Different Neural Network Depths for Deep Reinforcement Learning

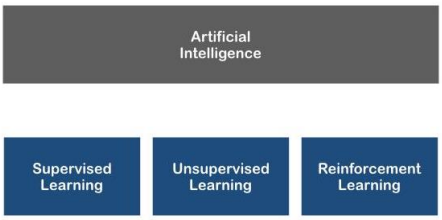
Jonathan Sadighian, Nadia Sjöstedt, Rhea Moubarak
Computer Science Master's Program, École Pour l'Informatique et les Techniques Avancées

Abstract

In recent history, there have been many advances in artificial intelligence, and more specifically, deep reinforcement learning. In this paper, we explore and test the performance of different neural network architectures with a fixed neuron budget for deep reinforcement learning. We use the DDQN algorithm with prioritized experience replay for the Atari game Space Invaders to evaluate performance.

1. Introduction

Machine learning is an application of artificial intelligence where information is automatically learned and not explicitly programmed. As of today, there are three machine learning paradigms: supervised learning, unsupervised learning and reinforcement learning.



In *supervised* learning, learning is derived from a training set of labeled examples provided by a knowledgeable external supervisor. The goal of supervised learning is to extrapolate, or generalize either a label or a correct action to take for a given situation. Supervised learning is the kind of learning studied in most current research in the field of machine learning [16].

In *unsupervised* learning, learning consists of finding structure hidden in collections of unlabeled data. Moreover, under this paradigm learning does not focus on selecting a correct action for a given situation and does not have a specified goal.

In *reinforcement learning*, learning is discovered through trial-and-error. Although reinforcement learning is similar to unsupervised learning in that labeled data is not provided for training, it differs in that the goal is to maximize a reward signal, rather than finding hidden structure.

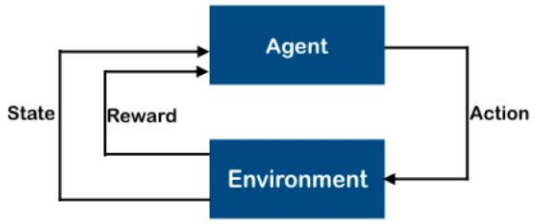
1.1 Markov Decision Process

Reinforcement learning uses an approach called Markov Decision Process (MDP) to

make decisions. The idea behind this framework is to capture the most import aspects of a real problem facing a learning agent interacting over time with its environment to achieve a goal.

Figure 1 is a visual example of the MDP framework.

Figure 1



Furthermore, an MDP captures the environment in a grid, and divides it into actions, states, and rewards. Under the MDP framework, an agent uses its policy to select the best action for a given state. Moreover, the policy acts as a function taking the state as an input and produces the action as an output. The purpose of Reinforcement Learning is to solve the MPD.

1.2 Neural Network

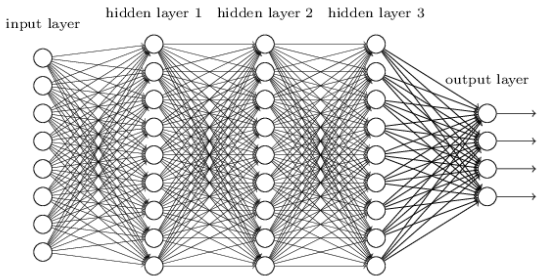
Neural networks are a family of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. In deep learning, a network composed of several layers and these layers are made of nodes. A node is where computation happens, loosely patterned on a neuron in the human brain, which fires when it encounters sufficient stimuli. A node combines input from the data with a set of coefficients or weights, that assigns significance to inputs for the task the algorithm is trying to learn.

These input-weight products are summed and the sum is passed through a node, activation function, which leads to an act of classification.

A multilayer perceptron (“MLP”) is a type of forward feed artificial neural network. Deep learning networks often used MLP’s in the past, but have recently been replaced with convolutional and long-short term networks [8].

Figure 2 is a visual example of an MLP neural network.

Figure 2



Earlier versions of neural networks were shallow (only one hidden layer). The term “Deep” in neural networks means more than one hidden layer. Hence, adding more layers and advancing further into the neural net can make the nodes recognize more complex features since they aggregate and combine features from the previous layer.

1.3 DQN

Deep Q-learning (DQN) is a type of temporal-difference reinforcement learning algorithm which uses a neural network as the function approximator (i.e., the best action for given observation). Another distinguishing factor of DQN is memory replay: saving previous states, actions, rewards, and proceeding states into a

buffer, which is sampled in batches to compute gradients. Research has proven that experience replay improves data efficiency and reduces correlation amongst sampling [8].

1.4 Dueling DQN

Dueling DQN (DDQN) algorithm proposed by [2] modifies the standard DQN neural network architecture by splitting a state's Q-value into its parts:

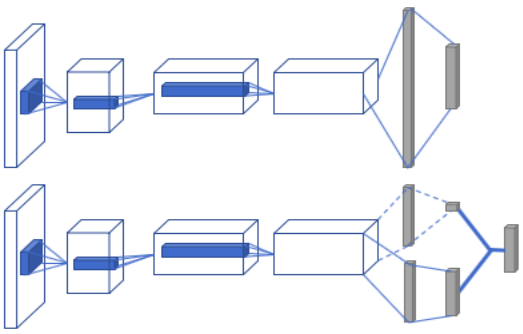
$$Q(s,a) = A(s,a) + V(s)$$

- **V(s):** the value of being at that state
- **A(s,a):** the advantage of taking that action at that state

By decoupling the action-state and state values, an agent is able to focus on selecting a better action for a given state. After splitting the two values, their outputs are aggregated together find the value for the state-action pair (i.e., Q-value). [2] has proven that this architecture helps accelerate training and find more reliable Q-values for each action.

Figure 3 is sets forth the differences between DQN vs. Dueling DQN architectures.

Figure 3



1.5 Prioritized Experience Replay

Prioritized experience replay modifies experience replay to focus on experiences may be more important than others for our training, but might occur less frequently. [3] has shown that prioritized experience replay accelerates learning and outperforms non-prioritized replay learning.

2. Related work

There has been a lot of research in recent history applying deep reinforcement learning algorithms to solve difficult tasks [7]. [8] used Deep Q-Learning to achieve superhuman level performance when playing Atari games.

Others [1, 6, 9] have proposed novel methods for solving reinforcement learning problems, which have outperformed [8]'s DQN benchmark. As of this time, it is well known that the reinforcement learning model we used in our experiment is not the latest and greatest [10, 11, 12].

Since DQN was published in 2013, there has been relatively little research evaluating deep reinforcement learning neural network architecture under a fixed neuron budget. [13] experiments with the number of layers in an MLP, but applied to a supervised learning problem. To the best of our knowledge, we are the first to explicitly experiment with the impacts of depth versus width performance using a fixed neuron budget applied to deep reinforcement learning.

3. The Experiment

Our experiment consists of training, testing, and evaluating the performance of three different neural networks using the same

DQN algorithm enhanced with prioritized replay [3, 4] and a dueling network [2].

3.1 The Simulator

We use OpenAI Gym for the simulator in our experiment. OpenAI Gym is an environment toolkit used for developing and comparing RL algorithms. The Gym library is a set of environments (e.g., Algorithmic, Atari, Robotics, etc.) with a common interface for testing RL algorithms.

In our experiment, we use the Atari environment for evaluation. Atari games have multiple versions of how environment data is processed and rendered. The frame-skip option gives researchers the option to reduce the number of frames-per-second that are fed to the environment, which can improve gameplay performance [14]. Alternatively, we use the "NoFrameSkip" option, which ensures that our agent steps through all frames in an episode. More specifically, we use "SpaceInvadersNoFrameSkip-v4" as our environment. V4 indicates that our environment provides our agent a stack of four frames, which follows [8]'s approach.

3.2 The Agent

We use OpenAI Baselines for the agent in our experiment. OpenAI Baselines are a set of implementations of reinforcement learning algorithms with the intention of providing researches with templates to ensure consistent experimentation.

In our experiment, we use the "DeepQ" algorithm provided by OpenAI Baselines for our agent with all the default hyperparameters.

3.3 Neural Network

OpenAI has many neural network architectures available through their Baseline implementations, such as convolutional, LSTM, and MLPs.

In our experiment, we use the "MLP" model with a fixed neuron budget of 256 neurons. Using this budget, we test three different architectures: two-layer network, with 128 neurons per layer, four-layer network, with 64 neurons per layer, and eight-layer network, with 32 neurons per layer.

4. Results

We train our 3 agents for 10 million frames (or steps in the environment), which takes approximately 16 hours per agent on our testing computer. We follow [8] to prepare our input data and train our agent using the default hyperparameters [15].

We denote our baseline, shallow and deep agents as four, two, and eight, representing their respective number of layers. We compare the different networks' learning rate, measured by game play high score.

Table 1

Agent	**Avg Score**
Baseline (four)	242
Shallow (two)	247
Deep (eight)	250

As Table 1 shows, the agent with the deep 8-layer MLP performed the best, but the outcome is relatively similar across all agents.

Figure 4

As Figure 4 shows, all agents appear to learn quickly within the first 200,000 frames before leveling off around 250-to-260 in-game points for the next 800,000 frames of training. The baseline and deep agents appear to be exploring different policies around 700,000 and 900,000 frames, respectively.

Unfortunately, during the 10 million frames of training, the agents were unable to discover a superior strategy enabling them to advance to more challenging levels in the game.

5. Conclusion

We show that under a fixed neuron budget, deeper MLPs are able to achieve a higher score on Space Breakers compared to our baseline and shallow networks.

Although we note that performance and learning rates for all our agents are relatively similar after being trained for 10 million frames, we think it would be interesting to extend our experiment and evaluate neural network performance after being trained on more frames (e.g., 100 million frames). Additionally, we would like to run more experiments using a full hyperparameter sweep and different Atari levels.

References

1. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T.P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. ICML.
2. Wang, Z., Freitas, N.D., & Lanctot, M. (2016). Dueling Network Architectures for Deep Reinforcement Learning. ICML.
3. Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized Experience Replay. CoRR, abs/1511.05952.
4. Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., & Freitas, N.D. (2016). Sample Efficient Actor-Critic with Experience Replay. CoRR, abs/1611.01224.
5. Lira, A.S., Machado, M.C., & Bowling, M.H. (2018). Generalization and Regularization in DQN. CoRR, abs/1810.00123.
6. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. CoRR, abs/1707.06347.
7. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T.P., Simonyan, K., & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science, 362, 1140-1144.
8. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M.A. (2013). Playing Atari with Deep Reinforcement Learning. CoRR, abs/1312.5602.
9. Babaeizadeh, M., Frosio, I., Tyree, S., Clemons, J., & Kautz, J. (2016). Reinforcement Learning through Asynchronous Advantage Actor-Critic on a GPU.
10. Hessel, M., Modayil, J., Hasselt, H.V., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M.G., & Silver, D. (2018). Rainbow: Combining Improvements in Deep Reinforcement Learning. AAAI.
11. Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., & Kavukcuoglu, K. (2018). IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. ICML.
12. Gruslys, A., Azar, M.G., Bellemare, M.G., & Munos, R. (2017). The Reactor: A Sample-Efficient Actor-Critic Architecture. CoRR, abs/1704.04651.
13. Kanhabua, N., Ren, H., & Moeslund, T.B. (2016). Learning Dynamic Classes of Events using Stacked Multilayer Perceptron Networks. CoRR, abs/1606.07219.
14. Braylan, Alex & Hollenbeck, Mark & Meyerson, Elliot & Miikkulainen, Risto. (2015). Frame Skip Is a Powerful Parameter for Learning to Play Atari.
15. Openai. (2019, January 31). Openai/baselines. Retrieved from https://github.com/openai/baselines
16. Sutton, R. S., Barto, A. G. (2018). Reinforcement Learning: An Introduction. The MIT Press.