Università degli studi di
Milano-Bicocca

Decision Models

Final Project

Hit or miss: ANN agent and RL to play Breakout

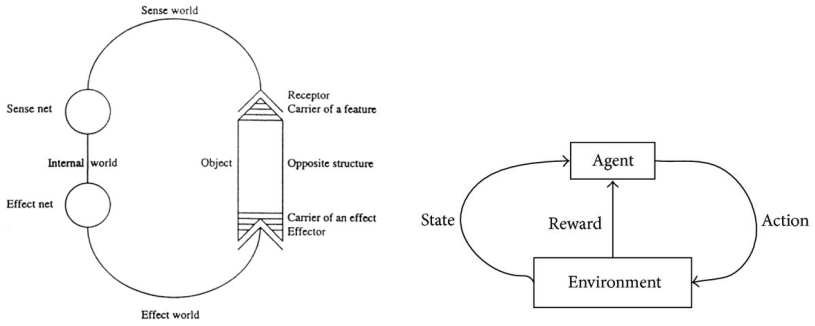
Authors:
Ennio Antonio Guzman
Louis Fabrice Tshimanga
September 15, 2019

Abstract

Reinforcement Learning and Neural Networks Learning are two frameworks in computer science related to biology, ethology and games. In this work an agent was developed through reward and punishment after trial and error, to play the videogame of Atari Breakout at near human level with no prior knowledge, behaving according to a simple neural net with a sensory layer, a processing one and an effector neuron. A multi-staged training protocol was shown to be effective and quick in obtaining peak performances for the shallow neural network, breaking most records of other non-deep learning approaches.

1 Introduction

Reinforcement Learning and Artificial Neural Networks are two blooming fields in computer science and artificial intelligence research, related to biological and ethological models, and connected in recent applications. Both the theoretical frameworks upon which modern results are developed are apparently simple and they were sketched in the 50s and revised in the 80s of last century, but only in recent years they have been jointly applied and technically improved. One of the breakthroughs has been DeepMind's agent for Atari games [1], a Deep Neural Network-fuelled automaton capable of playing vintage videogames, after training itself to explore each game's action space. It recorded outstanding results, paving the way for a new generation of agents able to break human records also on games of abstract mechanics like Go, bringing specific levels of competiotions between machines now out of reach for human experts. The central idea in Reinforcement Learning (RL) is condensated in Thorndike's Law of Effect, stating that in the animal, for repeating situations (other things being equal) actions that are accompanied or closely followed by satisfaction will be more firmly connected and likely to recur with the situation, while actions accompanied or closely followed by discomfort undergo the opposite process [2]. Modern approaches to RL consist of an agent trying less and less random actions in a training environment, acquiring varying information on the state[space] of the outside world as of the self, together with a numerical quantity representing positive and negative rewards subsequent to the actions performed in the many different settings. A wide range of complex behaviours arise from the optimization of rewards. Famous examples are the Q-Learning and SARSA algorithms.



(a) A biological model of perception, the effect circle generating "mental models", from J. Von Uexkull

(b) An outline of the Reinforcement Cycle, where the agent acts on and sense the world, receiving reward

Figure 1: Theoretical biology models and RL, a comparison

Artificial Neural Networks (ANNs) are biologically inspired mathematical models, where each neuron outputs a function of others neurons' output. ANNs are considered feature extraction machines, capable of abstracting patterns layer by layer. They are problematic insofar they are considered black box systems, opaque in the effect of each neuron and parameter in the behaviour of the whole system, but they're powerful approximators of very general functions.

The present work follows the unifying framework popularized by DeepMind's Deep-Q-Learning (DQL) system, and creates a lighter, simpler ANN agent to play Atari Breakout at human level, i.e. better than classic RL approaches, and with less computational and time resources than DQL. The sensory-to-effector neural network can be viewed as a classifier trying at each step to tag the observed game situation as one in which it should move either right or left. This classifier framework is at the core of the backpropagation algorithm updating the weights according to the misclassification errors and the rewards received.

2 Background

DeepMind is a team of researchers working in artificial intelligence and machine learning, brought to prominence in 2013 with a research paper exemplif-

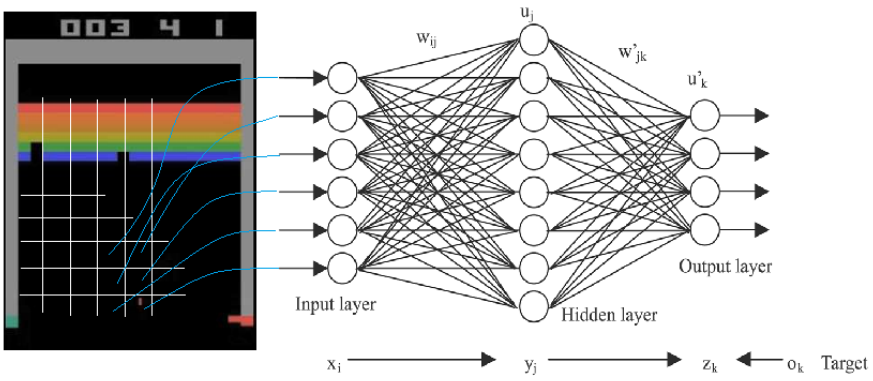


Figure 2: A display of the screen as sensory input, processed by a hidden layer and then converted into classification/action output (please note that the actual ANN has only 1 output neuron

ing Deep Reinforcement Learning efficacy in training an agent to play 7 Atari vintage videogames [1], one of which is Breakout, the target of the present studies. The architecture implemented by the team is a Convolutional Neural Network (CNN), a kind of net inspired by visual processing in the animal and hence reknowned for its performance in visual classification tasks [3]. In fact, the Arcade Learning Environment (ALE) replicating the Atari console offers proper image data to the agent, which processes features from the visual input and adapts action policies based on video patterns, according to the rewards assigned. In recent years Elon Musk promoted the establishment of a common environment for developers [4], a collection of games and problems to be accessed through a unique interface for Python programs. The efforts led to the gym toolkit for developing and comparing reinforcement learning algorithms, compatible with any numerical computation library, and accessible for interaction through standardized and documented commands and protocols. The game of Breakout consists of a brickwall above a moving paddle, and a ball bouncing on the paddle and destroying bricks in the wall.

Bricks destruction is the event improving the score, and thus the switch for assigning a reward (equal to the score improvement by default). Bounce after bounce the ball accelerates, with implications for the best strategy to follow and hit (or miss) the ball. The basic gym-ALE environment allows 4 actions: calling for the ball; do nothing; move right; move left. Performing the action let the agent receive the state of the screen, the score, whether the game is finished, and informations regarding the lives (starting from 5 and decreased at every miss)

3 Datasets

The present study did not require any dataset in the classical sense. The computers running the programs had a built-in Windows system (Windows 10), upon which ubuntu's application was installed to properly build gym and its environments. The programs were written nonetheless on Jupyter notebooks accessed by browser open on Windows, while the kernel ran in the unix system (see Appendix). The environment chosen for the main training of the agent was *BreakoutDeterministic-v4*, with the name referring to the type of game, the deterministic performance of the action chosen (while the simple version has a 25% chance of repeating last action regardless of the choice from the agent), and the sampling nature of the image rendered (i.e. 1 every 4 frames). Every observation of the environment is a colored image saved as a *210x160x3* array. The Python libraries essential to the study were gym and numpy, while matplotlib, pickle and time were useful additions for plotting and managing purposes.

4 The Methodological Approach

The hypothesis tested questioned the feasibility of a computationally affordable and consistent ANN approach to RL, through a 3-staged training.

4.1 Preprocessing

The information gained from a single frame is both insufficient and redundant for the scopes of the agent. Each frame was firstly cropped on the first axis, in a *35:195* slice containing the playing field without the section of the screen showing the score and lives (irrelevant to the agent, as long as it gets a signed

reward for scoring or losing life, which is done extracting the information through the gym interfacing function). The image arrays were then sampled and decolorized before being fed to the ANN input layer. The sampling rate of 2 on the geometric dimensions saved a *80x80x3* array, which was then flattened to gray by setting the RGB dimension to *0*. After the first frame, each preprocessed array was actually used to compute and save in memory only the difference between itself and its predecessor. This lightens the impact of the actual processing and adds time-dependant information missing in the frame per se.

4.2 ANN Architecture, Dynamics and Parameters

The network is a 1-hidden layer perceptron, with a structure of:

- an input layer of *80x80=6400* sensory neurons;
- a hidden layer of *200* processing neurons;
- an output layer of a single neuron, the classifier or effector which translates the processing in a moving probability;

The hidden neurons have a *ReLU* activation function, while the output neuron has a *sigmoid* activation function, apt to express a classic normalized probability. The learning rate of the ANN was set to *1e-4* for the first and second training stages, and to *1e-5* for the third one, with batches of size equal to *10* complete games. An extra-training was performed with the smallest learning rate and batches of *62*. The backpropagation algorithm used was RMSProp, as implemented in [5].

4.3 RL Salience, Game Mechanics and Rewards

RL requires a system of rewards of positive (negative) sign, given to the agent at the completion of (un)desired actions or sequences of actions, which will be valued by said parameter. In the context of Breakout, the goal of the game would be to demolish the rainbow wall above the paddle loosing as few lives as possible in the process, i.e. hitting the ball whenever possible. A random agent might be intended to learn how to hit the ball first, then how to hit the wall in the most proficuous way: higher bricks give higher scores and an expert could carve a tunnel in the wall and have the ball repeatedly

hit it from above. However, the interface with the ALE only assigns positive rewards equal to the points scored per hit. This means there's no negative reward by default, furthermore the positive reward traditionally discounted would value more the few steps done after hitting the ball, rather than those immediately before. The training pipeline developed was designed to overcome this problem, without implementing non-linear discount functions. One reward system assigned a reward of *+1* to every action that obtained a positive score, regardless of the magnitude, in the pursuit of learning to hit. A second system rewarded every step of survival (i.e. every step in which a life wasn't lost) a *+0.01* reward, so that the close-to-average amount of steps per game of a naïve or random agent might provide a total reward close to 0, given a *-1* reward for every life lost. The goal of this system was letting the agent identify the specific bad moves, yet inevitable on the long run, that bring to missing the ball. The third rewarding system assigned a *+2* for every first hit of a possible combo, and multiplied the eventual following hits by the length of the ongoing combo (i.e. the third hit in a row brings *2*3 = 6*, the *n*-th brings *2*n*). The loss of a life, in this system, was associated with either a *-1*, or a *-2* in case of the breaking of a combo. The objective was to promote the agent fluent and consistent playing style and partially account for the rising rewards obtained for multiple hits straight.

4.4 The Training Pipeline

As mentioned above, the process of learning went through three different rewarding systems. A first stage focused on generally hitting the ball and the wall, a second stage promoted survival and non-avoidance of the ball, and the last stage was set to boost all aspects of the playing style, regardless of nominal score associated to the actions. The first stage performed *1000* iterations, updating formerly random initialized weights every 10 episodes. The second stage was applied with identical parameters upon the trained weights. Finally, *200* iterations with new batch size and learning rate (see Section 4.2) were performed. A most promising agent was then selected and further trained for 16h30, on a 3.6GHz processor, in batches of *64* and learning rate of *1e-5* to extend the capabilities of the agent and the net architecture itself.

5 Results and Evaluation

In the next figures, results from 10 complete training processes will be presented. Tentative protocols and unsuccessful trials were aborted, respectively bringing results non significantly different from random agents and RL results from the literature preceding DeepMind's approach. These same results are summarized in the Figures 3,4 and 5, openly accessible at [6] and [1]:

Figure 3: excerpt from Table 4 in Bellemare, Naddaf, Veness, & Bowling

Game	Basic	BASS	DISCO	LSH	RAM	Random	Const	Perturb
Breakout	3.3	**5.2**	3.9	2.5	4.0	1.5	3.0	2.9

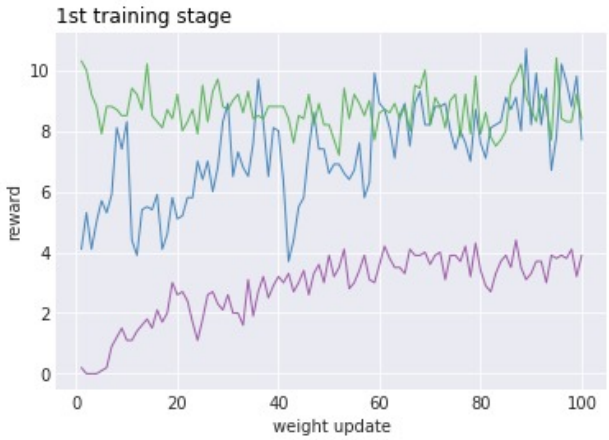
Figure 4: excerpt from Table 5 in Bellemare, Naddaf, Veness, & Bowling

Game	Full Tree	UCT	Best Learner	Best Baseline
Breakout	1.1	**364.4**	5.2	3.0

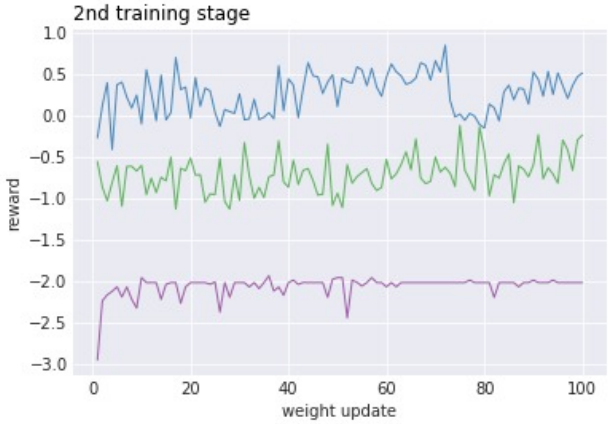
Figure 5: excerpt from Table 4 in Minh & al., of DeepMind

	Breakout
Random	1.2
Sarsa [3]	5.2
Contingency [4]	6
DQN	**168**
Human	31

Figure 6 shows the 3 different stages for 3 agents, the overall best and worst, and an average player from the agent's pool



(a) stage 1



(b) stage 2

(c) stage 3

Figure 6: Worse, Best and average scorers in 3 stages of training

A bird's eye view of the three-staged training process in 10 trials from 10 different random initializations for the ANN architecture is shown in Figure 6, where the different rewarding systems clearly appear as 3 phases of different magnitude for updates 1-100, 100-200, and 200-220 (please, note that an update was performed every 10 iterations of a complete game).

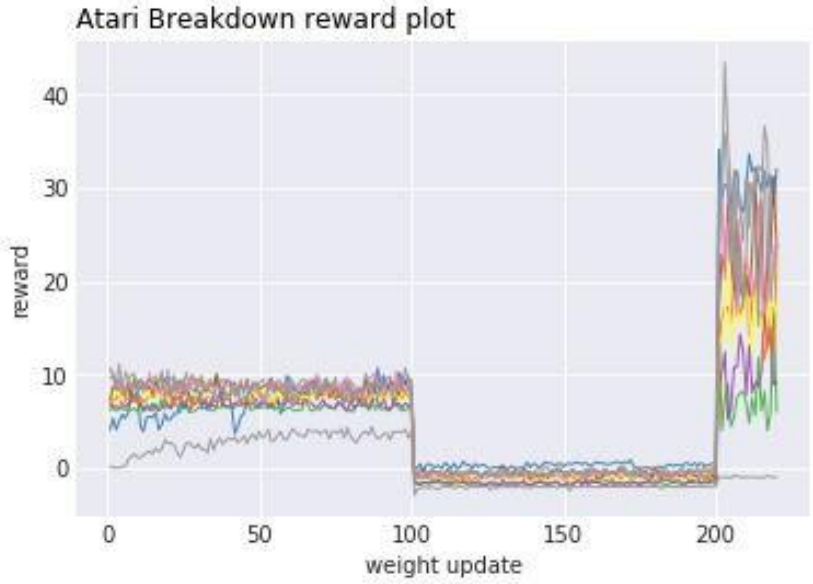


Figure 7: Reward history for 10 agents

The best player agent obtained an average of *22.0267* points in 150 games, and was further trained to reach a mean of *28.0533*, and a median of *30*, also its maximum, scoring at least *7* points in the worst game.

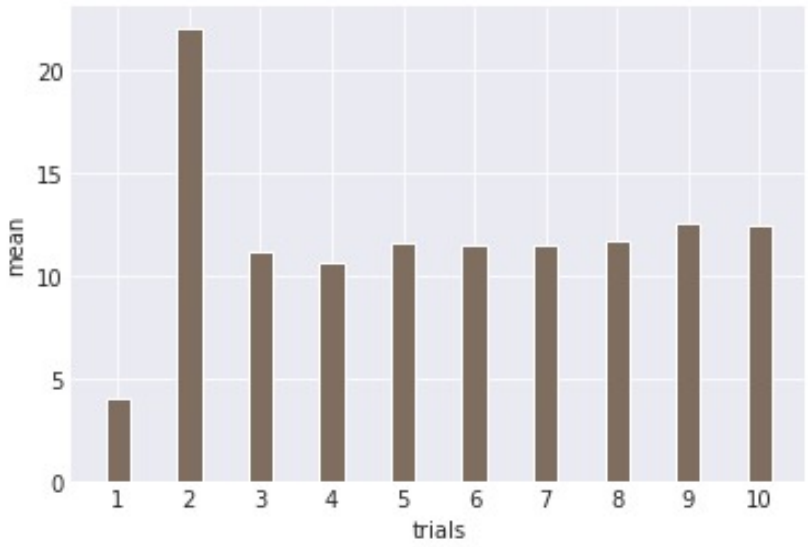


Figure 8: Means of the scores after the training

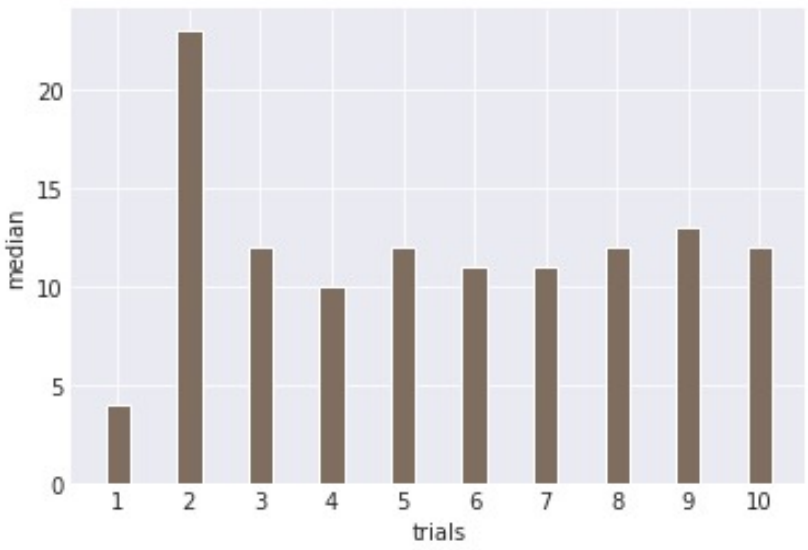


Figure 9: Medians of the scores after the training

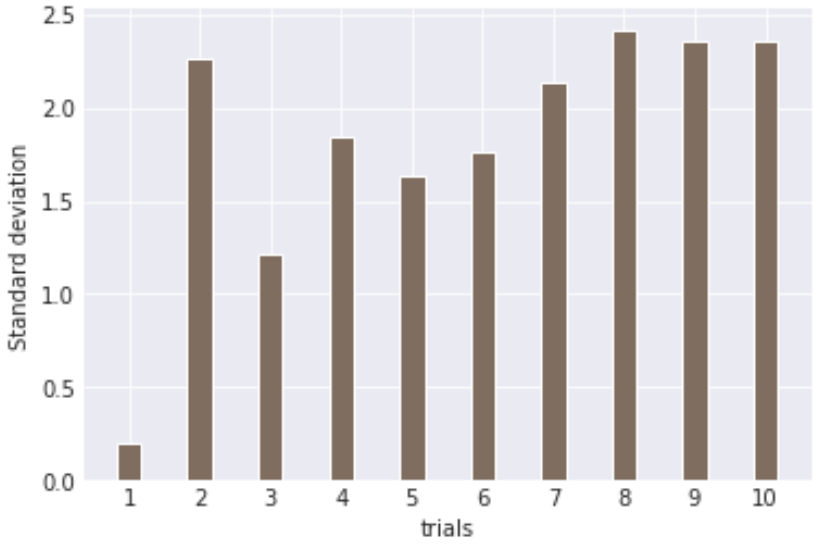


Figure 10: Standard deviations of the scores after the training

After every complete training, mean, median and standard deviation of the scores over 150 games were calculated. Data shown in Figure 8, 9 and 10.

6 Discussion

The limited computational and temporal resources forced the research for a quick and efficient neural network architecture and training protocol. Explorative combinations of hyperparameters (included no. of iterations) were attempted, and observational notes taken, with regards to the agents playing style. The progressively accelerating ball, not rendered in the printout of the single iterations, makes it impossible for a non-omniscent and sensory-imperfect agent to follow the horizontal ball movement and hit it whenever possible. Furthermore, this isn't necessarily a good scoring practice, since the layered brickwall awards higher scores for higher layer bricks, so that hitting the same zone is the most effective strategy. The best scorers in the study were indeed capable of playing on the corners of the game field, hitting the ball a handful times in a row, with a focus on the opposite corner of the brickwall and the above corner, often with a risky but rewarding move of waiting the last call to hit the ball with the side of the paddle, for an early

bounce on the same side and the opposite side (eventually already carved by past hits) as a target. The consistency of this strategy across the sensibly good player agents, together with the worsening of performances after eccessive iterations, made it plausible that this strategy pertains to peculiar regions of the cost function's surface of the ANN, and might suggest that the best performances obtained might be at least close to the best absolute performances the present architecture could bring. A wide metaheuristic approach could test difference sets of hyperparameters and of the phase space of the system, but the convenience of such approach should be tested against brute force, or better, more complex but less supervised studies, as in the now famous Deep-Q-Learning approach. It should be noted that the learning and testing environment in use, the only unpredictability for the agent's point of view consisted in the stochastic nature of the action choice: the output neuron computed what was a probability of moving right, compared at each step to a random number; the non-deterministic environment instead overrides the action of choice with a probability of 25% to just repeat the last action, bringing a further error component which would unbalance the reward counting and discounting at each step.

7 Conclusions

The study under scrutiny presented an original application of classic and simple ANNs as a means for an agent to explore and success in a RL problem. The results were very favorable both on the resources account, compared to similar approaches used as benchmark and didactic examples in Deep-Q-Learning introductions, and on the matter of points scored, since the agents consistently came close to average human performance and at best reached human expert levels, developing on the run an apparently sensible strategy and tactics.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXiv:1312.5602 [cs]*, Dec. 2013, arXiv: 1312.5602. [Online]. Available: http://arxiv.org/abs/1312.5602

[2] S. McLeod, "Edward Thorndike: The Law of Effect," p. 2, 2018. [Online]. Available: https://www.simplypsychology.org/edward-thorndike.html

[3] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, Apr. 1980. [Online]. Available: http://link.springer.com/10.1007/BF00344251

[4] S. Shead, "Elon Musk's $1 billion AI company launches a 'gym' where developers train their computers," *Business Insider*, p. 1, Apr. 2016. [Online]. Available: https://www.businessinsider.com/openai-has-launched-a-gym-where-developers-can-train-their-computers-2016-4?IR=T

[5] S. Ruder, "An overview of gradient descent optimization algorithms," Jan. 2016. [Online]. Available: http://ruder.io/optimizing-gradient-descent/index.html

[6] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The Arcade Learning Environment: An Evaluation Platform for General Agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, Jun. 2013, arXiv: 1207.4708. [Online]. Available: http://arxiv.org/abs/1207.4708