole task is between 32 and 64 [1].

Furthermore, our results indicate that at small batch sizes, the increase in the batch size positively impacts the performance of the agent, but this improvement in performance plateaus as the batch size increases to a larger amount. We also observe how increasing the batch size increases the training time taken, which further supports our inference that a moderate batch size would be optimal for training the agent.

However, our results are not as statistically significant as one could hope. Running more experiments, despite the computational cost, would provide more significant data and more certainty in the finding. We could also increase the number of episodes the agents are trained for to see more long-term performance. Additionally, we could have incorporated the variance and confidence intervals into our visualization and analysis. Another way to improve significance and reproducibility is to seed the network parameters.

Some other possibilities to extend our work would include more thoroughly tuning the other hyper-parameters (learning rate, discount factor etc.) to obtain the best DQN overall for the given task. Furthermore, we initially planned on extending Choi's analysis to a new environment but were unable to due to time and computational constrains. Hence, we would like to further extend Choi's work to new environments, such as Atari games (Breakout, Pac Man and more), to investigate if different tasks require different amounts of 'working memory' to be learned effectively.

References

[1] Choi, M.. (2019). *An Empirical Study on the Optimal Batch Size for the Deep Q-Network*. In: Kim JH. et al. (eds) Robot Intelligence Technology and Applications 5. RiTA 2017. Advances in Intelligent Systems and Computing, vol 751. Springer, Cham.

[2] Fan, J., Wang, Z., Xie, Y. & Yang, Z.. (2020). *A Theoretical Analysis of Deep Q-Learning.* Proceedings of the 2nd Conference on Learning for Dynamics and Control, in PMLR 120:486-489

[3] Mnih, V., Kavukcuoglu K., Silver D., Graves A., Antonoglou I., Wierstra D., & Riedmiller M.. (2013). *Playing Atari with Deep Reinforcement Learning.* NIPS Deep Learning Workshop 2013, arXiv:1312.5602