NOTEBOOK

DeepMind Game Bot

Create Snale Game AI

Learn

#1

Introduction

DeepMind Technologies is a British artificial intelligence subsidiary of Alphabet Inc. and research laboratory founded in September 2010. DeepMind was acquired by Google in 2014. The company is based in London, with research centres in Canada, France, and the United States. In 2015, it became a wholly owned subsidiary of Alphabet Inc, Google's parent company.

The start-up was founded by Demis Hassabis, Shane Legg, and Mustafa Suleyman in 2010. Hassabis and Legg first met at University College London's Gatsby Computational Neuroscience Unit.

During one of the interviews, Demis Hassabis said that the start-up began working on artificial intelligence technology by teaching it how to play old games from the seventies and eighties, which are relatively primitive compared to the ones that are available today. Some of those games included Breakout, Pong, and Space Invaders. AI was introduced to one game at a time, without any prior knowledge of its rules. After spending some time learning the game, AI would eventually become an expert in it. "The cognitive processes which the AI goes through are said to be very like those a human who had never seen the game would use to understand and attempt to master it." The goal of the founders is to create a general-purpose AI that can be useful and effective for almost anything.

#2

Achievements

Learn to Walk

Teaching a machine to walk has proven tricky because there are so many variables involved. Companies like former Google subsidiary Boston Dynamics have succeeded in creating programs that tell robots how to walk, but you can't account for all the possible situations. When such a system encounters a new obstacle, it might have no idea how to navigate it.

The team used simulations in a complex world filled with obstacles, but the goal for the AI was simple: Make it as far as possible as fast as possible. The parkour course contained walls, cliffs, hurdles, and tilting floors. The "reward" for the AI drove the simulations to discover new ways to traverse the terrain, and none of the movements were provided programmatically — this is all emergent behavior.

This research shows that complex problems can be solved with very little input from humans. Just offer a learning AI an opportunity to solve the problem, and it can develop surprisingly complex behaviors.

#2

Achievements

AlphaGo - Defeated Go player

Go is considered much more difficult for computers to win than other games such as chess because its much larger branching factor makes it prohibitively difficult to use traditional AI methods such as min-max with alpha-beta pruning, and tree traversal.

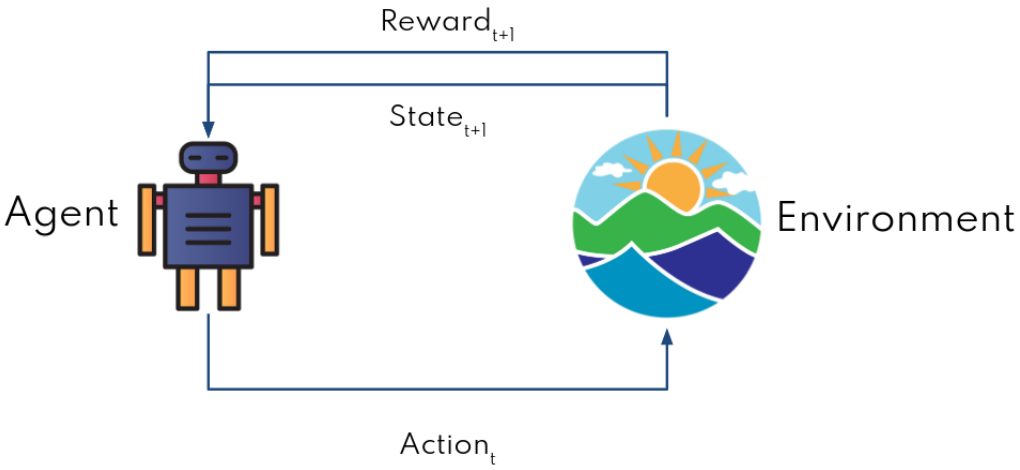
AlphaGo is a computer program that plays the board game Go. In October 2015, the distributed version of AlphaGo defeated the European Go champion Fan Hui, a 2-dan (out of 9 dan possible) professional, five to zero. This was the first time a computer Go program had beaten a professional human player on a full-sized board without handicap. The announcement of the news was delayed until 27 January 2016 to coincide with the publication of a paper in the journal Nature describing the algorithms used.

AlphaGo also won against Lee Sedol, ranked 9-dan, one of the best players at Go.

#3

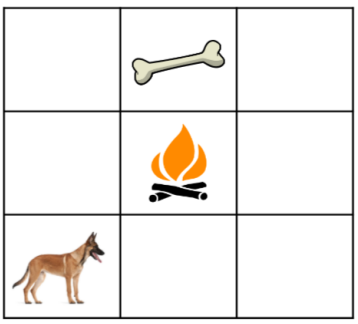
Reinforcement Learning

Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. In this type of learning, we have an agent, which is allowed to interact with the environment. After any action taken by the agent in the environment, the environment will return a reward to the agent, and a new state of the environment.



The agent will explore the whole environment using random moves in the beginning, and will gain some knowledge of the environment. After training, the agent will only make moves that will return high rewards.

Example:
Consider an environment of size 3x3, and the agent is a dog whose goal is to get to the bone and avoid the fire.

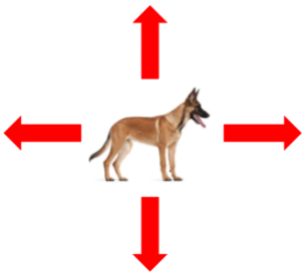


#3

Reinforcement Learning

First, we need to define actions that can be taken by our agent, states in the environment, and rewards.

Action



State

S1	S2	S3
S4	S5	S6
S7	S8	S9

Reward

-10 +10

Our agent, the dog can move up, right, down, or left. There are 9 states, two of them are end states (Fire and Bone). The dog will get a reward of 10 points if he successfully gets to the bone, and a reward of -10 if he falls in the fire.

Q Values, and Q Learning

Q value or Quality value is used to determine the quality of an action at a state. Q value depends on the state and action. A high Q value for an action means that the end reward will be high if we take that action.

$$Q(State, Action) = Q\ Value$$

#3

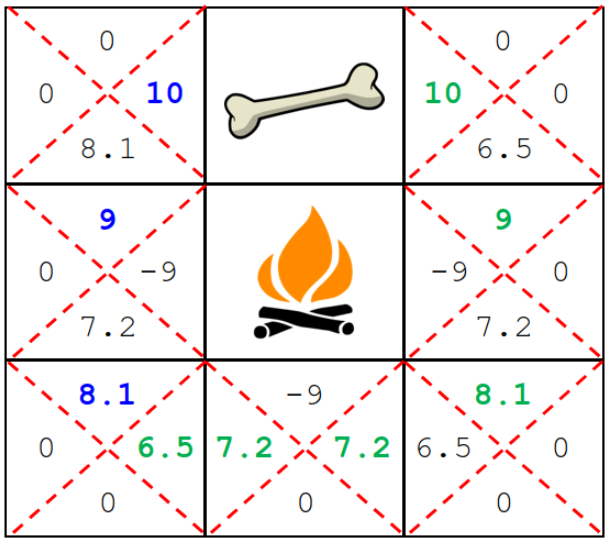
Reinforcement Learning

We need to learn these Q values for all the state-action pairs. The algorithm which is used to learn these values is called Q Learning. The skeleton learning equation in Q Learning is as follows:

To get the new Q value for a state-action pair, we add the old value with the reward of that action, and the max Q value of the next state.

We also use discounting reward which means that our agent wants to reward as soon as possible. Thus our end reward value of +10 will decrease with every step away from the end state.

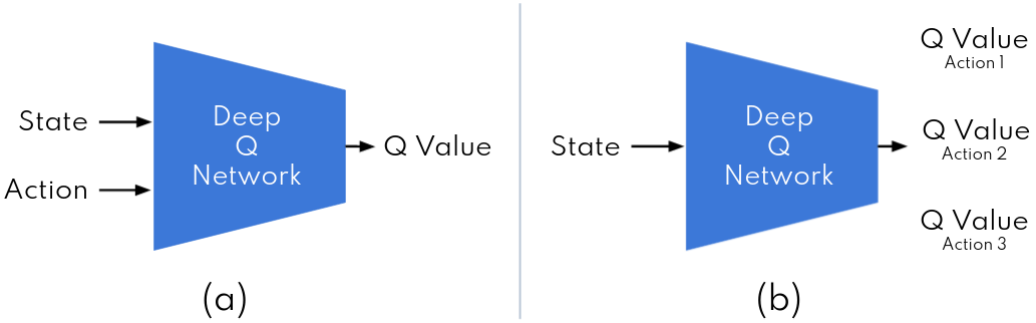
After using Q Learning, and applying a discount of reward we will get Q values as showing in the picture below. Now the dog will always take the shortest route to the bone.



#4

Deep Q Network

For games with many states, we use a neural network to get Q Value for a state-action pair. This network is known as Deep Q Network or DQN. We can either of the following architecture of DQN, but the second one is preferred.



If we use first architecture, then for at a given state, we have to iterate through DQN n times, where n is the number of actions available for the agent.

If we use second architecture, then we have to iterate through DQN only one time, and we will get Q Values for all actions, among them we will choose the action with the highest Q value.

We want to iterate through DQN only one time because iterating through a Neural Network is a computationally expensive step, and it takes time to get output. Thus less iteration is preferred.

#5

Snake Game

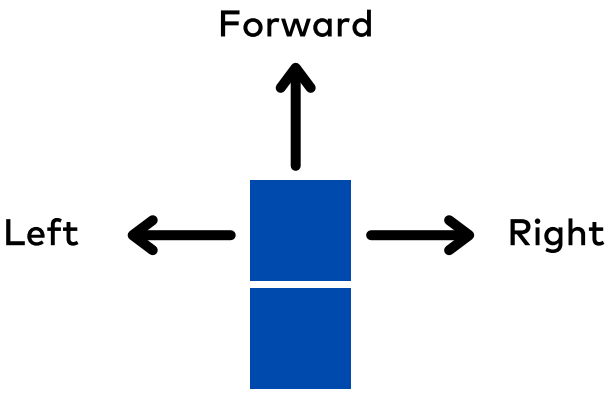
We have to create a bot for the Snake game. In the game of snake, the snake has to eat food to grow, and avoid hitting walls and eating itself. First, we have to define the actions, state, and rewards.

Snake Body

Food

<u>Actions</u>

Snake can only move left, forward, or right. Snake can't move backward.



<u>State</u>

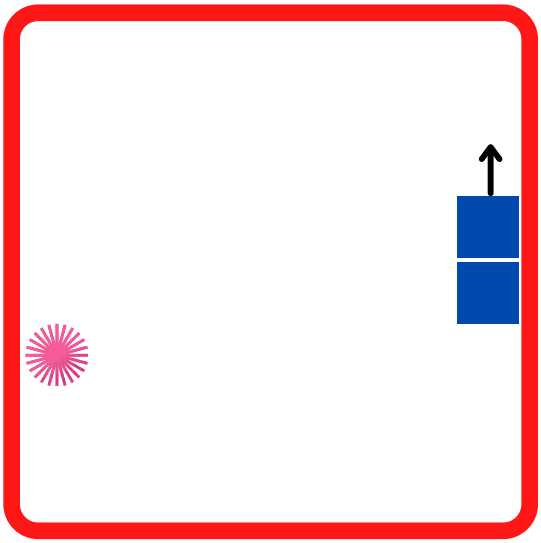
We can get the state of the game by just answering the following things:

- Is there a wall or snake body on the left?
- Is there a wall or snake body in front?
- Is there a wall or snake body on the right?
- Was snake moving up?
- Was snake moving right?
- Was snake moving down?
- Was snake moving left?
- Is the food left or right?
- Is the food up or down?

#5

Snake Game

Assume the following state when snake is moving up.



- Is there a wall or snake body on the left? No
- Is there a wall or snake body in front? No
- Is there a wall or snake body on the right? Yes
- Was snake moving up? Yes
- Was snake moving right? No
- Was snake moving down? No
- Was snake moving left? No
- Is the food left or right? Left
- Is the food up or down? Down

Reward

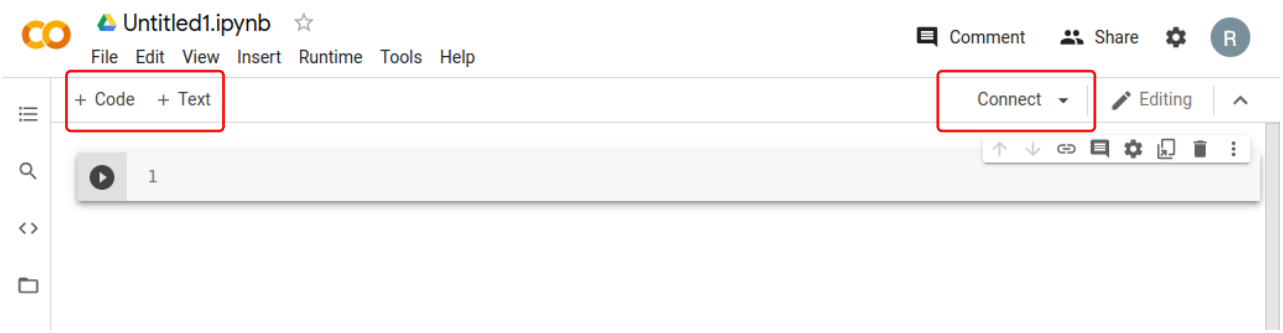
The agent will get +10 reward when the snake eats food, and -10 when the snake eats itself or hits the wall.

#Extra

Introduction to Colab

Nearly all machine learning and deep learning algorithms require good hardware. What if ... you don't have good hardware? Should you drop your dream to be a data scientist? No, there's an alternative Let us introduce you to Colab.

Colab is a service provided by Google which lets you access a virtual machine hosted on google servers. These virtual machines have dual-core Xeon processors, with 12GB of RAM. You can even use GPU for your neural networks. Colab is an interactive python notebook (ipynb), which means that with writing python code, you can also write normal text, include images.
To create a new colab notebook, just go to "https://colab.research.google.com", and create a new notebook. You will get something like below



You can connect to runtime by clicking the "Connect" button in the right corner. You can add a new Code cell, or text cell using respective buttons in the toolbar.

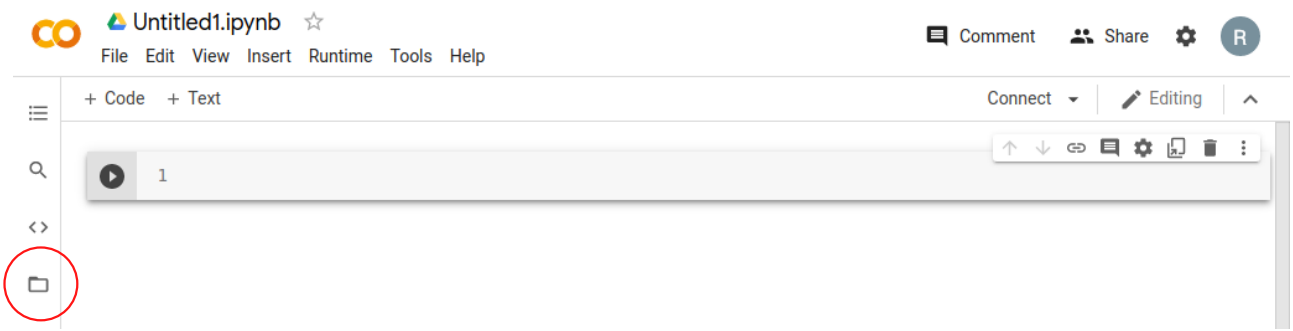
#Extra

Introduction to Colab

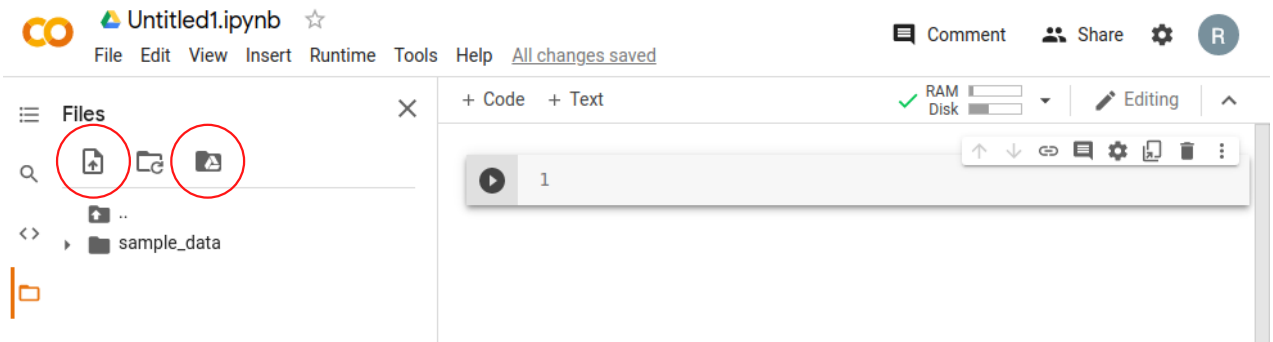
Uploading files

Sometimes you need to use a file from your PC. For that, you can upload the required files to google colab. Colab provides 100 GB in a colab session. If you want to access some files from your drive, you can even do so by connecting the drive to colab.

To upload something, open file pane from left toolbar.



Now, just upload the file using first button, and you can also mount google drive using last button



#7

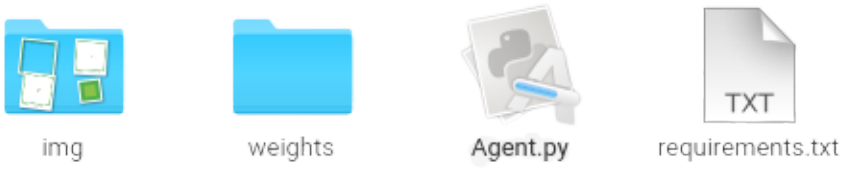
Code

Upload and unzip

We have also shared a zip folder and a colab file. Upload the zip to the shared colab file runtime.

```
!unzip -q GameBot\ \(Colab\).zip
```

After unzipping you will get the following 4 files and folders.



img/	-	images for a snake, food, and game boundaries
weights/	-	weights from a trained DQN
Agent.py	-	DQN Agent to play the game of snake
requirements.txt	-	libraries required for this task

Install required libraries and then restart the runtime.

```
!pip install -r requirements.txt
```

You can change the parameters defined in function define_parameters. After that run all the cells.

Learn

Knowledge Discovery through Brand Stories

Visit Us at :
techlearn.live

Email : support@techlearn.live
Phone : +91-9154796743