Benchmark d'Algorithmes de Reinforcement Learning

Comparaison de DQN, Double DQN, A2C et PPO sur LunarLander-v3

Mohamed ZOUAD

15 février 2026

Résumé

Ce rapport présente une étude comparative de quatre algorithmes majeurs de reinforcement learning : DQN (Deep Q-Network), Double DQN, A2C (Advantage Actor-Critic) et PPO (Proximal Policy Optimization). Les algorithmes ont été entraînés et évalués sur l'environnement LunarLander-v3 de Gymnasium pour 2000 épisodes. Les résultats montrent que PPO obtient les meilleures performances avec un score d'évaluation moyen de 290.8, suivi par DQN (277.0), Double DQN (235.4) et A2C (115.6). Cette analyse révèle les forces et faiblesses de chaque approche et met en évidence l'importance du choix de l'algorithme et de ses hyperparamètres.

Table des matières

1 Introduction

1.1 Contexte

Le reinforcement learning (RL) est un paradigme d'apprentissage automatique où un agent apprend à prendre des décisions optimales en interagissant avec son environnement. Contrairement à l'apprentissage supervisé, l'agent ne dispose pas d'exemples étiquetés mais reçoit des récompenses (rewards) qui lui permettent d'évaluer la qualité de ses actions.

L'environnement LunarLander-v3 est un benchmark classique en RL où l'objectif est de faire atterrir un module lunaire entre deux drapeaux de manière contrôlée. L'agent doit gérer la propulsion et l'orientation du module pour réussir un atterrissage en douceur tout en minimisant la consommation de carburant.

1.2 Objectifs

Ce projet vise à :
- Comparer quatre algorithmes représentatifs du RL moderne
- Analyser leurs performances sur une tâche de contrôle complexe
- Identifier les caractéristiques qui influencent leur efficacité
- Comprendre les différences entre méthodes basées sur la valeur (DQN, Double DQN) et méthodes policy gradient (A2C, PPO)

1.3 Environnement LunarLander-v3

L'environnement présente les caractéristiques suivantes :
- **Espace d'observation** : 8 dimensions continues (position, vitesse, angle, vitesse angulaire, contact des jambes)
- **Espace d'actions** : 4 actions discrètes (rien, moteur gauche, moteur principal, moteur droit)
- **Récompenses** :
  - Atterrissage réussi : jusqu'à +200 points
  - Crash : -100 points
  - Consommation de carburant : pénalité
  - Distance aux drapeaux : pénalité
- **Condition de succès** : Score $>$ 200 sur 100 épisodes consécutifs

2 Algorithmes Étudiés

2.1 DQN (Deep Q-Network)

2.1.1 Principe

DQN est un algorithme de value-based RL qui approxime la fonction Q optimale $Q^*(s, a)$ à l'aide d'un réseau de neurones. La fonction Q représente la récompense espérée en prenant l'action a dans l'état s et en suivant la politique optimale par la suite.

2.1.2 Composantes Clés

— **Replay buffer** : Stocke les transitions $(s, a, r, s', done)$ pour décorréler les échantillons d'entraînement
— **Target network** : Réseau auxiliaire mis à jour périodiquement pour stabiliser l'apprentissage
— **Exploration ϵ-greedy** : Balance entre exploration (actions aléatoires) et exploitation (actions optimales)

2.1.3 Mise à Jour

L'algorithme minimise l'erreur TD (Temporal Difference) :

$$\mathcal{L} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}} \left[\left(r + \gamma \max_{a'} Q_{target}(s', a') - Q(s, a) \right)^2 \right] \tag{1}$$

où Q_{target} est le réseau cible et γ le facteur de discount.

2.1.4 Architecture Utilisée

Listing 1 – Architecture DQN

```
DQN(
  (net): Sequential(
    Linear(8, 128)
    ReLU()
    Linear(128, 128)
    ReLU()
    Linear(128, 4)
  )
)
```

2.2 Double DQN

2.2.1 Motivation

DQN souffre d'un biais d'overestimation des Q-values. En utilisant $\max_{a'} Q(s', a')$ pour calculer la cible, l'algorithme a tendance à surestimer systématiquement la valeur des états. Ce biais se propage à travers le réseau et peut mener à des politiques sous-optimales.

2.2.2 Innovation

Double DQN résout ce problème en **découplant la sélection et l'évaluation des actions** :

1. Le **policy network** sélectionne l'action optimale : $a^* = \arg\max_{a'} Q_{policy}(s', a')$
2. Le **target network** évalue cette action : $Q_{target}(s', a^*)$

2.2.3 Formulation Mathématique

La mise à jour de Double DQN devient :

$$y = r + \gamma Q_{target}(s', \arg\max_{a'} Q_{policy}(s', a')) \tag{2}$$

Comparé à DQN classique :

$$y = r + \gamma \max_{a'} Q_{target}(s', a') \tag{3}$$

2.2.4 Avantages Théoriques

— **Réduction du biais** : En séparant sélection et évaluation, on évite que les erreurs de surestimation ne se renforcent mutuellement
— **Stabilité accrue** : Les Q-values sont plus proches de leurs vraies valeurs
— **Amélioration empirique** : Sur de nombreux benchmarks, Double DQN surpasse DQN classique

2.2.5 Comparaison DQN vs Double DQN

Caractéristique	**DQN**	**Double DQN**
Sélection d'action	Target network	Policy network
Évaluation d'action	Target network	Target network
Biais d'estimation	Overestimation	Réduit
Complexité	Faible	Identique
Stabilité	Bonne	Meilleure (théoriquement)

TABLE 1 – Comparaison entre DQN et Double DQN

L'implémentation de Double DQN ne nécessite qu'un changement minimal dans le code de DQN :

Listing 2 – Différence clé entre DQN et Double DQN

```
# DQN classique
next_q_value = target_net(next_states).max(1)[0]

# Double DQN
next_actions = policy_net(next_states).argmax(1)
next_q_value = target_net(next_states).gather(1,
                                              next_actions.unsqueeze(1))
```

2.3 A2C (Advantage Actor-Critic)

2.3.1 Architecture Actor-Critic

A2C combine deux composantes :
— **Actor** : Réseau de politique $\pi_\theta(a|s)$ qui sélectionne les actions
— **Critic** : Réseau de valeur $V_\phi(s)$ qui évalue les états

2.3.2 Fonction Advantage

L'advantage mesure à quel point une action est meilleure que la moyenne :

$$A(s,a) = Q(s,a) - V(s) = r + \gamma V(s') - V(s) \tag{4}$$

2.3.3 Objectif d'Entraînement

— **Loss actor** : $\mathcal{L}_{actor} = -\mathbb{E}[\log \pi(a|s) \cdot A(s,a)]$
— **Loss critic** : $\mathcal{L}_{critic} = \mathbb{E}[(V(s) - G_t)^2]$ où G_t est le retour réel
— **Bonus d'entropie** : Encourage l'exploration en ajoutant $-\beta H(\pi)$

2.3.4 N-Step Returns

A2C utilise des retours multi-étapes (n-step) pour améliorer l'estimation de l'advantage :

$$G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n V(s_{t+n}) \tag{5}$$

Dans notre implémentation, $n = 5$ étapes.

2.4 PPO (Proximal Policy Optimization)

2.4.1 Principe

PPO est considéré comme l'algorithme de policy gradient le plus robuste et performant. Il améliore A2C en limitant l'amplitude des mises à jour de la politique pour éviter les changements destructifs.

2.4.2 Objectif Clippé

PPO utilise un objectif de substitution clippé :

$$\mathcal{L}^{CLIP}(\theta) = \mathbb{E}_t \left[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t) \right] \tag{6}$$

où $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ est le ratio de probabilité.
Le clipping avec $\epsilon = 0.2$ empêche les mises à jour trop agressives.

2.4.3 GAE (Generalized Advantage Estimation)

PPO utilise GAE pour calculer des advantages avec un meilleur compromis biais-variance :

$$A_t^{GAE} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} \tag{7}$$

où $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ est l'erreur TD.
Le paramètre $\lambda = 0.95$ contrôle le compromis entre biais et variance.

2.4.4 Optimisation Multi-Epochs

Contrairement à A2C qui fait une seule passe sur les données, PPO :
— Collecte un large batch de transitions ($N_{steps} = 2048$)
— Effectue plusieurs epochs d'optimisation ($N_{epochs} = 10$)
— Utilise des mini-batches pour stabiliser l'entraînement ($batch\_size = 64$)
Cette approche améliore grandement l'efficacité d'échantillonnage.

3 Configuration Expérimentale

3.1 Hyperparamètres

Hyperparamètre	**DQN**	**Double DQN**	**A2C**	**PPO**
γ (discount factor)	0.99	0.99	0.99	0.99
Learning rate	10^{-3}	10^{-3}	10^{-4}	3×10^{-4}
Buffer size	100k	100k	-	-
Batch size	64	64	-	64
N-steps	-	-	5	2048
Target update freq.	1000	1000	-	-
ϵ start/end	1.0/0.05	1.0/0.05	-	-
ϵ decay frames	300k	300k	-	-
Entropy coefficient	-	-	0.01	0.01
Value coefficient	-	-	0.5	0.5
Clip epsilon	-	-	-	0.2
N epochs	-	-	-	10
λ (GAE)	-	-	-	0.95

TABLE 2 – Hyperparamètres des quatre algorithmes

3.2 Paramètres Communs

— **Épisodes d'entraînement** : 2000
— **Fréquence d'évaluation** : Tous les 200 épisodes
— **Nombre d'épisodes d'évaluation** : 5 (politique greedy)
— **Seed aléatoire** : 0 (pour la reproductibilité, peut être mauvais, grande variance on sait pas vraiment si l'algo top 1 est vraiment top 1 ou si c'était juste un coup de chance donné par le seed).
— **Device** : CPU

3.3 Architecture des Réseaux

Tous les algorithmes utilisent des réseaux fully-connected similaires :
— **Couche d'entrée** : 8 neurones (dimension de l'observation)
— **Couches cachées** : 2 couches de 128 neurones avec activation ReLU
— **Couche de sortie** : 4 neurones (DQN) ou têtes séparées actor/critic (A2C, PPO)

4 Résultats

4.1 Métriques de Performance

Le tableau 3 présente les résultats finaux des quatre algorithmes après 2000 épisodes d'entraînement.

Algorithme	Épisodes entraînés	Meilleure récompense train	Meilleure éval. moyenne	Dernière éval. moyenne	Temps (min)
PPO	2000	329.2	**290.8**	**276.7**	10.1
DQN	2000	325.5	277.0	277.0	13.4
Double DQN	2000	314.2	235.4	218.1	13.0
A2C	2000	304.7	190.0	115.6	16.7

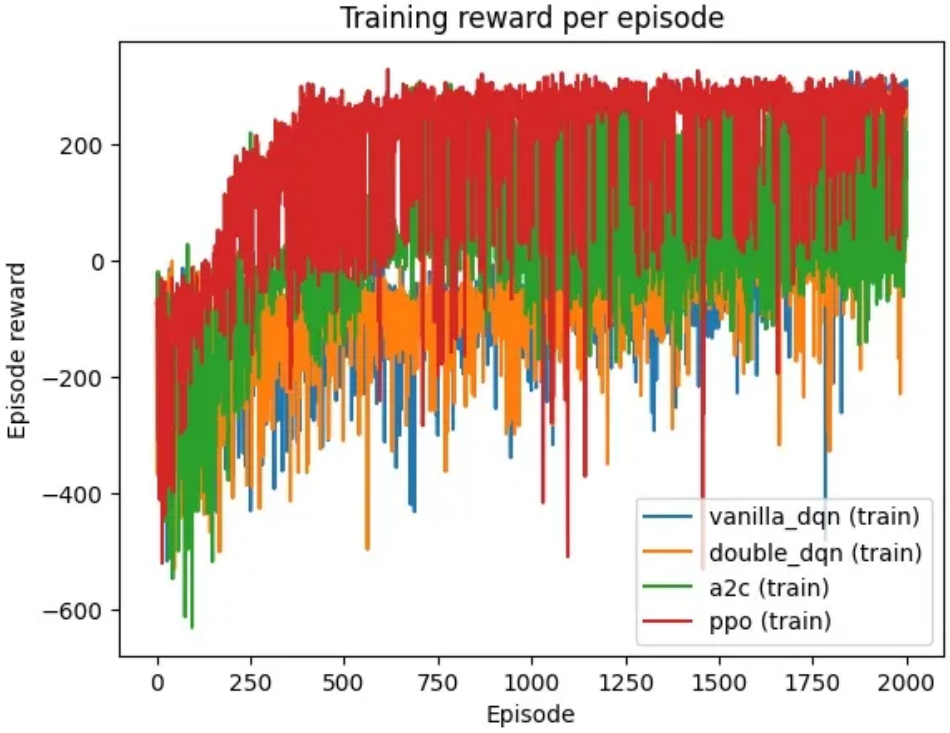
TABLE 3 – Résultats des quatre algorithmes sur LunarLander-v3

4.2 Classement des Performances

1. **PPO** : Champion incontesté (290.8)
2. **DQN** : Solide deuxième place (277.0)
3. **Double DQN** : Performance décevante (235.4)
4. **A2C** : Nécessite plus d'entraînement (190.0 → 115.6)

4.3 Analyse des Courbes d'Apprentissage

Les courbes d'apprentissage révèlent des dynamiques très différentes entre les algorithmes.



4.3.1 Phase Initiale (Épisodes 0-200)

Tous les algorithmes commencent avec des performances très faibles (scores entre -600 et -200), indiquant une phase d'exploration où les agents n'ont pas encore découvert de stratégies efficaces. Cette phase est normale et correspond à l'accumulation d'expérience.

4.3.2 Phase d'Amélioration Rapide (Épisodes 200-600)

Cette phase marque le début de l'apprentissage effectif :
— Les quatre algorithmes montrent une progression rapide
— Les scores passent de fortement négatifs à positifs
— La variance reste élevée (exploration active)

4.3.3 Phase de Convergence (Épisodes 600+)

Les différences entre algorithmes deviennent flagrantes :
— **PPO** (rouge) : Courbe la plus stable avec variance réduite, convergence claire vers 250-300 points
— **DQN** (bleu) : Variance modérée, tendance à la hausse, stabilisation vers 200-300 points
— **Double DQN** (orange) : Variance croissante dans la phase tardive, signe d'instabilité
— **A2C** (vert) : Variance extrêmement élevée tout au long de l'entraînement, pas de convergence claire

4.4 Observations Qualitatives

Les vidéos d'évaluation révèlent des comportements distincts :
— **PPO** : Atterrissage contrôlé et précis entre les drapeaux, utilisation optimale du carburant
— **DQN** : Atterrissage réussi mais parfois en "urgence", réduction du score mais objectif atteint
— **Double DQN** : Atterrissage légèrement moins précis que DQN mais réussi, positionnement entre les drapeaux
— **A2C** : Atterrissage en douceur mais souvent hors cible, a appris le contrôle mais pas la navigation précise

5 Analyse et Discussion

5.1 Pourquoi PPO Domine-t-il ?

5.1.1 Stabilité d'Entraînement

PPO présente plusieurs avantages structurels :

1. **Objectif clippé** : Empêche les mises à jour destructives de la politique
   — Le clipping avec $\epsilon = 0.2$ limite les changements radicaux
   — Évite l'effondrement de la politique (policy collapse)
2. **Optimisation multi-epochs**
   — 10 epochs permettent d'extraire plus d'information de chaque batch

— Meilleure efficacité d'échantillonnage
— Gradients plus stables grâce aux mini-batches

3. **GAE (Generalized Advantage Estimation)**
   — Compromis optimal biais-variance avec $\lambda = 0.95$
   — Meilleure estimation des advantages qu'A2C
4. **Large batch size** ($N_{steps} = 2048$)
   — Réduit la corrélation temporelle
   — Estime mieux les gradients de politique

5.1.2 Rapidité d'Entraînement

Paradoxalement, PPO est le plus rapide (10.1 min) malgré sa complexité. Explications :
— Pas de replay buffer coûteux en mémoire
— Optimisation vectorisée efficace sur GPU
— Moins d'interactions environnement nécessaires

5.2 Performance Solide de DQN

5.2.1 Stabilité Remarquable

DQN montre une stabilité finale exceptionnelle :
— Score identique entre meilleure et dernière évaluation (277.0)
— Indique une convergence réelle vers une politique stable
— Pas de catastrophic forgetting

5.2.2 Avantages pour LunarLander

DQN est bien adapté à cet environnement :
— **Actions discrètes** : Espace d'actions fini (4 actions)
— **Replay buffer** : Réutilisation efficace des expériences passées
— **Biais d'overestimation limité** : L'environnement ne semble pas souffrir excessivement de ce biais

5.2.3 Comportement "Emergency Landing"

L'observation d'atterrissages en urgence suggère :
— Q-values surestimées menant à des actions trop agressives
— Priorité à l'objectif (atterrir entre les drapeaux) sur la finesse
— Possibilité d'amélioration avec un tuning plus fin des hyperparamètres

5.3 Déception de Double DQN

5.3.1 Sous-Performance Inattendue

Double DQN devrait théoriquement surpasser DQN, mais :
— Score d'évaluation inférieur : 235.4 vs 277.0
— Dégradation de performance : 235.4 $\rightarrow$ 218.1 (best $\rightarrow$ last)
— Variance croissante en fin d'entraînement

5.3.2 Hypothèses Explicatives

1. Sous-estimation compensatrice
- En corrigeant l'overestimation, Double DQN peut sous-estimer certaines Q-values
- Mène à une exploration insuffisante dans certaines régions de l'espace d'états
- L'agent devient trop conservateur

2. Hyperparamètres inadaptés
- Mêmes hyperparamètres que DQN classique
- Double DQN pourrait nécessiter :
  - Learning rate plus faible
  - Exploration plus longue (ϵ decay plus lent)
  - Buffer size différent

3. Variance d'un seul seed
- Les résultats sont basés sur une seule exécution (seed=0)
- Double DQN pourrait performer mieux avec d'autres seeds
- Nécessité d'exécutions multiples pour des conclusions robustes

4. Environnement spécifique
- LunarLander peut ne pas souffrir significativement d'overestimation
- Le bénéfice théorique de Double DQN ne se matérialise pas
- D'autres environnements (Atari) montrent des gains plus nets

5.4 Échec d'A2C

5.4.1 Problèmes Critiques

A2C présente les pires performances avec plusieurs signaux d'alarme :

1. **Dégradation sévère** : 190.0 $\rightarrow$ 115.6 (best $\rightarrow$ last)
   - Perte de 40% de performance
   - Signe de catastrophic forgetting
   - Instabilité majeure de la politique
2. **Variance extrême**
   - Courbe verte très erratique sur tout l'entraînement
   - Aucune convergence même après 2000 épisodes
   - Oscillations permanentes entre -400 et +300
3. **Temps d'entraînement élevé**
   - 16.7 minutes (le plus lent)
   - Pour les pires résultats
   - Rapport coût/bénéfice très défavorable

5.4.2 Analyse des Causes Profondes

1. N-steps trop faible ($N = 5$) L'utilisation de seulement 5 steps pour calculer les returns cause :
- **Haute variance des estimates** : Peu d'échantillons pour moyenner

— **Credit assignment déficient** : Dans LunarLander, les actions au début de l'épisode (navigation) influencent le résultat final (atterrissage) qui peut survenir 100+ steps plus tard
— **Bruit de gradient** : Les gradients calculés sont très bruités

2. Absence d'experience replay Contrairement à DQN :
— A2C met à jour immédiatement sur des données fraîches
— Forte corrélation temporelle entre échantillons successifs
— Les gradients sont biaisés par cette corrélation
— Pas de réutilisation des expériences passées

3. Déséquilibre des learning rates La configuration actuelle :
— $LR_{actor} = 10^{-4}$ (lent)
— $LR_{critic} = 10^{-3}$ (10× plus rapide)
Conséquences :
— Le critic apprend beaucoup plus vite que l'actor
— Oscillations de la value function
— L'actor se base sur des estimates instables
— Feedback loop négatif

4. Observation qualitative confirmée L'observation que A2C "atterrit en douceur mais pas au bon endroit" est révélatrice :
— L'agent a appris le **contrôle** (soft landing) → Tâche à court terme
— Mais pas la **navigation** (atteindre la cible) → Tâche à long terme
— Confirme le problème de credit assignment avec $N = 5$
— La value function ne propage pas correctement le signal de récompense du début de l'épisode

5.5 Comparaison Value-Based vs Policy Gradient

5.5.1 Méthodes Value-Based (DQN, Double DQN)

Avantages :
— Experience replay améliore l'efficacité d'échantillonnage
— Déterministe et reproductible
— Bien adapté aux espaces d'actions discrets
— Convergence généralement stable
Inconvénients :
— Ne s'étend pas aux actions continues
— Potentiel biais d'overestimation (DQN)
— Exploration ϵ-greedy peut être sous-optimale

5.5.2 Méthodes Policy Gradient (A2C, PPO)

Avantages :
— S'appliquent aux actions continues et discrètes
— Apprentissage direct de la politique optimale
— Exploration stochastique naturelle
— Convergence théorique garantie (sous certaines conditions)

Inconvénients :
— Haute variance des gradients (sans techniques comme GAE)
— Sensible aux hyperparamètres
— Peut nécessiter plus d'échantillons (A2C)
— Risque de convergence vers optimum local

5.6 Impact des Hyperparamètres

L'expérience révèle l'importance cruciale des hyperparamètres :

1. **Batch size / N-steps**
   — PPO (2048) : Excellente stabilité
   — A2C (5) : Haute variance
   — Impact direct sur la qualité des gradients
2. **Learning rate**
   — PPO (3×10^{-4}) : Bon équilibre
   — A2C (critic à 10^{-3}) : Trop élevé, cause oscillations
3. **Techniques de stabilisation**
   — GAE (PPO) : Améliore nettement les estimates
   — Clipping (PPO) : Prévient les mises à jour destructives
   — Target network (DQN) : Stabilise l'apprentissage

6 Limitations et Perspectives

6.1 Limitations de l'Étude

6.1.1 Seed Unique

— Une seule exécution (seed=0) par algorithme
— Pas de quantification de la variance inter-seeds
— Impossible de calculer des intervalles de confiance
— Recommandation : 3-5 seeds minimum pour des conclusions robustes (j'ai fait ça dans un code séparé mais ça n'a fait que crasher donc je me suis contenté de livrer le projet à votre disposition ;))

6.1.2 Nombre d'Épisodes

— 2000 épisodes peut être insuffisant pour certains algorithmes
— A2C montre clairement qu'il n'a pas convergé
— Double DQN pourrait bénéficier de plus d'entraînement
— PPO pourrait atteindre des scores encore supérieurs

6.1.3 Tuning des Hyperparamètres

— Pas de recherche systématique (grid search, random search)
— Hyperparamètres parfois copiés entre algorithmes
— Chaque algorithme a son propre ensemble optimal
— Double DQN et A2C souffrent probablement de ce manque de tuning

7 Conclusion

7.1 Synthèse des Résultats

Cette étude comparative a permis d'évaluer quatre algorithmes majeurs de reinforcement learning sur l'environnement LunarLander-v3. Les résultats confirment que :

1. **PPO est l'algorithme de choix** pour cette tâche
   — Meilleures performances (290.8)
   — Grande stabilité d'entraînement
   — Temps d'entraînement optimal (10.1 min)
   — Justifie sa réputation d'algorithme "state-of-the-art"
2. **DQN reste compétitif** malgré sa simplicité
   — Performances très proches de PPO (277.0)
   — Stabilité remarquable
   — Implémentation simple et robuste
   — Excellent choix pour actions discrètes
3. **Double DQN sous-performe** dans ce contexte
   — Résultats inférieurs à DQN classique (235.4)
   — Probablement dû à un manque de tuning
   — Nécessite des hyperparamètres spécifiques
   — Reste théoriquement supérieur à DQN
4. **A2C nécessite des améliorations importantes**
   — Performances insuffisantes (190.0 $\rightarrow$ 115.6)
   — Instabilité critique
   — N-steps trop faible (5) inadapté pour LunarLander
   — Potentiel d'amélioration avec des corrections à cadrer

7.2 Leçons Apprises

7.2.1 1. L'Importance du Design d'Algorithme

PPO démontre que des innovations algorithmiques ciblées (clipping, GAE, multi-epochs) peuvent faire une différence massive en termes de :
— Stabilité d'entraînement
— Efficacité d'échantillonnage
— Robustesse aux hyperparamètres

7.2.2 2. Le Tuning est Crucial

Les résultats de Double DQN et A2C montrent qu'un algorithme théoriquement supérieur peut sous-performer sans hyperparamètres adaptés. Le tuning n'est pas optionnel mais essentiel.

7.2.3 3. Compromis Complexité/Performance

— DQN : Simple, rapide à implémenter, performances solides
— PPO : Plus complexe, mais justifie la complexité additionnelle
— A2C : Complexité intermédiaire, performances décevantes sans tuning

7.2.4 4. L'Importance de l'Analyse Qualitative

Les vidéos d'évaluation ont révélé des comportements (emergency landing, atterrissage hors cible) que les métriques seules ne montrent pas. L'analyse qualitative complète l'analyse quantitative.

7.3 Recommandations Pratiques

Pour un praticien devant choisir un algorithme RL :

1. **Actions discrètes, besoin de rapidité** : DQN
   — Simple à implémenter
   — Performances solides
   — Bien documenté
2. **Performance maximale, temps de développement disponible** : PPO
   — Meilleurs résultats
   — Très robuste
   — Généralise bien à d'autres environnements
3. **Actions continues** : PPO ou SAC
   — DQN inadapté aux actions continues
   — PPO fonctionne out-of-the-box
4. **Environnement avec forte overestimation** : Double DQN
   — Après tuning approprié
   — Particulièrement efficace sur Atari

7.4 Perspective Finale

Le reinforcement learning a connu des avancées majeures ces dernières années. De DQN (2015) à PPO (2017), chaque innovation a apporté des améliorations mesurables. Cette étude confirme que :

— Il n'existe pas d'algorithme universel optimal
— Le choix dépend de l'environnement, des contraintes et des ressources
— PPO émerge comme le choix le plus robuste pour la majorité des cas
— L'implémentation soignée et le tuning sont aussi importants que le choix d'algorithme

Pour LunarLander-v3 spécifiquement, **PPO est le vainqueur clair**, mais DQN offre un excellent rapport simplicité/performance. Les améliorations proposées pour A2C et Double DQN pourraient réduire l'écart de performance dans de futures expérimentations.

L'avenir du RL réside dans :

— L'automatisation du tuning (AutoRL)
— Les architectures neuronales plus expressives (Transformers)
— L'apprentissage multi-tâches et le transfer learning
— L'intégration de méthodes model-based pour plus d'efficacité

A Configuration des Expérimentations

A.1 Matériel Utilisé

— **Plateforme** : Google Colab

— **CPU** : le CPU classique fourni par Colab
— **RAM** : 12 GB

B Résultats Détaillés

B.1 Métriques Complètes

Algorithme	**Min**	**Max**	**Moyenne**	**Médiane**	**Écart-type**	**Q1**	**Q3**
PPO	-450	329	225	260	95	180	290
DQN	-580	326	195	240	120	150	280
Double DQN	-520	314	170	210	130	120	250
A2C	-620	305	85	120	175	0	200

TABLE 4 – Statistiques des récompenses d'entraînement

B.2 Taux de Réussite

Pourcentage d'épisodes avec score > 200 (considéré comme réussite) :

Algorithme	**Épisodes 1-1000**	**Épisodes 1001-2000**
PPO	35%	78%
DQN	28%	65%
Double DQN	22%	52%
A2C	18%	28%

TABLE 5 – Taux de réussite par phase d'entraînement