Cours DQN – Section 2.2

2.2 – Le rôle du réseau de neurones dans l'estimation des valeurs Q

Dans DQN, le réseau de neurones joue le rôle d'approximateur de la fonction de valeur $Q(s, a)$. Cette section détaille la manière dont le réseau traite les entrées, génère des sorties, et apprend à partir de ses erreurs.

1. Entrée du réseau : représentation de l'état

L'entrée du réseau dépend du type d'environnement :

— Dans un jeu comme **Pong**, l'entrée est une image de l'écran (souvent 84x84 pixels), éventuellement en niveaux de gris.

— On empile généralement plusieurs frames (ex. : 4 images consécutives) pour permettre au réseau de percevoir le mouvement.

Format d'entrée typique :

$$\text{Tensor de taille } 84 \times 84 \times 4$$

2. Couches intermédiaires : extraction de caractéristiques

Le réseau contient généralement des couches **convolutionnelles**, qui sont excellentes pour traiter des images.

Ces couches permettent de :

— Détecter des motifs simples : bordures, formes, déplacement d'objets.

— Réduire la dimensionnalité tout en gardant l'information pertinente.

— Préparer les données pour la prise de décision.

3. Sortie du réseau : estimation des valeurs Q

La couche de sortie du réseau contient autant de neurones que d'actions possibles. Chaque neurone donne une estimation de la **valeur Q** pour l'action correspondante, dans l'état donné en entrée.

Exemple : Pour un jeu où l'agent peut effectuer trois actions (gauche, droite, rester), la sortie est un vecteur :

$$[Q(s, \text{gauche}), \quad Q(s, \text{droite}), \quad Q(s, \text{rester})]$$

L'agent choisira l'action avec la valeur Q maximale.

4. Mise à jour par rétropropagation

Après chaque interaction avec l'environnement, le réseau est mis à jour pour améliorer ses prédictions.

— On calcule une **valeur cible** pour l'action choisie (avec la formule du Q-learning).
— On mesure l'erreur entre la sortie du réseau et cette cible.
— On applique la **descente de gradient** pour ajuster les poids du réseau.

Fonction de perte typique : erreur quadratique moyenne (MSE) :

$$\mathcal{L} = (Q_{\text{prédit}} - Q_{\text{cible}})^2$$

5. Entraînement avec des mini-lots

Pour une meilleure stabilité :

— Le réseau est entraîné sur des **mini-batchs** d'expériences extraites aléatoirement d'une mémoire (Replay Memory).
— Cela améliore la convergence et évite l'apprentissage biaisé.

Conclusion

Le réseau de neurones dans DQN :

— Remplace la table Q traditionnelle.
— Est capable de traiter des entrées riches (images, signaux...).
— Apprend à approximer la fonction de valeur $Q(s, a)$ de manière efficace.

Dans le prochain module, nous verrons les trois idées techniques clés qui permettent à DQN de fonctionner correctement : mémoire de répétition, réseau cible, et échantillonnage aléatoire.