Développeur d'Intelligence Artificielle Appliquée

Cours #7

www.impactia.org

Structure de la formation

- **#1: Introduction**
- **#2: Vision**
- **#3: Vision**
- **#4: Vision**
- **#5: Renforcement**
- **#6: Renforcement**
- **#7: Renforcement**

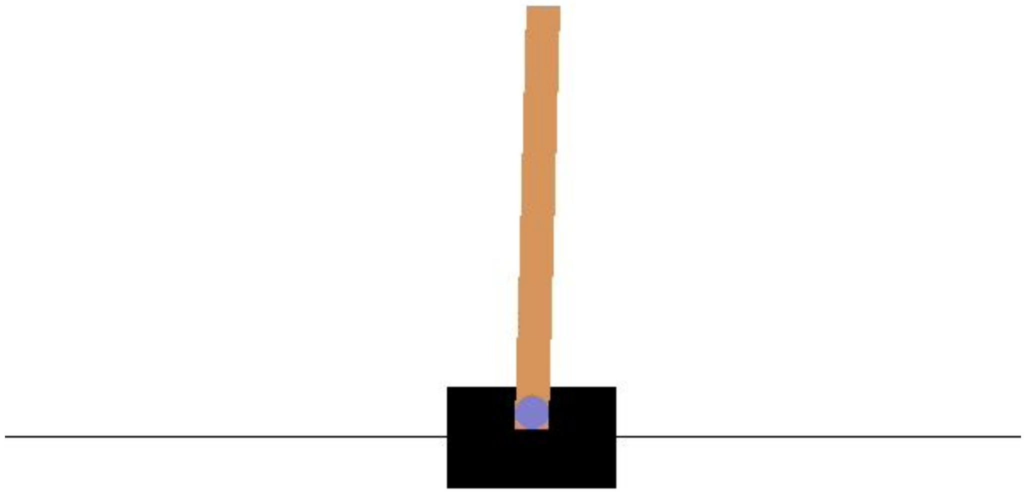
- #8: Langage
- #9: Langage
- #10: Langage
- # 11: Génération d’images
- #12: Génération d’images
- #13: Génération d’images
- #14: Projet
- #15: Projet

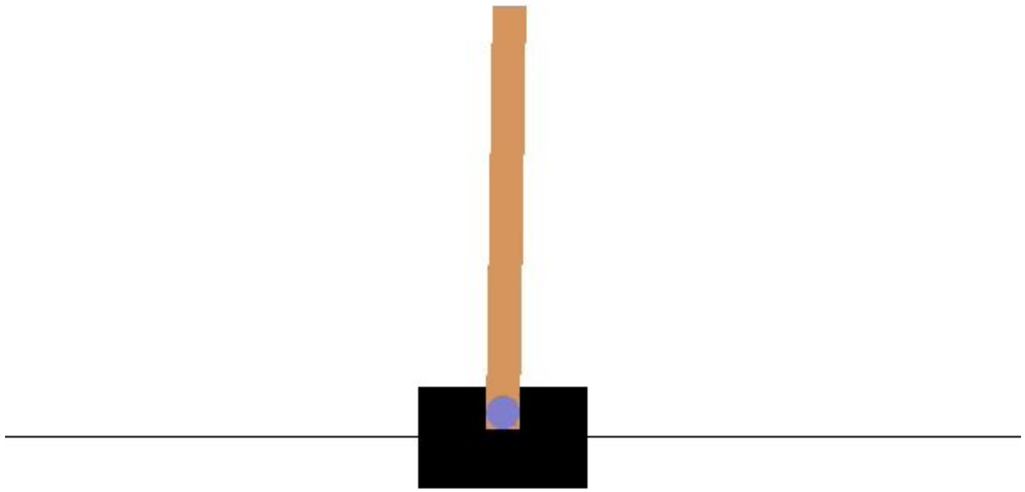
Semaine dernière : Cours #6

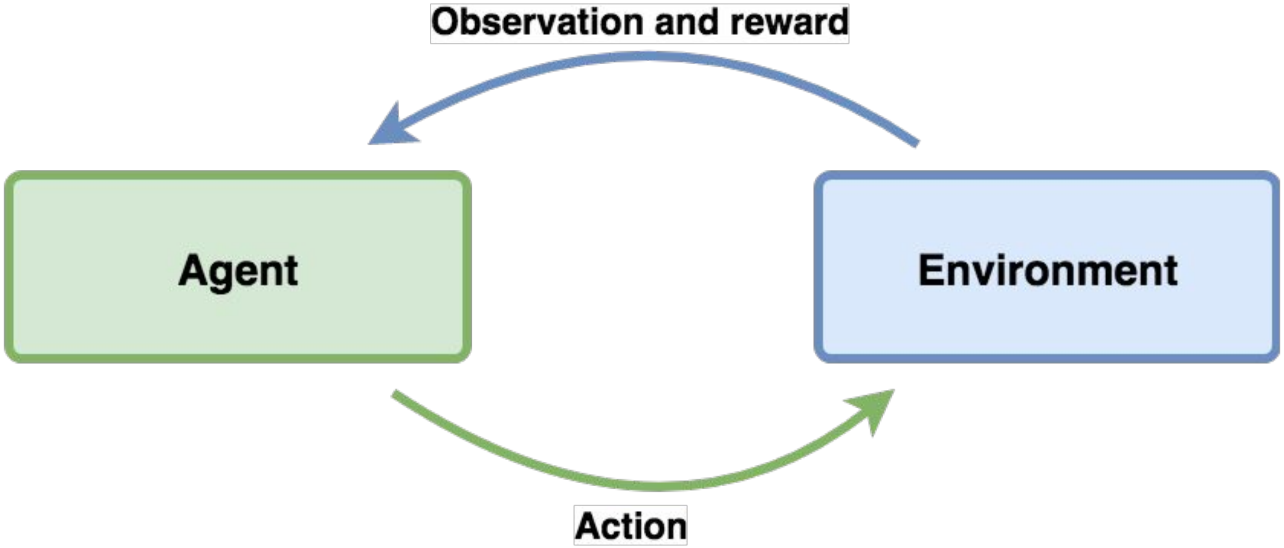
- Théorie
  - L'environnement droneRL
  - Q-learning
- Pratique
  - Entrainement des premiers agents droneRL

Aujourd’hui : Cours #5

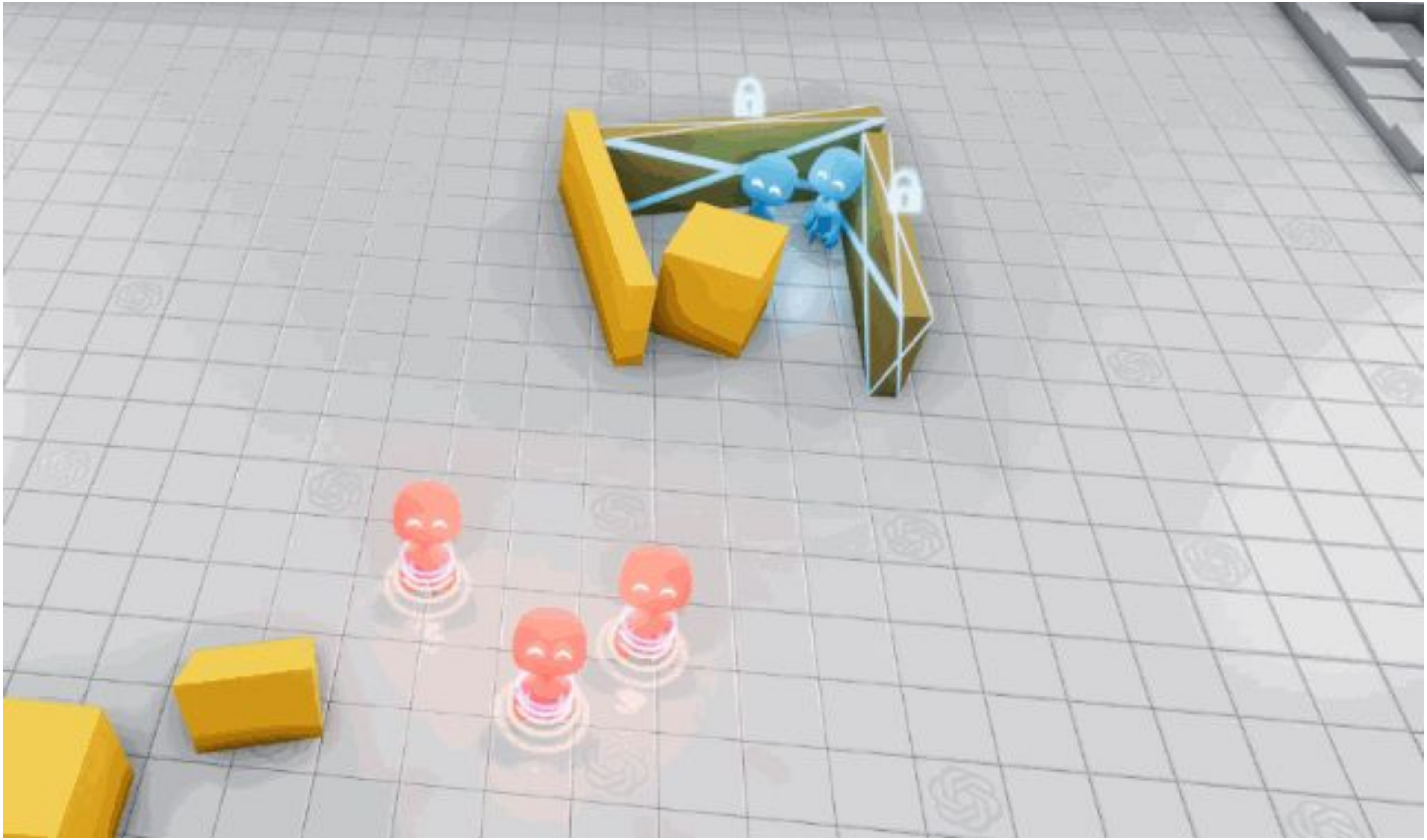
- Théorie
  - Deep Q-Network (DQN)
  - Apprentissage prioritisé
- Pratique
  - Entrainement d’agents droneRL
  - Challenge droneRL ! 💪



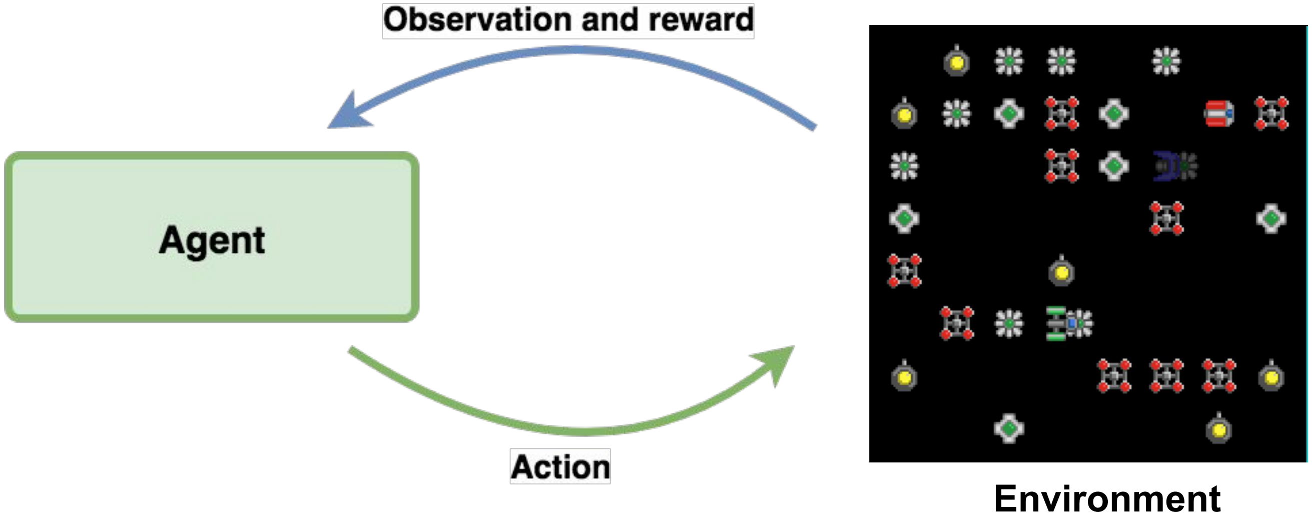


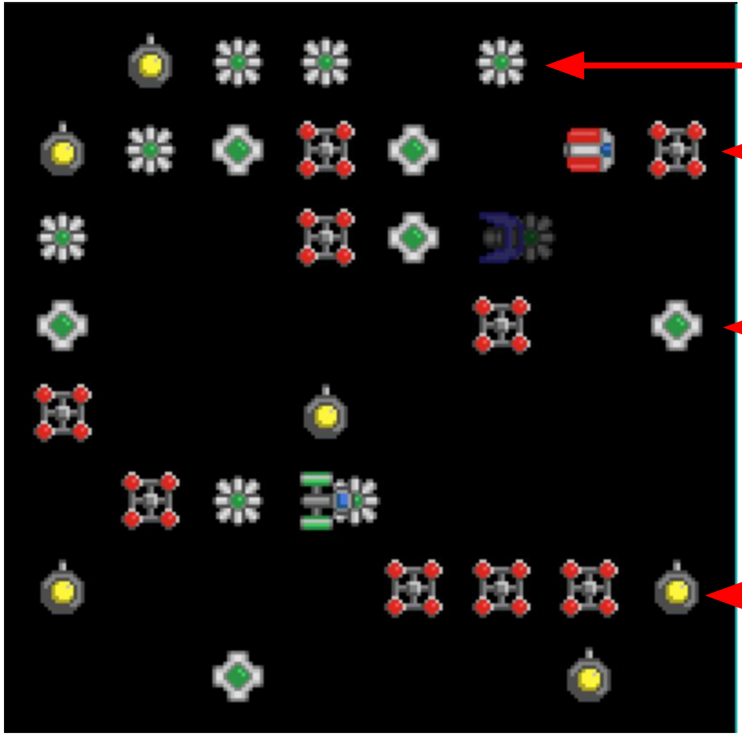


But :
Maximiser la somme des *récompenses* (“rewards”)
jusqu’à la fin de l’*épisode*



L’environnement *DeliveryDrones*


Observation and reward
Agent
Action
Environment



colis

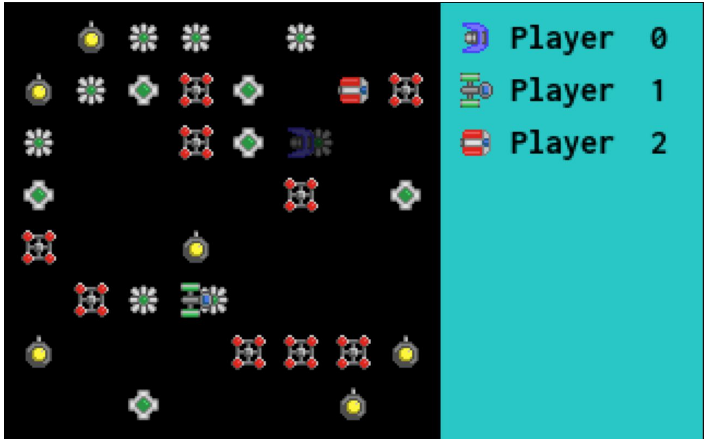
gratte-ciel

point de dépot

chargeur

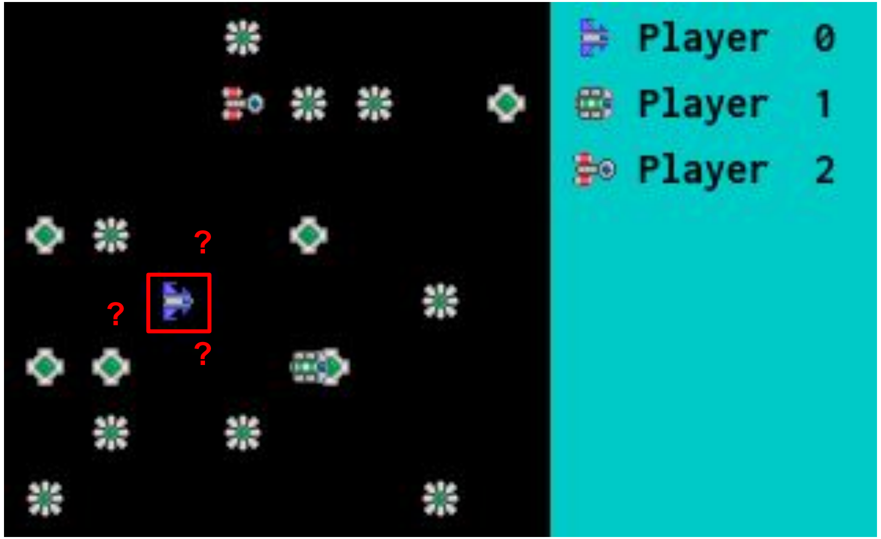
```python
# paramètres de l'environnement
self.env_params = {
    'pickup_reward': 0,
    'delivery_reward': 1,
    'crash_reward': -1,
    'charge_reward': -0.1,
    'discharge': 10,
    'charge': 20,
    'drone_density': 0.05,
    'packets_factor': 3,
    'dropzones_factor': 2,
    'stations_factor': 2,
    'skyscrapers_factor': 3,
    'rgb_render_rescale': 1.0
}

# wrapper

env = WindowedGridView(env, radius=3)
```

Paramètres utilisés pour l'évaluation sur AIcrowd !



Comment
“apprendre un comportement” ?

Apprendre un comportement



Une information à disposition

↖

5 actions possibles - que faire ?

LEFT **DOWN** **RIGHT** **UP** **STAY**

La “solution” d’un environnement

Si on connaît les valeurs de ce tableau,
il est facile de suivre un comportement optimal !

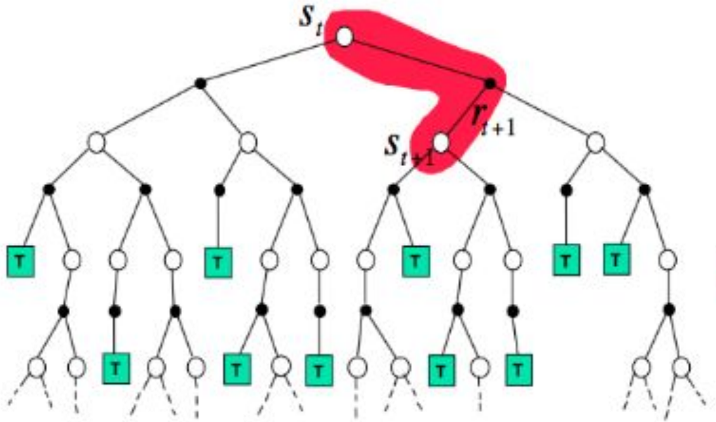
Actions

Observation

	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
↖	2.2	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2

“Q-table”

Comment apprendre la Q-table ?



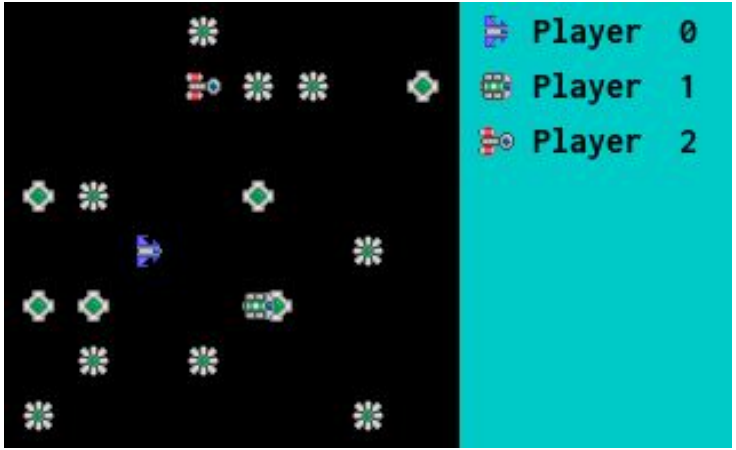
Q-learning

Actions

Observation

	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
↖	2.2	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2

“Q-table”
Somme attendue des récompenses

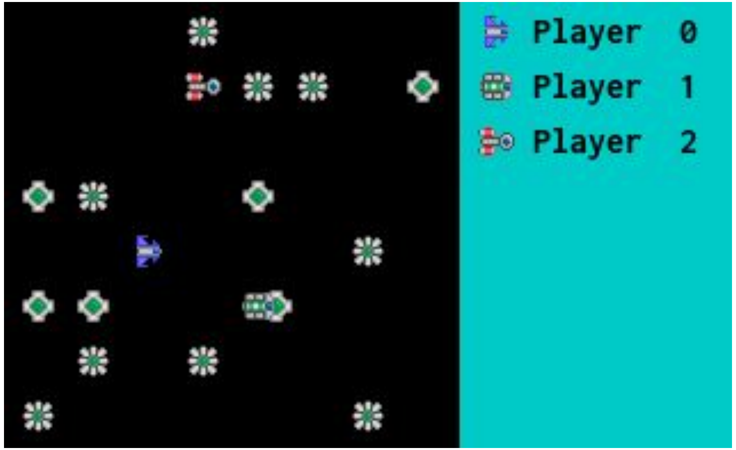


Actions

Observation

	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
↖	2.2	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2

“Q-table”
Somme attendue des récompenses



Actions

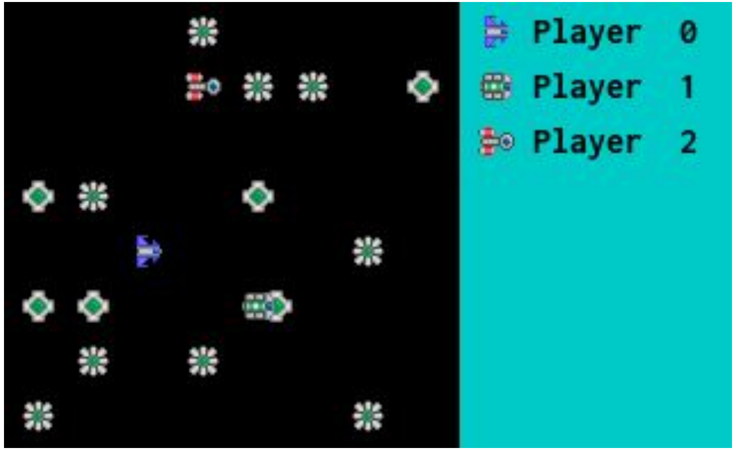
Observation

	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
↖	2.2	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2

"Q-table"
Somme attendue des récompenses

On sélectionne l'action qui maximise la somme attendue des récompenses

$$a = \text{argmax}_a \, Q(s, a)$$



On sélectionne l’action qui maximise la somme attendue des récompenses

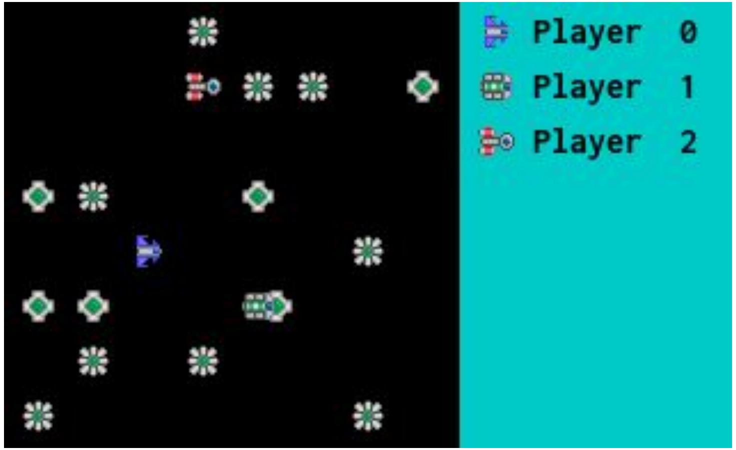
$$a = \text{argmax}_a\, Q(s, a)$$

Actions

Observation

	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
$\boxed{↖}$	$\boxed{2.2}$	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2

“Q-table”
Somme attendue des récompenses



Actions

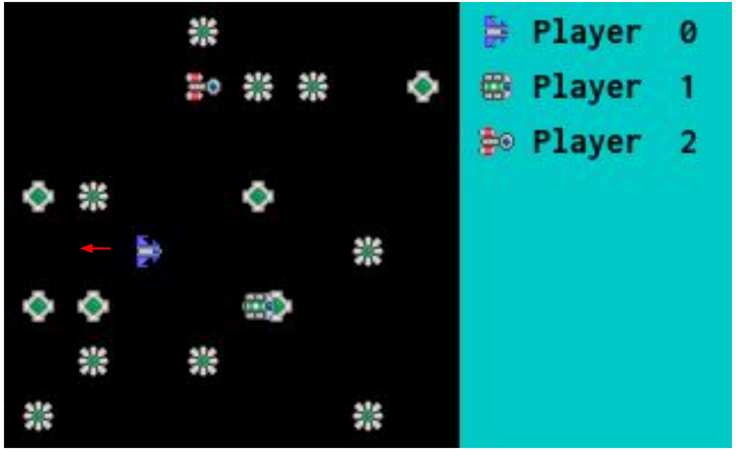
Observation

	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
↖	2.2	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2

"Q-table"
Somme attendue des récompenses

On sélectionne l'action qui maximise la somme attendue des récompenses

$$a = \text{argmax}_a\, Q(s, a)$$

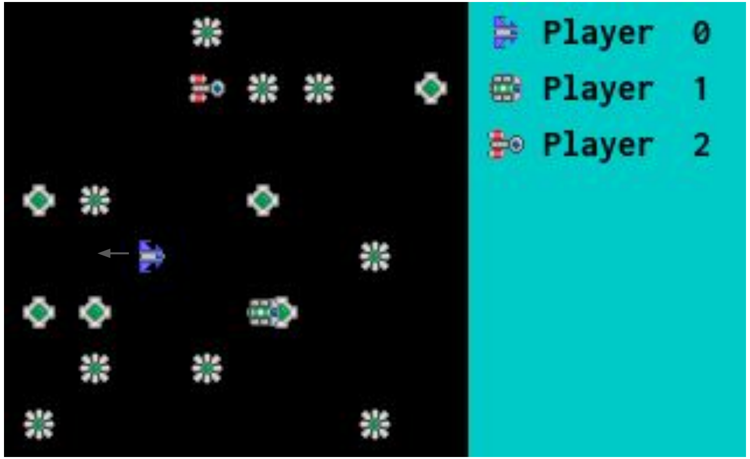


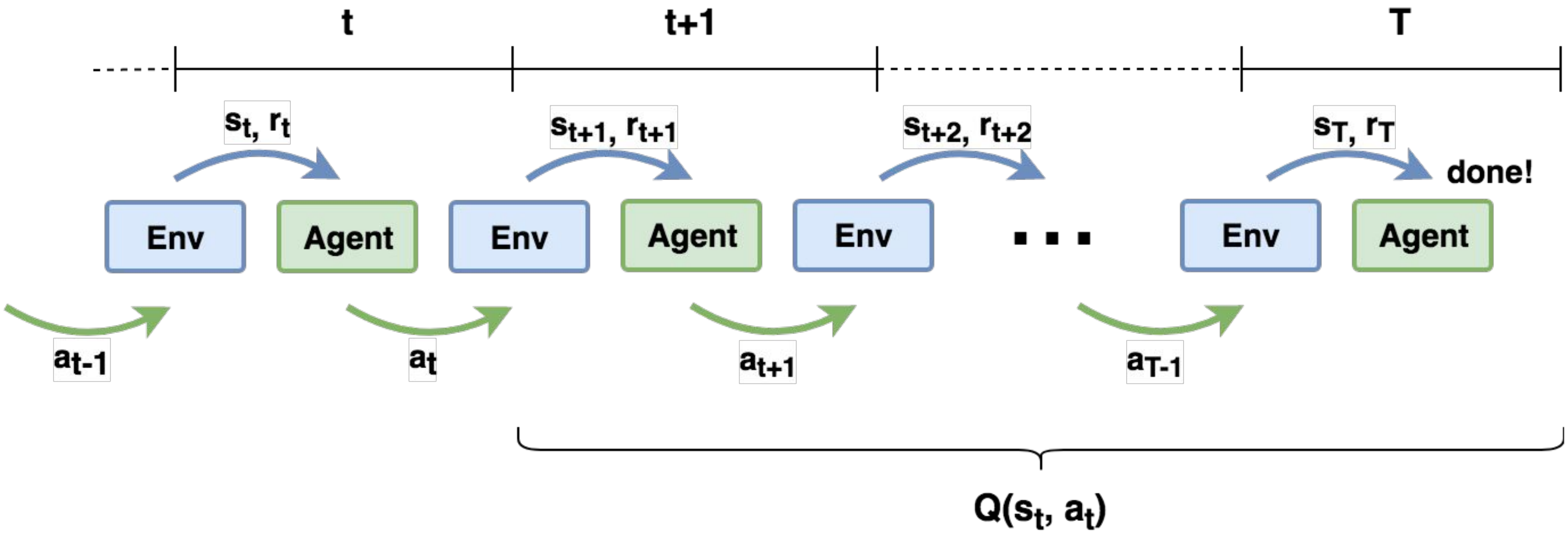
Q-learning

“Equation de Bellman”

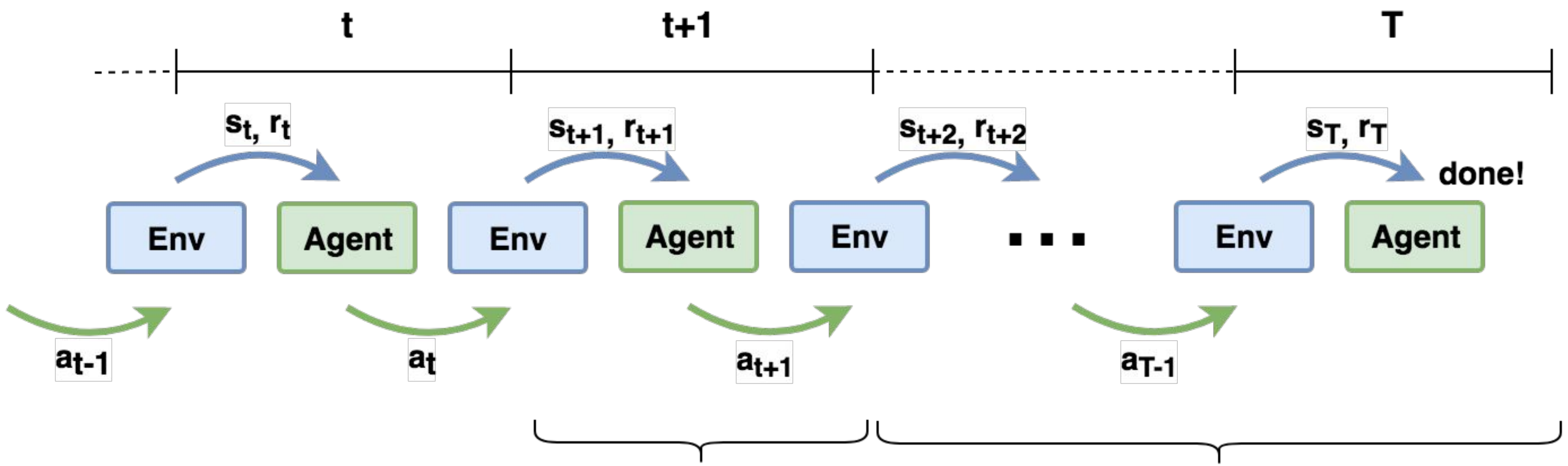
$$Q_{n+1}(s_t, a_t) = r_{t+1} + \gamma \max_a Q_n(s_{t+1}, a)$$

	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
↖	$\boxed{2.2}$	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2





$$= \Sigma\, r_{t+1} + r_{t+2} + \ldots + r_T$$



$$Q_{n+1}(s_t, a_t) = r_{t+1} + \max_a Q_n(s_{t+1}, a)$$

Q-learning

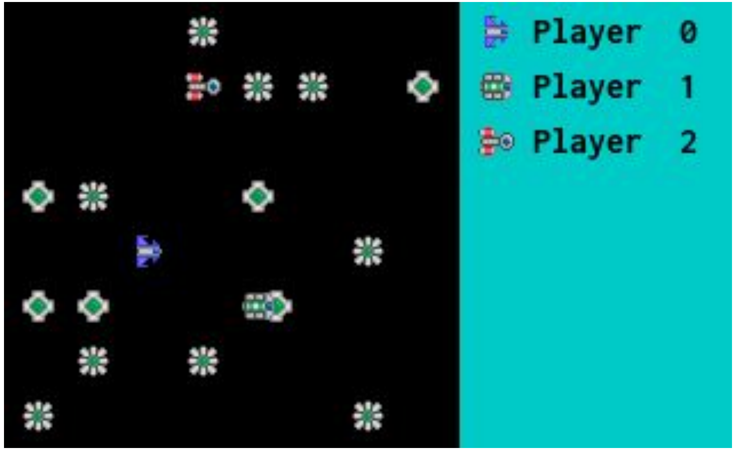
Actions

States

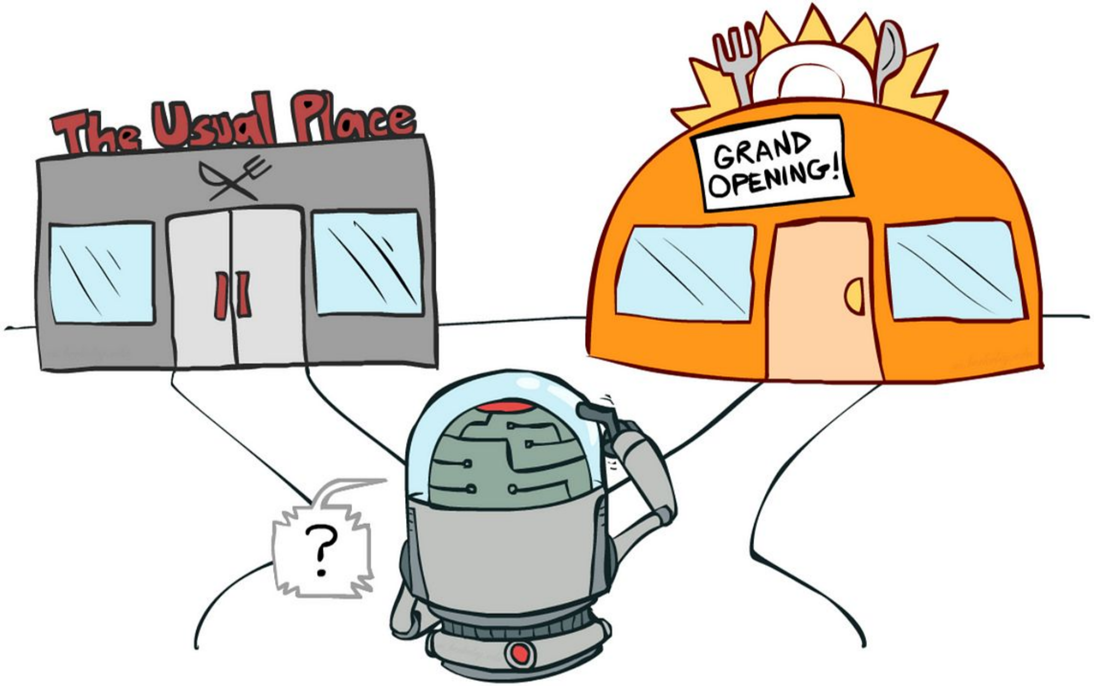
	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
↖	2.2	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2

$a = \text{argmax}_a\, Q(s_t, a)$

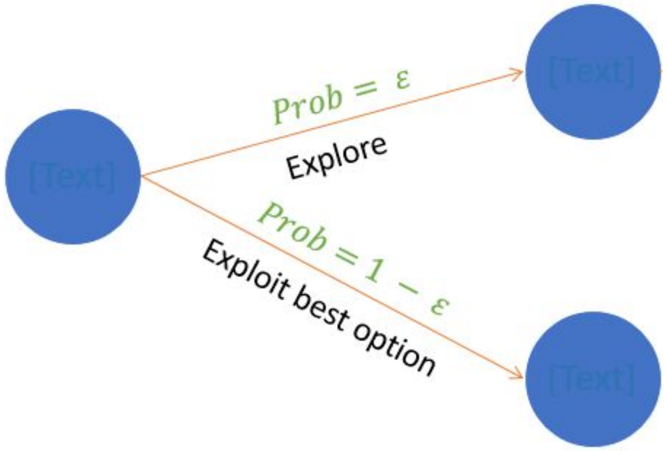
Equation de Bellman



“Explorer” ou “exploiter” ?



Explorer… parfois !
Méthode “epsilon-greedy”



Q-learning

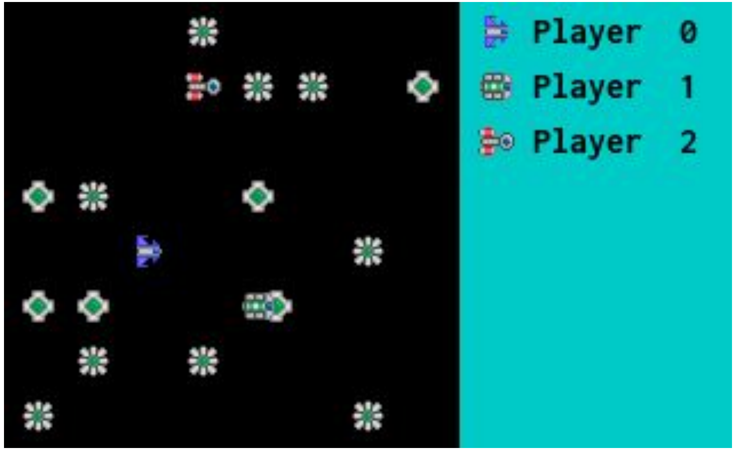
Actions

States

	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
↖	2.2	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2

$a = \mathrm{argmax}_a\, Q(s_t, a)$
+
epsilon-greedy

Bellman Equation



Q-learning

Fonctionne dans tous les cas !*

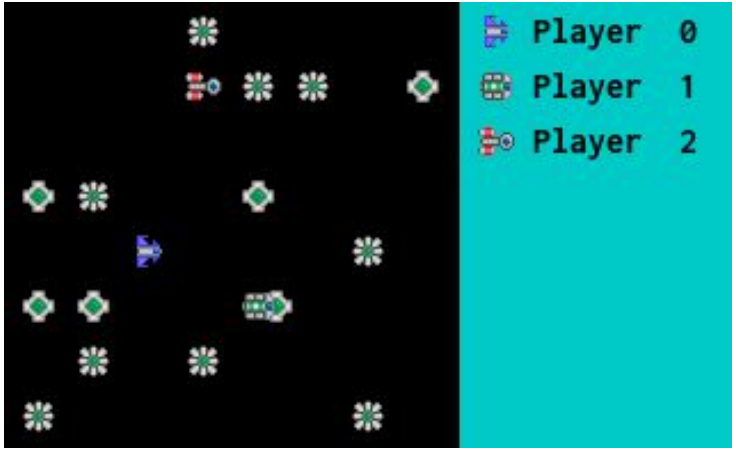
Actions

States

	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
↖	2.2	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2

$a = \text{argmax}_a\, Q(s_t, a)$
+
epsilon-greedy

Bellman Equation



Q-learning

Fonctionne dans tous les cas !*

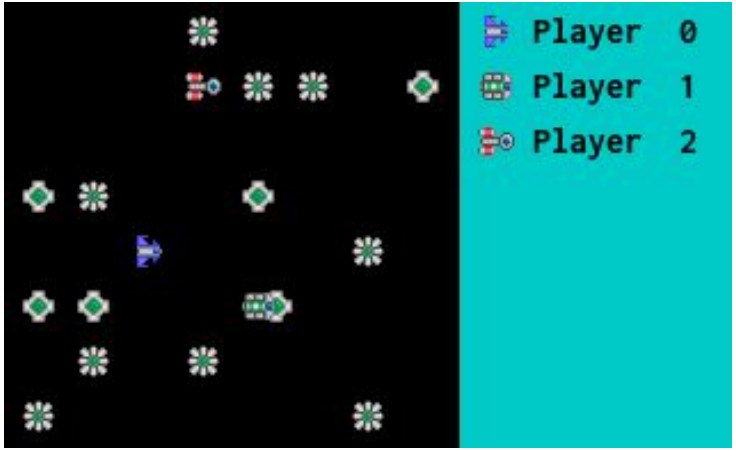
Actions

States

	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
↖	2.2	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2

$a = \text{argmax}_a\, Q(s_t, a)$
+
epsilon-greedy

Bellman Equation

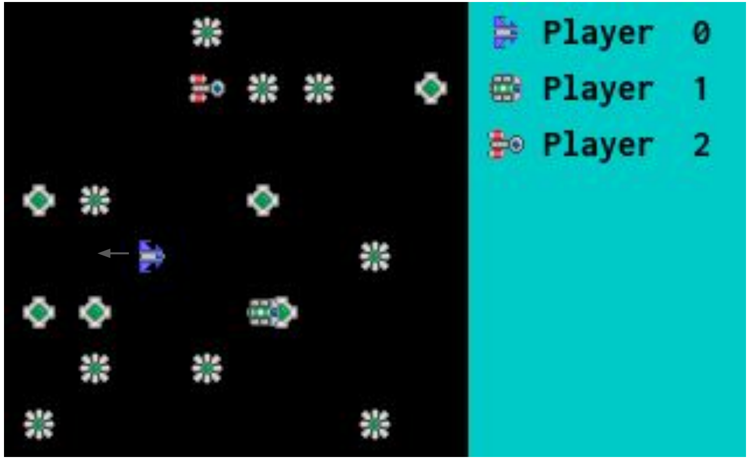


* peut prendre un temps infini 😅

	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
↖	$\boxed{2.2}$	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2

Bellman Equation

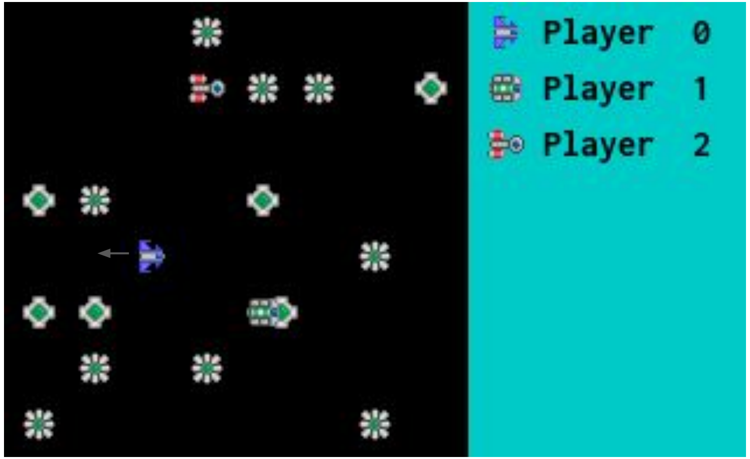
$$\underbrace{Q_{n+1}(s_t, a_t)}_{\text{new Q-value}} = r_{t+1} + \gamma \max_a Q_n(s_{t+1}, a)$$



	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
↖	2.2	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2

Bellman Equation

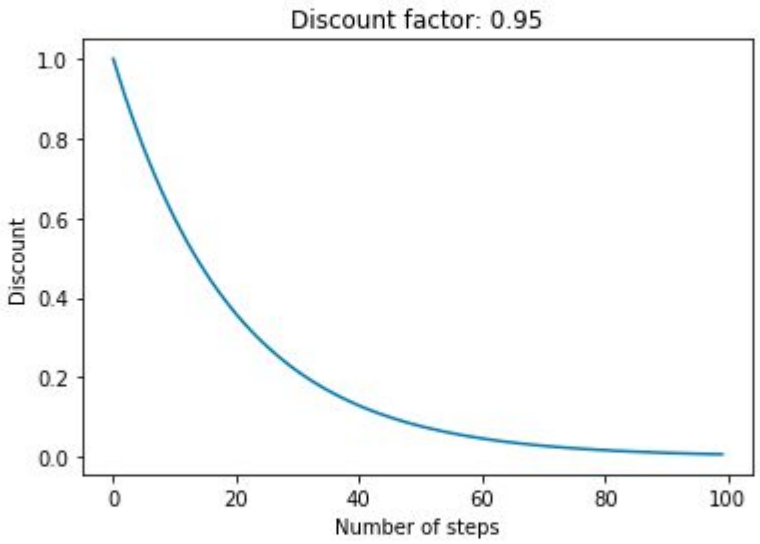
$$\underbrace{Q_{n+1}(s_t, a_t)}_{\text{new Q-value}} = r_{t+1} + \boxed{\gamma} \max_a Q_n(s_{t+1}, a)$$



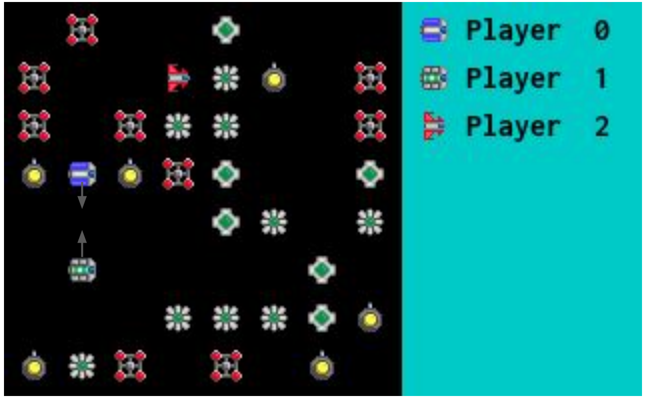
Equation de Bellman

$$Q_{n+1}(s_t, a_t) = r_{t+1} + \boxed{\gamma} \max_a Q_n(s_{t+1}, a)$$

Discount factor
(“facteur d’actualisation”)



Learning rate



$$\underbrace{Q_{n+1}(s_t, a_t)}_{\text{new Q-value}} = (1-\alpha)Q_n(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q_n(s_{t+1}, a))$$

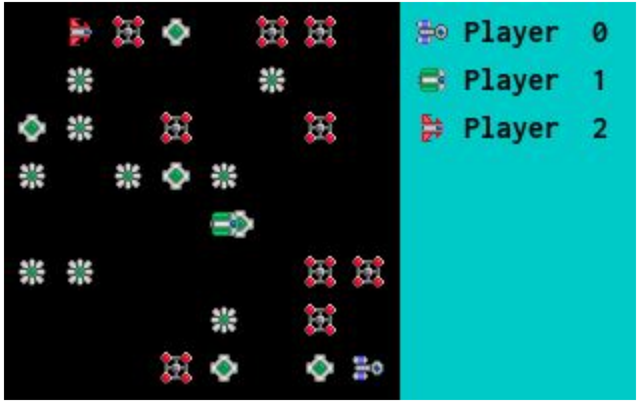
```
alpha = 0   # No update
alpha = 1   # Full update
alpha = 0.5 # Mean between old/new
```

Mise en pratique

Introduction à Q-learning

https://colab.research.google.com/drive/1KSei3ZdjNyyUYsPMqTyli4G2rm9v-P2E

Q-table size




```
{'target_dir': 2, 'lidar', [0, 0, 0, 0, 1, 1, 1, 1]}
```

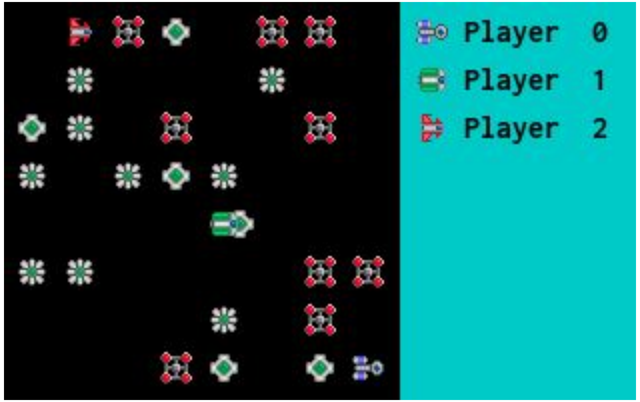
↓

```
Dict(target_dir:Discrete(8), lidar:MultiBinary(8))
```

↓

```
n_states = 8 * (2**8) = 2048
```

Q-table size




```
{'target_dir': 2, 'lidar', [0, 0, 0, 0, 1, 1, 1, 1]}
```

↓

```
Dict(target_dir:Discrete(8), lidar:MultiBinary(8))
```

↓

```
n_states = 8 * (2**8) = 2048
```

DQN
Deep Q-Network

Rappel : Q-learning

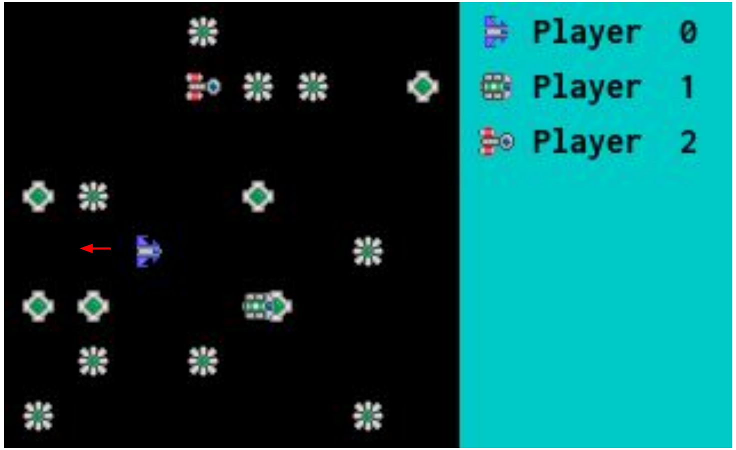
On sélectionne l'action qui maximise la somme attendue des récompenses

$$a = \text{argmax}_a \, Q(s, a)$$

Actions

Observations

	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
↖	2.2	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2

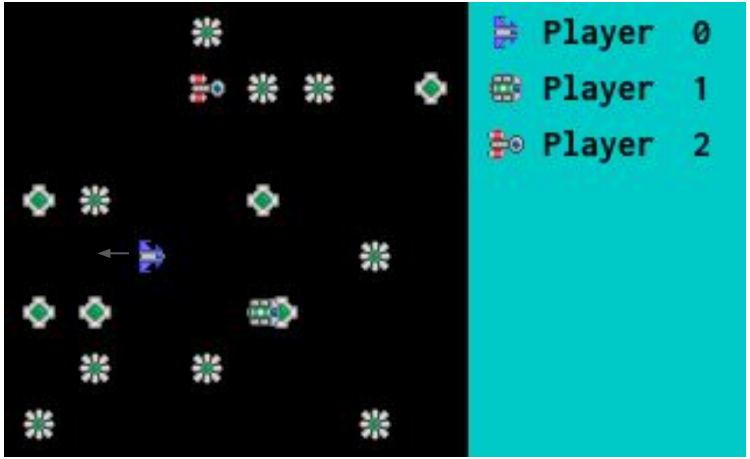


Rappel : Q-learning

	LEFT	DOWN	RIGHT	UP	STAY
→	1.7	1.4	2.6	0.86	1.5
↘	1.3	1.6	2.4	0.78	1.5
↑	1.9	1.6	1.4	2.5	1.7
↗	1.4	0.58	1.8	1.6	1.6
↖	$\boxed{2.2}$	1.3	0.97	1.6	1.5
↙	2.5	2	1.3	0.9	1.5
↓	1.8	2.9	1.7	1.2	2.1
←	2.6	1.5	2.1	1.2	2.2

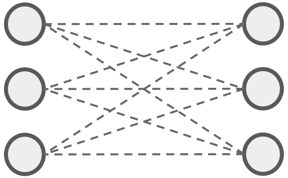
"Equation de Bellman"

$$Q_{n+1}(s_t, a_t) = r_{t+1} + \gamma \max_a Q_n(s_{t+1}, a)$$

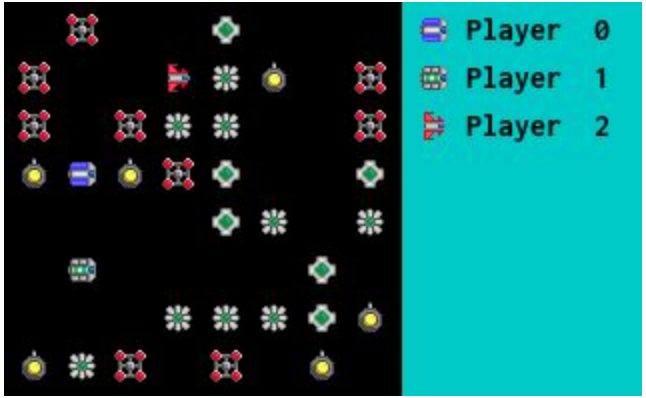


Deep Q-network (DQN)

charge: 16.18%
lidar_left: 0
lidar_down: 1
...



Q(Up) = 0.33
Q(Right) = 3.14
Q(Stay) = 2.71
...



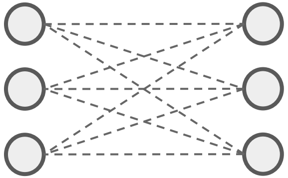
Deep Q-network (DQN)

charge: 16.18%

lidar_left: 0

lidar_down: 1

...

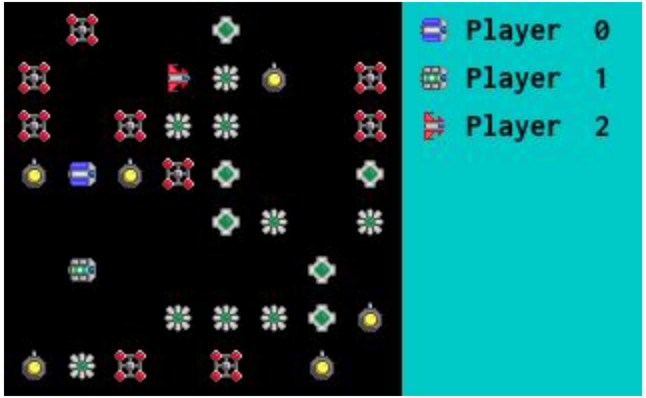


Q(Up) = 0.33

Q(Right) = 3.14

Q(Stay) = 2.71

...

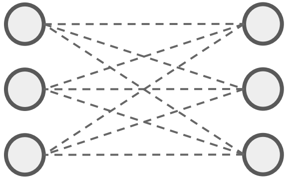


"Q-*network*"

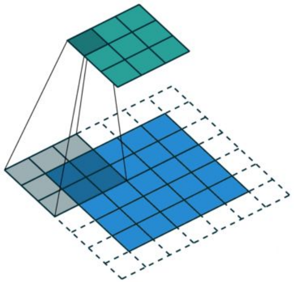
$$\delta = \boxed{Q_n}(s_t, a_t) - \left(r_{t+1} + \gamma \max \boxed{Q_n}(s_{t+1}, a)\right)$$

δ est l'erreur à minimiser

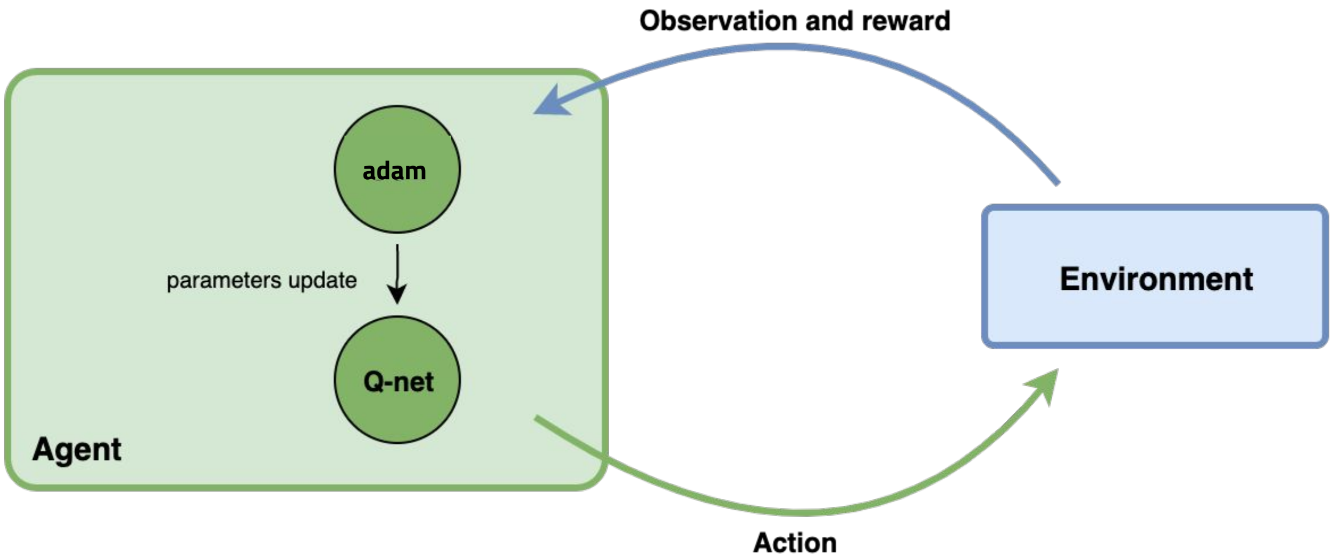
Deep Q-network (DQN)

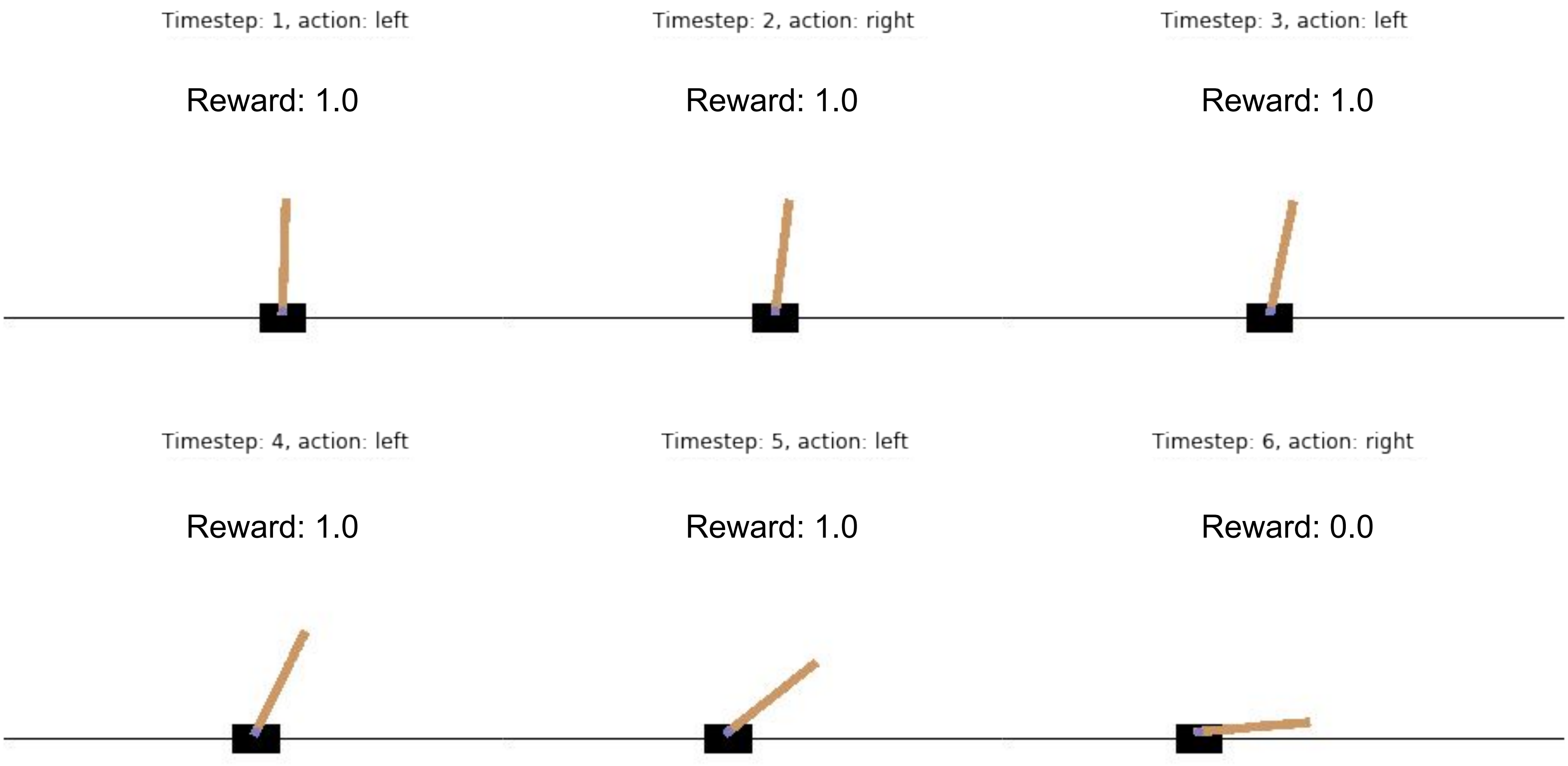


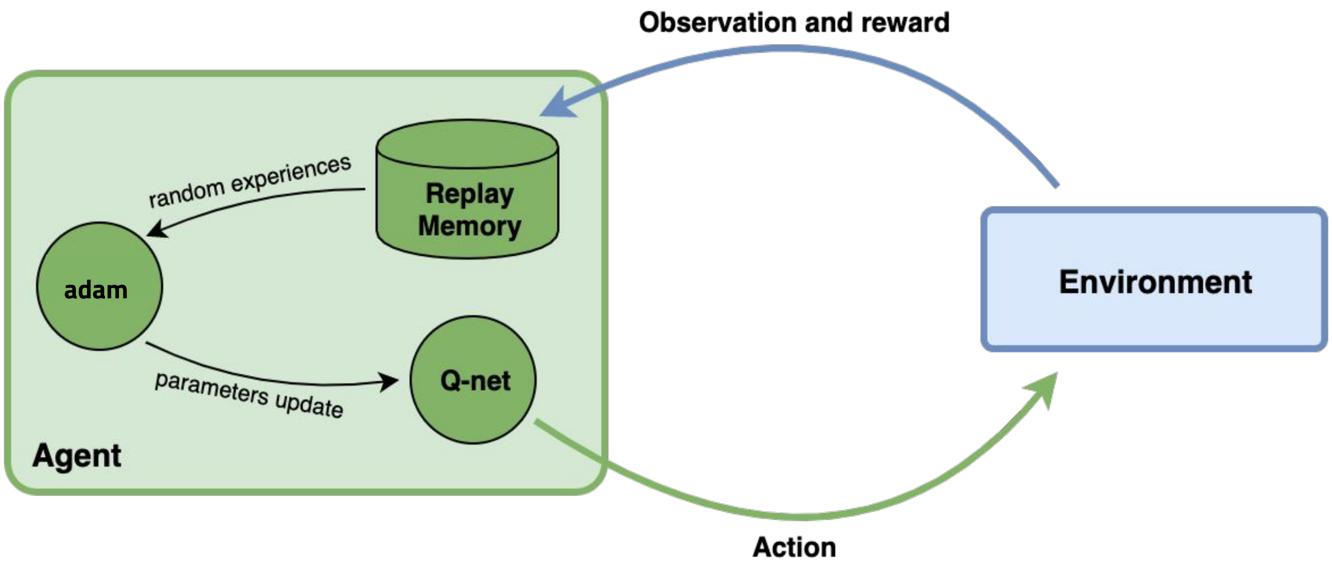
Couche dense (“fully connected”)

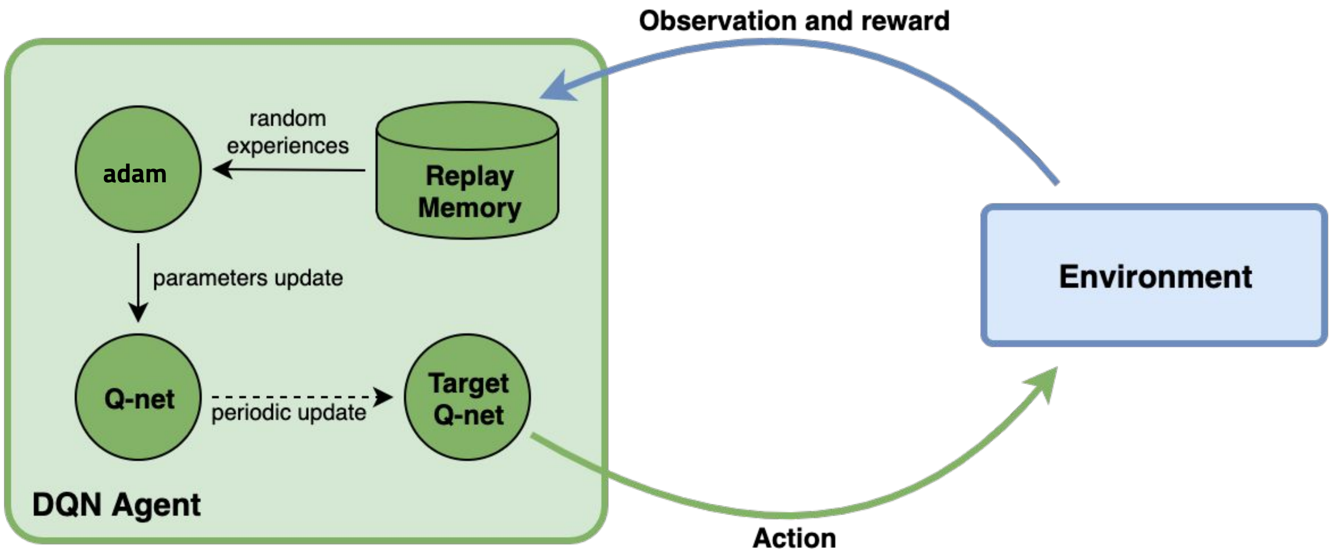


Couche à convolutions


Observation and reward
adam
parameters update
Q-net
Agent
Environment
Action


Timestep: 1, action: left
Reward: 1.0
Timestep: 2, action: right
Reward: 1.0
Timestep: 3, action: left
Reward: 1.0
Timestep: 4, action: left
Reward: 1.0
Timestep: 5, action: left
Reward: 1.0
Timestep: 6, action: right
Reward: 0.0


Observation and reward
random experiences
Replay
Memory
adam
parameters update
Q-net
Agent
Environment
Action


Observation and reward
random
experiences
adam
Replay
Memory
parameters update
Environment
Q-net
periodic update
Target
Q-net
DQN Agent
Action

A vous de jouer !

Mise en pratique

Deep Q-Network

https://colab.research.google.com/drive/1R9X7vTaoTzDI1QGoArBMnMwgPB-BXK2C

Peut-on faire mieux ?


DISK STORAGE
RD
RE

Les difficultés de l’AR

- Entrainement long
- Sensibilité des paramètres
- Manque de généralisation
- Dilemme exploration/exploitation

Trucs et Astuces

- "Reward engineering"
  - Ajustement à la main des récompenses
  - Mais, risque d'introduire des biais !
- Entraîner dans différents environnements
  - Appeler plusieurs fois `trainer.train()`
- Faire varier le paramètre epsilon
  - Le faire descendre puis remonter de façon cyclique ?
- Entraîner plus n'améliore pas toujours les choses !
  - Évaluer régulièrement, puis garder le meilleur agent ?

Les difficultés de l’AR

- Entrainement long
- Sensibilité des paramètres
- Manque de généralisation
- Dilemme exploration/exploitation

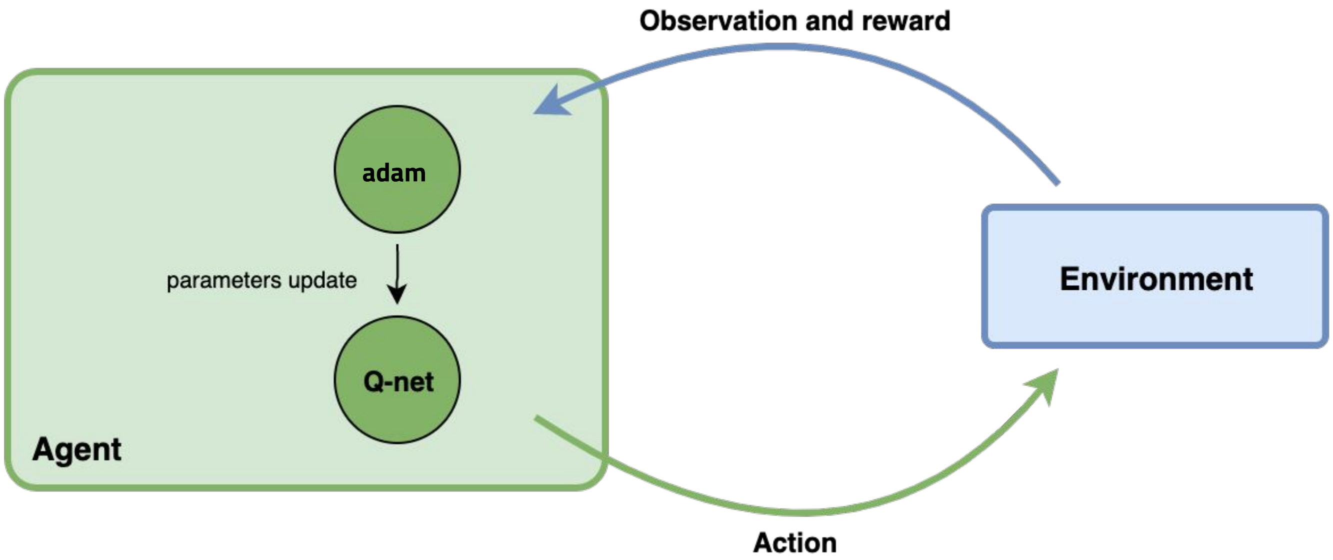
Apprentissage Prioritisé !

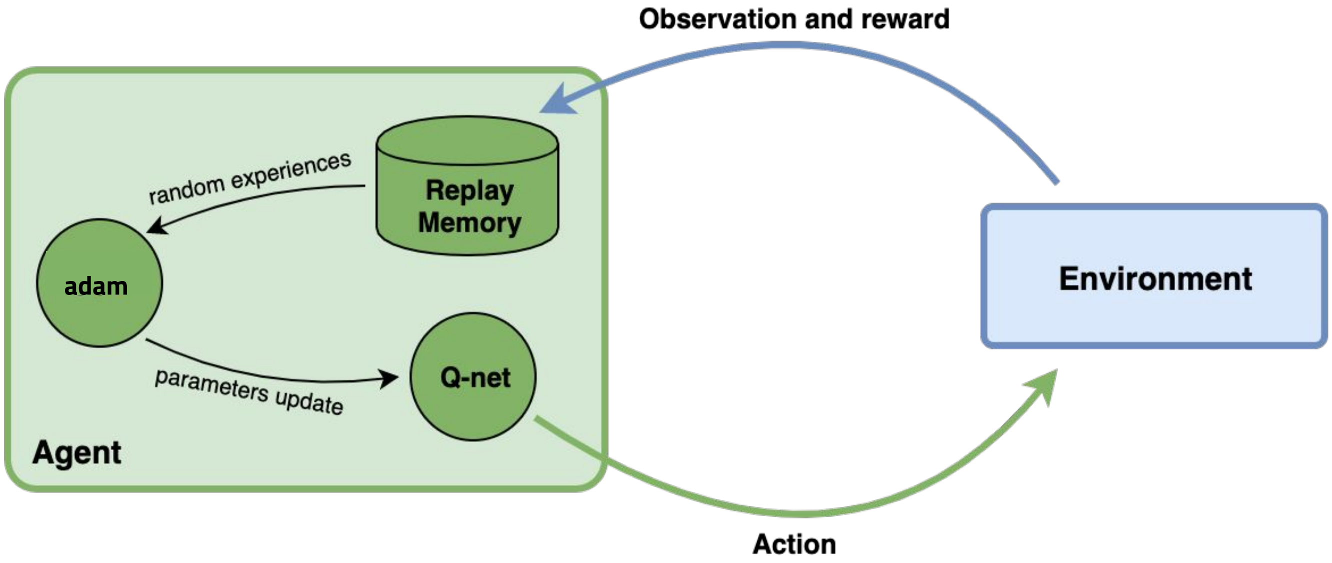
Prioritized Experience Replay
(“Rejeu d'Expériences Priorisées”)

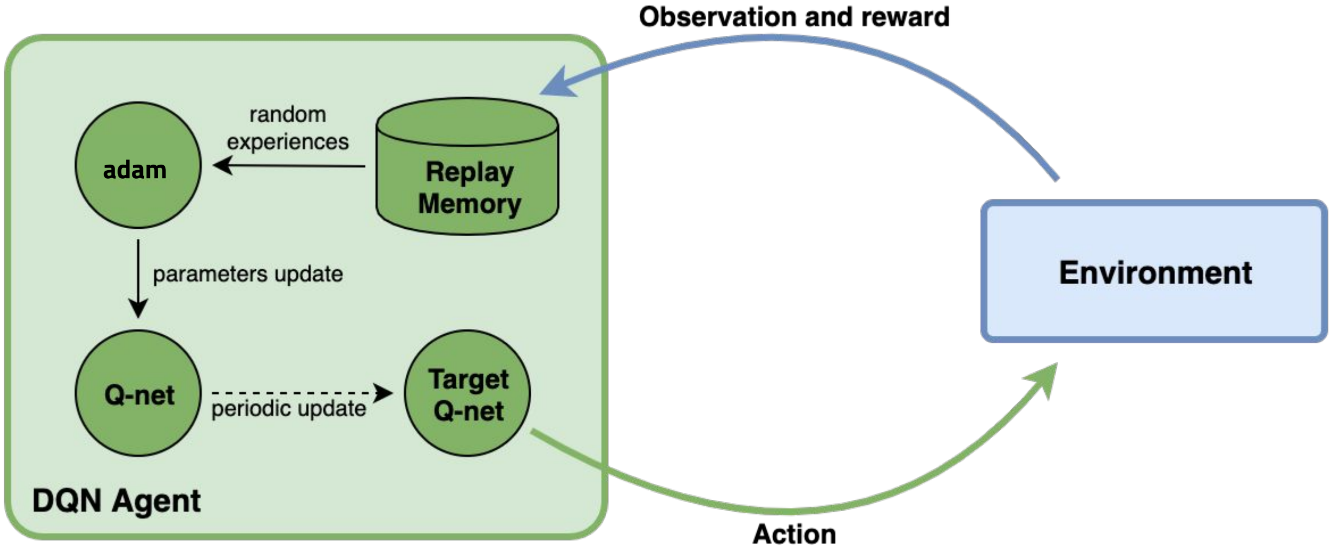
Schaul et al. 2015
https://arxiv.org/abs/1511.05952

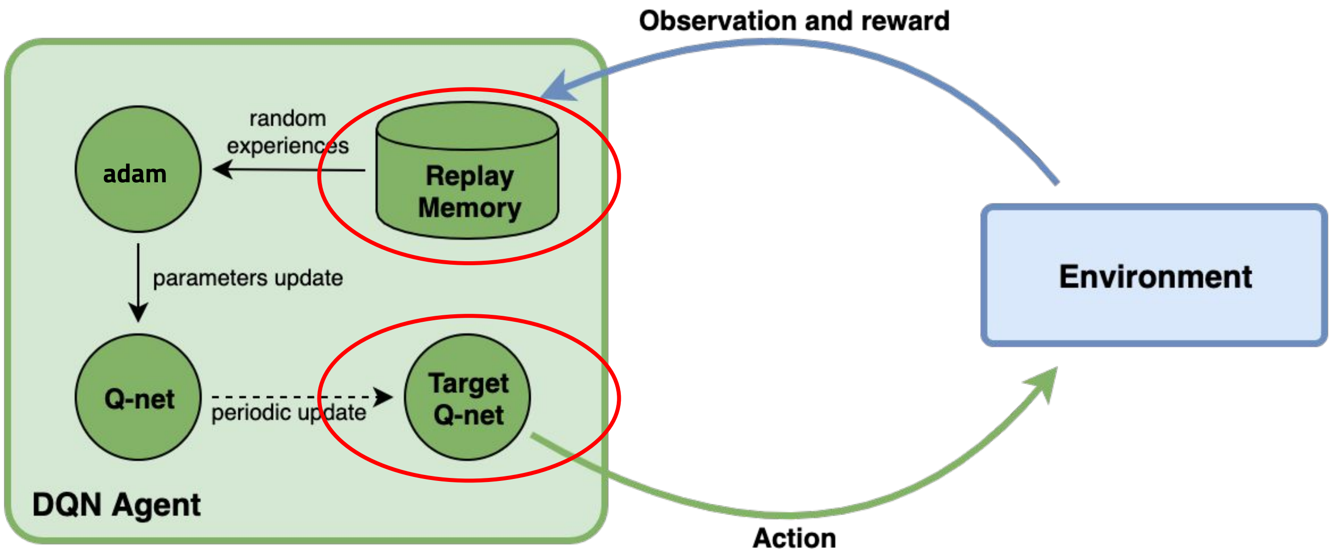
But :

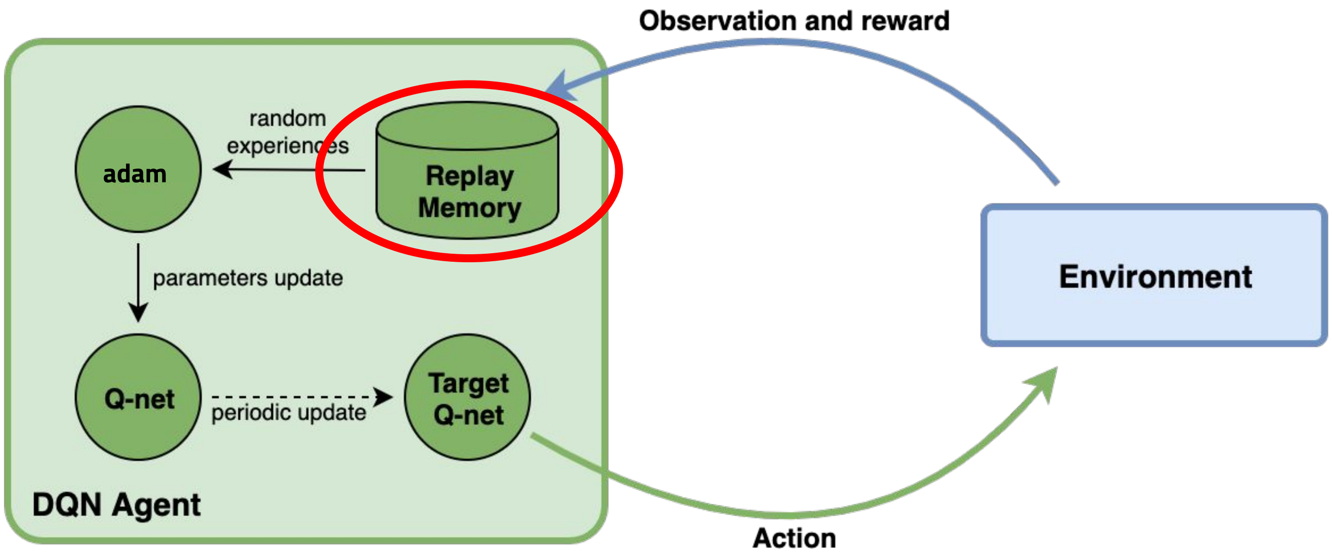
apprendre plus efficacement


Observation and reward
adam
parameters update
Q-net
Agent
Environment
Action


Observation and reward
random experiences
Replay
Memory
adam
parameters update
Q-net
Agent
Environment
Action

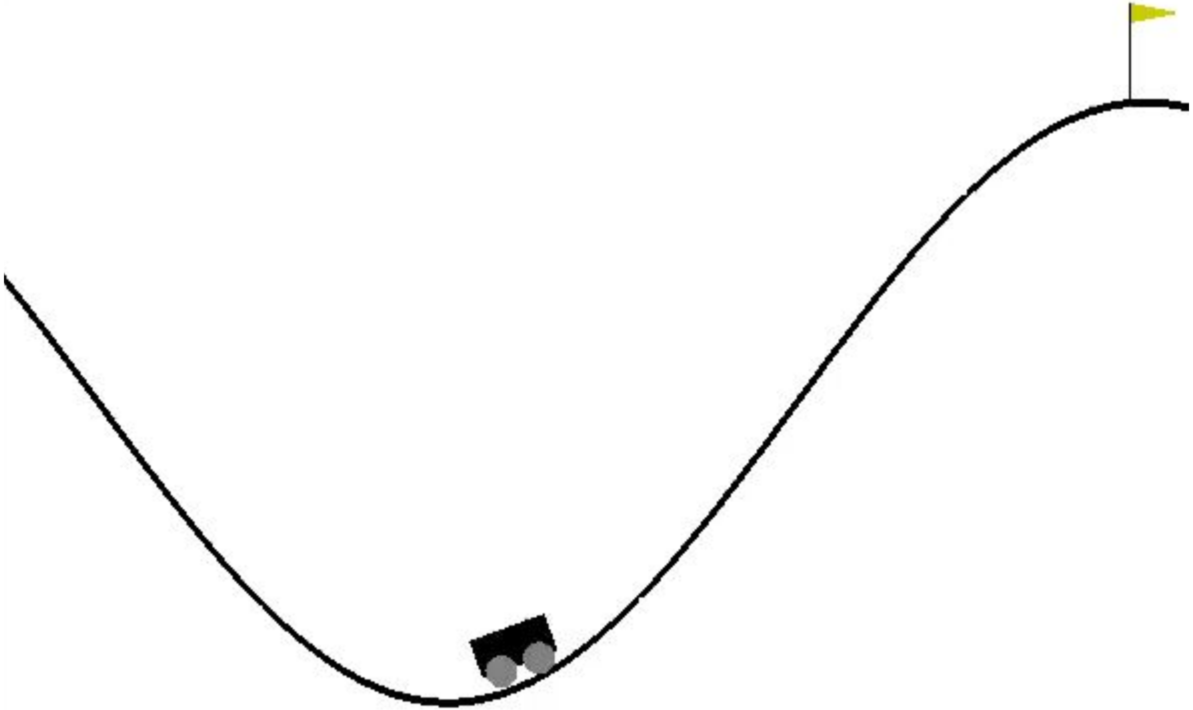

Observation and reward
random
experiences
adam
Replay
Memory
parameters update
Q-net
periodic update
Target
Q-net
DQN Agent
Environment
Action

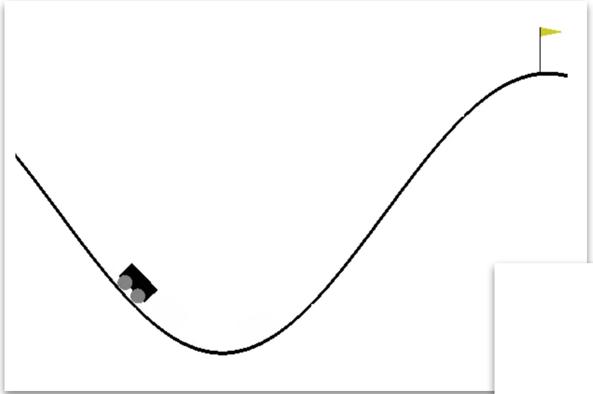

Observation and reward
random experiences
adam
Replay Memory
parameters update
Environment
Q-net
periodic update
Target Q-net
DQN Agent
Action

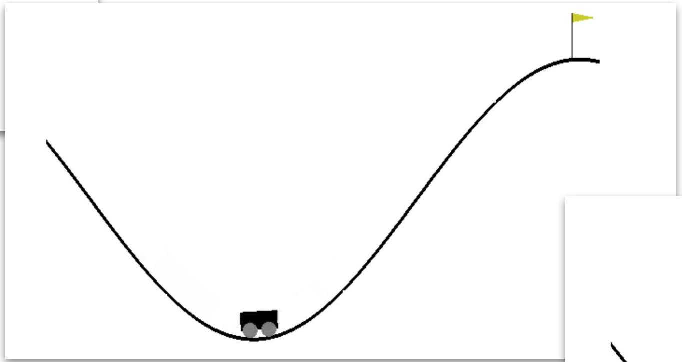

Observation and reward
random
experiences
adam
Replay
Memory
parameters update
Q-net
periodic update
Target
Q-net
DQN Agent
Environment
Action

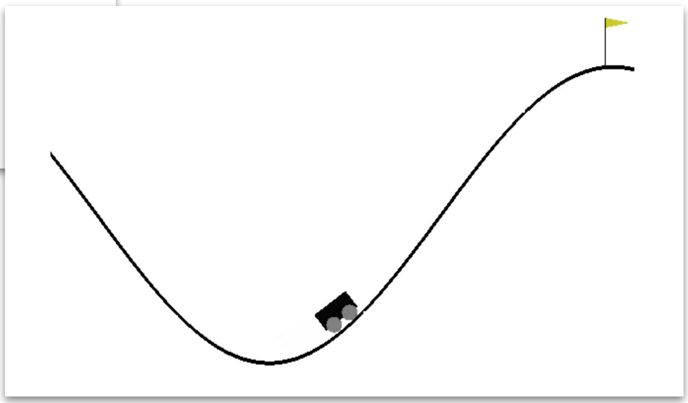
“Toutes les expériences ne sont pas égales”

A Katharopoulos

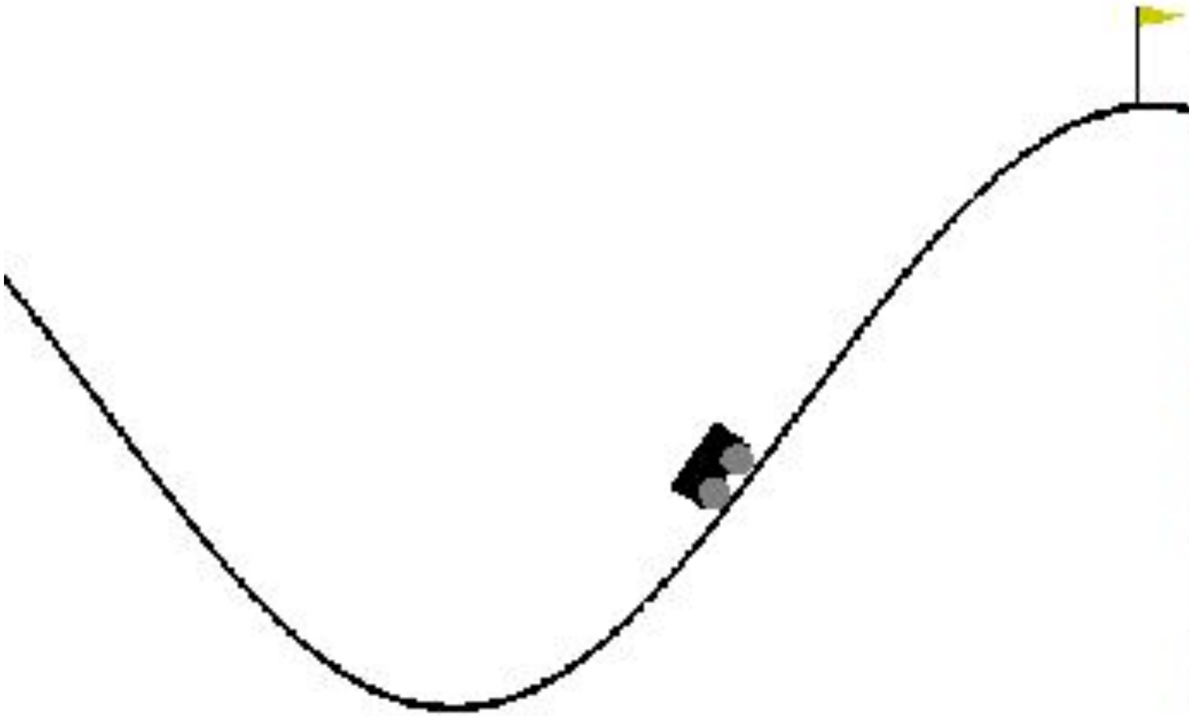


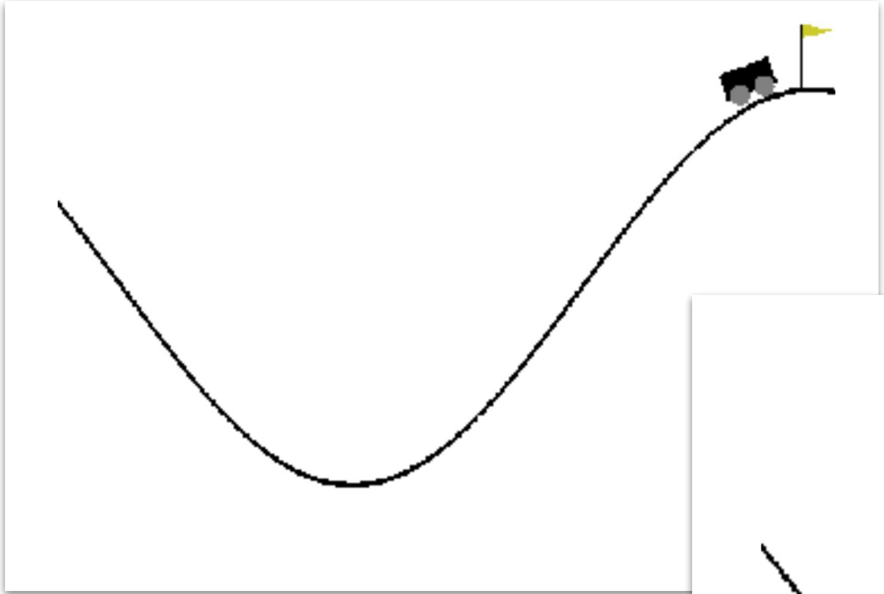


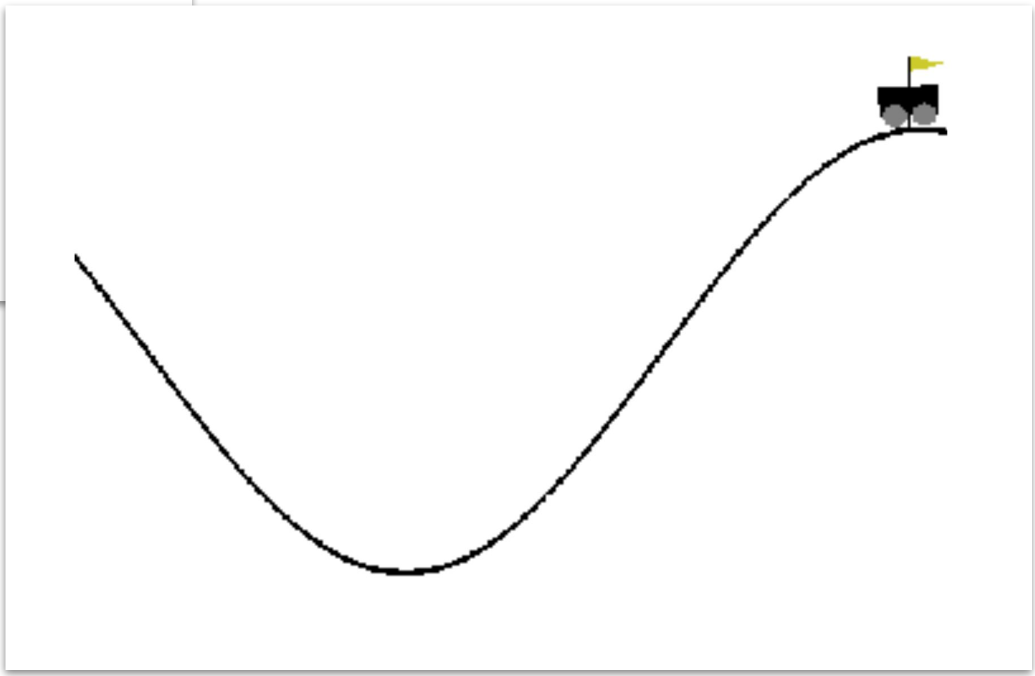




99% du replay buffer

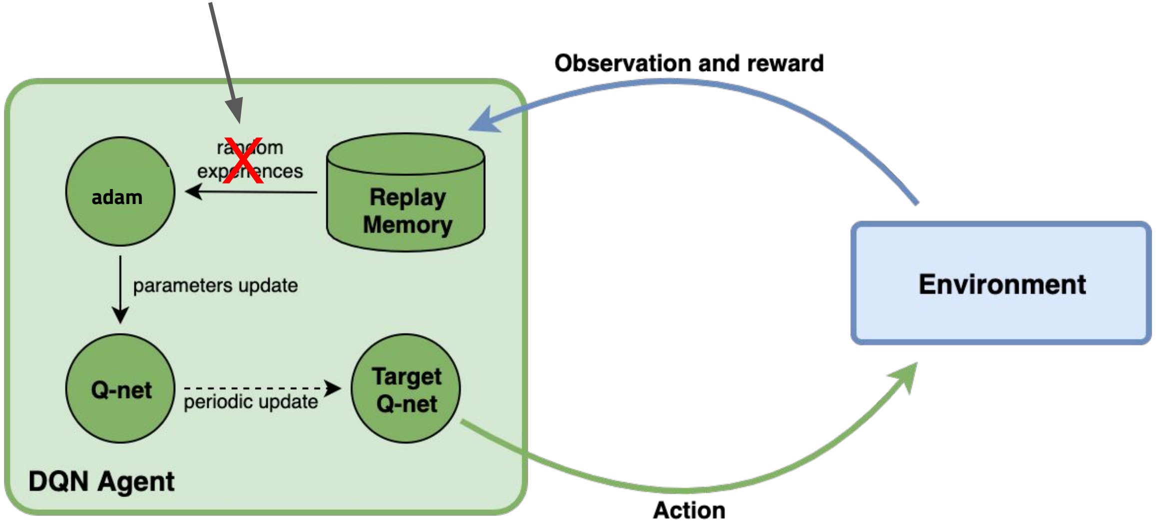


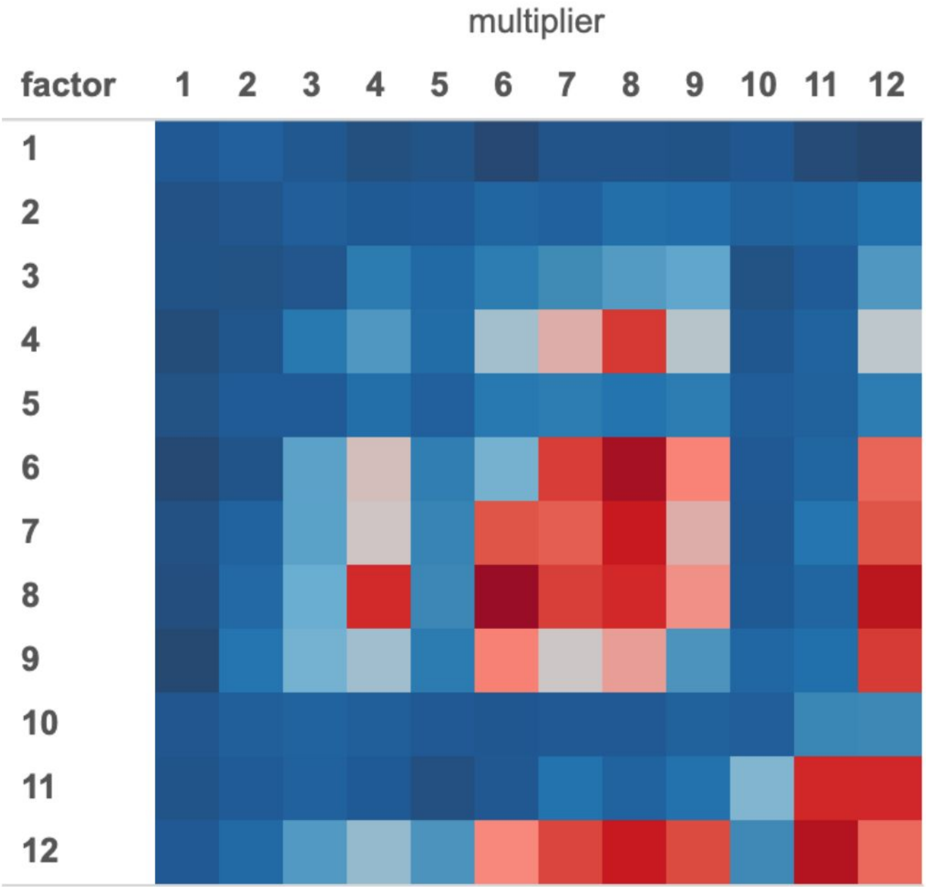




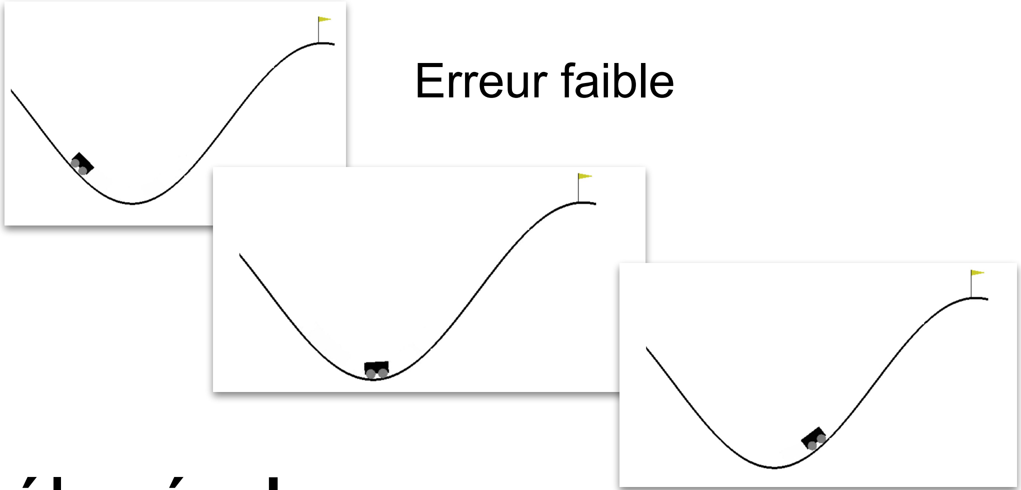
Les 1% !

Sample in smarter way!



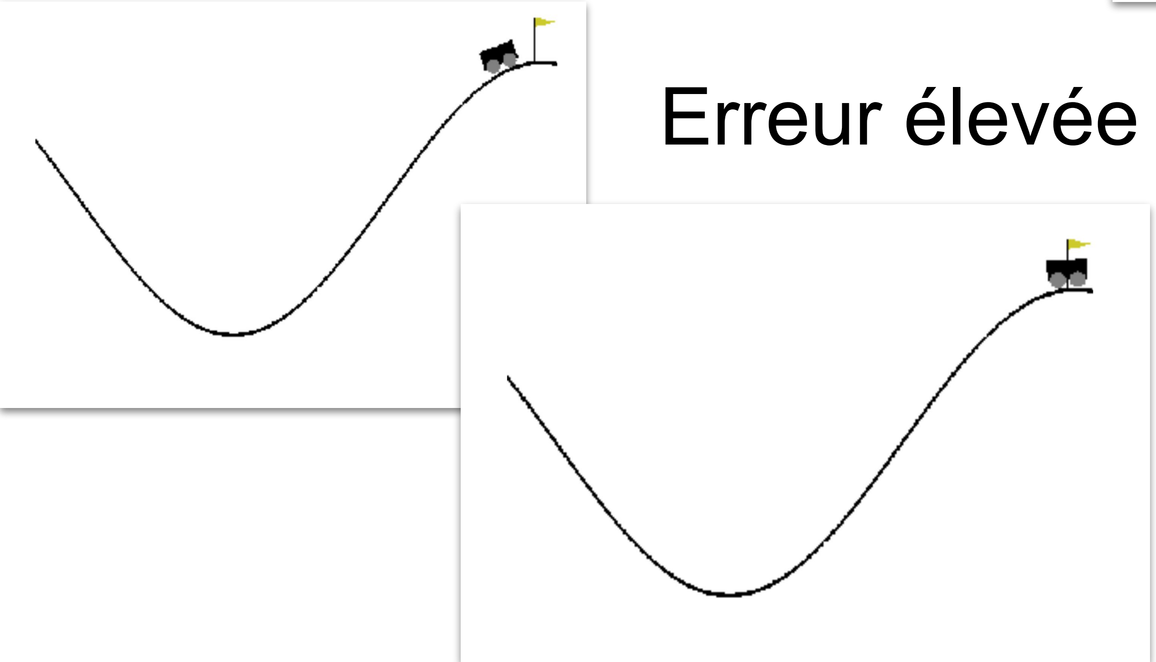


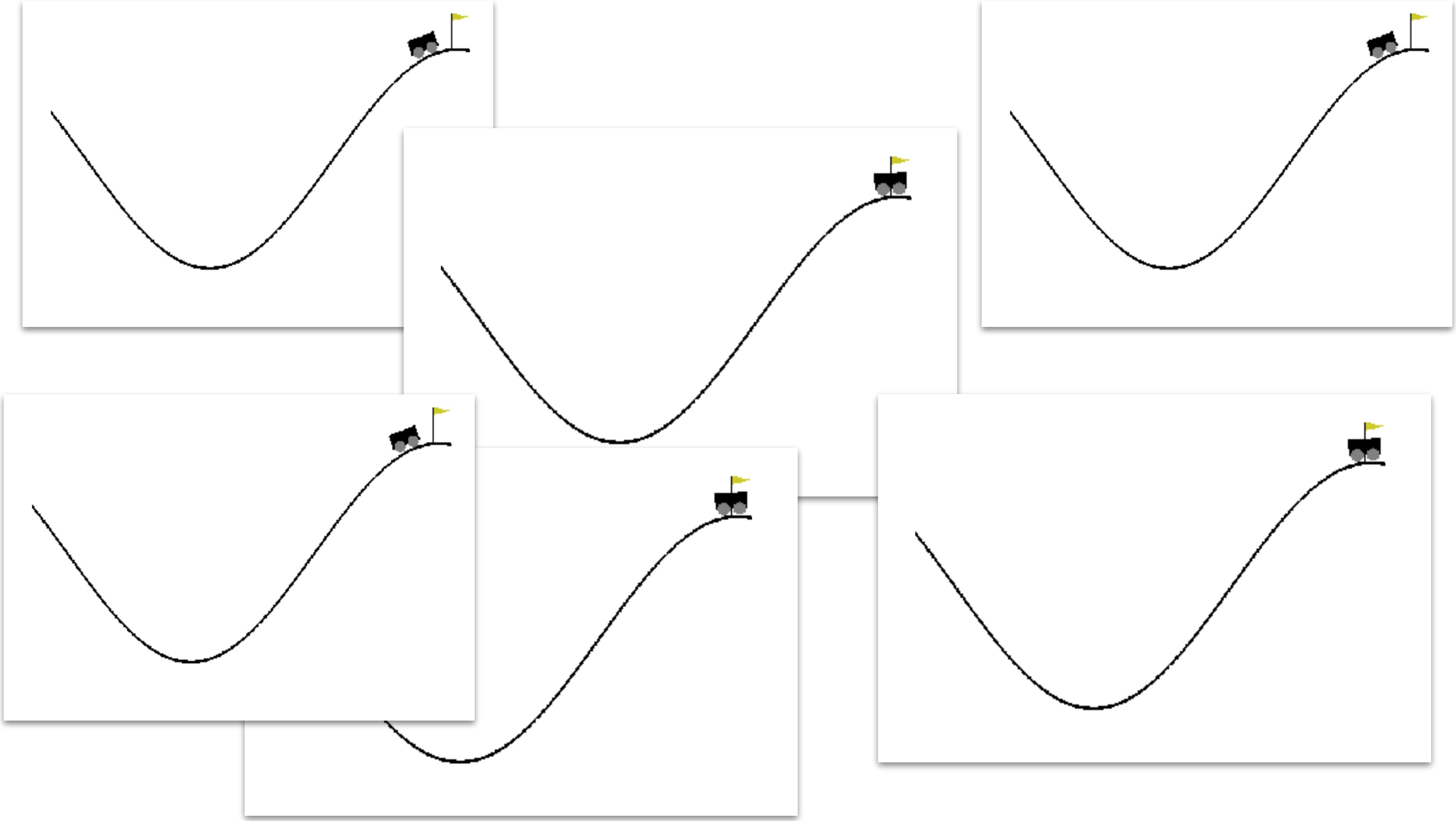
https://www.theguardian.com/news/datablog/2013/may/31/times-tables-hardest-easiest-children



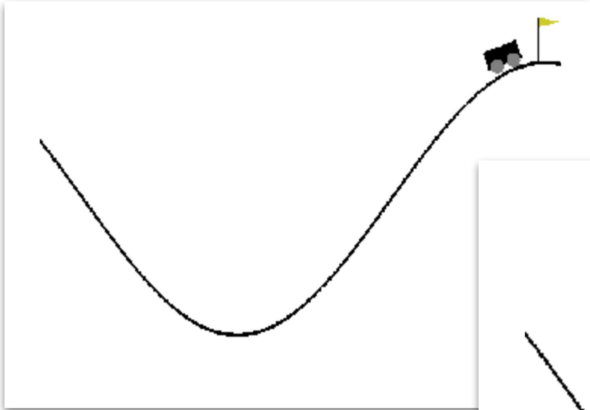
Erreur faible

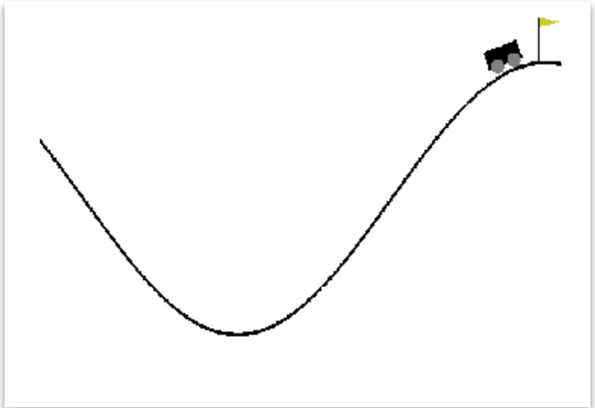
Erreur élevée !

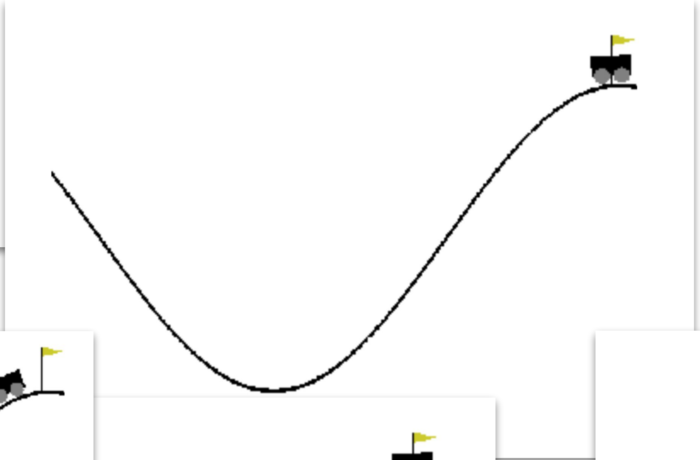


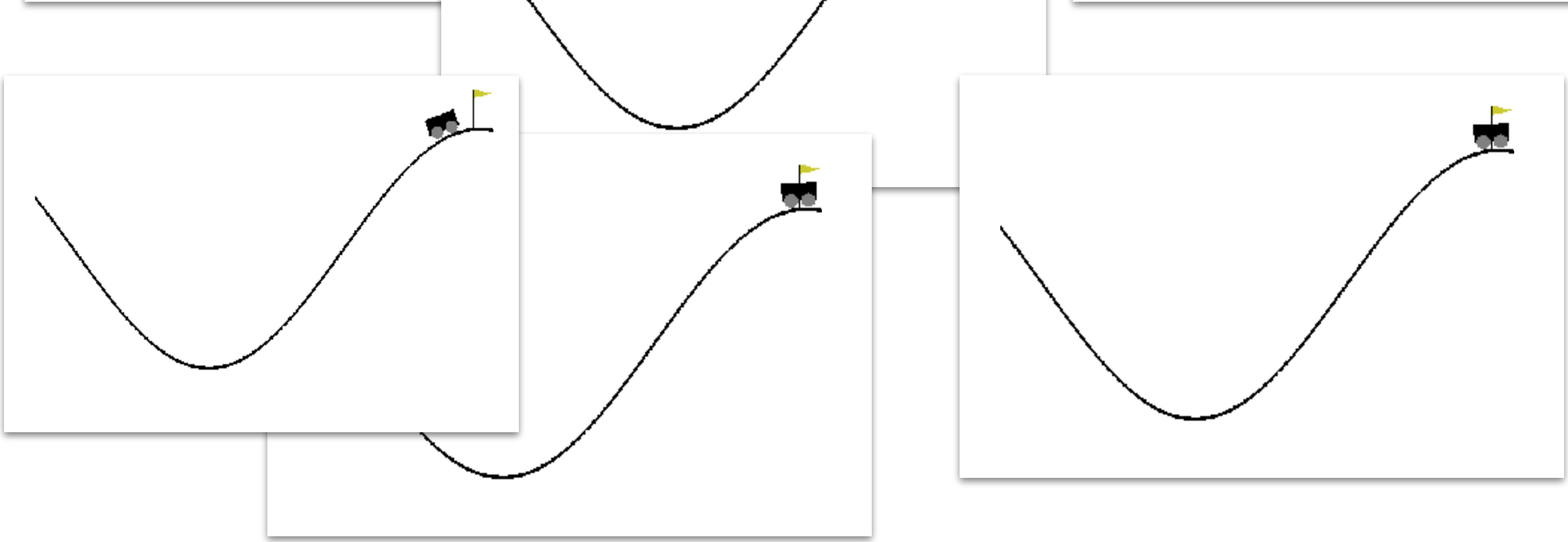


Trop de biais!







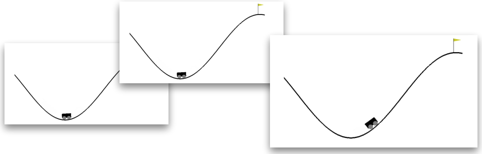


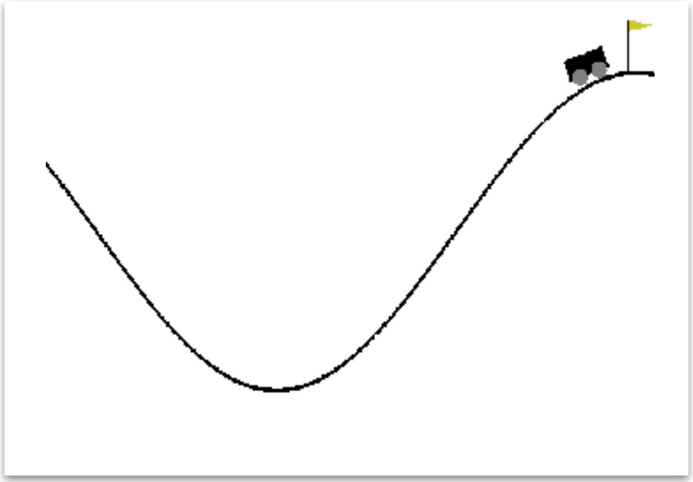
Réduisons le biais…

α

Débiaisage

Avant

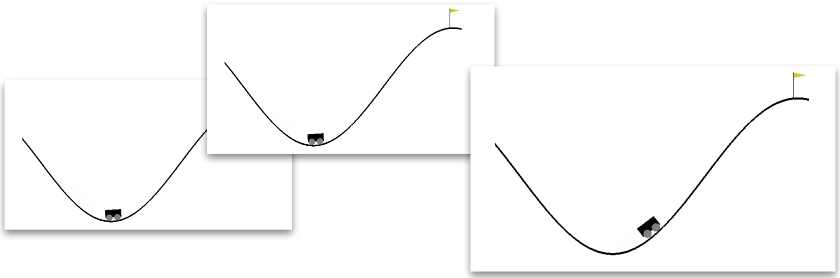


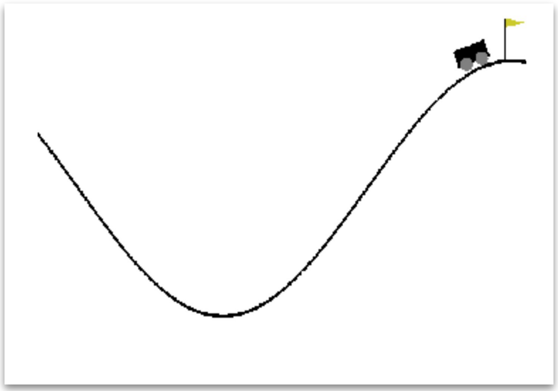


Réduisons le biais…

α Débiaisage

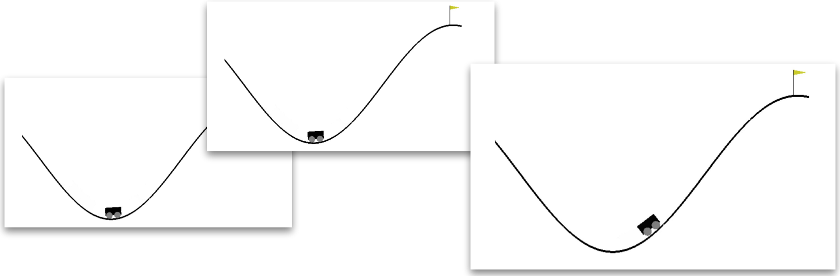
Après





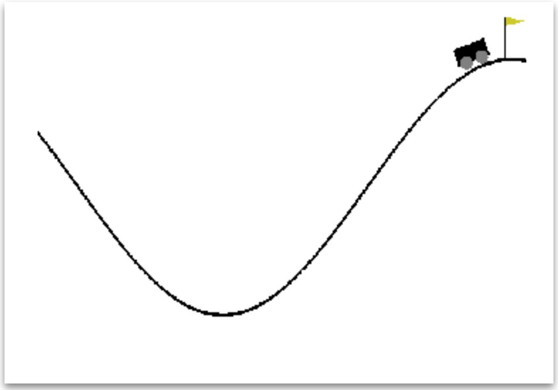
Réduisons le biais…

Après



α Débiaisage

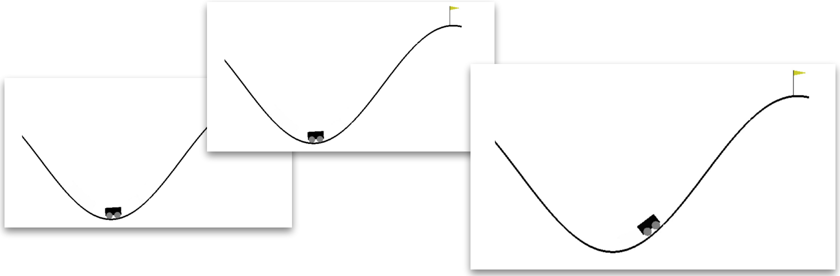
$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

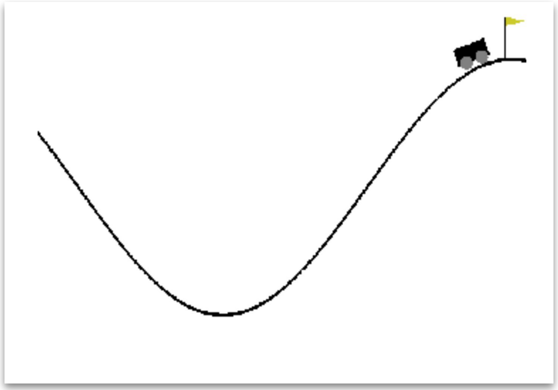


Réduisons le biais…

α Débiaisage

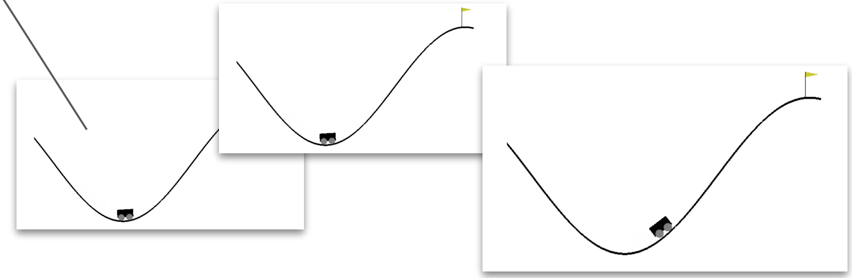
β Correction de l'échantillonnage





Réduisons le biais…

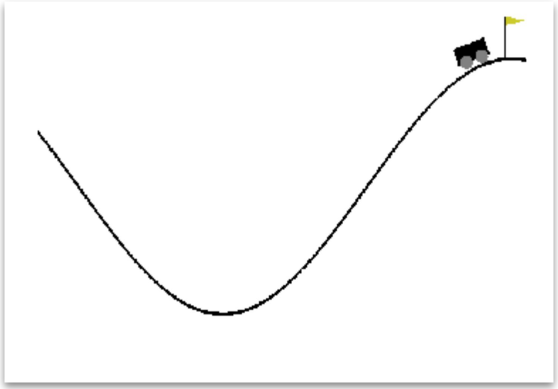
Plus d’effet



α Débiaisage

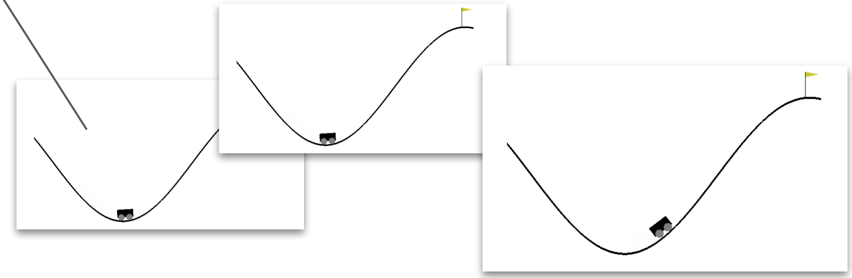
β Correction de l'échantillonnage

Moins d’effet



Réduisons le biais…

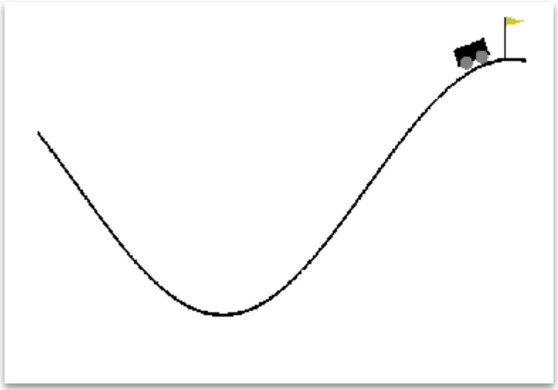
Plus d’effet

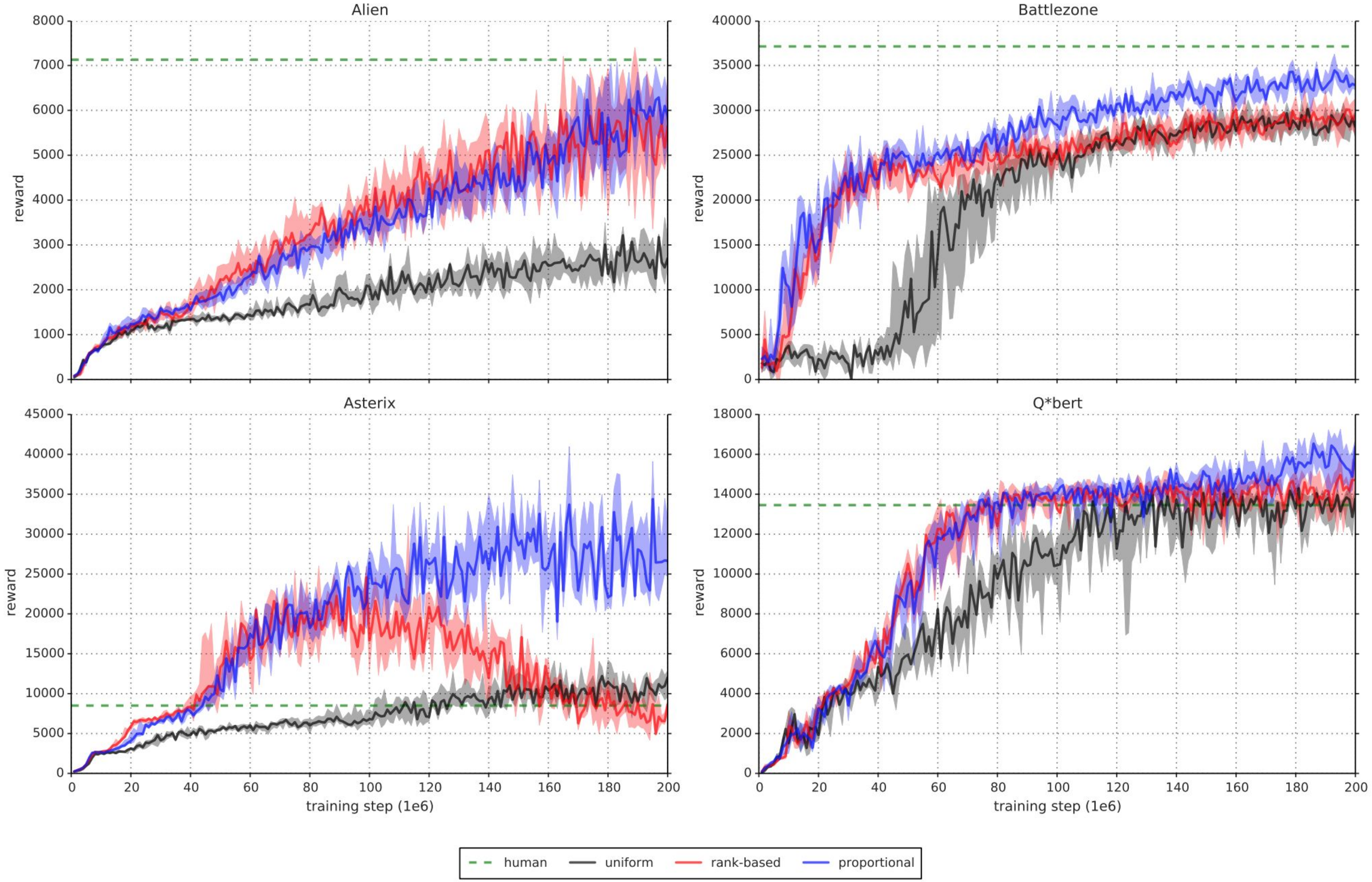


α Débiaisage
= 0.6

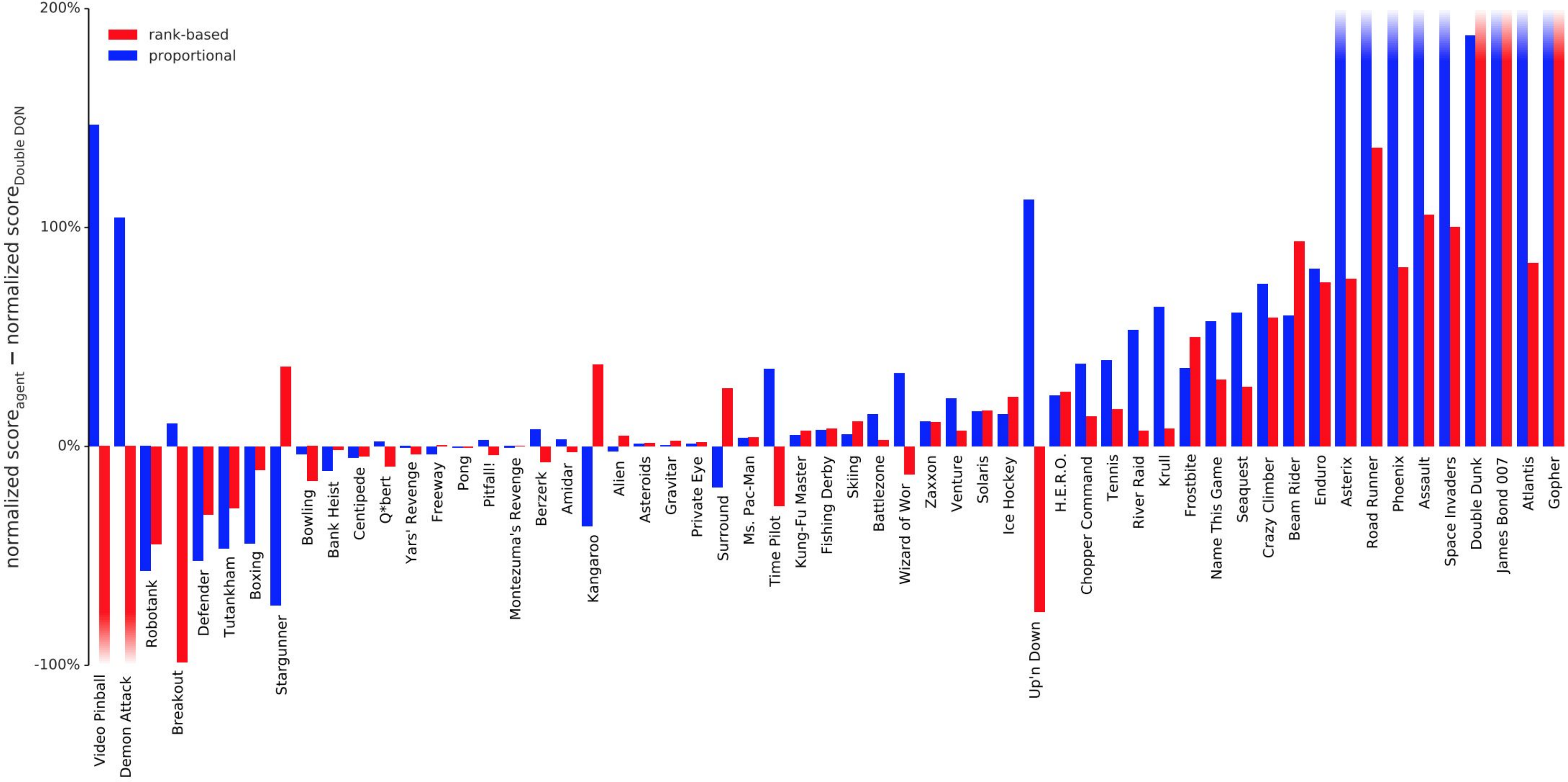
β Correction de l'échantillonnage
= 0.4

Moins d’effet



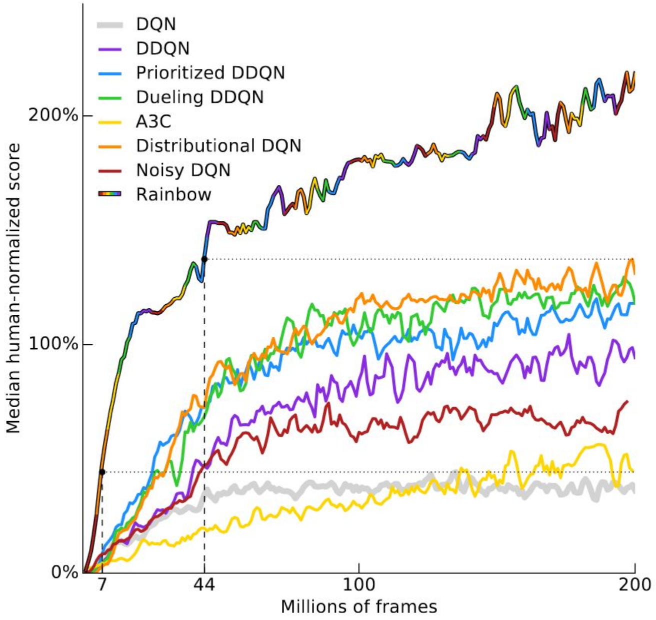


Alien
Battlezone
Asterix
Q*bert
reward
training step (1e6)
human
uniform
rank-based
proportional



rank-based
proportional
200%
100%
0%
-100%
normalized score$_{agent}$ − normalized score$_{Double\ DQN}$
Video Pinball
Demon Attack
Robotank
Breakout
Defender
Tutankham
Boxing
Stargunner
Bowling
Bank Heist
Centipede
Q*bert
Yars' Revenge
Freeway
Pong
Pitfall!
Montezuma's Revenge
Berzerk
Amidar
Kangaroo
Alien
Asteroids
Gravitar
Private Eye
Surround
Ms. Pac-Man
Time Pilot
Kung-Fu Master
Fishing Derby
Skiing
Battlezone
Wizard of Wor
Zaxxon
Venture
Solaris
Ice Hockey
Up'n Down
H.E.R.O.
Chopper Command
Tennis
River Raid
Krull
Frostbite
Name This Game
Seaquest
Crazy Climber
Beam Rider
Enduro
Asterix
Road Runner
Phoenix
Assault
Space Invaders
Double Dunk
James Bond 007
Atlantis
Gopher

Rainbow DQN



DQN
DDQN
Prioritized DDQN
Dueling DDQN
A3C
Distributional DQN
Noisy DQN
Rainbow
200%
100%
0%
Median human-normalized score
7
44
100
200
Millions of frames

A vous de jouer !

Mise en pratique

Apprentissage Prioritisé

https://colab.research.google.com/drive/156LKB0aPnKab1loUjeEKwV3VTmL7fguW