Chapter 3

**AI: Convolutional Q-Learning**

**Game of DOOM**

Deep Convolutional Q-Learning Intuition

Eligibility Trace

**3.1 Objectives**

1. **Deep Convolutional Q-Learning Intuition:** We'll talk about deep convolutional Q-Learning and the intuition behind it.

* Why deep convolutional Q-Learning is so powerful than deep Q-Learning.
* How deep Q-Learning is just a basic building block of deep convolutional Q-Learning.

1. **Eligibility Trace (N-step Q-learning):** Here we'll talk about Eligibility-Trace (N-step Q-learning) a very powerful addition to the whole concept of deep Q-learning.

* And we'll talk about the intuition behind that.
* So that we can understand, how the agent can actually handle some complex environments and navigate them successfully.
* Before dive into above topics, it's good review CNN to understand better
* How images are processed by neural networks in order to look for features and what's the
* whole convolutional layers,
* pooling Layers,
* the flattening layers are and how all those works
* In order to understand how to come up with some parameters that describe the environment (or that describe that image) and therefore we're going to be using those as inputs into our *neural networks* inside of a vector.

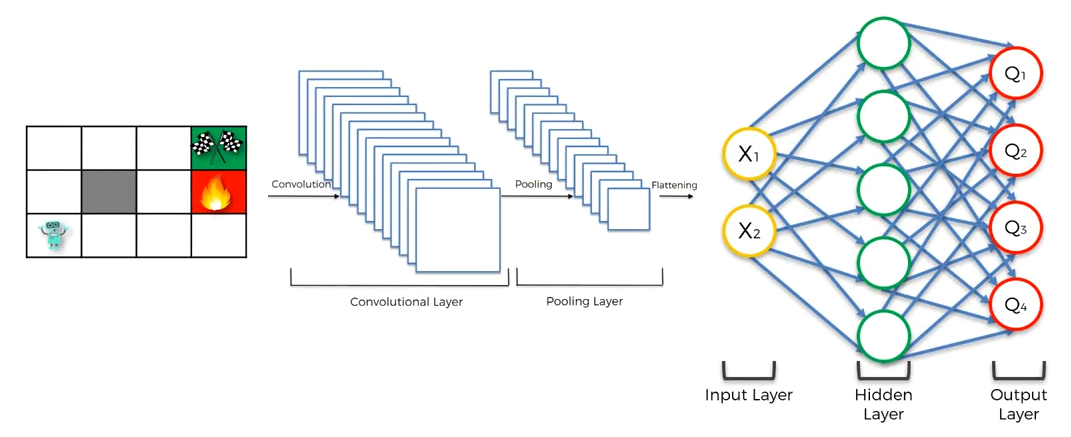
**3.2 Deep Convolutional Q-Learning**

Previously we talked about Deep Q-learning. We had an agent in an environment where and we had a vector *describing that environment* then the vector was fed into a Neural Network and at the end we got the Predicted Q-values as our output.

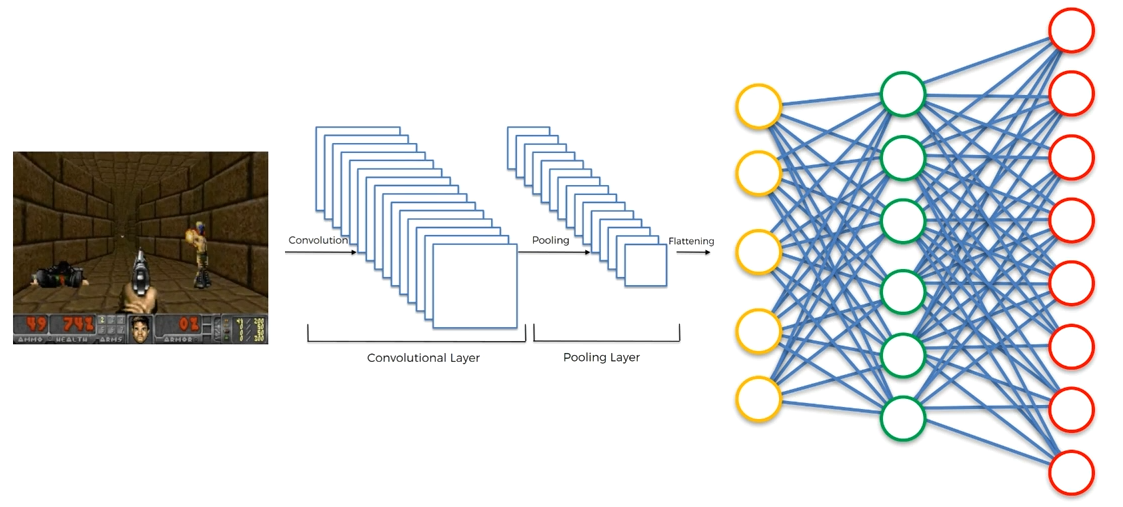
And also there were different parts:

* Learning
* Acting
* Experience Replay
* Action Selection Policies

|  |  |
| --- | --- |
| * The input vector: But the key point for all of this is: how do represent the actual Environment to our Neural Network as a Vector (input vectors). (Remember, we represented our Maze-example using coordinates.) * In the real life we cannot do that easily. In real life we use our senses, our eyes and understand what's going on in an environment. We are not getting the information about the environment as a vector. * Here CNN come into play, we are adding visual recognition to our agent. So that it can see an environment (A maze or a video-game) and take an action. * Since most of the information that we're getting about the world around us comes through our sight. * And that is why we're going to change that little arrow with a Convolutional Neural Network (CNN). |  |



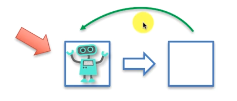
* The agent is now looking at the image of the Maze. So it is an image of the environment. And the agent is actually looking (seeing) at the environment.
* Let's say he's playing this on a computer and he can see this environment whatever a human would see.
* When the agent seeing an image, it goes through a convulsion layer.
* In CNN we've got the convulsion layer for the convolutional operation; then there are also pooling-layer; flattening-layer;
* After the flattening is complete, we have inputs which go into the Deep-Neural-Network and gives the output (action).
* This is a way more realistic because the agent has to use their sights and has to process images of the environment (just like humans).
* Moreover, it allows us to process much more complex environments.
* For instance playing a video game: Doom or other games like IGI. Instead of just getting a vector of information for a specific environment, our agent has now vision, which make them able to interact with any environment.
* The agent will actually see this exact picture (the pixels) as a human can see.
* **That's the beauty of it:** the agent can now operate any kind of environment that you attach it to because as long as there's a visual representation of environment, it's already got the whole structure is ready to process that.
* So that's what Deep-Convolutional-Q-learning is all about.
* We're adding convolution Layers into our agents brain and we're making it even more complex and therefore we can able to solve even more complex challenges.

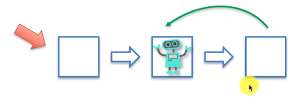


**3.3 Eligibility Trace (N-step Q-Learning)**

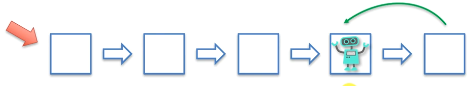
Here we'll discuss about Eligibility Trace or N-step Q-Learning.

* Here we've got two agents and they're navigating the same environment and we're going to see how these two agents work.
* *First one* is going to work without eligibility trace.
* *Second one* is going to work with eligibility trace and hopefully we'll see why the *second one* is going to be so much *more* *powerful* than the *first one*.

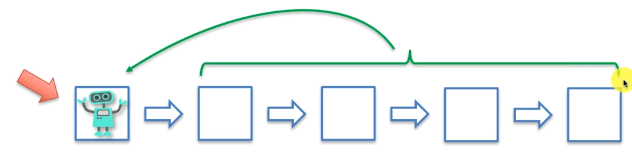






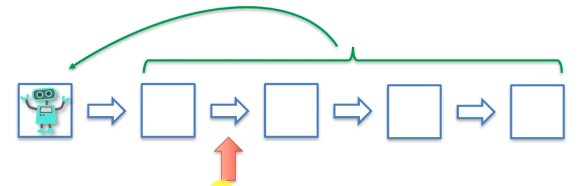


* Above represents the DEEP Q-LEARNING we've learned so far.
* Where the agent is going to take a step or an action to move into a new state, gets a certain reward, put that through algorithm, update the neural network, then takes a new step.
* This process is repeated until it reaches to the finish-line.
* We're going to get some quite good results from this approach.
* Notice that, the first-agent takes single steps at a time. It cannot keep trace of "group-of-actions" that takes it to the finish-line.
* Now the agent-2 going to navigate the *same environment* and it's going to use eligibility trace.



* He's going to take **N-steps** (in this figure four steps) is and after taking these steps he'll **calculate** the **total rewards** that he got from those steps and
* Then he will put it through his network that's governing the decision making process and then the neural network will learn from that.
* Then he decides whether those steps "N-steps" were good steps or not.
* By taking **N-steps** the second agent actually knows what's at the end.
* The first-agent takes a single-step and decides whether this *step is good or not* he's only looking at single-reward that he's getting from a single-step. And he's only guided by these rewards.
* Whereas for the second-agent, if he takes **N-steps** to get to the finish line then he consider that **combination of steps** was good.
* Or, if he ended up in the fire-pit (*car didn't get to the finish line* or *car crossed the sand-wall-border* or *losing in a video game*) and it got the final penalty. Then he decides that the **whole combination of steps is bad**.
* N-step-Q-learning: The benefit of ***taking N-steps*** instead of ***single step*** is that

1. The agent will get more information.
2. For the **earlier steps** of that group the agent will have **more insights**, like ***"what's the outcome of after taking next 4 or 5 steps from the current step".***

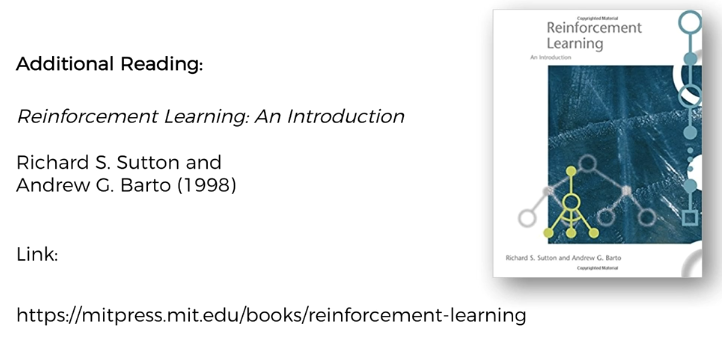


1. It keeps the trace: That's why it's called Eligibility Trace because during this process the agent *not only looks at the* cumulative reward and then the cumulative loss which then used to update the NN, but also he keeps the **trace of eligibility**.

* This **Eligibility Trace** is kept in the algorithm which ***keeps trace of group-of-steps*** that is ***eligible for a punishment*** ***or reward***.
* For example if we get a punishment (negative reward) then the algorithm will keep trace which of *these steps* is most likely to be eligible for that punishment.
* So the agent not only gets the **whole pattern or combination of steps** but also keep a trace of eligibility which *steps are we going to update if we get a reward*.
* Eligibility trace kept for both positive reward and negative reward.
* The ***algorithm*** helps us to ***keep track*** what actions and ***steps are eligible*** to be ***updated*** based on that ***reward*** that we get. And that's why it's called ***eligibility trace***.

And so that's the basic intuition behind eligibility trace.

* Additional reading 1 : If you'd like to delve further into Eligibility Traces or N-step Q-learning then there is a wonderful book called REINFORCEMENT LEARNING by Richard Sutton and Andrew Barto 1998.
* This is the most common or popular book on Reinforcement Learning.
* The Eligibility Traces discussed in Chapter 7.



* You'll get lots of detail. There are also lots of *pictures* with *intuitive explanations* about *artificial intelligence* and *reinforcement* *learning*.
* ***Forward-Backward eligibility traces*** and also how
* ***Integral temporal difference***
* ***Monte-Carlo methods***
* Link between ***eligibility traces***, ***temporal differences*** & ***Monte-Carlo methods***.
* Additional reading 2: The second reference is Google Deep Mind research paper called asynchronous methods for deep reinforcement learning by .
* We'll discuss it in the next chapter for A3C-model.
* We'll also discuss how they implemented Eligibility Traces in this paper and we're going to be using this for the practical side of things.

