المدرسة الوطنية للذكاء الاصطناعي والرقمنة - بركان

ÉCOLE NATIONALE DE L'INTELLIGENCE ARTIFICIELLE ET DU DIGITAL - BERKANE

MACHINE LEARNING 2

Pr. BOUTAHIR Mohamed Khalifa

2024 – 2025

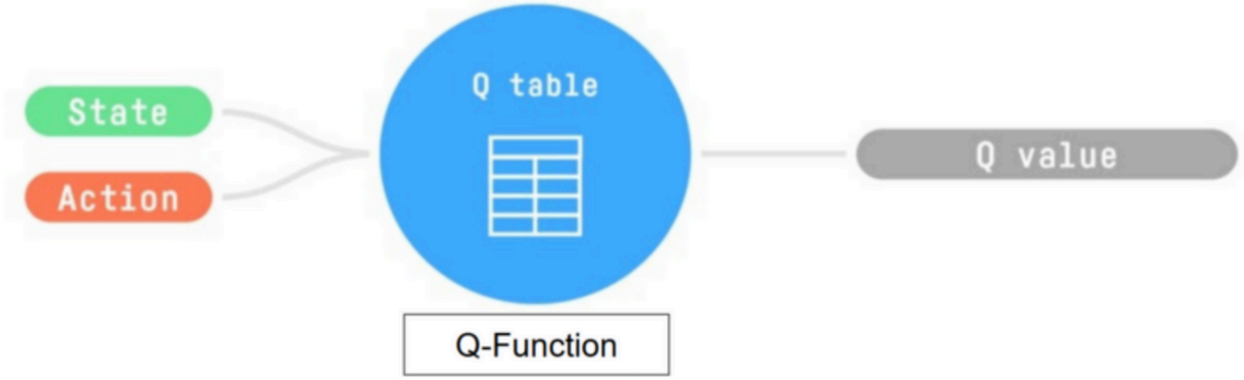
○ OBJECTIFS DE LA SESSION

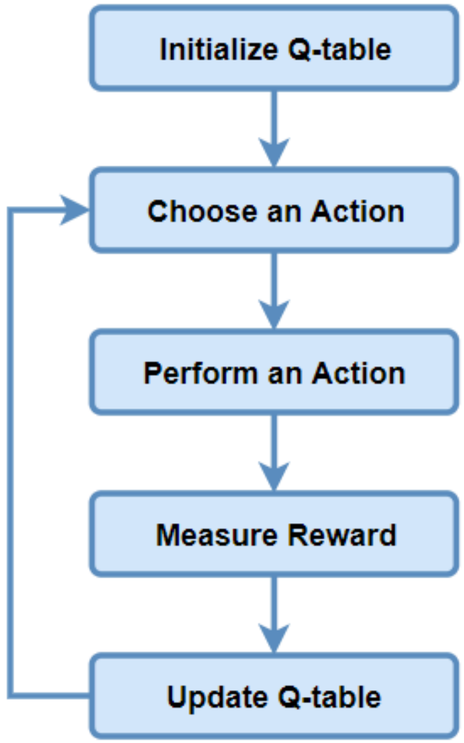
Dans cette session, nous allons approfondir l'apprentissage par renforcement profond (Deep Reinforcement Learning - DRL). Introduire les réseaux de neurones profonds pour l'apprentissage Q (Deep Q-Networks - DQN), une approche clé du DRL. Et aussi l'utilisation des frameworks TensorFlow et PyTorch pour le développement et l'entraînement des agents intelligents. L'objectif est d'acquérir une compréhension pratique et théorique des outils essentiels pour concevoir et entraîner des agents intelligents capables de prendre des décisions optimales dans des environnements complexes.

- Explorer le fonctionnement des Deep Q-Networks (DQN) et leur architecture
- Implémenter un agent basé sur DQN
- Comprendre le rôle de TensorFlow et PyTorch dans les applications RL

○ Du Q-Learning au Deep Q-Learning

Nous avons appris que le **Q-Learning** est un algorithme que nous utilisons pour entraîner notre **fonction Q**, une fonction **action-valeur** qui détermine la **valeur** d'être à un **état** particulier et d'effectuer une **action** spécifique à cet **état.**



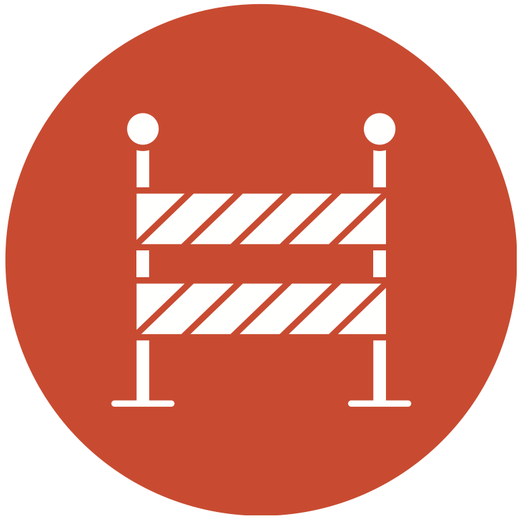


Du Q-Learning au Deep Q-Learning

Le **Q-Learning** est une **méthode tabulaire**, ce qui signifie qu'il stocke toutes les **valeurs Q** dans une **table Q**. Cela fonctionne bien lorsque l'espace **d'états** et **d'actions** est petit, mais devient irréalisable dès que ces espaces deviennent trop grands.

Q-Learning est :

- **Non évolutif :** Impossible de stocker et gérer une table Q lorsque l'espace d'états est trop vaste.
- **Généralisation limitée :** Il ne peut pas apprendre efficacement à partir d'expériences passées pour de nouveaux états.
- **Trop gourmand en mémoire :** Avec un grand nombre d'états, la table Q devient trop grande pour être stockée.
- **Inefficace sur des environnements complexes :** Il fonctionne bien sur des petits environnements comme :
  - FrozenLake (16 états)
  - CartPole (4 états)

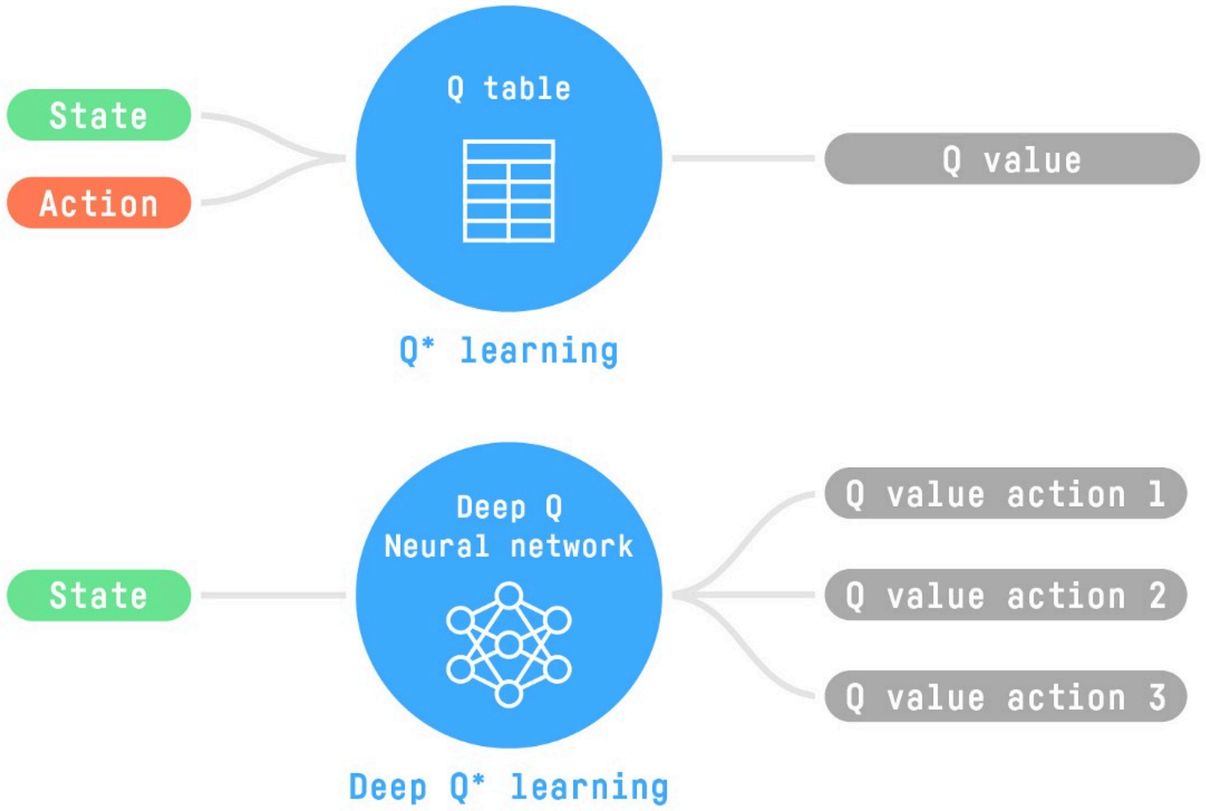


○ Du Q-Learning au Deep Q-Learning

les environnements complexes tels que **les environnements d'Atari** ont un espace d'observation très grands, par exemple sur l'image, nous avons la forme (210, 160, 3), contenant des valeurs allant de 0 à 255, ce qui nous donne **256^(210×160×3)=256^(100800)** observations possibles.

Avec un espace d'observation vaste, stocker les valeurs Q de chaque état dans une table Q devient **impossible** en raison de la mémoire requise et du nombre d'observations possibles.
C'est pourquoi nous avons besoin **d'un modèle basé sur les réseaux de neurones comme Deep Q-Network (DQN) pour approximer la fonction Q**.

○ Du Q-Learning au Deep Q-Learning



○ Du Q-Learning au Deep Q-Learning

Deep Q-Learning (DQN) utilise un **réseau de neurones profond** (**Deep Neural Network - DNN**) pour **approximer** la **fonction Q,** au lieu d'une table.

En **Deep Q-Learning (DQN), approximer les valeurs** signifie que, au lieu d'avoir une table géante, **le modèle apprend une fonction qui estime les valeurs Q pour chaque action possible dans un état donné.**

En d'autres termes :

- Nous ne stockons pas chaque état individuellement, mais nous apprenons une représentation qui peut être généralisée à de nouveaux états.
- On entraîne un réseau de neurones pour prédire les valeurs Q optimales sans avoir besoin de stocker chaque combinaison possible d'état-action.
- L'approximation permet de gérer des espaces d'états immenses, comme les images d'Atari (210×160×3 pixels).

○ Du Q-Learning au Deep Q-Learning

DQN = Q-Learning + Deep Neural Network (CNN for images) + Experience Replay + Target Network

1- Utilise un Deep Neural Network (au lieu d'une table Q)

- Le DQN remplace le table Q Simple par un réseau neuronal (souvent un CNN pour les environnements basés sur l'image comme Atari).

2- Experience Replay (mémoire tampon pour la stabilité de l'apprentissage)

- DQN utilise une mémoire tampon pour stocker les expériences passées et les réutiliser plus tard, plutôt que d'apprendre immédiatement après chaque action. Cela permet d'éviter que l'agent ne se base uniquement sur des événements récents, ce qui pourrait rendre l'apprentissage instable.

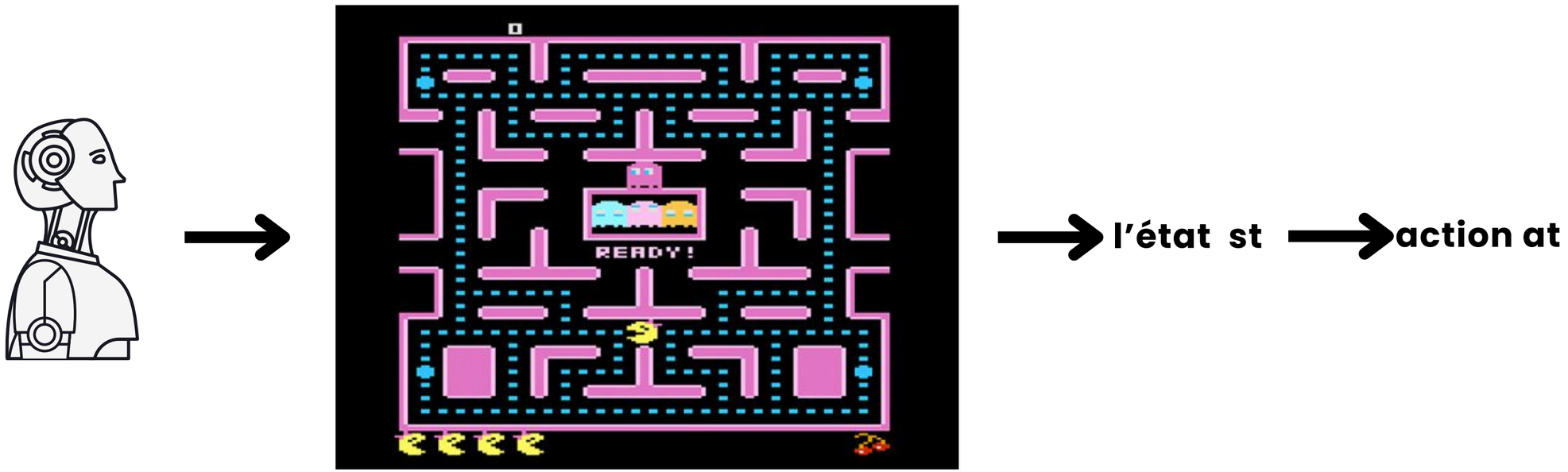
3- Target Network (stabilisation des mises à jour de la valeur Q)

- Dans Q-Learning standard, nous mettons à jour les valeurs Q en utilisant le même réseau, ce qui conduit à l'instabilité.
- DQN maintient un « réseau cible » séparé qui se met à jour lentement pour rendre l'apprentissage plus stable.

- les étapes du DQN

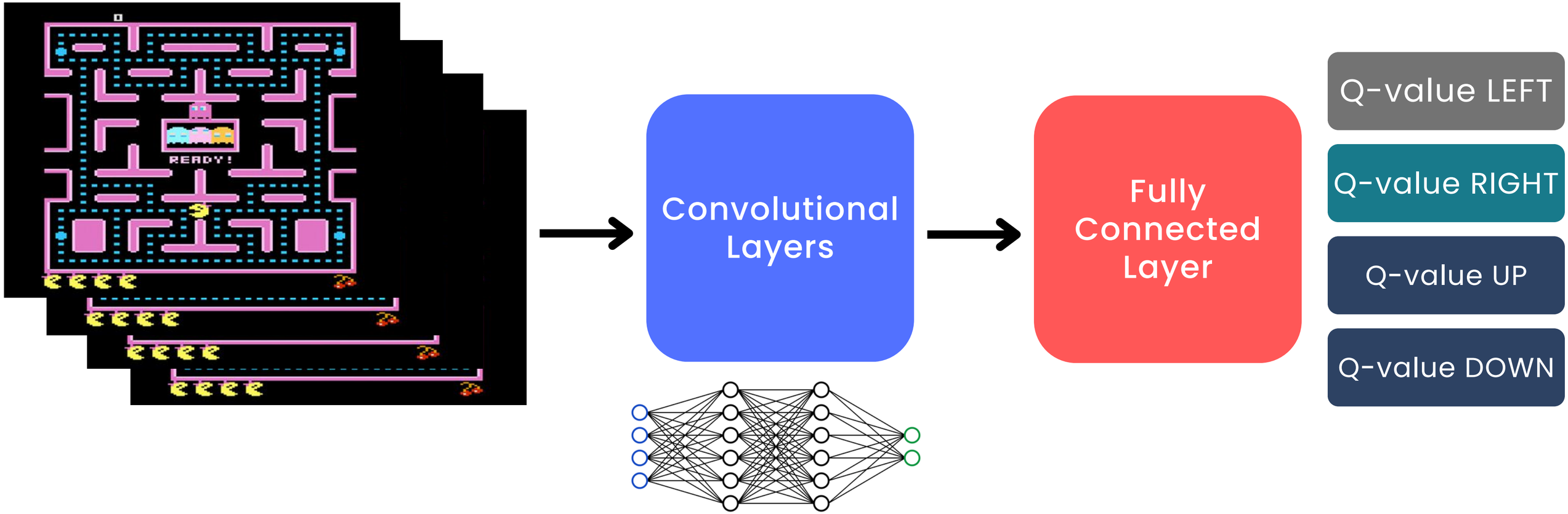
1 L'agent explore l'environnement

- L'agent commence sans aucune connaissance.
- Il observe l'état actuel st (ex: une image du jeu).
- L'objectif est de choisir une action at est l'appliquer sur l'état st.



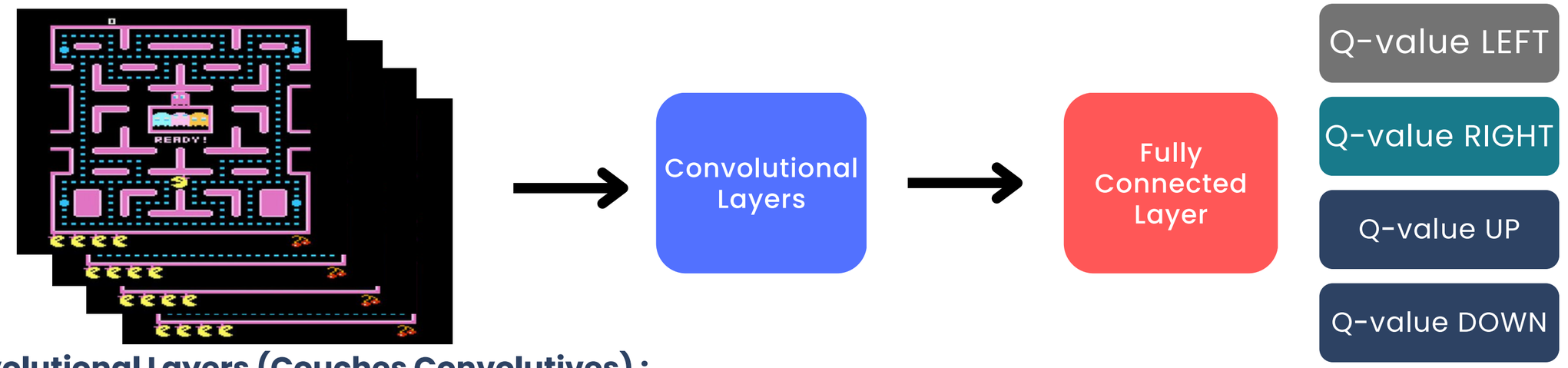
- les étapes du DQN

1 L'agent explore l'environnement



- les étapes du DQN

1 L'agent explore l'environnement



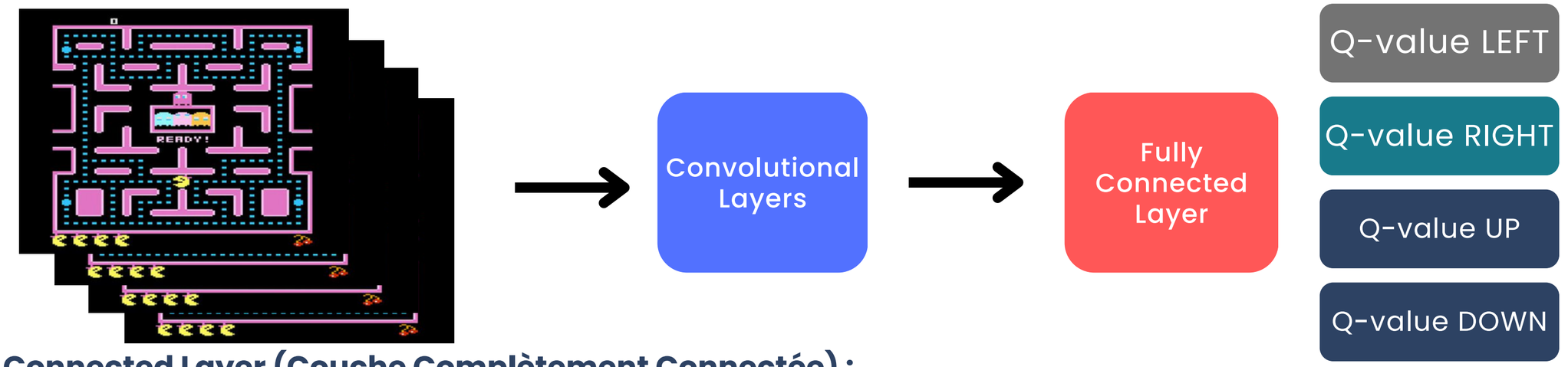
Convolutional Layers (Couches Convolutives) :

- Ces couches permettent à l'agent d'analyser l'image (ex: Pac-Man, les fantômes, les murs).
- Elles détectent les formes, les textures, et les objets dans le jeu.
- Chaque couche extrait des caractéristiques importantes pour comprendre l'environnement.

Imagine comme que l'agent a une loupe intelligente qui analyse l'image par petits morceaux pour comprendre où sont les obstacles et les éléments du jeu.

- les étapes du DQN

1 L'agent explore l'environnement



Fully Connected Layer (Couche Complètement Connectée) :

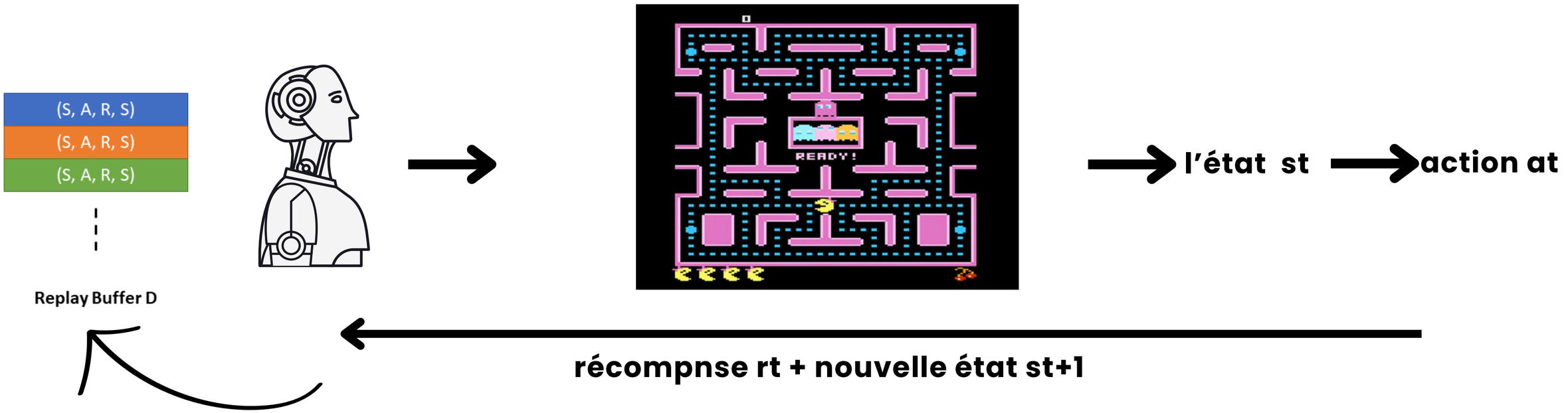
- Après avoir analysé l'image, cette couche prend toutes les caractéristiques extraites et essaie de prédire la meilleure action à prendre.
- Elle associe chaque état observé à des valeurs Q pour chaque action possible.

Imagine un chef d'orchestre qui reçoit toutes les informations des musiciens (couches convolutives) et décide quelle est la meilleure musique à jouer (quelle action choisir).

- les étapes du DQN

2 Stockage de l’expérience dans la mémoire (Experience Replay Buffer)

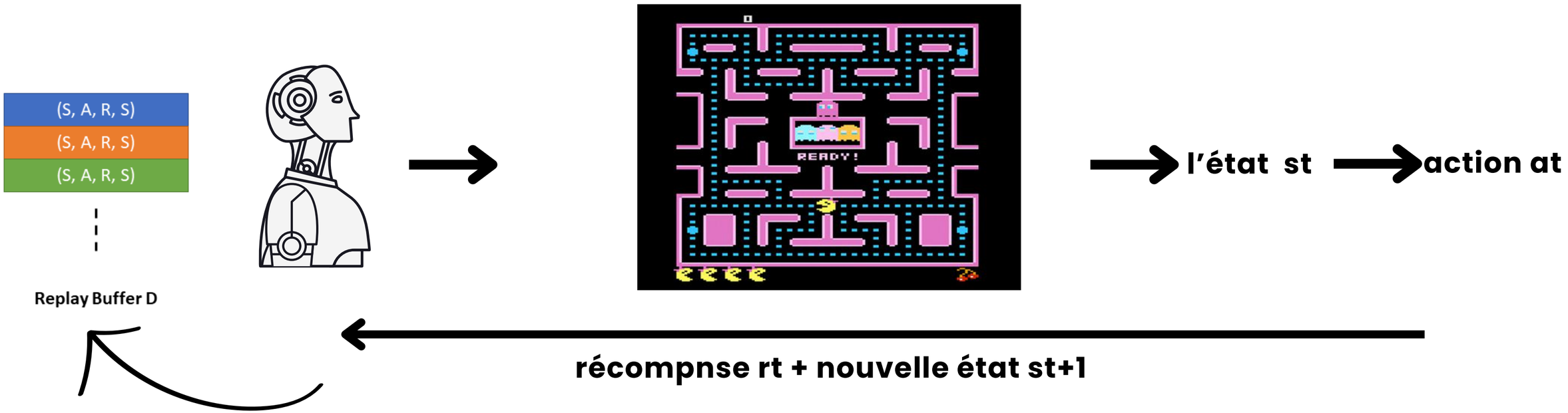
- On enregistre l’expérience sous forme de transition : (st,at,rt,st+1)
- Cette transition est ajoutée à la mémoire pour un usage futur.
- La mémoire permet de réutiliser des expériences passées de façon aléatoire pour éviter que l’agent ne soit trop biaisé par les événements récents.



- les étapes du DQN

3 Échantillonnage aléatoire de la mémoire

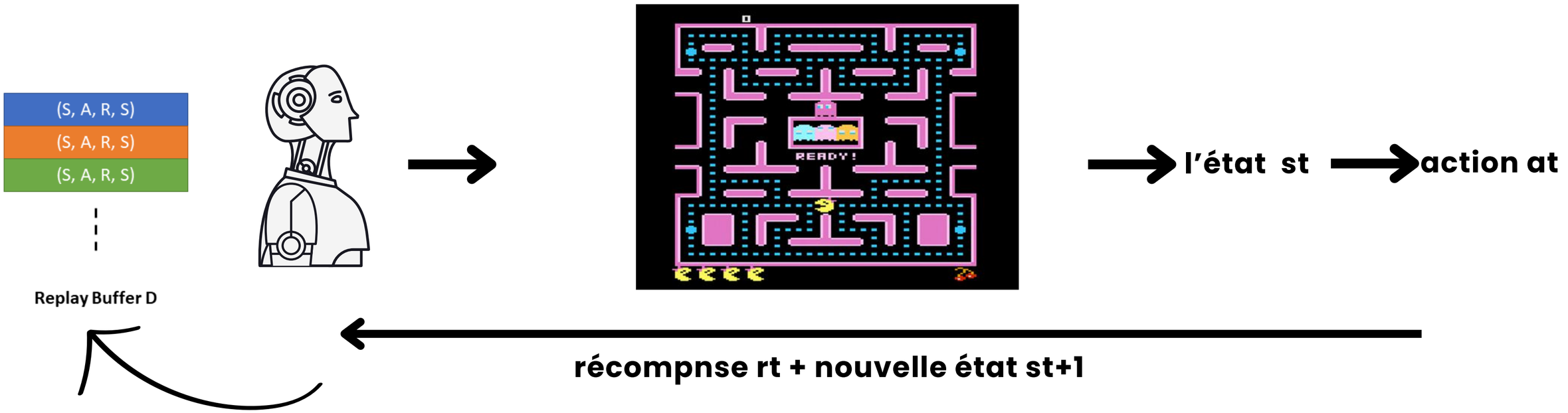
- Après avoir collecté assez de données, on sélectionne au hasard un lot (mini-batch) d'expériences depuis la mémoire.
- Cela empêche l'agent d'apprendre uniquement sur des événements récents et améliore la stabilité de l'apprentissage.



◦ les étapes du DQN

4 Entraînement du réseau de neurones principal (Online Network)

- Le réseau de neurones prend Les états St du mini-lot extrait comme entrée et prédit les valeurs Q.
- Il est entraîné à minimiser l'erreur entre ses prédictions et la cible par l'utilisation de l'équation : $y_t = r_t + \gamma \max Q(s', a')$
- On utilise la descente de gradient et la rétropropagation pour ajuster les paramètres du réseau.
- Après plusieurs cycles, l'agent devient plus performant et prend de meilleures décisions.



- les étapes du DQN

5 Utilisation du Target Network pour stabiliser l'apprentissage

- On introduit un second réseau de neurones (Target Network) qui est une copie du réseau principal, mais qui est mis à jour moins fréquemment.
- Cela empêche le réseau de s'adapter trop vite à des valeurs Q instables.
- L'équation de mise à jour devient : $y_t = r_t + \gamma \max Q_{\text{target}}(s', a')$
- Au lieu d'utiliser le réseau principal pour estimer les valeurs Q futures, on utilise le réseau cible, qui est plus stable.

- les étapes du DQN

6 Fonction de Perte dans DQN

- Dans DQN, **la fonction de perte mesure MSE** l'écart entre la valeur Q prédite par le réseau principal (Online Network) et la valeur Q cible estimée par le réseau cible (Target Network).
- Formule :

$$L(\theta) = \mathbb{E}\left[(y_t - Q(s, a; \theta))^2\right]$$

$$y_t = r_t + \gamma \max Q_{\text{target}}(s', a')$$

Valeur cible, calculée par le Target Network

Valeur Q prédite par le Online Network