Navigation for the Banana Collector

Abstract

A simple deep-Q-learning algorithm was used to solve the “Banana Collector” game which was simulated in Unity. The vanilla agent, which was built with simple multi-layers perceptron with two hidden layers, 64, 64 neurons respectively, can achieve +13 scores on average at around 450 episodes and achieve +25 scores in some episodes. Several different model setting were used to compare agents’ performance. Comparing numbers of layers and batch sizes, slightly deeper network (for instance, 3 layers) or larger batch size (for instance, 512 per batch) will attain more stable rewards than vanilla one. However, when batch normalization (BN) was used in hidden layers, the agent crashed after thousands of episodes and recover to the average performance no more in the following episodes. The reasons of BN lead to crash may as follow: 1) large variation between sampled states 2) moving average converge to states that almost get the yellow banana while forget how to search / moving when banana is not shown or still far away. Building the agent with convolution neural network (CNN) may solve these falls of BN since actions guided by images will not change dramatically between adjacent states. Also, using images as input makes us easy to stack a few previous frames as current state. In summary, a simple mlp-like agent can solve the “Banana Collector” game easily, however, agents with CNN backbone may have greater change to get better and stable

performance since it reduce variations in input states.

- Project goal

In this project, we're going to use deep-Q-learning to train an agent. The goal of the agent is to collect correct things in the "Banana Collector Game". In this game, the agent is requested to collect "yellow" banana as many as it can and avoid "purple" banana as possible.

Method and experiments

Agent information

- State: 37 dimensions include agent's velocity, along with ray-based perception of objects around the agent's forward direction
- Action: 4 discrete actions: forward / backward / left / right
- Reward: Touch nothing: 0, touch purple banana: -1, touch yellow banana: +1

Model

Main structure: multi-layer perceptron network.

- Optimizer: Adam, learning rate: 5e-4
- Update per action. Soft-update (switch local net parameters to target net parameters) per 25 update.

Experiment

- Batch size (bz): 64 / 512
- Numbers of layers (nL): 2, 3, or 10 layers (64 units per layer)

- Use Batchnorm (BN): True / False

Result

bz	nL	BN	Max rewards	Avg.rewards of last 100 episodes
64	2	False	26.0	15.44
64	3	False	26.0	15.61
64	3	True	25.0	0.48
512	2	False	25.0	14.62
512	3	True	24.0	0.27
512	5	False	26.0	15.37
512	10	False	14.0	0.04
512	10	True	24.0	9.73

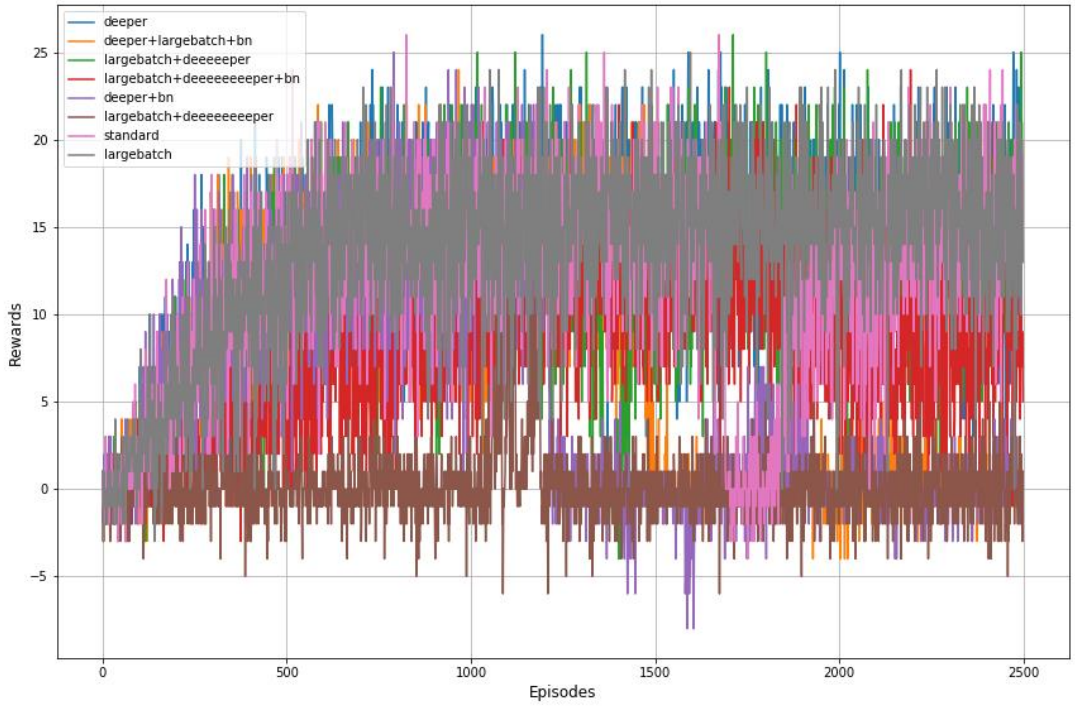

Figure1. Rewards of each experiment conditions.

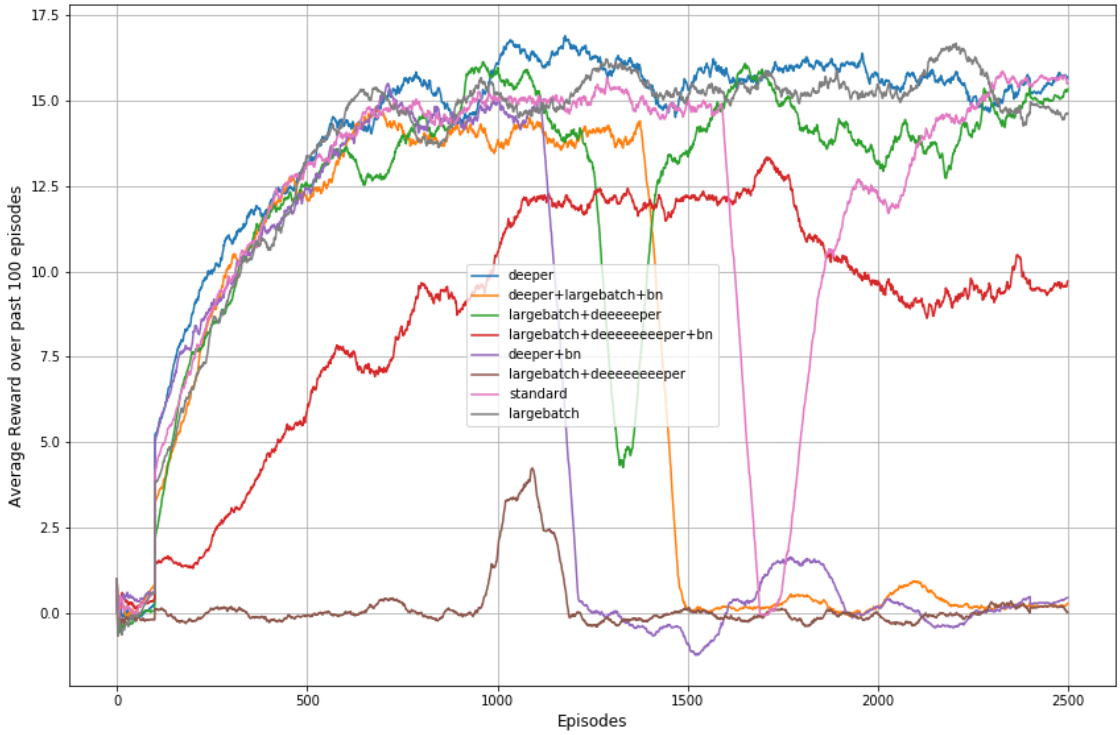

Figure2. Moving average rewards (window size = 100) of each experiment conditions.

Discussion

1. Deeper networks provide more non-linearity and model capacity to learn complex representations. Comparing numbers of layers, our results shows that slightly deeper networks gain more rewards than shallow one. However, the performance will decrease when the network becomes too deep, for instance, stack 10+ layers. Such phenomenon is due to gradient vanishing and may be solved by special design such as residual block in Convolution Neural Networks (CNN) or batch normalization. Our results shows that using 10 layers network with BN will make the agent be able to learn the game.
2. The batch normalization (BN) has been proved to improve the stability of the

network’s performance and widely used in modern networks. Using the BN in the “Banana Collector game”, however, leads the agent crash after about one thousands of episode. Reasons of such fails may due to 1) variations between states during agent exploring the environment and 2) mean/variance and moving average converge to states that know how to approach yellow bananas while loss its ability to explore environment when no bananas placed around.