Project Part 4
736 Neural Networks and Machine Learning
Nic Manoogian
Robert Bond III
Zach Lauzon

Executive Summary

During the course of this project, we've investigated two methods of training an agent to play *Super Mario Bros.* using reinforcement learning. The first, *Neuroevolution of Augmenting Topologies (NEAT)* (Stanley & Miikkulainen, 2002), was effectively employed to create an agent which is capable of beating several levels of the game. The second, *Deep Q Learning / Deep Q Network (DQN)* (Mnih et al., 2015), was employed to yield a slightly less-performant agent but with a novel style of play. Specifically, the DQN agent does not beat as many levels as the DQN agent, but demonstrates "skill" by performing multi-frame techniques.

Requirements and Specification

During the course of this paper, we will summarize, analyze, and compare the results of our experiments conducted on the NEAT and DQN agents. Such comparisons will include each algorithm's,

- performance in terms of levels beaten,
- resource consumption (CPU, GPU, Main Memory, and GPU Memory),
- and projected growth of performance over time.

Further, we will outline some of the challenges associated with empirically comparing these agents. And lastly, we will provide our own impressions of each agent using subjective measures such as "skill" and "style-of-play."

Experiments

While exploring *NEAT* and *DQN*, we conducted a series of experiments to test the hyper-parameters of each algorithm.

NEAT Experiment Overview

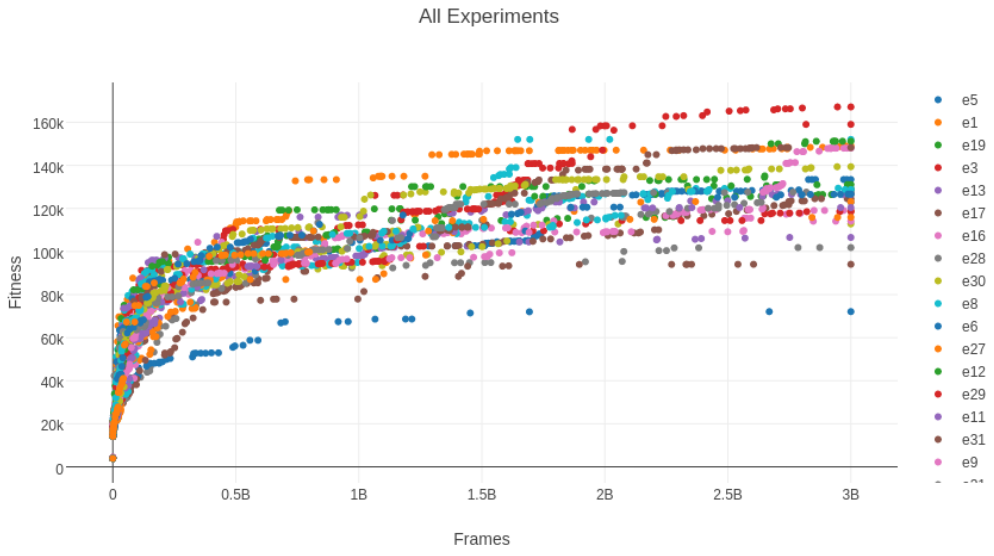
Recall, we generated 32 experiments (named $e0$ through $e31$) by combining all possible high and low values for the selected NEAT hyper-parameters. The list of hyper-parameters we tested are as follows: *Population, Max Stagnation, Add Link Chance, Add Node Chance, and Step Size.* For a breakdown of each experiment and the results we achieved, refer to our Project 2 paper's *Experiments* and *Results* section.

DQN Experiment Overview

Due to limited computation resources, we conducted fewer experiments on DQN. Recall, we generated 4 experiments (named $e0$ through $e3$) by testing different applications of the following hyper-parameters: *Discount Factor, Minibatch Size, Running Time, and Color.* For more detail on these hyper-parameters and their implications, see our Project 3 paper's *Experiments* and *Results* section.

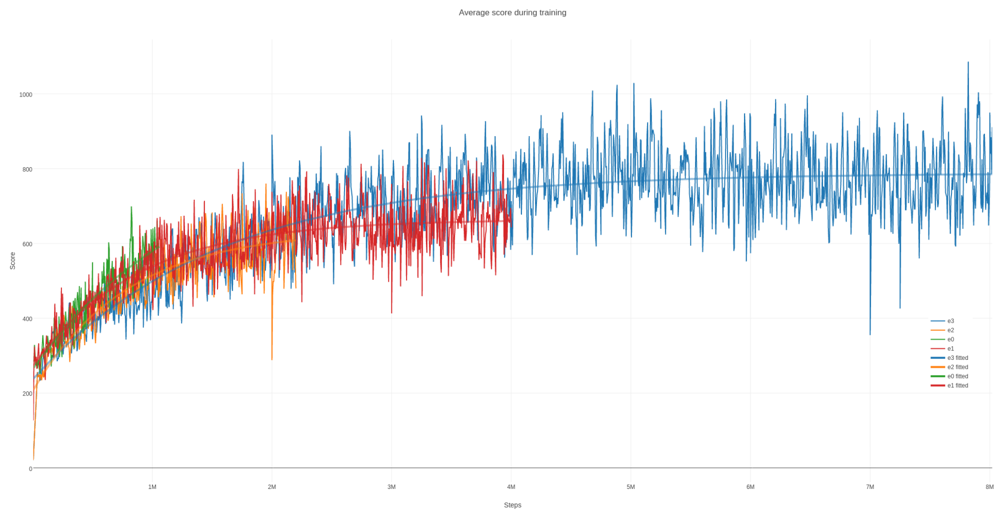
Results Analysis

NEAT Analysis Overview



In our Project 2 paper, we outline the specific hyper-parameters that were tested during our NEAT research. The figure above illustrates our NEAT agent's growth during training. As we see here, the different combinations of hyper-parameters changed the performance of the agent but the shape of the function appears to be the same in all cases. Indeed, all of our generated NEAT agents converge asymptotically to a fitness value between 40-thousand and 160-thousand. Our best NEAT agent beats 11 of the 22 tested levels. More specific findings are reported in our Project 2 paper.

DQN Analysis Overview



In our Project 3 paper, we again hyper-parameters that were tested during our DQN research. The figure above illustrates the DQN agent's growth during training. Similar to our NEAT agent, the different iterations of our DQN agent converge asymptotically with different performance scores. Our best DQN agent beat 2 out of the 22 tested levels. Again, more specific findings are reported in our Project 3 paper.

Comparison

It is difficult to compare the DQN and NEAT agents due to the type of reinforcement that we administer to each. The NEAT agent is evaluated (and thus rewarded) using the fitness function defined formally in the Project 2 paper. This function rewarded the agent for increasing its X position in the level and for beating the level while we punish the agent (a negative reward) for being defeated and spending time in the level. This fitness function was selected to encourage fast movement through the level. Without the negative rewards, NEAT algorithm would be very slow to create players that quickly move through the game.

The DQN algorithm does not need the same penalties to perform well. We simply provide the DQN agent with a reward for right-ward X movement and let the algorithm do the rest. The DQN agent is implicitly penalized for defeat because it can no longer move right.

Because these two agents are scored differently, we defined a new scoring function S which was formally defined in the Project 3 paper as the player's X position in the level. Our best NEAT agent reaches an average S score of 2136. Our best DQN agent reaches an average S score of 1068. However, it should be noted that the NEAT agent was allowed to train for far longer than the DQN agent.

Not captured by the S-score, the DQN agent also exhibits a few interesting patterns of behavior, that in many ways, are skillful. In particular, the agent:

- seems to "wait" to allow enemies to pass through critical areas of the level,
- will backtrack though parts of a level if no progress can be made on the agent's current path,
- and seems to learn techniques such as jumping on Koopas and kicking their shells to clear lines of enemies.

It should be noted that these behaviors were never exhibited by our NEAT agent. In other words, our NEAT agents appear to achieve goals "with luck" while the DQN agent appears to achieve goals "with skill." It is our overall sense that the DQN agent, although not currently the highest-scoring, has far greater potential than NEAT for this very reason.

Resource Consumption

The following table summarizes the resources consumed during training by both NEAT and DQN:

Method	CPU Utilization	RAM	GPU Utilization	GPU Memory
NEAT	100%	100MB	None	None
DQN	100%	1.6GB	80%	700MB

It should be noted that NEAT utilizes a cluster of computers, whereas DQN runs on a single machine.

During testing, DQN uses the same amount of resources, while NEAT uses 25% CPU and 5MB of RAM. It is clear that the DQN approach requires far more resources than NEAT, but we find the use reasonable given the nature of image-based problems. DQN's RAM usage is likely due to the *replay history* feature of deep Q learning.

Final Impressions

While our best NEAT agent does currently outperform our best DQN agent, there are some significant differences between the agents' styles of play.

Consider a particular part of *level 2-1*, depicted at `https://imgur.com/a/tKMJM`. As shown, the NEAT agent sprints directly through a tight area of the world; just barely avoiding enemies in its path. In the same part of the level, the DQN agent actually backtracks to avoid and attack these enemies. While a certain "style-of-play" cannot be quantified empirically, it is clear to a human observer that there is a distinct type of intelligence associated with the DQN agent in contrast to the NEAT agent. It is our impression that the DQN agent has more potential and capacity to improve with more training time.

The novel behaviors exhibited by DQN best reflect our goals which we described originally in the Project 1 paper. We were looking for a machine learning agent capable of playing *Super Mario Bros.* in a new way; DQN may very well be the solution that we set out to find.

References

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Hassabis, D. (2015, Feb 26). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529-533. Retrieved from `http://dx.doi.org/10.1038/nature14236` (Letter)

Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, *10*(2), 99-127. Retrieved from `http://nn.cs.utexas.edu/?stanley:ec02`