UNIVERSITÉ DE LORRAINE | BIBLIOTHÈQUES UNIVERSITAIRES

AVERTISSEMENT
DROIT D'AUTEUR

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document. Toute contrefaçon, plagiat ou reproduction illicite encourt une poursuite pénale.

Contact bibliothèque : ddoc-theses-contact@univ-lorraine.fr
(Cette adresse ne permet pas de contacter les auteurs)

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10

UNIVERSITÉ DE LORRAINE
34 COURS LÉOPOLD — CS 25233
54052 NANCY CEDEX
TÉL : 03 54 50 54 00
CONTACT@UNIV-LORRAINE.FR
WWW.UNIV-LORRAINE.FR





École doctorale IAEM Lorraine

Application de l'apprentissage par renforcement profond : Modélisation contrainte d'un opérateur humain

THÈSE

présentée et soutenue publiquement le 9 juillet 2026

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Idriss ABDALLAH

Composition du jury

Direction :	Patrick HÉNAFF	Professeur, Bretagne INP, Lab-STICC
	Laurent CIARLETTA	Maître de conférences, Université de Lorraine, LORIA
Rapporteurs :	David FILLIAT	Professeur, ENSTA, Directeur de recherche, AMIAD
	Sonia VANIER	Professeure, École Polytechnique
Président du jury :	Eddy BAJIC	Professeur, Université de Lorraine, CRAN
Examinateurs :	Stéphane DONCIEUX	Professeur, Université Sorbonne, ISIR
	Eva ARTUSI	Docteure, Naval Group
	Jonathan CHAMPAGNE	Docteur, Naval Group
Invité :	Matthieu BONAVENT	Ingénieur, Naval Group

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Corporate sensitivity
PUBLIC

© Naval Group SA 2026 tous droits réservés.

Ce document dans son contenu et dans sa forme est la propriété de Naval Group SA et/ou de tiers. Toute utilisation, reproduction, modification, traduction, communication ou représentation intégrale ou partielle du présent document qui n'a pas été préalablement autorisée par écrit par Naval Group SA est formellement interdite. Une telle utilisation, reproduction, modification, traduction, communication ou représentation intégrale ou partielle non autorisée, par quelque moyen que ce soit, constituerait une contrefaçon sanctionnée par la loi aux plans pénal et, d'une manière générale, une atteinte aux droits de Naval Group SA.

Mis en page avec la classe thesul.

Remerciements

Afin de réaliser ces travaux, 3 années et demi ont été nécessaires. Ce temps a été partagé entre Nancy et Saint-Tropez et je souhaite remercier toutes les personnes qui m'ont permis de mener à bien ce projet. Ainsi, ces quelques mots essaient de rendre compte de l'aide qui m'a été apporté au cours de ces travaux de recherche.

Tout d'abord, je souhaite remercier Patrick HÉNAFF et Laurent CIARLETTA de m'avoir accompagné depuis les débuts de ma formation aux Mines de Nancy, malgré les multiples difficultés personnelles et professionnelles rencontrées. Je tiens à exprimer ma reconnaissance à l'équipe SIMBIOT avec qui je n'ai finalement passé que peu de temps, mais qui a toujours été accueillante et présente lorsque j'en ai eu besoin. Je souhaite également mentionner le suivi effectué par Olivier BUFFET et Eddy BAJIC qui m'a permis de prendre du recul sur l'avancée du projet, ce qui s'est avéré particulièrement précieux dans un tel contexte. Je remercie également l'ensemble du jury qui a pris le temps d'évaluer mes travaux avec précision et permis de concrétiser l'obtention de ce diplôme par une soutenance riche en échanges.

Je souhaite exprimer ma reconnaissance envers Matthieu BONAVENT et Jonathan CHAMPAGNE, qui ont su à la fois m'apporter leur expertise mais ont également mis un point d'honneur à ce que je puisse travailler dans les meilleures conditions possibles (autant sur le plan matériel que sur le plan moral). Je tiens à remercier particulièrement les équipes en charge du développement du simulateur et du logiciel tactique, qui ont toujours accueilli avec bienveillance (parfois bien dissimulée certes) mes questions et mes requêtes. Je m'excuse auprès de mes relecteurs à qui j'ai infligé quelques migraines suite à des interprétations un peu trop libre de la langue française. Plus globalement, je tiens à remercier l'ensemble des personnes que j'ai rencontré à Naval Group avec qui j'ai pu tisser des liens d'amitié autour de cafés (et de bières) et qui, je l'espère, dureront encore.

Je souhaite également mentionner l'ensemble des personnes extérieures à Naval Group et au LORIA mais dont la présence dans ma vie au cours de ces années a fortement contribué à l'achèvement de ces travaux. Je souhaite donc remercier mes amis d'enfance qui savent toujours se rendre disponible, mes amis de CrossFit Grimaud qui m'ont permis de me vider la tête quotidiennement (parfois un peu trop peut-être) et mes amis d'école dont la légèreté et l'honnêteté m'apportent toujours une bulle d'air frais. Finalement, j'exprime toute ma gratitude envers ma famille qui, malgré la distance et mon temps de réponse au téléphone, est toujours présente et l'a toujours été peu importe la situation.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

Table des matières

Table des figures **ix**

Introduction générale **1**

1 Contexte 1
2 Organisation du manuscrit 1
3 Contributions de la thèse 2
3.1 Publications scientifiques 2
3.2 Réalisations industriels 2

Chapitre 1 Contexte Industriel **3**

1.1 Domaine d'activité 4
1.1.1 Description du groupe 4
1.1.2 Site de Saint-Tropez 4
1.2 La torpille lourde F21 5
1.2.1 Classification des données relatives à la F21 5
1.2.2 Description générale de la F21 6
1.2.3 Logiciel tactique 6
1.2.4 Développement du logiciel tactique de la F21 8
1.3 Le simulateur numérique 9
1.3.1 Modèles composant le simulateur 9
1.3.2 Fonctionnement du simulateur 10
1.4 Le modèle de l'opérateur 13
1.4.1 Représentativité du modèle opérateur 14
1.4.2 Modélisations 15
1.5 Motivations industrielles 15
1.6 Contraintes de développement liées à la classification du sujet 16
1.6.1 Sujet et données classifiées 16
1.6.2 Puissance de calcul fixe 16

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Chapitre 2 État de l'art **19**

2.1 Travaux similaires . . . 20
2.2 Apprentissage par renforcement . . . 20
2.2.1 Formalisme et définitions . . . 21
2.2.2 Résolution d'un MDP . . . 24
2.2.3 Dilemme Exploration-Exploitation . . . 26
2.2.4 *Off-Policy* et *On-Policy* . . . 26
2.2.5 Exemple : Q-Learning . . . 26
2.2.6 Fonction d'approximation . . . 27
2.2.7 Descente de gradient . . . 28
2.3 Réseaux de neurones . . . 30
2.3.1 Définition . . . 30
2.3.2 Architectures classiques . . . 33
2.4 Apprentissage par renforcement profond . . . 34
2.4.1 Problématiques induites . . . 34
2.4.2 Algorithmes . . . 35
2.5 Exploitation de connaissances extérieures . . . 42
2.5.1 Transfer Learning . . . 43
2.5.2 Utilisation de démonstrations . . . 43
2.5.3 Information privilégiée . . . 46
2.5.4 Structurer le problème . . . 46
2.5.5 Curriculum Learning . . . 47
2.6 Application de l'apprentissage par renforcement . . . 48
2.6.1 Benchmarks . . . 48
2.6.2 Librairies . . . 48
2.6.3 Création d'un environnement . . . 48
2.6.4 Comparaison . . . 50

Chapitre 3 Reformulation du sujet de thèse **53**

3.1 Définition du MDP . . . 54
3.1.1 Espace d'observation . . . 54
3.1.2 Espace d'action . . . 55
3.1.3 Diversité des scénarios . . . 55
3.1.4 Connaissances métiers sur l'environnement . . . 56
3.1.5 Performances de référence . . . 56
3.2 Développement . . . 57
3.2.1 Environnement de développement . . . 58

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

3.2.2 Ressources allouables . . . 58
3.3 Application du modèle . . . 58
3.3.1 Performances . . . 59
3.3.2 Comportemental . . . 60
3.3.3 Acceptabilité du modèle en production . . . 60

Chapitre 4 Architecture logicielle de l'environnement d'apprentissage 63
4.1 Interfaçage du *wrapper* avec le simulateur . . . 64
4.1.1 Initialisation du simulateur . . . 64
4.1.2 Communication au cours de la simulation . . . 65
4.2 Création de l'environnement d'apprentissage . . . 65
4.2.1 Communication avec le simulateur . . . 65
4.2.2 Fonction de récompense . . . 66
4.2.3 Transformée d'informations . . . 66
4.2.4 Politique . . . 66
4.2.5 Transformée d'action . . . 67
4.2.6 *Frame Skipping* . . . 67
4.3 Analyse du *wrapper* . . . 67
4.3.1 Librairies choisies . . . 67
4.3.2 Utilisation du *wrapper* . . . 68
4.3.3 Efficacité . . . 68

Chapitre 5 Définition du problème d'apprentissage 71
5.1 Difficultés inhérentes à la définition d'un problème d'apprentissage . . . 72
5.1.1 Typologie des difficultés rencontrées . . . 72
5.1.2 Impacts des difficultés rencontrées sur la création d'un modèle . . . 74
5.2 Choix de versions . . . 74
5.3 Espace d'observation . . . 75
5.3.1 Construction de l'espace d'observation . . . 75
5.3.2 Représentation des positions dans l'espace d'observation . . . 75
5.4 Espace d'action . . . 76
5.4.1 Analyse de la perte de performance potentielle . . . 76
5.4.2 Construction de l'espace d'action . . . 77
5.5 Fonction de récompense . . . 77
5.5.1 Terme terminal . . . 77
5.5.2 Termes de *shaping* . . . 78
5.5.3 Pondération des termes de *shaping* par rapport au terme terminal . . . 80

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

5.5.4 Terme comportemental 83
5.6 Fenêtre de contrôle 85

Chapitre 6 Choix algorithmiques **87**
6.1 Choix de la base algorithmique 88
6.2 Utilisation des ressources calculatoires 88
6.2.1 Choix de l'architecture de parallélisation 88
6.2.2 Comparaison aux *benchmarks classiques* 89
6.2.3 Méthodologie expérimentale 90
6.2.4 Étude de l'existence d'un biais 91
6.2.5 Existence d'un biais induit par un ralentissement dans une zone sur les données d'entraînement et sur la politique apprise. 92
6.2.6 Influence du nombre d'*actors* en parallèle sur le biais 93
6.2.7 Influence de l'exploration sur le biais 94
6.2.8 Influence de la position de la zone sur le biais 94
6.2.9 Conclusion sur l'étude d'un biais 96
6.3 Ajouts algorithmiques supplémentaires 96
6.3.1 *Prioritized Experience Replay* 97
6.3.2 Initialisation avec la politique de référence 97
6.3.3 *Noisy Net* 100
6.3.4 *Dueling Net* 102

Chapitre 7 Utilisation des spécificités de l'environnement **105**
7.1 Utilisations possibles de la structure de machines à états 106
7.1.1 Définitions 106
7.1.2 Couplages possibles avec une machine à états 106
7.1.3 Reconsidération de l'état de l'art sur le couplage MAE et DRL 107
7.1.4 Nouvelles possibilités de couplages identifiées 108
7.1.5 Conclusion partielle sur le couplage entre une MAE et un algorithme de DRL 110
7.2 Apprentissages séparés des actions 111
7.2.1 Analyse sur le scénario bâtiment de surface 111
7.2.2 Analyse sur le scénario sous-marin 114
7.3 Ajout de l'*action masking* 117
7.4 Apprentissage commun 117
7.4.1 Apprentissage de toutes les actions 118
7.4.2 Utilisation de QDagger à partir des modèles à une action 120

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

7.5 Information privilégiée ou non 124

Chapitre 8 Élaboration finale du modèle opérateur **129**

8.1 Élaboration d'un modèle 130

8.1.1 Différence entre la mise au point d'un processus d'apprentissage et l'élaboration d'un modèle 130

8.1.2 Intérêt de QDagger 130

8.2 Analyse des modèles 131

8.3 Robustesse aux changements du MDP 133

8.3.1 Ajout de délai 133

8.3.2 Changement de scénario 134

Chapitre 9 Conclusion et perspectives **137**

9.1 Conclusion générale 137

9.2 Perspectives 139

9.2.1 Amélioration des modèles actuels 139

9.2.2 Extension à des problèmes similaires 139

9.2.3 Aide à la décision 140

Annexes **141**

Annexe A Informations déductibles d'une performance blanchie par rapport à une valeur de référence **141**

A.1 Majorant / Minorant de performances par rapport à une référence 141

A.1.1 Ratio d'amélioration 141

A.1.2 Écart de performance 142

A.1.3 Écart relatif de performance 142

A.2 Informations données par l'intervalle de confiance 143

A.2.1 Cas limite avec seulement des échecs/réussite 143

A.2.2 Cas général 145

Annexe B Définition des différents termes de *shaping* **147**

B.1 Distance à la cible 147

B.1.1 DistQuant 147

B.1.2 DistQual+ 147

B.1.3 DistQual- 147

B.2 Orientation à la cible 147

B.2.1 OriQuant 148

B.2.2 OriQual+ 148

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

B.2.3 OriQual- 148
B.3 Proximité de piste à la cible 148
B.3.1 ProxQuant 148
B.3.2 ProxQual+ 148
B.3.3 ProxQual- 149
B.3.4 ProxQual++ 149
B.3.5 ProxQual-- 149

Annexe C Implémentation du ralentissement **151**
C.1 Planification des mises à jour 151
C.2 Architecture asynchrone simulée 151

Annexe D Paramètres utilisés pour Cartpole **155**

Bibliographie **157**

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

Table des figures

1.1 Évolution de Naval Group depuis 1624 [3]. 4
1.2 Illustration d'une F21 tirée à partir d'un sous-marin. 5
1.3 Représentation schématique d'un tir contre un bâtiment de surface avec la F21. . 6
1.4 Machine à états simplifiée du logiciel tactique de la torpille. 7
1.5 Description du déroulement d'un tir fictif simplifié. 8
1.6 Schéma de différents modèles mis en jeux dans la simulation numérique. 9
1.7 Illustration de l'ordre d'exécution des événements et de l'évolution du temps simulé (nombre de secondes simulées) et du temps réel (temps pris pour l'exécution de la simulation) pour une simulation à événements discrets. 11
1.8 Différents types de variabilité des temps de calcul au sein du simulateur industriel 12
1.9 Interactions du modèle opérateur au sein du simulateur. 13

2.1 Principe de l'apprentissage par renforcement 21
2.2 Schéma d'un processus de décision markovien 22
2.3 Algorithme d'apprentissage par renforcement. 24
2.4 Représentation de la valeur moyenne et de l'écart relatif pour plusieurs actions de la fonction valeur état-action et la fonction avantage. 30
2.5 Schéma d'un réseau de neurones à deux couches cachées de trois neurones, une couche d'entrée de dimension deux et une couche de sortie de dimension deux. . . 31
2.6 L'apprentissage par renforcement profond. 34
2.7 Algorithme d'apprentissage par renforcement avec le mécanisme de rejeu d'expérience. 37
2.8 Agencement des tâches effectuées par les différents processus avec une architecture synchrone pour un algorithme de DRL. 39
2.9 Agencement des tâches effectuées par les différents processus avec une architecture asynchrone pour un algorithme de DRL. 39
2.10 Deux architectures de réseaux de neurones pour estimer la fonction état-action Q pour l'algorithme DQN. 40
2.11 Schéma d'un processus de décision markovien partiellement observable. 42
2.12 Illustration des différentes métriques pour les méthodes de TL [67]. 44
2.13 Utilisation de l'*Action Masking* pour un réseau estimant la fonction q. 47

3.1 Description du problème de décision séquentielle à résoudre par le modèle opérateur dans le cadre de l'apprentissage par renforcement où les termes en bleu représentent la particularisation à la problématique industrielle des termes d'apprentissage en orange. 54

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

3.2 Catégorisation des tirs en fonction de leur faisabilité pour le logiciel tactique et pour l'opérateur. 57
3.3 Outil de visualisation permettant d'analyser des simulations de tirs. 58
3.4 Exemple de graphique blanchi présentant les performances de plusieurs modèles. 59
3.5 Exemple de graphique blanchi présentant la fréquence d'action de plusieurs modèles. 61

4.1 Initialisation du processus du *wrapper*, du processus de simulation numérique et de la communication socket entre les deux. 64
4.2 Communication entre le processus du *wrapper* et le processus de simulation numérique au cours d'un tir. 64
4.3 Ensemble des blocs fonctionnels composant le *wrapper* afin d'adapter le simulateur numérique en un environnement d'apprentissage par renforcement. 66
4.4 Fonctionnement du *Frame Skipping* pour $n_{skip} = 1$. 67
4.5 Fichier de configuration simplifié de l'environnement d'apprentissage 68

5.1 Description de la définition de la distance, de la circulaire et de l'élévation d'un objet P dans le repère de la torpille. 76
5.2 Dégradation des performances sur SM pour une politique choisissant aléatoirement entre l'action "Rien Faire" et une des quatre actions en fonction de la proportion de choix aléatoire. 77
5.3 Corrélation de l'ensemble des termes de *shaping* considérés avec la probabilité de réussite et la durée d'un tir pour les scénarios BS et SM sans action opérateur. . 78
5.4 Corrélation des termes de *shaping* choisis pour les scénarios BS et SM sans action opérateur. 79
5.5 Courbe d'apprentissage moyennée sur une fenêtre glissante de 400 000 pas de temps moyennée sur 5 germes avec l'écart-type sur les scénarios SM et BS pour différentes fonction de récompense (*shaping*, *shaping* + terminal, terminal). . . . 80
5.6 Performance et fréquence d'action sur SMR et SM$\bar{\text{R}}$ moyennées sur l'ensemble des modèles appris dans la figure 5.5 avec un intervalle de confiance à 95%. 81
5.7 Performance et fréquence d'action sur BS moyennées sur l'ensemble des modèles appris dans la figure 5.5 avec un intervalle de confiance à 95% 82
5.8 Performance et fréquence d'action sur SM moyennées sur l'ensemble des modèles en fonction de la pondération du terme comportemental avec un intervalle de confiance à 95%. La valeur de pondération de 0 est placée à la valeur 0.3 pour des raisons de lisibilité. 83
5.9 Performance et fréquence d'action sur BS moyennées sur l'ensemble des modèles en fonction de la pondération du terme comportemental avec un intervalle de confiance à 95%. La valeur de pondération de 0 est placée à la valeur 0.3 pour des raisons de lisibilité. 84
5.10 Performance sur SM moyennées sur 5 germes aléatoires avec un intervalle de confiance à 95% en fonction de la simulation de panne lors de l'apprentissage. . . 85

6.1 Application de l'architecture asynchrone sur les ressources de calculs. 89
6.2 Valeurs de plusieurs $IQR_{\alpha\%}$ pour les environnements Atari, Atari sans *frame skip*, Mujoco, Cartpole et le simulateur industriel. 90
6.3 Environnement Cartpole avec deux zones symétriques définies par la position du chariot, notée x, telles que la zone gauche est pour $x \in [-\infty; -x_{zone}]$ et la zone droite est pour $x \in [x_{zone}; +\infty]$. 92

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

6.4 Présence moyenne par épisode par zone pendant l'entraînement moyennée sur 1000 entraînements avec un intervalle de confiance à 95% en fonction du coefficient de ralentissement de la zone gauche. 92
6.5 Présence moyenne par épisode par zone sur 100 épisodes avec la politique finale moyennée sur 1000 entraînements avec un intervalle de confiance à 95% en fonction du coefficient de ralentissement de la zone gauche. 92
6.6 Présence moyenne par épisode par zone sur 100 épisodes avec la politique finale moyennée sur 1000 entraînements avec un intervalle de confiance à 95%, en fonction du nombre d'*actors* en parallèle pour plusieurs valeurs de coefficients de ralentissement de la zone gauche. 93
6.7 Ratio moyen entre la présence dans la zone droite et la zone gauche sur 100 épisodes avec la politique finale moyennée sur 1000 entraînements avec un intervalle de confiance à 95%, en fonction du nombre d'*actors* en parallèle pour plusieurs valeurs de coefficient de ralentissement de la zone gauche. 93
6.8 Présence moyenne par épisode par zone sur 100 épisodes avec la politique finale moyennée sur 1000 entraînements avec un intervalle de confiance à 95% en fonction du pas de temps auquel ε atteint sa valeur finale pour plusieurs valeurs de coefficients de ralentissement de la zone gauche. 94
6.9 Ratio moyen entre la présence dans la zone droite et la zone gauche sur 100 épisodes avec la politique finale moyennée sur 1000 entraînements avec un intervalle de confiance à 95% en fonction du pas de temps auquel ε atteint sa valeur finale pour plusieurs valeurs de coefficient de ralentissement de la zone gauche. 94
6.10 Présence moyenne par épisode par zone pendant l'entraînement moyennée sur 1000 entraînements avec un intervalle de confiance à 95% en fonction de la taille de la zone. 95
6.11 Présence moyenne par épisode par zone sur 100 épisodes avec la politique finale moyennée sur 1000 entraînements avec un intervalle de confiance à 95% en fonction de la taille de la zone des ralentissements pour plusieurs valeurs de coefficients de ralentissement de la zone gauche. 95
6.12 Ratio moyen entre la présence dans la zone droite et la zone gauche sur 100 épisodes avec la politique finale moyennée sur 1000 entraînements avec un intervalle de confiance à 95% en fonction de la taille de la zone de ralentissement pour plusieurs valeurs de coefficient de ralentissement de la zone gauche. 95
6.13 Schéma de l'évolution des proportions des différentes politiques utilisées au cours d'un entraînement pour différentes méthodes d'exploration. 98
6.14 Récompense moyenne par épisode au cours de l'apprentissage (sur une fenêtre de 400 000 pas de temps) avec une initialisation du *replay buffer* avec une politique aléatoire ou une politique de référence moyennée sur 5 apprentissages et l'écart-type. 98
6.15 Évaluation des modèles avec l'initialisation aléatoire ou utilisant la politique de référence issus des entraînements présentés dans la figure 6.14 sur les scénarios SM. 99
6.16 Récompense moyenne par épisode au cours de l'apprentissage (sur une fenêtre de 400 000 pas de temps) avec une exploration utilisant la méthode ε-*greedy* ou la méthode *Noisy Net* [52] moyennée sur 5 apprentissages et l'écart-type. 100
6.17 Évaluation des modèles avec et sans le *Noisy Net* issus des entraînements présentés dans la figure 6.16 sur les scénarios SM. 101

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

6.18 Utilisation de la RAM par les processus Python du *wrapper* pour l'évaluation de SM sur un noeud de calcul avec 256 environnements en parallèle pour une politique utilisant un MLP (entraîné par l'approche ε-*greedy*) ou un *Noisy Network* en fonction du temps. 102
6.19 Récompense moyenne par épisode au cours de l'apprentissage (sur une fenêtre de 400 000 pas de temps) avec et sans le *Dueling Net* [52] moyennée sur 5 apprentissages et l'écart-type. 102
6.20 Évaluation des modèles avec et sans le *Dueling Net* issus des entraînements présentés dans la figure 6.19 sur les scénarios SM. 103

7.1 Algorithme d'apprentissage par renforcement découpé en une phase exploratoire et une phase d'apprentissage. 106
7.2 Placements possibles d'une MAE en phase exploratoire. 107
7.3 Courbes d'apprentissage moyennée sur une fenêtre glissante de 400 000 pas de temps moyenné sur 5 germes avec l'écart-type sur le scénario BS pour les différentes actions considérées. 111
7.4 Performances et fréquence d'action sur l'ensemble des scénarios BS moyennés des modèles appris dans la figure 7.3 avec un intervalle de confiance à 95%. 112
7.5 Termes de *shaping* d'orientation qualitative - et de proximité qualitative – sur l'ensemble des scénarios BS moyennés des modèles appris dans la figure 7.3 avec un intervalle de confiance à 95%. 113
7.6 Proportion d'épisode qui se finissent par un "crash" de l'environnement au cours de l'apprentissage en fonction du scénario et de l'action à la disposition du modèle. 114
7.7 Courbes d'apprentissage moyenné sur une fenêtre glissante de 400 000 pas de temps moyenné sur 5 germes avec l'écart-type sur le scénario SM pour différentes actions. 114
7.8 Performances sur l'ensemble des scénarios SM moyennés des modèles appris dans la figure 7.7 avec un intervalle de confiance à 95%. 115
7.9 Évaluation des termes de *shaping* d'orientation qualitative - et de proximité qualitative -- sur l'ensemble des scénarios SM moyennés des modèles appris dans la figure 7.7 avec un intervalle de confiance à 95%. 116
7.10 Fréquence d'action par phase en fonction de l'action à la disposition du modèle. . 116
7.11 Performances sur l'ensemble des scénarios SM moyennés des modèles appris dans la figure 7.7 avec un intervalle de confiance à 95%. 117
7.12 Performances et fréquence d'action sur SM avec un intervalle de confiance à 95% pour les modèles **Action 1**, **DQN 2M** et **DQN 10M** 118
7.13 Termes de *shaping* d'orientation qualitative - et de proximité qualitative -- sur SM avec un intervalle de confiance à 95% pour les modèles **Action 1**, **DQN 2M** et **DQN 10M**. 119
7.14 Fréquence d'action par action et par phase pour les modèles **DQN 10M** avec un intervalle de confiance à 95%. 120
7.15 Proportion d'épisode se terminant par un *crash* au cours de l'entraînement pour l'apprentissage utilisant DQN et seulement l'action 2 sur BS, *QDagger* avec l'adaptation de l'équation 7.4 et *QDagger* avec l'adaptation de l'équation 7.4 ainsi qu'un terme terminal de pénalisation. 123
7.16 Performances et fréquence d'action sur SM avec un intervalle de confiance à 95% pour les modèles **DQN 10M**, **QDag Défaut**, **QDag Auto** et **QDag + Anti Crash**. 123

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

7.17 Performances et fréquence d'action sur SM avec un intervalle de confiance à 95% pour les modèles **DQN Priv**, **DQN Non Priv**, **NQDagger** et **QDagger**. . . . 125
7.18 Termes de *shaping* d'orientation qualitative - et de proximité qualitative -- sur SM avec un intervalle de confiance à 95% pour les modèles **DQN Priv**, **DQN Non Priv**, **NQDagger** et **QDagger**. . . . 126
7.19 Durée d'un tir sur SM avec un intervalle de confiance à 95% pour les modèles **DQN Priv**, **DQN Non Priv**, **NQDagger** et **QDagger**. . . . 126
7.20 Fréquence d'action par action par phase pour les modèles **DQN Non Priv** et **QDagger** avec un intervalle de confiance à 95%. . . . 127

8.1 Performances et fréquence d'action des modèles M^1_{SM}, M^2_{SM} et M_{BS} respectivement sur les scénarios SM, SM et BS avec un intervalle de confiance à 95% 131
8.2 Évaluation de la fréquence d'action par action et par phase du logiciel tactique pour les modèles M^2_{SM} et M_{BS}. . . . 132
8.3 Performances des modèles M^2_{SM} et M_{BS} respectivement sur les scénarios SM, SM et BS avec un intervalle de confiance à 95% en fonction du délai d'application de l'action. La valeur sans délai est affichée pour la valeur d'abscisse 0.5 pour des raisons d'affichage. . . . 133
8.4 Fréquence d'action des modèles M^2_{SM} et M_{BS} respectivement sur les scénarios SM, SM et BS avec un intervalle de confiance à 95% en fonction du délai d'application de l'action. La valeur sans délai est affichée pour la valeur d'abscisse 0.5 pour des raisons d'affichage. . . . 134
8.5 Performances des modèles M^2_{SM} et M_{BS} respectivement sur les scénarios SM, SM et BS avec un intervalle de confiance à 95% en fonction du scénario. . . . 135
8.6 Fréquence d'action des modèles M^2_{SM} et M_{BS} respectivement sur les scénarios SM, SM et BS avec un intervalle de confiance à 95% en fonction du scénario. . . . 136

C.1 Agencement des tâches effectuées par les différents processus avec une architecture asynchrone pour un algorithme de DRL avec des ralentissements artificiels. . . . 151
C.2 Apprentissage avec l'algorithme DQN et une architecture asynchrone sur l'environnement Cartpole avec un coefficient de ralentissement de 10 sur toutes les transitions et un planificateur de mise à jour moyenné sur 25 germes. . . . 152
C.3 Agencement des tâches effectuées par les différents processus avec une architecture asynchrone simulée pour un algorithme de DRL avec des ralentissements artificiels. 152
C.4 Gestion des tâches par la boucle événementielle pour l'architecture asynchrone simulée de la figure C.3. . . . 153

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

Introduction générale

1 Contexte

Ces travaux de recherche sont issus d'une collaboration entre Naval Group, groupe industriel français spécialisé dans la construction navale de défense et le LORIA (Laboratoire lorrain de recherche en informatique et ses applications), un laboratoire de l'Université de Lorraine, du CNRS et de l'INRIA, qui mène des recherches fondamentales et appliquées, au niveau international, dans le domaine des sciences et technologies de l'information et de la communication. Cette collaboration utilise le dispositif CIFRE (Convention industrielle de formation par la recherche).

Le contexte industriel s'inscrit dans le développement de la torpille lourde F21, une arme sous-marine. Une fois tirée, l'arme est autonome et s'appuie sur une intelligence embarquée (dit logiciel tactique). Cette intelligence embarquée est elle-même mis au point à l'aide d'un simulateur de combat naval capable de modéliser une multitude de théâtres opérationnels. Pour être le plus représentatif possible, Naval Group modélise l'ensemble des phénomènes physiques (par exemple : acoustique sous-marine) et les différents acteurs présents sur le théâtre opérationnel, comme l'opérateur à bord du lanceur supervisant le tir torpille. Comme tous autres éléments du théâtre opérationnel, il est donc nécessaire de modéliser l'opérateur. Cependant, les modèles d'opérateur développés jusqu'à présent ne sont pas satisfaisants.

L'objectif de cette thèse est donc d'élaborer un modèle réactif d'opérateur humain pour augmenter la représentativité du simulateur en utilisant l'apprentissage automatique.

2 Organisation du manuscrit

Tout d'abord, le chapitre 1 présente l'ensemble des éléments relatifs à la torpille F21 nécessaires pour comprendre la problématique industrielle. Ainsi, il décrit le fonctionnement du simulateur industriel et de l'intelligence embarquée. Il place le modèle opérateur au sein de cet environnement logiciel. En outre, les contraintes liées à la protection du secret de la Défense nationale sont explicitées.

Une fois le besoin industriel exprimé, le chapitre 2 présente un état de l'art sur des problématiques connexes et conduit finalement au choix de l'apprentissage par renforcement profond pour modéliser l'opérateur. L'apprentissage par renforcement et les réseaux de neurones sont présentés séparément. Cela permet ensuite d'expliciter l'ensemble des algorithmes d'apprentissage par renforcement profond utilisé dans ces travaux. Finalement, nous faisons un focus sur l'application de l'apprentissage par renforcement.

Le chapitre 3 permet de mettre en regard les chapitres précedents, c'est-à-dire de replacer le besoin industriel au sein de l'état de l'art de l'apprentissage par renforcement. Il présente les caractéristiques du problème de décision séquentielle à résoudre par le modèle opérateur et les traduit dans le formalisme de l'apprentissage par renforcement. Cela nous permet d'expliciter les

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

conditions d'acceptabilité d'un modèle sur les aspects de performance et de représentativité tout en respectant la protection du secret de la Défense nationale.

Nous présentons, dans le chapitre 4, l'architecture logicielle développée pour adapter le simulateur industriel en un environnement d'apprentissage modulaire. Nous décrivons la manière dont nous nous sommes interfacés avec le simulateur et l'agencement des différents éléments nécessaires aux algorithmes de DRL. Finalement, nous étudions l'impact de ces modifications sur les performances.

Le chapitre 5 présente la définition du processus de décision markovien à résoudre pour créer un modèle opérateur. Ainsi, nous discutons l'ensemble des choix faits pour : l'espace d'observation, l'espace d'action, la fonction de récompense et la fenêtre de contrôle laissée au modèle.

L'algorithme d'apprentissage par renforcement profond utilisé dans les travaux de cette thèse est décrit dans le chapitre 6. Nous commençons par justifier le choix de l'algorithme du *Deep Q-Learning*. Par la suite, nous choisissons et étudions l'architecture de parallélisation asynchrone. Finalement, nous présentons l'impact de différentes modifications algorithmiques sur les performances des modèles.

Le chapitre 7 présente plusieurs approches utilisant des spécificités du problème industriel pour améliorer la capacité de converger vers un meilleur modèle. Nous commençons par une étude théorique des possibilités de couplage entre une machine à états et un algorithme de DRL. Puis, nous présentons des approches permettant de tirer profit des spécificités de l'espace d'action et de l'espace d'observation.

Finalement, avant de conclure et de discuter les différentes perspectives de ces travaux de thèse, les meilleurs modèles obtenus sont présentés et analysés dans le chapitre 8. Une étude de sensibilité est alors réalisée visant à évaluer l'impact de modifications de certains éléments de l'environnement sur ces différents modèles.

3 Contributions de la thèse

3.1 Publications scientifiques

Abdallah Idriss, Ciarletta Laurent, Hénaff Patrick, Champagne Jonathan et Bonavent Matthieu, "Couplage entre un apprentissage par renforcement profond et une machine à états : approche théorique", *CAID - Conference on Artificial Intelligence for Defense*, DGA, Nov 2024, Rennes, France. [1]

Abdallah Idriss, Ciarletta Laurent, Hénaff Patrick, Champagne Jonathan et Bonavent Matthieu, "Effect of a slowdown correlated to the current state of the environment on an asynchronous learning architecture", *Reinforcement Learning Journal*, Août 2025, pages 384-398. [2]

3.2 Réalisations industriels

Développement d'un *framework* pour utiliser l'apprentissage par renforcement profond avec le simulateur industriel de combat naval.

Développement de modèles opérateur pour le simulateur industriel de combat naval.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

1

Contexte Industriel

Ce chapitre décrit le besoin exprimé par Naval Group ayant motivé la réalisation de ces travaux de thèse. Il commence par un focus successif à plusieurs échelles du groupe industriel, permettant d'introduire peu à peu les modalités du développement du logiciel tactique de la torpille lourde F21. Parmi ces dernières, une description plus détaillée est faite du simulateur numérique, et plus spécifiquement du modèle comportemental de l'opérateur sous-marinier supervisant l'arme. En effet, la simulation et l'évaluation du comportement d'un humain mis dans cette situation sont deux tâches difficiles. Les travaux portent donc sur l'utilisation de l'apprentissage automatique pour modéliser un opérateur humain à partir de l'ensemble des données disponibles. Il est à remarquer que les données manipulées sont soumises à la protection du secret de la Défense nationale, ce qui induit des contraintes supplémentaires dans la communication de ces travaux.

Sommaire

1.1 Domaine d'activité 4
1.1.1 Description du groupe 4
1.1.2 Site de Saint-Tropez 4
1.2 La torpille lourde F21 5
1.2.1 Classification des données relatives à la F21 5
1.2.2 Description générale de la F21 6
1.2.3 Logiciel tactique 6
1.2.4 Développement du logiciel tactique de la F21 8
1.3 Le simulateur numérique 9
1.3.1 Modèles composant le simulateur 9
1.3.2 Fonctionnement du simulateur 10
1.4 Le modèle de l'opérateur 13
1.4.1 Représentativité du modèle opérateur 14
1.4.2 Modélisations 15
1.5 Motivations industrielles 15
1.6 Contraintes de développement liées à la classification du sujet . . . 16
1.6.1 Sujet et données classifiées 16
1.6.2 Puissance de calcul fixe 16

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

1.1 Domaine d'activité

Naval Group est un industriel français spécialisé dans le naval de défense. Il est également le partenaire industriel à l'origine de cette thèse CIFRE.

1.1.1 Description du groupe

Naval Group est l'héritier direct des arsenaux de la Marine royale dont l'évolution est décrite par la figure 1.1. Cet historique explique que l'actionnaire majoritaire soit l'état français (62.25%), et que le second soit Thales (35%). Actuellement, il en découle donc que son principal client est la Marine nationale. Cependant, malgré son ouverture relativement récente aux marchés internationaux, il se place comme un acteur international majeur du naval de défense car il travaille pour plus de 50 marines. Ainsi, il emploie près de 17000 collaborateurs au sein de 17 pays répartis sur 5 continents.

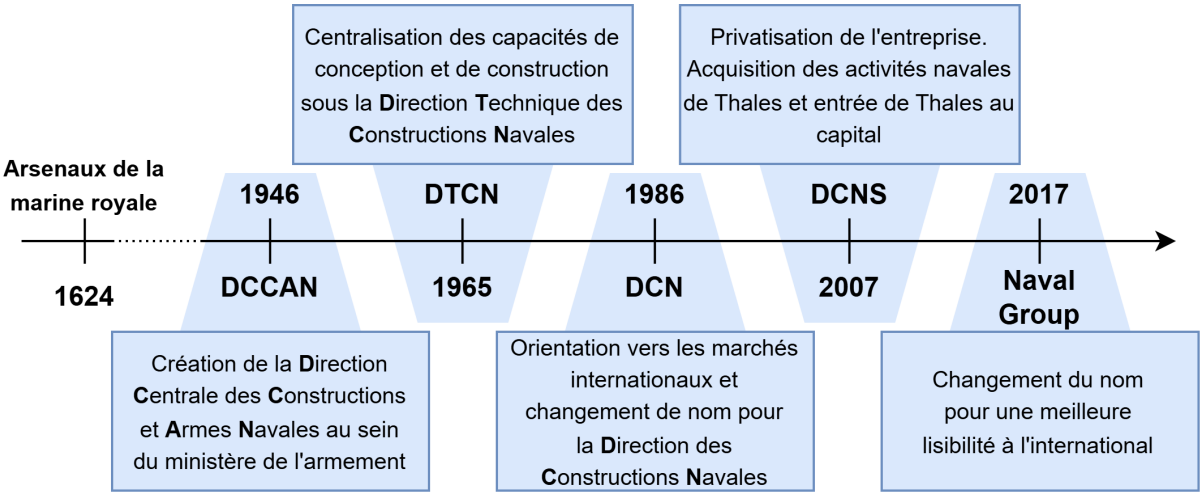


FIGURE 1.1 – Évolution de Naval Group depuis 1624 [3].

Son activité actuelle se répartie sur l'ensemble du cycle de vie des navires (conception, construction, maintenance en condition opérationnelle, démantèlement, etc.). Naval Group assure notamment le maintien en conditions opérationnelles :

- du Charles de Gaulle, un porte-avions à propulsion nucléaire qui est le bâtiment majeur de la Marine nationale.
- des quatres sous-marins nucléaires lanceurs d'engins (SNLE) de la classe Le Triomphant, éléments centraux de la stratégie de dissuasion nucléaire française.

En tant que systémier intégrateur, Naval Group s'occupe également de la production et de l'intégration des armes, des systèmes et des équipements sur ces différents navires.

1.1.2 Site de Saint-Tropez

Le site de Saint-Tropez est l'un des 10 sites de Naval Group situés dans la métropole française. Il a été fondé en 1912 par un conglomérat d'entreprises britanniques et se nommait initialement *Whitehead*. Il sera nationalisé et intégré à la Marine nationale en 1937. Depuis sa création, il produit toutes les torpilles françaises. Actuellement, il produit les torpilles armant les navires de la Marine nationale ainsi que d'autres équipements comme :

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

— La Canto®, un système de contre-mesure pour la défense anti-torpille basé sur le principe dilution/confusion.
— La F21®, une torpille lourde disposant d'une grande autonomie et d'un logiciel tactique avancé.
— La MU90 impact®, une torpille légère capable d'être lancée depuis un avion, un hélicoptère ou un navire.

La F21 est le programme de torpille lourde le plus récent du monde occidental, ce qui en fait la torpille lourde la plus autonome et endurante du marché. Ces différents équipements sont centraux dans les stratégies déployées au cours d'un combat naval.

1.2 La torpille lourde F21

La F21 est une torpille lourde conçue pour être lancée à partir d'un sous-marin (voir figure 1.2). Depuis 2019, elle remplace progressivement la F17 (ancienne génération de torpille lourde) à bord des SNLE et SNA (sous-marin nucléaire d'attaque) de la Marine nationale [4]. À l'export, elle équipe également la marine brésilienne.

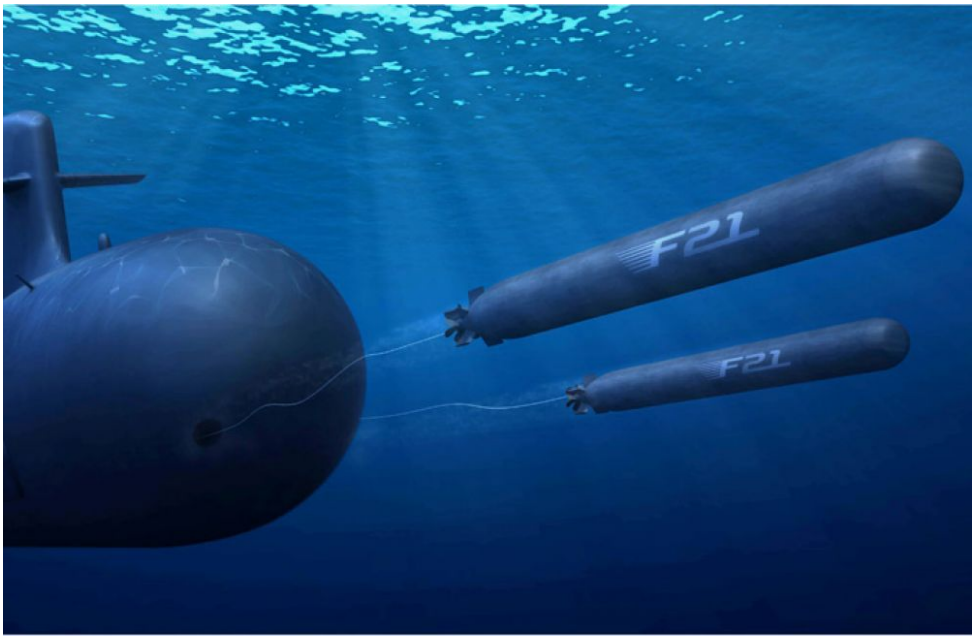


FIGURE 1.2 – Illustration d'une F21 tirée à partir d'un sous-marin.

1.2.1 Classification des données relatives à la F21

L'ensemble des travaux effectués sur la F21 sont donc régis par la protection du secret de la Défense nationale [5]. C'est la raison pour laquelle, il n'est pas possible d'exprimer toutes les données techniques de l'arme. Ainsi, l'ensemble des informations contenues et présentées dans la suite de ce document, concernant la F21, sont à la fois blanchies (i.e. rendues non-sensible) et incomplètes. Cependant, elles suffisent pour décrire et introduire les différentes notions nécessaires à la compréhension des difficultés rencontrées et des résultats.

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity
PUBLIC

1.2.2 Description générale de la F21

La F21 est composée de plusieurs sections (propulsion, charge de combat, énergie, tête acoustique, etc.) dont la coordination au cours d'un tir est assurée par un logiciel embarqué. Le logiciel tactique est le logiciel responsable de l'élaboration des consignes de guidage et de configuration des capteurs. La F21 est conçue pour attaquer deux types de cibles : les bâtiments de surface (voir figure 1.3) et les sous-marins. Une fibre optique est délovée entre le sous-marin lanceur et la torpille au cours du tir. Une communication bidirectionnelle entre l'opérateur au sein du sous-marin et la torpille est établie et permet :

— de la torpille vers l'opérateur : une communication sur l'état courant de l'environnement perçue et des décisions du logiciel tactique.
— de l'opérateur vers la torpille : l'opérateur peut réaliser une action influant sur les décisions du logiciel tactique de la torpille.

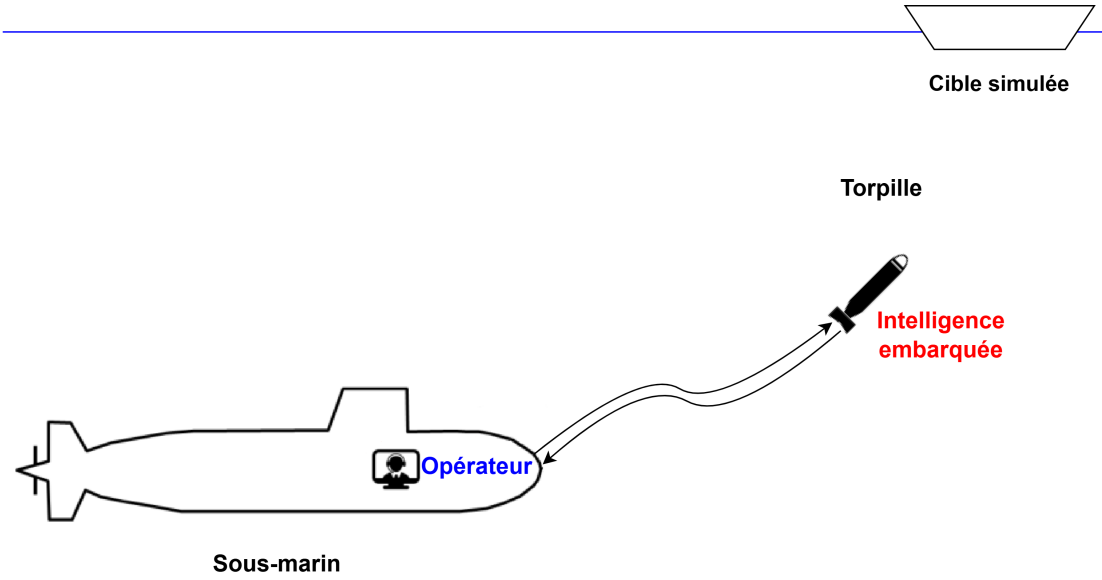


FIGURE 1.3 – Représentation schématique d'un tir contre un bâtiment de surface avec la F21.

La performance globale du tir est assurée par le duo composé de la F21 et de l'opérateur. Cet ensemble est appelé le système d'arme torpille lourde (SATL). Grâce à son logiciel tactique et aux actions opérateurs, le SATL est également capable de faire face aux réactions de sa cible. En effet, les stratégies déployées au cours d'un combat naval incluent des stratégies de lutte anti-torpille comme des manœuvres cinématiques ou bien l'utilisation de leurres ou de contre-mesures.

1.2.3 Logiciel tactique

Au sein de la F21, plusieurs logiciels sont développés pour assurer l'ensemble de ses capacités (pilote, communication, logiciel tactique). Les travaux présentés dans ce manuscrit ont seulement interagi avec le logiciel tactique.

Afin de décrire l'ensemble des éléments d'un tir, il convient de définir les termes suivants :

— Théâtre opérationnel : zone géographique où se déroule le tir, incluant tous les bâtiments présents.
— Cible : bâtiment que la torpille doit détruire (surface ou sous-marin).
— Lanceur : bâtiment à partir duquel la torpille est lancée.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

— Désignation d'objectif (DO) : Mesure bruitée de la cible obtenue par l'opérateur (grâce aux capteurs du lanceur) et transmise au cours du tir.
— Alarme : détection acoustique obtenue par le sonar de la F21.
— Piste : résultat du traitement des alarmes réalisé par le logiciel tactique visant à estimer mathématiquement l'état (position, vitesse) d'un objet présent sur le théâtre opérationnel (ex : cible, contre-mesure, ...).

Son fonctionnement se base sur une machine à états (MAE) dont chacun des états répond à un objectif spécifique. La figure 1.4 présente une version très simplifiée et non classifiée qui sera utilisée pour illustrer les résultats dans la suite du manuscrit. Les différents états sont :

— Recherche : Se diriger vers la DO dans le but d'obtenir des alarmes sur la cible, permettant in fine, de créer et d'entretenir une piste cohérente avec la cible désignée.
— Poursuite : Se rapprocher de la piste de la cible et affiner l'estimation de son état (position, vecteur vitesse).
— Attaque : Réaliser une trajectoire maximisant les chances de neutraliser la cible (i.e. de maximiser la probabilité de destruction).

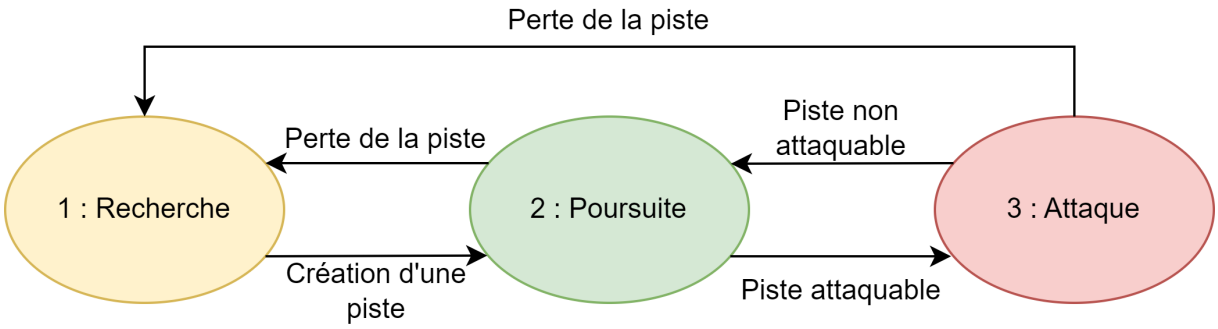


FIGURE 1.4 – Machine à états simplifiée du logiciel tactique de la torpille.

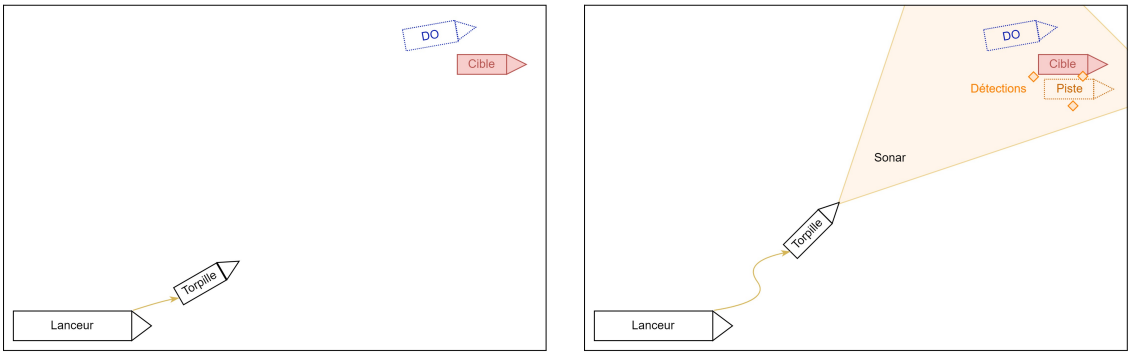
Les figures 1.5 illustrent les instants clefs d'un tir de F21 en totale autonomie (i.e. sans intervention de l'opérateur). Dans la figure 1.5a, le logiciel amorce une phase de **Recherche**, la torpille active son sonar en direction de la DO. Elle obtient alors des alarmes qui semblent cohérentes avec la DO. Une piste est alors créée. Comme cette piste est cohérente avec la cible, le logiciel tactique transite vers l'état **Poursuite**. Dans la figure 1.5b, la torpille s'approche de la piste et obtient de nouvelles alarmes. Elle affine alors l'état de la piste (on parle de recalage de la piste). Comme la piste est cohérente avec la cible et que la torpille est assez proche, le logiciel tactique transite vers l'état **Attaque**. Dans la figure 1.5c, la torpille réalise une trajectoire pour neutraliser la piste. L'état estimé par la piste étant très proche de celui de la cible, le tir est une réussite et permet donc de neutraliser la cible.

Il est important de noter que la situation illustrée par la figure 1.5 est extrêmement simplifiée pour des besoins de pédagogie et de classification. En effet, dans cette illustration :

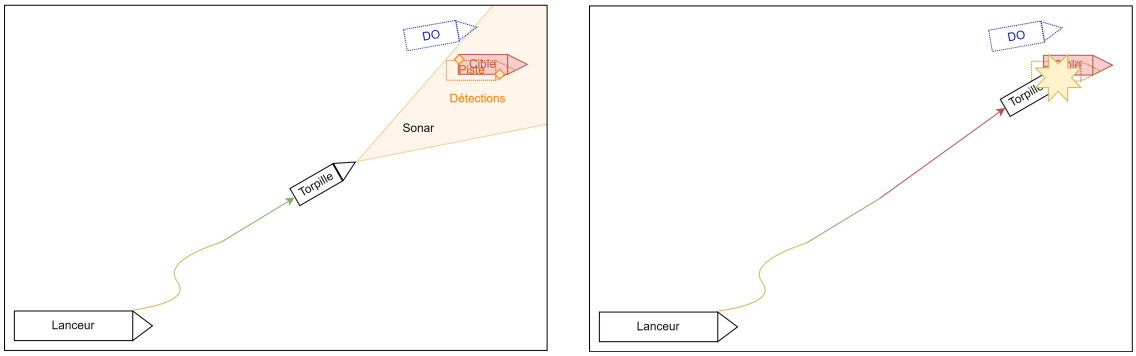
— La représentation est en 2 dimensions.
— La cible est immobile et ne réagit pas.
— La torpille se dirige effectivement sur la cible grâce à une DO relativement précise (i.e. proche de la cible).
— La torpille obtient uniquement des bonnes alarmes sur la cible, i.e. pas de fausses alarmes sur des éléments de l'environnement.
— Le logiciel tactique crée une piste cohérente avec la cible désignée à partir des alarmes.

En générale, dans les simulations de tirs de la F21, la plupart de ces simplifications sont

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity
PUBLIC



(a) Phase de recherche



(b) Phase de poursuite

(c) Phase d'attaque

FIGURE 1.5 – Description du déroulement d'un tir fictif simplifié.

remises en question. Cela donne donc un aperçu du haut niveau de complexité d'un tir torpille.

1.2.4 Développement du logiciel tactique de la F21

Afin de mettre au point ce logiciel, des données provenant de différentes sources sont utilisées. Leur représentativité est inversement proportionnelle à leur coût d'obtention. L'ensemble des sources par représentativité croissantes est :

- Simulateur numérique : Simulation fine de tous les éléments d'un tir (propagation acoustique, cinématique des acteurs du théâtre opérationnel, différentes sections de la torpille, opérateur environnement, ...). Il est possible de faire plusieurs centaines de milliers de simulation par jour.
- Simulation hybride : Utilisation de composants matériels (gouvernes, cartes électroniques, sonar, ...) au sein de la boucle de simulation numérique. Une simulation hybride est donc en temps réel. Il est donc moins aisé de réaliser des simulations en quantité, mais le niveau de représentativité de la simulation augmente fortement.
- Conditions réelles : Mise en place d'un lanceur, d'une cible, d'un opérateur et d'une torpille d'exercice. La représentativité est donc maximale mais la quantité de données disponibles est très faible.

Toutes ces données sont utilisées afin d'améliorer de façon continue la F21. Leurs coûts respectifs influent directement sur leurs fréquences d'obtention. Ainsi, le simulateur numérique est utilisé pour le développement quotidien, afin de tester rapidement des améliorations. Les autres types de données permettent de valider et de remonter des problèmes qui n'ont pas été

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

vus à cause du manque de représentativité du simulateur. Les données enregistrées permettent également d'ajuster les modèles du simulateur numérique. En outre, un nouveau phénomène physique observé lors de tirs en conditions réelles peut être modélisé puis intégré au simulateur.

1.3 Le simulateur numérique

Afin d'évaluer les performances du SATL, un simulateur numérique (appelé "simulateur" dans la suite) a été développé et continue d'être amélioré à partir du retour d'expériences des essais à la mer (voir section 1.2.4). Son objectif est de modéliser un théâtre opérationnel afin de permettre le développement, le test et l'amélioration du SATL en limitant le nombre des tirs réels.

1.3.1 Modèles composant le simulateur

La validité des performances évaluées s'appuie sur la représentativité du simulateur. La figure 1.6 illustre l'ensemble des modèles nécessaires à la simulation d'un tir torpille. Il est important de noter que le logiciel tactique est intégré avec sa version utilisée en condition réelle. Parmi les modèles physiques, on peut noter la propagation des ondes acoustiques qui permet de créer les alarmes obtenues par le logiciel tactique à travers son sonar. Il y a également la cinématique des différents acteurs (torpille, lanceur, cible, ...). La représentativité des modèles des différentes sections de la torpille est particulièrement primordiale afin que le guidage en condition réelle soit adéquat. En outre, des modèles comportementaux sont nécessaires pour chacun des acteurs autre que la torpille. Pour la cible, cela implique des réactions comme une manœuvre cinématique et/ou l'utilisation de matériel de lutte anti-torpille (contre-mesure, leurre, ...).

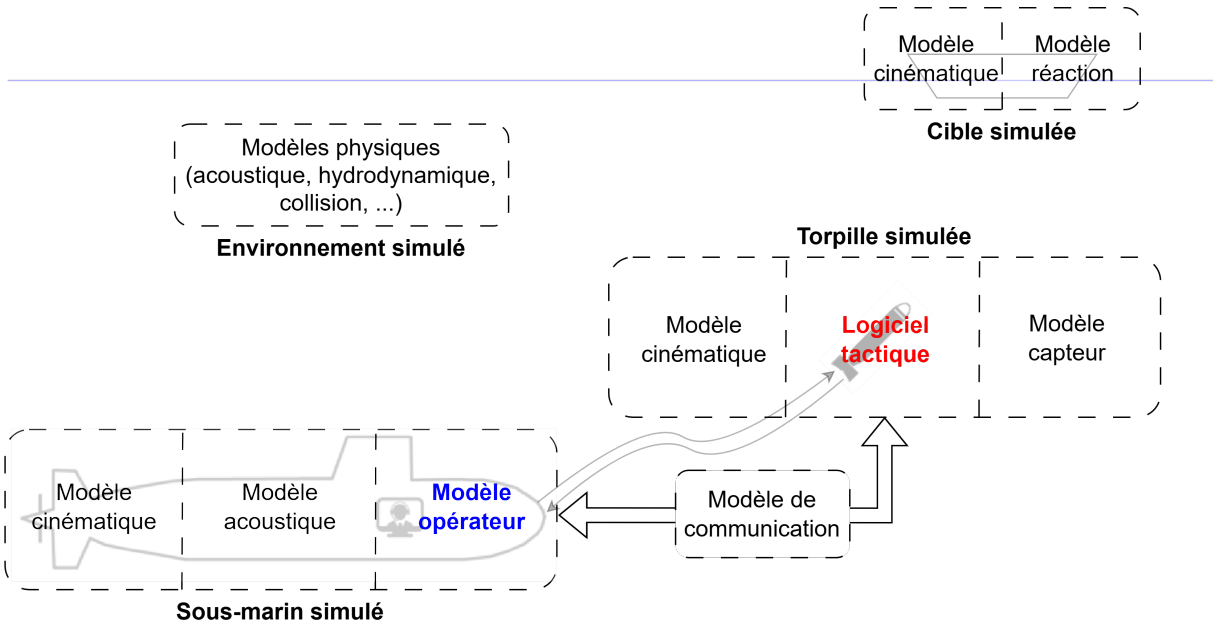


FIGURE 1.6 – Schéma de différents modèles mis en jeux dans la simulation numérique.

Un plan d'expérience avec plusieurs centaines de milliers de tirs à été conçu pour évaluer le système. Il définit un ensemble de variations du théâtre opérationnel : type de cible, réaction de

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity
PUBLIC

la cible, conditions initiales. Cela permet d'évaluer les performances par une méthode de Monte Carlo. Pour la suite du manuscrit nous considérerons 4 scénarios différents comportant :

- BS$\bar{\text{R}}$: cible de type bâtiment de surface sans aucune réaction.
- BSR : cible de type bâtiment de surface avec une réaction anti-torpille.
- SM$\bar{\text{R}}$: cible de type sous-marin sans aucune réaction.
- SMR : cible de type sous-marin avec une réaction anti-torpille.

Par exemple, le tir décrit par la figure 1.5 est un scénario sans réaction car la cible n'utilise aucun équipement de lutte anti-torpille et ne change pas de trajectoire à l'approche de la torpille. Si l'on considère que la cible est un bâtiment de surface, ce tir ferait donc partie du scénario BS$\bar{\text{R}}$.

1.3.2 Fonctionnement du simulateur

Le simulateur est basé sur le principe de la simulation à événements discrets, décrit dans la figure 1.7. Elle consiste en une modélisation informatique où le changement de l'état d'un système, au cours du temps, est une suite d'évènements discrets [6]. Chacun des acteurs du théâtre opérationnel ajoutent des événements dans la boucle événementielle. Chaque évènement est assigné à un instant donné et correspond à une modification de l'état du système. Ils sont alors exécutés par ordre chronologique à la date à laquelle ils sont assignés, ce qui permet de faire avancer le temps simulé.

Un point crucial pour un simulateur est le temps pris pour exécuter une simulation. En effet, cela permet, pour un temps donné, de définir le nombre de simulations exécutables. Ce nombre influe fortement sur l'intervalle de confiance des performances évaluées, et donc sur la fiabilité des résultats obtenus. Dans le cas du simulateur industriel considéré, la variabilité du temps de calcul par simulation est très importante. En fonction du scénario joué, plusieurs facteurs influent sur :

- Le nombre de secondes simulées par simulation. En effet, en fonction du scénario joué (distance initiale, réaction), le nombre de secondes simulées peut fortement varier. Or, augmenter le nombre de secondes simulées augmente le nombre d'événements à exécuter et donc le temps de calcul nécessaire à la simulation. Par exemple, parmi les différentes variations considérées au sein des scénarios, la distance initiale entre le lanceur et la cible influe sur le temps simulé nécessaire à la torpille pour s'en approcher.
- Le nombre d'événements par seconde simulé. Le fonctionnement par événement implique que le nombre d'événements par seconde simulé est variable en fonction de l'état courant du théâtre opérationnel. Par exemple, le nombre d'acteur présents sur le théâtre opérationnel augmente le nombre d'événements nécessaires pour les mettre à jour et les faire interagir entre eux (propagation acoustique).
- La temps de calcul moyen par événement. Le coût calculatoire peut être très différent entre les types d'événements, et même entre deux événements de même nature. En effet, il peut être fortement dépendant de l'état courant de l'environnement. Par exemple, pour la variabilité entre deux événements différents, il est souvent plus rapide de mettre à jour la position d'un élément en mouvement rectiligne uniforme que de simuler les signaux acoustiques reçus par un sonar. Pour la variabilité d'un même événement, le temps de calcul pour simuler les signaux acoustiques reçus par un sonar, dépend fortement du nombre de rayons à propager et donc du nombre d'éléments détectés dans la zone de portée acoustique du sonar (aussi appelé cône sonar, représenté en jaune dans la figure 1.5) au moment de l'émission acoustique.

Les figures 1.8 présentent des mesures permettant d'illustrer ces différentes variabilités de

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

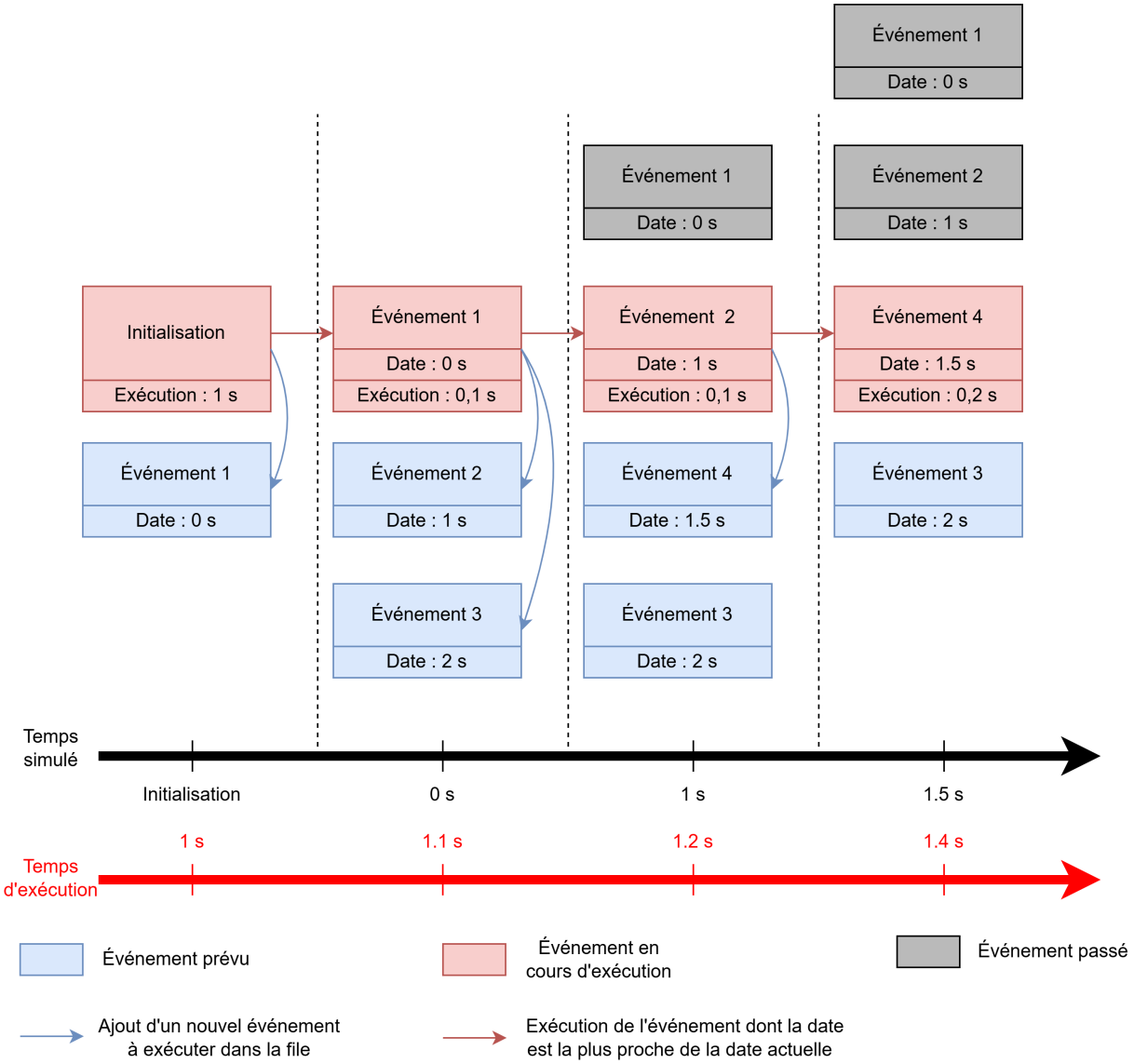
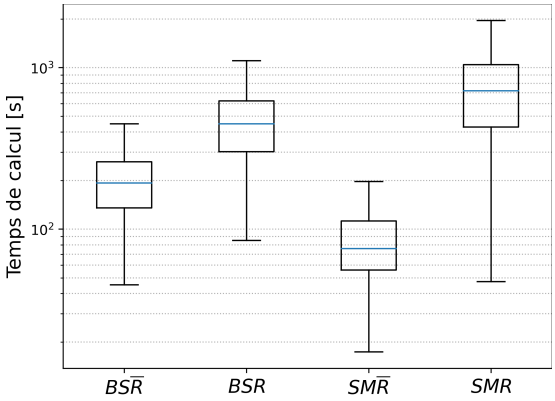


FIGURE 1.7 – Illustration de l'ordre d'exécution des événements et de l'évolution du temps simulé (nombre de secondes simulées) et du temps réel (temps pris pour l'exécution de la simulation) pour une simulation à événements discrets.

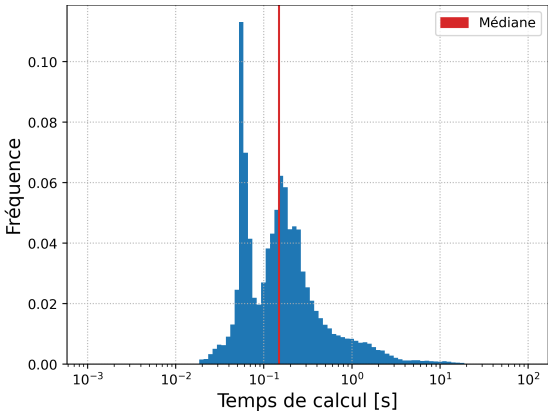
temps de calcul. La figure 1.8a présente les différences des temps de simulations entre les différents scénarios considérés. On remarque que lorsque la cible réagit (SMR et BSR) le temps de calcul augmente (peu importe le type de cible). En effet, cela est causé par une complexité plus grande du scénario. Donc, le temps simulé est plus grand car il est plus difficile d'atteindre la cible, le nombre d'acteurs est plus grand donc il y a plus d'événements et les événements sont plus coûteux. On peut voir que le temps de calcul dépend du type de cible. L'augmentation du temps de calcul due aux réactions est lui aussi différent avec un ratio de 2 entre les médianes sur bâtiment de surface (entre $\overline{\text{BSR}}$ et BSR) contre 10 sur sous-marin (entre $\overline{\text{SMR}}$ et SMR).

La figure 1.8b présente le temps moyen de calcul pour une fenêtre de temps simulé pour un ensemble de simulation fait avec le scénario SMR. On peut voir que la variabilité pour une fenêtre fixe de temps simulé est très forte avec un ratio de 10^3 entre les plus rapides et les plus

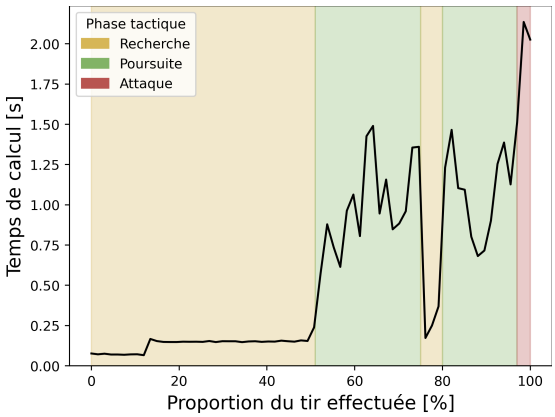
© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity
PUBLIC



(a) Démonstration de variabilité de temps de calcul inter scénarios. Répartition du temps de calcul moyen par simulation en fonction des différents scénarios considérés.



(b) Démonstration de variabilité de temps de calcul intra-scénario. Histogramme du temps d'exécution pour une fenêtre de temps simulé donné sur l'ensemble de SMR.



(c) Démonstration de variabilité de temps de calcul au sein d'une simulation. Temps d'exécution pour une fenêtre de temps simulé au cours d'un tir $SM\overline{R}$.

FIGURE 1.8 – Différents types de variabilité des temps de calcul au sein du simulateur industriel

lentes.

La figure 1.8c montre la variabilité du temps de calcul au sein d'une même simulation. On peut voir que lors de la première moitié du tir, les temps de calcul sont très rapides, cela correspond à la première phase de recherche, où la torpille n'a pas encore d'alarmes et donc il y a très peu d'interactions. Une fois que la torpille a ses premières alarmes, ce qui correspond à la fin de la première phase de recherche et au début de sa première phase de poursuite, on voit un fort ralentissement de la vitesse de calcul. Vers 75% du tir, la torpille perd sa cible, cela se traduit par une accélération du temps de calcul car la situation acoustique est beaucoup plus simple. Finalement, les dernières phases de poursuite et d'attaque sont numériquement plus couteuses, car elles impliquent une situation acoustique dense (nombreuses alarmes) et des traitements algorithmiques poussés.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

1.4 Le modèle de l'opérateur

Dans la précédente section, nous avons vu que le simulateur numérique est constitué de nombreux modèles (voir figure 1.3). Dans cette section, nous nous focalisons sur l'un d'entre eux : le modèle de l'opérateur (appelé modèle opérateur par la suite). Ce dernier occupe une place différente car il représente une partie du SATL et influe donc fortement sur la représentativité globale des performances évaluées.

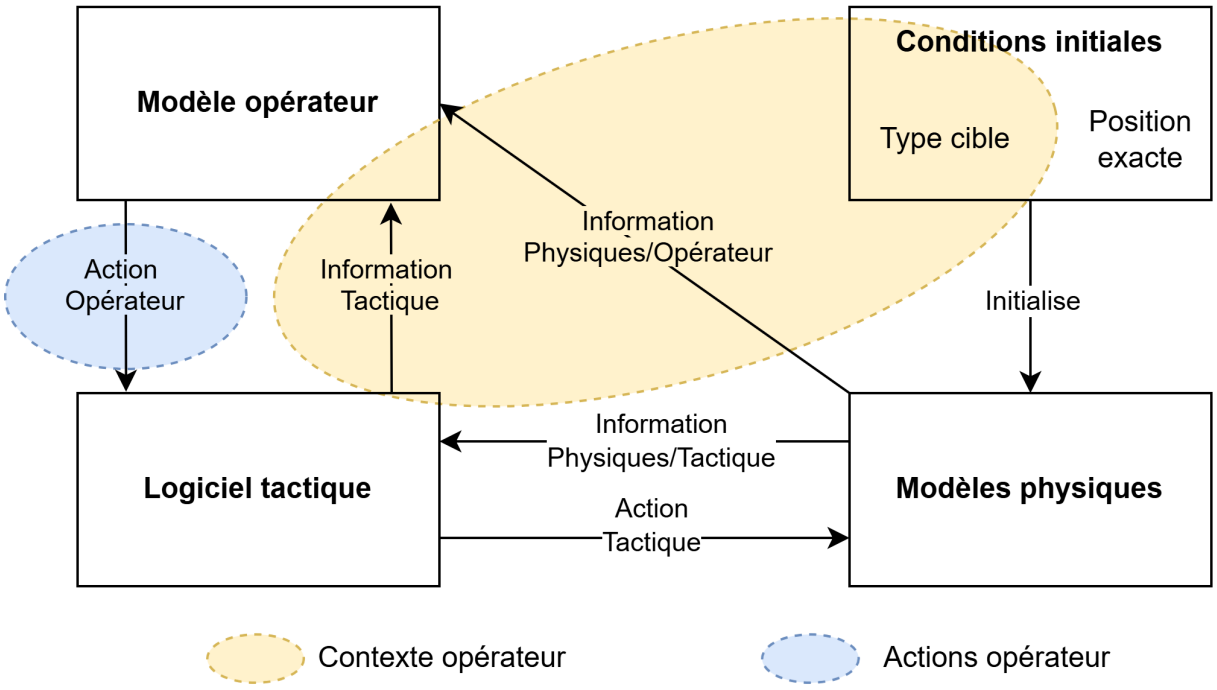


FIGURE 1.9 – Interactions du modèle opérateur au sein du simulateur.

La figure 1.9 décrit les interactions du modèle opérateur avec les autres éléments du simulateur. On définit le contexte opérateur comme l'ensemble des éléments à la disposition de l'opérateur pour prendre une décision. Il est composé :

- d'une partie des conditions initiales. En effet, l'opérateur possède des informations sur la cible qu'il souhaite neutraliser (type de cible, position initiale approximative, etc). Ces informations sont donc inchangées et disponibles tout au long du tir.
- d'informations "Physiques/Opérateur". C'est l'ensemble des données qui sont transmises directement à l'opérateur depuis les modèles physiques. Elles correspondent aux données issues de la perception du théâtre opérationnel par les capteurs simulés du lanceur. Ces informations sont donc dynamiques tout au long du tir. Il est important de noter que ces informations ne sont pas disponibles pour le logiciel tactique, qui lui, dispose des données perçues par le sonar simulé de la torpille.
- d'informations "Tactique". Ce sont toutes les données qui sont transmises directement à l'opérateur depuis le logiciel tactique. Elles sont issues des données remontées par la liaison filaire entre la torpille et le lanceur, comme l'état de la piste, les alarmes ou la phase courante du logiciel tactique. Cet ensemble de données est également dynamique au cours du tir.

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity
PUBLIC

Le modèle opérateur interagit de façon indirecte avec les autres modèles à travers le logiciel tactique. En effet, les actions disponibles au modèle opérateur sont les actions "Opérateur" traitées par le logiciel tactique. Ce dernier les prend en compte pour modifier ses décisions et donc les actions "Tactique" qui sont envoyées.

1.4.1 Représentativité du modèle opérateur

Le modèle opérateur doit simuler un comportement humain réagissant en temps réel aux informations du contexte opérateur. Cela implique plusieurs difficultés :

- Il n'existe pas d'équation permettant de décrire ce qui doit être fait par le modèle opérateur. Contrairement à des phénomènes connus, comme l'hydrodynamique, il est difficile de spécifier par des règles un comportement humain de façon générique.
- Ce modèle est fortement dépendant du contexte du théâtre opérationnel. Le nombre de situations dans lesquelles l'opérateur peut se retrouver est très grand de part les possibilités de positionnement et de l'hétérogénéité des éléments. Ainsi, l'opérateur n'a qu'une vision partielle de cet environnement.

En outre, évaluer la représentativité d'un modèle opérateur est difficile. En effet, les contraintes sur le modèle sont à la fois sur la performance globale du SATL mais aussi sur la manière dont il agit. Ces contraintes de représentativité sont :

- les performances attendues. En effet, l'objectif du modèle opérateur, contrairement au logiciel tactique, n'est pas de maximiser les performances obtenues en simulation numérique. Par construction du SATL, il est attendu que les actions opérateurs apportent un gain de performance, mais ce gain doit rester semblable par rapport aux conditions réelles.
- la réalisabilité du comportement. Le modèle doit être en adéquation avec les capacités physiques de l'opérateur. En effet, la fréquence d'action et le temps de réaction sont deux composantes du modèle qui doivent être humainement vraisemblables.
- la vraisemblance du comportement. Les actions prises par le modèle doivent être vraisemblables par rapport à celles d'un opérateur humain. Cela peut contraindre le modèle à ne pas faire des actions qui sont bénéfiques pour les performances mais dont le lien de cause à effet est trop compliqué ou inattendu.
- le contexte opérationnel. Un opérateur en condition réelle est en réalité dans un contexte difficile à modéliser car il subit une chaîne de commandement et un environnement stressant ce qui peut provoquer des imprécisions, voire des erreurs dans son raisonnement ou dans la saisie de l'action.

Au vu de la complexité modéliser un comportement humain plausible, il semblerait logique d'utiliser les données de tir réels. Cependant, la quantité de données utilisables est très faibles pour plusieurs raisons :

- Le coût de génération des données. En effet, faire un tir implique la mobilisation de plusieurs équipes et de matériel (voir section 1.2.4).
- Le contexte de génération des données : les essais à la mer ne sont pas forcément faits dans un contexte opérationnel parfait. Par exemple, certains sont réalisés à partir d'un lanceur différent, comme le catamaran industriel (le "Pégase" [7]), ou bien avec un opérateur qui n'est pas un marin.
- L'objectif de génération des données : la F21 étant un système complexe, tous les tirs ne sont pas faits pour tester uniquement le logiciel tactique en condition réelle mais plutôt de manière à récolter des données en utilisant des versions logicielles "instrumentées". Certains tirs ne sont donc pas faits dans des conditions et des réglages d'utilisation réelle.
- La date de génération des données : plus la date du tir est ancienne, plus la version du

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

logiciel utilisée et la façon d'utiliser l'arme peuvent être éloignées du présent. En effet, le logiciel tactique s'inscrit dans une démarche d'amélioration incrémentale.
— L'utilisation adéquate de l'arme : certains tirs sont des tirs d'entraînement pour les marins afin de se former, il est donc possible que certaines données réelles ne soient pas en adéquation avec ce qu'il est attendu d'un bon opérateur.

1.4.2 Modélisations

Jusqu'à ce jour, Naval Group a créé trois modèles opérateur différents.

Modèle actuel

Le modèle préexistant est régi par un ensemble réduit de règles simples. La partie traitée est assez représentative d'un opérateur mais son périmètre fonctionnel est fortement restreint par rapport à l'ensemble des possibilités mises à la disposition de l'opérateur. En effet, le modèle n'utilise qu'un nombre très restreint des actions possibles. Il n'est donc pas représentatif car très réduit dans ses capacités, et la logique humaine n'est pas bien représentée.

Modèle basé sur de la logique booléenne

Ce modèle utilise la logique booléenne à partir de règles métiers. Il est fait pour compléter le modèle actuel et étendre le périmètre des actions traitées. Suite à une phase de réglage itérative de 3 mois, plusieurs conclusions ont été faites :
— Les performances n'étaient pas satisfaisantes.
— Il n'a pas été possible de décrire tous les aspects comportementaux (voir section 1.4.1) à l'aide d'un ensemble de règles simples.
— L'utilisation de règles pour décrire le comportement d'un être humain nécessite de les intriquer fortement pour capturer sa complexité. Or, la doctrine d'emploi de l'arme évolue régulièrement forçant donc des mises à jour du modèle opérateur. Cette intrication rend le maintien de ce modèle sur le long terme non pérenne.

L'ensemble de ces problèmes a conduit à l'abandon de ce modèle.

Modèle basé sur une machine à états

Ce modèle utilise une machine à états construite à base de connaissances métiers. Il est également fait pour compléter le modèle actuel et étendre le périmètre des actions traitées. L'objectif est de représenter l'état de confiance de l'opérateur par rapport à la situation tactique par des états. Ainsi, chaque état conduit à des réactions différentes sur l'évolution de la situation tactique. Ce modèle est le plus performant et le plus représentatif mais reste non satisfaisant pour les mêmes raisons que le modèle à base de règles booléennes. Il n'est donc pas utilisé.

1.5 Motivations industrielles

Suite aux difficultés rencontrées pour modéliser un opérateur humain avec des méthodes de programmation conventionnelles, l'objectif des travaux de cette thèse est donc d'utiliser l'ensemble des données pour :

La modélisation de l'opérateur humain par une approche d'apprentissage automatique.

Une modélisation fidèle de l'opérateur permettrait donc d'améliorer fortement la représentativité actuelle du simulateur numérique et donc l'évaluation des performances du SATL.

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

En outre, la création d'un modèle opérateur représentatif peut permettre de mettre en place de l'aide à la décision pour un opérateur en condition réelle.

1.6 Contraintes de développement liées à la classification du sujet

Naval Group étant un fournisseur de la Marine nationale, plusieurs de ses activités sont soumises à des restrictions liées à la protection du secret de la Défense nationale. Le programme F21 en fait partie.

1.6.1 Sujet et données classifiées

La plupart des données industrielles manipulées au cours de ces travaux de thèse ne peuvent être divulguées pour des raisons de classification ou bien de besoin d'en connaitre. Cela implique plusieurs difficultés supplémentaires :

- Communication : les informations pouvant être diffusées sont fortement contrôlées. L'ensemble des informations ou résultats présentés doivent être blanchis (c'est le cas de l'ensemble de ce document). Cela conduit à un surplus de temps dans le traitement et la présentation des résultats ainsi qu'une réduction de l'impact des résultats. La présentation du logiciel tactique faite dans la section 1.2.3 en est un exemple.
- Processus : les travaux de recherche sont soumis à des processus lourds assurant la sûreté des données manipulées. Cela crée des surcoûts temporels lors du développement comme la validation et l'installation d'un environnement de travail ou bien l'accès à Internet.
- Connaissances : le sujet traité étant classifié du point de vue de la protection du secret de la Défense nationale, il l'est également sensible pour les autres pays. La littérature scientifique concernant des sujets connexes est donc très restreinte. Autrement dit, il est très difficile de trouver une publication scientifique autour de la torpillerie.

1.6.2 Puissance de calcul fixe

Du fait de la classification des données manipulées, les travaux ont été menés majoritairement dans un système d'information isolé de l'extérieur. Cela induit un accès à une quantité de puissance calcul de fixe. En effet, l'accès à des ordinateurs extérieurs au système d'information dans lequel le développement a été fait, est totalement impossible. En outre, l'homologation de nouveaux ordinateurs est longue. Cela implique qu'au cours de la thèse, l'ensemble de la puissance de calcul disponible a été :

- Un ordinateur fixe possédant :
  - 48 Go de RAM
  - un CPU avec 20 coeurs
  - un GPU avec 16 Go de VRAM
- 4 fermes de calcul possédant :
  - 256 Go de RAM
  - 256 coeurs

Contrairement à l'ordinateur dédié aux travaux de thèse, les fermes de calcul sont des ressources partagées par tous les utilisateurs du système d'information ce qui implique une disponibilité partielle au cours des travaux de thèse.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

Conclusion

Naval Group est un des industriels internationaux majeurs du naval de défense. Son activité est la conception, la production, l'intégration et le maintien en conditions opérationnelles de navires maritimes militaires. Parmi les différents produits qu'il met au point pour ces derniers, il assure notamment le développement de la F21, une torpille lourde équipant les SNA et SNLE de la Marine nationale. Un logiciel tactique embarqué, utilisant une machine à états, confère à l'arme un certain niveau d'autonomie et de performance. Néanmoins, l'opérateur dans le sous-marin est en liaison bidirectionnelle avec la torpille grâce à une liaison en fibre optique. Cette dernière lui permet d'obtenir des informations et d'agir sur la tactique choisie. Le duo formé par le logiciel tactique d'une part, et l'opérateur d'autre part, forme le Système d'Arme Torpille Lourde (SATL). Le développement du SATL repose sur l'utilisation de plusieurs types de données (simulation numérique, simulation hybride, essais mer) dont la représentativité est inversement proportionnelle au coût d'obtention (financier et temporel).

Parmi les différents moyens d'obtenir des données, le moins coûteux est le recours à un simulateur numérique. Il met en jeu un ensemble de modèles dans une boucle événementielle discrète dont le but est d'émuler un théâtre opérationnel représentatif afin d'évaluer et d'améliorer le logiciel tactique. Cela permet notamment de mettre en jeu la torpille dans des centaines de milliers de configurations opérationnelles, regroupées en différents types de scénarios. En outre, par construction, la rapidité du simulateur est fortement variable et dépend de l'état courant de l'environnement.

Au sein du simulateur numérique, l'opérateur est particulièrement difficile à modéliser car le but est de retranscrire un comportement humain. Il dispose d'informations hétérogènes et dynamiques pour choisir des actions dont les effets ne sont qu'indirects sur l'environnement. En effet, les leviers à la disposition de ce modèle sont des modifications à travers le logiciel tactique. De plus, la notion de représentativité est difficile à définir car il faut allier gain de performances et vraisemblance du comportement humain. Plusieurs travaux ont été menés via de la programmation symbolique pour modéliser cet opérateur, mais à ce jour, aucun n'a été jugé satisfaisant.

Enfin, les travaux menés dans cette thèse sont en parties soumises à la protection du secret pour la Défense nationale. Cela a eu plusieurs conséquences dont des restrictions sur la communication, un alourdissement des processus, une quasi-absence de ressources bibliographiques similaires et un accès à une puissance de calcul limitée.

Suite à l'exposition du contexte industriel dans ce chapitre, le chapitre 2 présente un état de l'art de l'apprentissage par renforcement profond orienté sur l'ensemble des algorithmes utilisés au cours de ces travaux. En effet, c'est cette approche qui a été retenu pour créer un modèle opérateur.

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity
PUBLIC

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

2

État de l'art

Ce chapitre fait un état de l'art de l'apprentissage par renforcement profond. L'objectif n'est pas d'être exhaustif mais néanmoins d'introduire l'ensemble des algorithmes et des concepts nécessaires pour la compréhension des chapitres suivants. L'apprentissage par renforcement et les réseaux de neurones sont présentés séparément. Cela permet ensuite la description des algorithmes d'apprentissage par renforcement profond, ainsi que l'étude des approches et difficultés liées à son application à un nouveau problème.

Sommaire

2.1 Travaux similaires 20
2.2 Apprentissage par renforcement 20
2.2.1 Formalisme et définitions 21
2.2.2 Résolution d'un MDP 24
2.2.3 Dilemme Exploration-Exploitation 26
2.2.4 *Off-Policy* et *On-Policy* 26
2.2.5 Exemple : Q-Learning 26
2.2.6 Fonction d'approximation 27
2.2.7 Descente de gradient 28
2.3 Réseaux de neurones 30
2.3.1 Définition 30
2.3.2 Architectures classiques 33
2.4 Apprentissage par renforcement profond 34
2.4.1 Problématiques induites 34
2.4.2 Algorithmes 35
2.5 Exploitation de connaissances extérieures 42
2.5.1 Transfer Learning 43
2.5.2 Utilisation de démonstrations 43
2.5.3 Information privilégiée 46
2.5.4 Structurer le problème 46
2.5.5 Curriculum Learning 47
2.6 Application de l'apprentissage par renforcement 48
2.6.1 Benchmarks 48
2.6.2 Librairies 48
2.6.3 Création d'un environnement 48
2.6.4 Comparaison 50

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

2.1 Travaux similaires

Le guidage de missile (ou plus précisément de torpille) est un problème de décision séquentielle auquel l'apprentissage automatique a déjà été appliqué. Parmi les différentes méthodes possibles, l'apprentissage par renforcement a été utilisé de nombreuses fois en simulation sur cette problématique. Elle a par exemple été utilisé pour créer des lois de guidages pour des missiles [8] [9] [10] [11], pour des missiles d'interception [80] [13] et pour des torpilles de suivi de sillage [14].

La modélisation d'un opérateur humain est un sujet issu du domaine des facteurs humains. En effet, on peut notamment citer le modèle SHEL (*Software*, *Hardware*, *Environment* et *Liveware*) [15] qui identifie un ensemble d'éléments influant sur la décision finale d'un opérateur humain. Ainsi, la compréhension d'un comportement humain est un problème multi-factoriel dont nous avons estimé qu'il était en dehors du périmètre de ces travaux. Nous avons donc choisi de créer un modèle opérateur en apprenant une solution au problème qu'il doit résoudre plutôt qu'en essayant de reconstruire la façon de penser d'un opérateur. Pour cela, l'apprentissage par renforcement est reconnu comme une méthode d'apprentissage automatique proche du comportement humain [16] [17]. En effet, l'apprentissage par renforcement profond a été utilisé pour modéliser un humain dans un environnement industriel [19], dans des jeux vidéo [20] ou un contexte de conduite [18].

Nous avons donc choisi l'approche de l'apprentissage par renforcement pour modéliser l'opérateur à partir du simulateur numérique.

2.2 Apprentissage par renforcement

L'ensemble de cette section s'inspire de [17].

L'apprentissage par renforcement, abrégé RL pour *Reinforcement Learning*, est une méthode d'apprentissage automatique qui se base sur l'évolution d'un agent dans un environnement. Le but de cet agent est d'apprendre le comportement qui maximise un scalaire obtenu au cours de son évolution. Pour le RL, il n'est donc pas nécessaire d'avoir des exemples ou un modèle complet du fonctionnement de son environnement. Il suffit que la maximisation du scalaire donnée au cours de ses interactions soit alignée avec la tâche que l'agent doit apprendre.

Les trois composantes principales d'un problème d'apprentissage par renforcement sont :

- Un agent, qui a connaissance de son état et qui peut agir sur un environnement au travers d'actions ;
- Un environnement, qui permet à partir d'un état et d'une action de l'agent, de transiter vers un nouvel état et de communiquer à l'agent son nouvel état ;
- Une fonction de récompense, qui, à chaque transition de l'environnement, calcule le scalaire évaluant le comportement de l'agent et qui doit donc être maximisé.

La figure 2.1 illustre ces interactions successives entre l'agent qui choisit une action, et l'environnement qui communique alors la valeur de la récompense et le nouvel état de l'agent. Ce processus continue indéfiniment ou bien jusqu'à une condition d'arrêt interne à l'environnement.

La problématique de l'apprentissage par renforcement est de réussir à découvrir le comportement maximisant l'espérance de la somme des récompenses reçues par l'agent au cours du temps.

L'apprentissage par renforcement permet de résoudre des problèmes de décision séquentielle. C'est-à-dire une suite de décisions successives dans le temps. Par exemple, des problématiques de contrôle robotique [78, 23], des jeux de sociétés [24, 25] ou des jeux vidéo [26, 27].

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

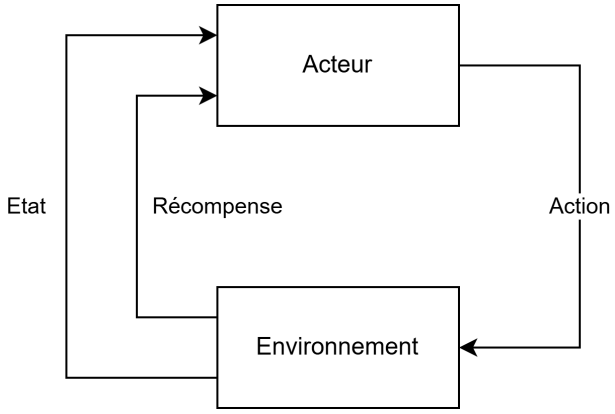


FIGURE 2.1 – Principe de l'apprentissage par renforcement

2.2.1 Formalisme et définitions

Cette section introduit les objets mathématiques utilisés pour l'apprentissage par renforcement.

Problème de décision de Markov

Le formalisme mathématique utilisé pour représenter un tel problème est le processus de décision markovien, abrégé MDP pour *Markov Decision Process*.

Un MDP est défini par :

— $\mathcal{S}$, un ensemble d'états qui décrit la perception de l'environnement par l'agent ;

— $\mathcal{A}$, un ensemble d'actions qui décrit l'ensemble des possibilités à la disposition de l'agent pour interagir avec l'environnement ;

— $\mathbf{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, une fonction de transition qui décrit les probabilités de transition d'un état vers un autre état sachant qu'une action a été prise ;

— $\mathbf{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, une fonction de récompense (ici considérée déterministe), qui permet de récompenser l'agent de façon positive ou négative.

Dans notre cas, on considérera que le MDP décrit un processus où le temps est discret et la fonction de récompense est déterministe.

Politique

Une fonction de politique π est définie par $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. Cette fonction représente les choix faits par l'agent en fonction de l'observation qu'il a à sa disposition. La définition précédente sous-entend qu'elle peut être probabiliste, mais il est également possible qu'elle soit déterministe.

Trajectoire

A partir d'un MDP et d'une politique, on peut créer une chaîne de Markov. C'est-à-dire une suite de variable aléatoire avec la propriété de Markov. Cette propriété indique que les variables aléatoires sont indépendantes de l'historique, autrement dit, connaître des informations sur les états antérieurs n'apporte aucune information supplémentaire sur l'état actuel. Une réalisation d'une telle chaîne de Markov est appelée une trajectoire.

Nous utiliserons seulement les MDP à horizon fini. C'est-à-dire qu'il existe un ensemble d'états qui sont des états terminaux indiquant la fin de la trajectoire et que toute trajectoire atteint

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity
PUBLIC

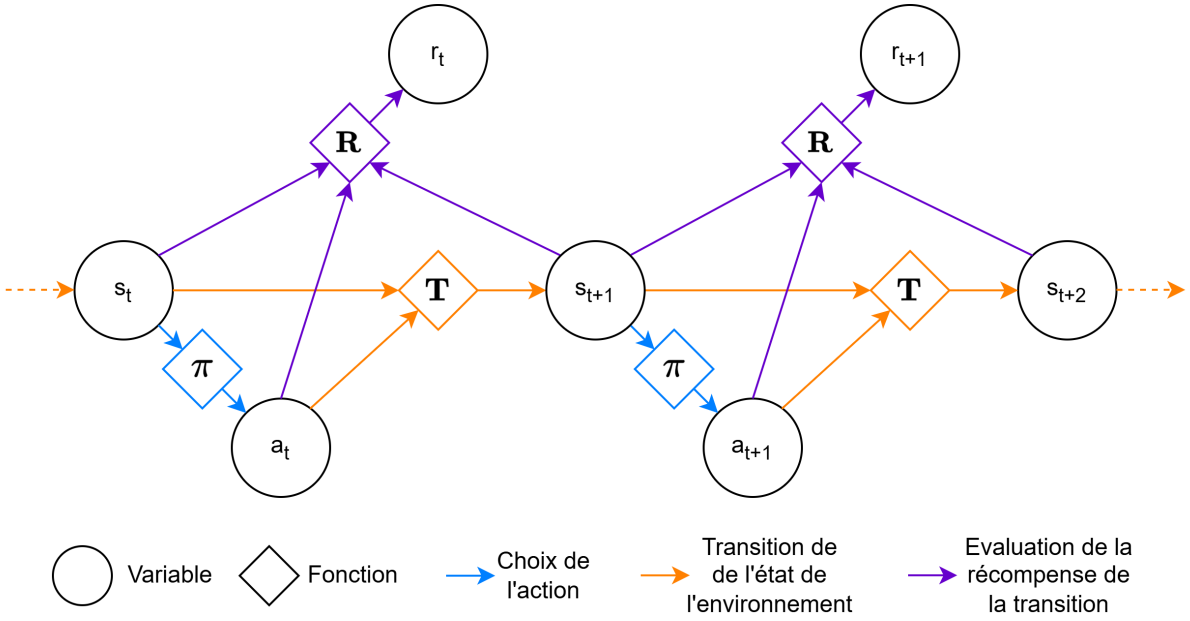


FIGURE 2.2 – Schéma d'un processus de décision markovien

un état terminal. L'indice terminal sera noté T. Cette chaîne implique une suite de variables aléatoires qui sont notées par :

— les états $(S_t)_{t\in[\![0,T]\!]} \in \mathcal{S}^{T+1}$;
— les actions $(A_t)_{t\in[\![0,T-1]\!]} \in \mathcal{A}^T$;
— les récompense $(R_t)_{t\in[\![0,T-1]\!]} \in \mathbb{R}^T$.

La figure 2.2 montre les liens entre les différentes variables aléatoires, qui sont régis par la fonction de transition, la fonction de récompense et la politique.

Une trajectoire est donc composée d'une suite de réalisations de variables aléatoires qui sont :

— les états observés $(s_t)_{t\in[\![0,T]\!]} \in \mathcal{S}^{T+1}$;
— les actions prises $(a_t)_{t\in[\![0,T-1]\!]} \in \mathcal{A}^T$;
— les récompenses obtenues $(r_t)_{t\in[\![0,T-1]\!]} \in \mathbb{R}^T$.

On définit alors la transition à l'instant t par le tuple : (s_t, a_t, r_t, s_{t+1}).

Fonctions de valeurs

On définit le gain cumulé réduit à l'instant t par :

$$G_t = \sum_{k=0}^{T-1-t} \gamma^k R_{t+k} \tag{2.1}$$

Avec $0 \leq \gamma \leq 1$, le facteur de réduction.

Le gain cumulé réduit prend en compte les récompenses futures de façon dégressive. La maximisation de cette quantité permet d'éviter un effet de myopie pour l'agent. En effet, s'il se contente d'optimiser ses actions pour une récompense instantanée, cela peut être néfaste pour la situation future. Le facteur γ permet de moduler l'importance accordée au futur par rapport au présent. Par exemple, avec $\gamma = 0$, on obtient un gain égal à la récompense à l'instant t et $\gamma \approx 1$, un gain pondérant autant les récompenses futures que la présente.

Corporate sensitivity PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

En développant l'équation 2.1, on observe que le gain à un instant donné est relié aux gain des instants suivants par :

$$G_t = R_t + \gamma G_{t+1} \tag{2.2}$$
$$G_t = R_t + \gamma R_{t+1} + \gamma^2 G_{t+2} \tag{2.3}$$
$$\forall n \in [\![0, T-t-1]\!],\ G_t = \sum_{k=0}^{n-1} \gamma^k R_{t+k} + \gamma^n G_{t+n} \tag{2.4}$$

Les fonctions de valeurs évaluent, à politique fixée, l'espérance du gain cumulé réduit à partir d'un état ou d'un couple état-action.

On définit la fonction de valeur état V, $\forall \pi$, où π est une politique, $\forall s \in \mathcal{S}$:

$$V_\pi(s) = \mathbb{E}_\pi(G_t | S_t = s) \tag{2.5}$$

La fonction de valeur état-action Q, $\forall \pi$, où π est une politique, $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$:

$$Q_\pi(s,a) = \mathbb{E}_\pi(G_t | S_t = s, A_t = a) \tag{2.6}$$

Une façon intuitive de définir une politique à partir d'une fonction de valeur état-action Q, est la méthode gourmande (ou *greedy*) qui consiste à prendre à partir de n'importe quel état, l'action dont la valeur de la fonction état-action est la plus haute. Autrement dit :

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A},\ \pi(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}(Q(s,a)) \tag{2.7}$$

Équation de Bellman

L'équation de Bellman tire profit de l'équation 2.2 pour définir un lien récursif entre les valeurs de la fonction de valeur états à l'instant présent et à l'instant suivant par :

$$V_\pi(s) = \mathbb{E}_\pi(R_t + \gamma G_{t+1} | S_t = s) \tag{2.8}$$
$$= \sum_{a \in \mathcal{A}} \pi(a,s) \sum_{s' \in \mathcal{S}} \mathbf{T}\left(s,a,s'\right)\left(\mathbf{R}\left(s,a,s'\right) + \gamma \mathbb{E}_\pi\left(G_{t+1} | S_{t+1} = s'\right)\right) \tag{2.9}$$
$$= \sum_{a \in \mathcal{A}} \pi(a,s) \sum_{s' \in \mathcal{S}} \mathbf{T}\left(s,a,s'\right)\left(\mathbf{R}\left(s,a,s'\right) + \gamma V_\pi\left(s'\right)\right) \tag{2.10}$$

La fonction de valeur V_π est l'unique solution de cette équation. L'équation de Bellman peut également se formuler de façon similaire pour la fonction de valeur état-action. Elle peut être étendue pour relier un instant présent à un instant futur arbitraire avec l'équation 2.4.

Formulation du problème

On définit la politique optimale par la politique qui choisit l'action dont le gain cumulé réduit espéré est le plus haut à partir de n'importe quel état.

Le problème de l'apprentissage par renforcement peut se formuler avec ces définitions par la recherche de la politique optimale.

Il est possible qu'il existe plusieurs politiques optimales, cependant, elles partagent toute la même fonction de valeur état, notée V_* :

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

$$\forall s \in \mathcal{S},\ V_*(s) = \max_{\pi}(V_\pi(s)) \tag{2.11}$$

On peut définir de façon similaire la fonction de valeur état-action optimale.

Toute politique qui est gourmande par rapport à ces fonctions de valeurs optimales, est une politique optimale.

2.2.2 Résolution d'un MDP

Les fonctions de valeur dépendent de la politique utilisée. Si on utilise les fonctions de valeurs pour améliorer la politique, on obtient alors une interdépendance. L'approche *Generalized Policy Iteration* (GPI) permet de créer une politique dans un MDP en alternant un processus d'évaluation et d'amélioration.

Generalized Policy Iteration

Le principe est la concurrence de deux processus. L'**évaluation**, qui a pour but de converger vers une évaluation correcte du gain espéré (avec une fonction de valeur) de la politique actuelle. L'**amélioration**, qui change la politique actuelle afin de prendre de meilleures actions au sens de la récompense. Ces deux processus sont donc opposés car le premier cherche à converger pour une politique donnée et le second modifie la politique. Cependant, la convergence théorique vers le point fixe (politique optimale, fonction de valeur optimale), est prouvée dans le cas où chaque état est visité un nombre infini de fois lors de l'apprentissage.

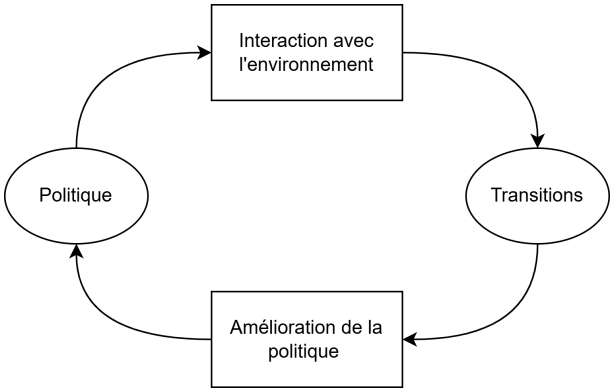


FIGURE 2.3 – Algorithme d'apprentissage par renforcement.

La figure 2.3 représente un algorithme d'apprentissage par renforcement classique utilisant le principe de GPI. C'est donc un processus itératif avec une succession de phases d'interactions entre la politique et l'environnement, et une phase d'amélioration de la politique. L'évaluation est faite grâce à l'analyse des récompenses obtenues au cours des interactions avec l'environnement. L'amélioration consiste alors en l'amélioration de la politique pour les futures interactions.

Model-based

Une méthode d'apprentissage par renforcement est dite *model-based* lorsque qu'on a connaissance de la fonction **T** et qu'on l'utilise.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

On peut alors utiliser une résolution par programmation dynamique en se basant sur l'équation de Bellman car tous les termes sont connus. Elle se base sur une résolution de l'équation 2.10 jusqu'à convergence pour évaluer la fonction de valeur.

Model-free

Une méthode d'apprentissage par renforcement est dite *model-free* lorsque $\mathbf{T}$ n'est pas connue mais il est tout de même possible de simuler le processus de transition et d'obtenir des tirages aléatoires de cette distribution à partir d'un couple état-action.

L'évaluation se base sur les récompenses obtenues lors de la simulation d'expériences.

Monte-Carlo

La méthode la plus simple pour évaluer les fonctions de valeur est celle de Monte Carlo, où des trajectoires sont simulées à partir d'un état donné et les gains obtenus sont moyennés pour obtenir une estimation de la fonction de valeur de l'état de départ.

En générant une trajectoire $(s_t, a_t, r_t)_{t\in[0,T]}$ à partir d'une politique, une règle de mise à jour directe est :

$$V(s_t) \leftarrow \alpha(g_t - V(s_t)) \tag{2.12}$$

Avec α, un taux d'apprentissage modulant la rapidité des mises à jour et g_t est la valeur obtenue du gain cumulé réduit lors de la trajectoire.

Temporal-Difference Learning

La méthode du *Temporal-Difference Learning* a été introduite par [28]. Cette méthode est *model-free*. Elle utilise l'équation 2.2 pour faire de la programmation dynamique. Ainsi, il est possible d'utiliser l'estimation de la fonction de valeurs de l'état suivant pour l'estimation de la fonction de valeur de l'état présent. Ce concept est appelé *bootstrapping*.

En générant une trajectoire $(s_t, a_t, r_t)_{t\in[\![0,T]\!]}$ à partir d'une politique, la règle de mise à jour est :

$$\forall t \in [\![0, T-1]\!],\ V(s_t) \leftarrow \alpha(r_t + \gamma V(s_{t+1}) - V(s_t)) \tag{2.13}$$

Avec α, le taux d'apprentissage.

Il est possible d'étendre cette formule avec l'équation 2.4. Ainsi, on obtient une règle de mise à jour qui généralise l'équation 2.12 et l'équation 2.13 :

$$\forall t \in [\![0, T-1]\!],\ \forall n \in [\![0, T-t-1]\!],\ V(s_t) \leftarrow \alpha\left(\sum_{k=0}^{n-1} \gamma^k R_{t+k} + \gamma^n V(s_{t+n}) - V(s_t)\right) \tag{2.14}$$

L'entier n permet de moduler le nombre de récompenses réellement expérimentées à prendre en compte avant d'utiliser. Cela permet d'avoir une variable de maîtrise entre le biais et la variance pour l'estimation de la fonction de valeur état. Plus n est petit, plus le biais est fort car on utilise plus l'estimation courante de la fonction de valeur mais la variance est plus faible car on utilise moins de récompenses expérimentées. Au contraire, plus n est grand, plus la variance est forte et moins le biais est fort.

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

2.2.3 Dilemme Exploration-Exploitation

Le dilemme Exploration-Exploitation est de réussir à allier le besoin d'expériences nouvelles pour explorer et évaluer correctement la fonction de valeur tout en utilisant la politique actuelle afin de tirer profit des connaissances acquises grâce aux expériences passées.

Un comportement favorisant seulement l'exploration tend vers une convergence lente, car l'agent n'utilise que très peu ses connaissances acquises pour orienter son exploration et a donc peu de chances de découvrir la meilleure politique.

Un comportement favorisant seulement l'exploitation aura une convergence plus rapide, mais aura beaucoup plus de chances de conduire à un comportement sous-optimal, car il n'aura pas exploré l'ensemble des possibilités.

Par exemple, la méthode gourmande crée une politique dans le paradigme de l'exploitation. En effet, l'agent prend à chaque fois l'action qui est considérée comme la meilleure.

Pour encourager l'exploration lors de l'apprentissage, on peut agir directement sur la politique en choisissant des actions considérées comme sous-optimales ou bien indirectement en agissant par exemple sur les fonctions de valeurs.

Une approche directe pour ajouter une part d'exploration à la méthode *greedy* est la méthode ε-*greedy*. Elle choisit l'action de plus grande valeur avec une chance de $1 - \varepsilon$ et sinon choisit aléatoirement une action. Le paramètre ε permet donc de moduler la part allouée à l'exploration et celle allouée à l'exploitation. La valeur du paramètre est souvent dynamique

Pour les méthodes indirectes, un exemple est la méthode *Upper-Confidence-Bound* qui augmente la fonction de valeur des actions de façon inversement proportionnelle à la fréquence avec laquelle elles sont choisies.

2.2.4 *Off-Policy* et *On-Policy*

Les méthodes *On-Policy* utilisent la politique optimisée pour interagir avec l'environnement lors de l'apprentissage. Les méthodes *Off-Policy* utilisent une politique différente de celle qui est optimisée pour interagir avec l'environnement. Pour ces dernières, on parlera donc de politique décisionnelle, pour celle qui interagit avec l'environnement, et de politique cible, pour celle qui est mise à jour à partir des données et optimisée.

La dissociation introduite par les méthodes *Off-Policy* peut permettre d'aider pour le dilemme exploration-exploitation. En effet, la politique décisionnelle peut être créée dans un but de générer les trajectoires qui maximisent l'intérêt pour l'apprentissage, tandis que la politique cible garde l'objectif d'être la meilleure possible.

Les méthodes *Off-Policy* introduisent un décalage entre la politique décisionnelle et la politique cible. En effet, certains choix d'action faits par la politique décisionnelle n'auraient peut-être pas été fait par la politique cible. Ce décalage peut conduire à une instabilité lors de l'apprentissage. Par exemple, pour endiguer ce décalage, il est possible de pondérer les mises à jour en fonction de la vraisemblance que la politique cible prenne cette action (méthode de l'*Importance Sampling*), ce qui permet de réduire la variance lors de l'apprentissage. Ainsi, tous les algorithmes d'apprentissage par renforcement ne sont pas capable d'utiliser des données *Off-Policy*.

2.2.5 Exemple : Q-Learning

L'algorithme 1 décrit le Q-Learning, un algorithme d'apprentissage par renforcement introduit par [29]. Son objectif est d'approximer la fonction de valeur état-action ce qui permet donc d'avoir une politique grâce à la méthode gourmande décrite dans la section 2.2.1. Comme on le

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

voit ligne 6, le choix de l'action pour les interactions avec l'environnement utilise l'approche *ε-greedy*, décrite dans la section 2.2.3. La ligne 8 montre le processus de mise à jour de l'estimation de la fonction de valeur. Il utilise l'équation de Bellman, d'une façon similaire à équation 2.13, mais pour une fonction de valeur état-action. Un point notable est que l'équation n'utilise pas la fonction de valeur état-action de la politique courante mais l'action dont la valeur est maximum sur l'ensemble des actions ce qui en fait une méthode *Off-Policy*.

Entrées : Taux d'apprentissage α, Taux d'exploration ε, Facteur de réduction γ
Sorties : Tableau de valeur état-action $Q[.,.]$
1 **Initialisation :** Initialisation de Q pour chacun des couples état-action possibles;
2 **répéter**
3 **Initialiser l'environnement** : donne s_0;
4 $s := s_0$
5 **tant que** s *n'est pas terminal* **faire**
6 **Choisir une action** : choix de a avec la méthode *ε-greedy* à partir de la valeur courante de Q.;
7 **Transition de l'environnement** : obtention du nouvel état s' et de la récompense r.;
8 $Q[s,a] := Q[s,a] + \alpha \left(r + \gamma \max_{a' \in \mathcal{A}} (Q[s',a']) - Q[s,a] \right)$;
9 $s := s'$
10 **fin**
11 **jusqu'à** Q *a convergé*;

Algorithme 1 : Algorithme Q-Learning

L'algorithme du *Q-Learning* a des garanties théoriques de convergence vers la politique optimale.

2.2.6 Fonction d'approximation

Les fonctions de valeurs et de politiques sont définies sur l'ensemble des états possibles de l'environnement. Il est donc nécessaire d'apprendre une valeur pour chacun des états ou couple état-action possibles. Traiter tous ces états indépendamment est irréalisable si le nombre d'états est trop grand (combinatoire des états possibles de l'environnement comme la disposition possibles des 32 pièces sur un échiquier de 64 cases pour le jeu d'échecs) ou bien lorsque l'espace d'état est continu (nombre d'états infini).

Une solution à ce problème est l'utilisation de fonctions d'approximation. Le concept est d'utiliser des familles de fonctions paramétrées définies sur l'ensemble de l'espace d'état. La phase d'apprentissage devient alors l'optimisation des paramètres pour réduire l'erreur sur les états expérimentés et non plus une estimation indépendante pour chacun des états possibles. Cela permet de généraliser un comportement aux états similaires vus lors de l'apprentissage sans nécessité d'observer tous les états.

L'utilisation de fonctions d'approximation engendre la perte des garanties de convergence vers une politique optimale des algorithmes classiques.

En outre, si l'espace d'action est continu, cela crée également des problèmes similaires pour évaluer le gain cumulé réduit de chacune des actions possibles.

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

2.2.7 Descente de gradient

Un mécanisme d'optimisation permettant d'obtenir des paramètres d'une fonction d'approximation est la descente de gradient. La descente de gradient se fait en définissant une fonction de coût, notée $\mathbf{L}$, qui dépend du vecteur de paramètres, noté $\boldsymbol{\theta}$, de la fonction d'approximation. La minimisation de cette fonction de coût doit induire que la fonction d'approximation se rapproche de la fonction cible.

Il existe deux familles d'algorithmes utilisant la descente de gradient. La différence fondamentale entre les deux est la fonction apprise pour créer la politique. La première famille utilise une fonction de valeur état-action, donc elle est à valeur dans $\mathbb{R}^{\mathcal{A}}$. La seconde apprend directement une politique, donc elle est à valeur dans $\mathcal{A}$.

Approximation de la fonction de valeur

L'algorithme du Q-Learning avec une fonction d'approximation est utilisé comme exemple. La fonction de valeur à approximer est donc $Q_{\boldsymbol{\theta}}$ et la fonction cible associée sera Q_*. On peut étendre l'équation de mise à jour du Q-Learning (voir algorithme 1) pour créer une fonction de coût :

$$\mathbf{L}(\boldsymbol{\theta}) = \mathbb{E}_{(s,a,r,s')\sim\pi}\left(\left(Q_{\boldsymbol{\theta}}(s,a) - \hat{Q}(r,s')\right)^2\right) \tag{2.15}$$

$$\hat{Q}(r,s') = \left(r + \gamma \max_{a'\in\mathcal{A}} Q_{\boldsymbol{\theta}}(s',a')\right) \tag{2.16}$$

Où l'espérance se fait sur des transitions $(s,a,r,s') \in \mathcal{S}\times\mathcal{A}\times\mathbb{R}\times\mathcal{S}$, issues de l'utilisation d'une fonction de politique π, avec $\hat{Q}$ la cible. Le terme $\left(Q_{\boldsymbol{\theta}}(s,a) - \hat{Q}(s')\right)^2$ est appelé l'erreur TD, en référence à la *Temporal-Difference*.

Ce principe s'incorpore alors au sein d'un processus de type GPI en définissant une suite de paramètres $(\boldsymbol{\theta}_i)_{i\in\mathbb{N}}$ qui doivent tendre vers les paramètres optimaux, $\boldsymbol{\theta}_*$. Ces paramètres optimaux sont tels que $Q_{\boldsymbol{\theta}_*}$ est le plus proche (au sens de la fonction de coût) possible de Q_* parmi l'ensemble des politiques représentables par la famille de fonction utilisée.

Pour cela, la fonction de coup est minimisée successivement à chaque phase d'évaluation, avec pour valeur :

$$\mathbf{L}(\boldsymbol{\theta}_i) = \mathbb{E}_{(s,a,r,s')\sim\pi_{\boldsymbol{\theta}_i}}\left(Q_{\boldsymbol{\theta}_i}(s,a) - \hat{Q}_{\boldsymbol{\theta}_i}(s')\right)^2 \tag{2.17}$$

$$\hat{Q}_{\boldsymbol{\theta}_i}(s') = \left(r + \gamma \max_{a'\in\mathcal{A}} Q_{\boldsymbol{\theta}_i}(s',a')\right) \tag{2.18}$$

Cependant, comme $\hat{Q}_{\boldsymbol{\theta}_i}$ dépend de $\boldsymbol{\theta}_i$, la méthode du semi-gradient est utilisée. Elle consiste à considérer la valeur de $\hat{Q}_{\boldsymbol{\theta}_i}$ comme constante par rapport à $\boldsymbol{\theta}_i$. On obtient alors la valeur du gradient utilisée pour chacune des phases d'évaluation :

$$\nabla_{\boldsymbol{\theta}_i}\mathbf{L}_i(\boldsymbol{\theta}_i) \propto \mathbb{E}_{(s,a,r,s')\sim\pi_{\boldsymbol{\theta}_i}}\left(Q_{\boldsymbol{\theta}_i}(s,a) - \hat{Q}_{\boldsymbol{\theta}_i}(s')\right)\nabla_{\boldsymbol{\theta}_i}Q_{\boldsymbol{\theta}_i}(s,a) \tag{2.19}$$

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

Approximation de la politique

Les méthodes approximant directement la politique utilise le théorème du *Policy Gradient* afin d'obtenir une expression du gradient des paramètres de la politique. Ainsi, il est possible d'optimiser directement une politique.

En définissant la fonction de politique par $\pi_{\boldsymbol{\theta}}$, où $\boldsymbol{\theta}$ est le vecteur de paramètres de la politique, la fonction à optimiser par rapport à $\boldsymbol{\theta}$, notée J, est l'espérance du gain à partir du premier état s_0 :

$$\begin{aligned} \mathrm{J}(\boldsymbol{\theta}) &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}(G_0|S_0 = s_0) && (2.20)\\ &= V_{\pi_{\boldsymbol{\theta}}}(s_0) && (2.21) \end{aligned}$$

Le théorème de distribution stationnaire démontre que pour une marche aléatoire suffisamment longue à travers une chaîne de Markov, la distribution selon les états possibles est indépendante de l'état initial. Cette distribution stationnaire sera notée $d_{\pi_{\boldsymbol{\theta}}}$. Cela permet de reformuler J par :

$$\begin{aligned} \mathrm{J}(\boldsymbol{\theta}) &= \sum_{s\in\mathcal{S}} d_{\pi_{\boldsymbol{\theta}}}(s) V_{\pi_{\boldsymbol{\theta}}}(s) && (2.22)\\ &= \sum_{s\in\mathcal{S}} d_{\pi_{\boldsymbol{\theta}}}(s) \sum_{a\in\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) Q_{\pi_{\boldsymbol{\theta}}}(s,a) && (2.23) \end{aligned}$$

Le théorème du *Policy Gradient* permet alors de prouver que :

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \sum_{s\in\mathcal{S}} d_{\pi_{\boldsymbol{\theta}}} \sum_{a\in\mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) Q_{\pi_{\boldsymbol{\theta}}}(s,a) && (2.24)\\ &\propto \sum_{s\in\mathcal{S}} d_{\pi_{\boldsymbol{\theta}}} \sum_{a\in\mathcal{A}} Q_{\pi_{\boldsymbol{\theta}}}(s,a) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) && (2.25) \end{aligned}$$

Ainsi, une descente du gradient de J par rapport à $\boldsymbol{\theta}$ est équivalent à augmenter la probabilité de choisir une action dans un état donné proportionnellement à une estimation de la fonction de valeur état-action.

L'application directe du gradient précédent donne empiriquement de mauvais résultats. En effet, comme dans le montre la figure 2.4, il est difficile d'estimer la différence de gain entre des trajectoires dont l'écart de gain est faible devant leurs valeurs. Dans cette optique, on définit alors la fonction d'avantage :

$$A(s,a) = Q(s,a) - V(s) \tag{2.26}$$

Cette fonction permet d'estimer le gain supplémentaire espéré en utilisant une certaine action dans un état par rapport au gain espéré si la politique courante est utilisée. La figure 2.4 illustre que cette fonction à une valeur moyenne beaucoup moins grande devant l'écart relatif. En outre, retirer une valeur de base qui ne dépend pas de a ne change pas la formule de l'équation 2.25. On obtient alors une nouvelle valeur du gradient en remplaçant la fonction de valeur état-action par la fonction d'avantage :

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \sum_{s\in\mathcal{S}} d_{\pi_{\boldsymbol{\theta}}} \sum_{a\in\mathcal{A}} A_{\pi_{\boldsymbol{\theta}}}(s,a) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s) \tag{2.27}$$

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

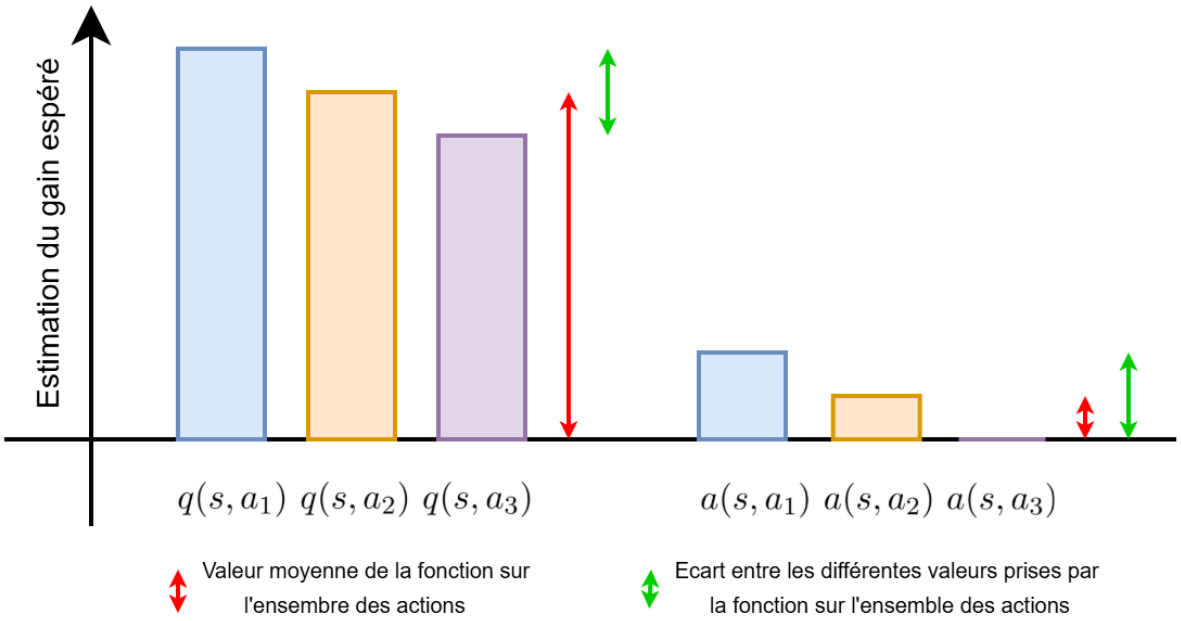


FIGURE 2.4 – Représentation de la valeur moyenne et de l'écart relatif pour plusieurs actions de la fonction valeur état-action et la fonction avantage.

En utilisant cette expression du gradient, la convergence est beaucoup plus stable. Cependant, il devient alors nécessaire d'estimer la fonction de politique et la fonction d'avantage. Ces méthodes sont appelées algorithmes *Actor-Critic*, *Actor* fait référence à la fonction de politique apprise et *Critic* fait référence à la fonction d'avantage apprise qui permet d'évaluer les actions et d'orienter les actions à prioriser.

2.3 Réseaux de neurones

Les réseaux de neurones sont inspirés du fonctionnement du cerveau humain. C'est une famille de fonction paramétrées qui peuvent approximer n'importe quelle fonction. Ils sont utilisés dans l'apprentissage automatique comme pour des problèmes de classification ou de régression.

2.3.1 Définition

Un réseau de neurones estime une fonction f qui a comme entrée une donnée x^{in} et une sortie x^{out} [30]. Il est organisé en plusieurs couches de neurones où chacune d'entre elles utilise un ensemble de poids qui les lient aux neurones de la couche précédente par une combinaison affine à laquelle on applique une fonction d'activation. On appelle "couches cachées" d'un réseau de neurones, toutes les couches qui ne sont ni la première, qui utilise la donnée d'entrée (x^{in}), ni la dernière, qui donne la valeur de sortie du réseau (x^{out}). La figure 2.5 représente un réseau de neurones dont les dimensions d'entrée et de sortie sont de 2 et qui a deux couches cachées de 3 neurones.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

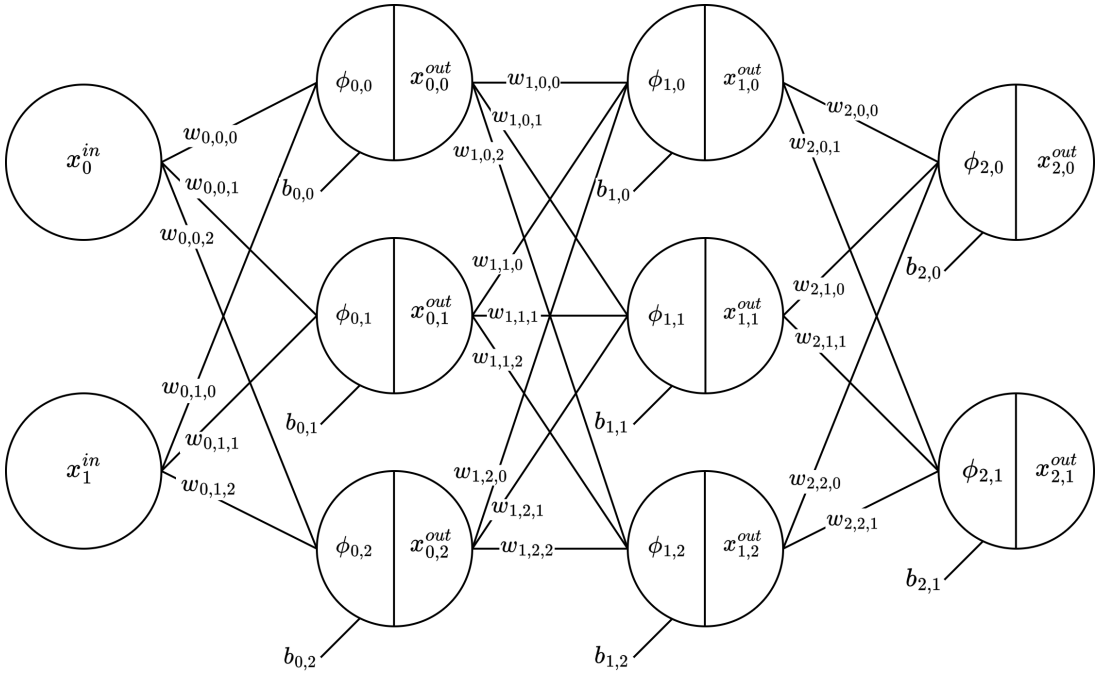


FIGURE 2.5 – Schéma d'un réseau de neurones à deux couches cachées de trois neurones, une couche d'entrée de dimension deux et une couche de sortie de dimension deux.

Évaluation

On note d_k la dimension de la sortie de la couche k, $x_{k,i}^{out}$ la composante i de la sortie de la couche k, $w_{k,i,j}$ le poids j de la combinaison affine du neurone i de la couche k, $b_{k,i}$ le biais de la combinaison affine du neurone i de la couche k et $\phi_{k,i}$ la fonction d'activation du neurone i de la couche k. La figure 2.5 montre l'agencement de ces différents objets pour un réseau de neurones. Le résultat de la composition de la combinaison affine par la fonction d'activation pour le neurone numéro i de la couche k d'un réseau est :

$$x_{k,i}^{out} = \phi_{k,i}\left(\sum_{j=0}^{d_{k-1}} w_{k,i,j} x_{k-1,j}^{out} + b_{k,i}\right) \tag{2.28}$$

L'architecture du réseau de neurones, c'est-à-dire le nombre de neurones de chacune des couches, ainsi que les fonctions d'activation, est choisie au préalable en fonction du type de fonction à estimer. Ce sont les poids des combinaisons affines de chaque neurone qui sont modifiés lors de la phase d'apprentissage. On obtient donc un type de fonction Φ définie par une architecture et paramétrée par l'ensemble des $w_{k,i,j}$ et des $b_{k,i}$.

Le théorème d'approximation universelle [31] démontre qu'un réseau de neurones à une seule couche cachée contenant un nombre fini de neurones avec une fonction d'activation non polynomiale peut approximer des fonctions continues sur des sous-ensembles compacts de $\mathbb{R}^n$. Ce théorème justifie théoriquement la capacité de généralisation et d'approximation des réseaux de neurones.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Rétropropagation du gradient

Lors de l'entraînement d'un réseau de neurones, la valeur d'une fonction de coût est minimisée [30]. La fonction de coût est censée être à son minimum lorsque le réseau a approximé au plus proche possible la fonction souhaitée. Cette minimisation est faite par une mise à jour itérative de la valeur des paramètres du réseau en utilisant le gradient de la fonction de coût.

La rétropropagation du gradient est un algorithme pour évaluer efficacement la valeur du gradient de la fonction de coût par rapport à chacun de ses paramètres. Cet algorithme calcule récursivement la valeur du gradient de l'erreur de la fonction de coût de la dernière couche vers la première.

En notant $\mathbf{L}$ la fonction de coût et K l'indice de la dernière couche, la relation entre la valeur du gradient des différentes couches est :

$$\frac{\partial\, \mathbf{L} \circ \Phi}{\partial w_{k,i,j}}\left(x^{in}\right) = x^{out}_{k-1,j}\delta_{k,i} \tag{2.29}$$

$$\frac{\partial\, \mathbf{L} \circ \Phi}{\partial b_{k,i}}\left(x^{in}\right) = \delta_{k,i} \tag{2.30}$$

Pour la dernière couche, $\delta_{K,i}$ vaut :

$$\delta_{K,i} = \phi'_{K,i}\left(\sum_{j=0}^{d_K} w_{K,i,j}x^{out}_{K,j} + b_{K,i}\right) * \mathbf{L}(x^{out}_{K,i}) \tag{2.31}$$

Pour toutes les autres couches :

$$\delta_{k,i} = \phi'_{k,i}\left(\sum_{j=0}^{d_k} w_{k,i,j}x^{out}_{k,j} + b_{k,i}\right) * \sum_{j=0}^{d_{k+1}} w_{k+1,i,j}\delta_{k+1,j} \tag{2.32}$$

La rétropropagation du gradient tire fortement profit des processeurs graphique pour accélérer son temps d'exécution car il repose majoritairement sur des calculs matriciels de grande dimension. Une fois le gradient calculé, on peut alors utiliser plusieurs algorithmes pour mettre à jour les poids comme la descente de gradient stochastique ou Adam [32].

Entraînement

L'entraînement du réseau est la phase d'ajustement des paramètres. Elle se fait grâce à l'évaluation du gradient de la fonction de coût sur un ensemble de données puis par une mise à jour des poids du modèle.

La qualité du modèle dépend fortement de la qualité des données, c'est-à-dire de leur capacité à représenter l'ensemble du problème, de la présence de biais, d'erreurs, de leur stabilité face à des situations similaires.

Plusieurs méthodes et hyperparamètres permettent d'améliorer la qualité et la rapidité de l'entraînement. Ils agissent à tous les niveaux de fonctionnement du réseau (données, mise à jour des poids, architecture du réseau) et peuvent être combinées.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

2.3.2 Architectures classiques

Définir l'architecture d'un réseau de neurones est le fait de choisir le nombre de couches de neurones, le nombre de neurones par couche ainsi que la façon dont il se connectent entre eux. On peut imposer des contraintes sur certains poids, comme être égaux ou être nuls. Cela permet d'une part de réduire le nombre de paramètres et d'autre part de forcer la façon d'interpréter les données (invariance temporelle, spatiale, ou cohérence locale). On obtient alors un réseau qui a plus de facilité à apprendre, car il a moins de paramètres et qu'il extrait intrinsèquement de l'information d'une façon prédéfinie et adaptée.

Perceptron multi-couche

Un perceptron multi-couche est un réseau où aucune contrainte n'est appliquée sur les poids et sur l'architecture, c'est-à-dire qu'il se compose uniquement de couches dites *fully connected*. Une couche est considérée comme *fully connected* lorsque chacun de ses neurones a un poids apprenable qui le relie à la sortie de chaque neurone de la couche précédente. Les couches du réseau décrit dans la figure 2.5 sont *fully connected*.

Convolutionnal neural networks

Un *Convolutionnal Neural Networks* (CNN) est un réseau qui contient des couches de convolution. Une couche de convolution utilise des noyaux de convolution, c'est-à-dire une matrice de poids de taille fixe qui est appliquée sur toutes les données. Cela permet d'extraire des caractéristiques localement, car chaque neurone est n'uniquement affecté que par un sous-ensemble des neurones de la couche précédente et de façon invariante selon les dimensions où le noyau se déplace.

Les réseaux convolutionnels permettent de réduire le nombre de poids à apprendre tout en forçant une compréhension locale des phénomènes de façon invariante selon la dimension de translation du noyau. Ils s'utilisent généralement pour le traitement d'image (convolution 2D pour une invariance spatiale) ou de séries temporelles (convolution 1D pour une invariance temporelle).

Recurrent neural networks

Un *Recurrent Neural Networks* (RNN) est un réseau de neurones ou des neurones sont connectés à eux-mêmes. C'est-à-dire que la même architecture est appliquée à chaque pas de temps et renvoie à la fois une valeur de sortie, mais également un vecteur d'information pour le prochain pas de temps. Cette architecture est majoritairement utilisée pour l'interprétation de données avec une cohérence temporelle (trajectoires, séries temporelles).

L'entraînement d'un réseau récurrent est assez complexe et instable, car l'implication des paramètres au sein d'une évaluation est beaucoup plus complexe. En effet, à chaque pas de temps, le même réseau est appliqué, avec comme vecteur d'entrée, la donnée à l'instant présent ainsi que le résultat précédent. Cela implique des problèmes de stabilité numériques avec potentiellement la dissipation ou l'explosion du gradient au fur et à mesure du parcours des appels du RNN. Le réseau oublie alors progressivement les informations du passé ou au contraire, une explosion des valeurs des instants passés se produit.

Deux architectures de RNN sont couramment utilisées : le *Long Short-Term Memory* (LSTM) [33] et le *Gated Recurrent Unit* (GRU) [34]. Ces deux architectures utilisent des systèmes de portes pour gérer la mémoire à court et à long terme.

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

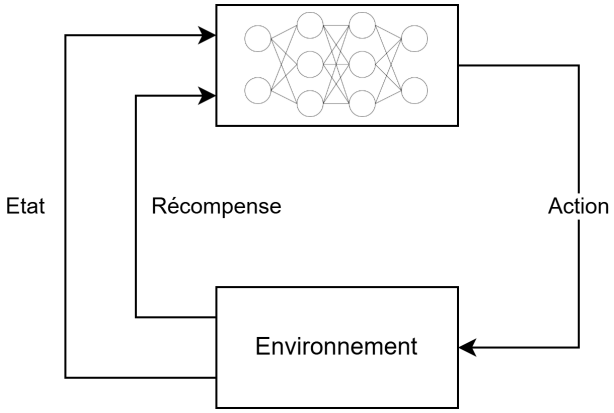


FIGURE 2.6 – L'apprentissage par renforcement profond.

Transformers

Cette architecture a été introduite pour la première fois par [35]. Elle se base sur l'interprétation des inter-dépendances entre différents éléments d'une séquence de longueur indéterminées.

Cette architecture permet notamment de traiter des problèmes où les données sont temporelles, de façon similaire aux RNN. Cependant, elle a l'avantage de ne pas traiter les informations de façon séquentielle ce qui la rend beaucoup plus rapide en temps de calcul. C'est notamment ces architectures qui sont à la base des modèles de l'état de l'art pour le traitement du langage naturel, par exemple les différentes versions de GPT (*Generative Pretrained Transformer*) [36].

2.4 Apprentissage par renforcement profond

L'apprentissage par renforcement profond, abrégé DRL pour *Deep Reinforcement Learning*, décrit l'ensemble des algorithmes qui utilisent un réseau de neurones comme fonction d'approximation dans un problème d'apprentissage par renforcement. Ainsi, la politique apprise est un réseau de neurones, comme représenté par la figure 2.6. Le vecteur d'entrée du réseau est l'observation de l'environnement, la sortie du réseau permet de choisir une action et la récompense est utilisée pour le calcul de la fonction de coût et donc du gradient permettant de mettre à jour les poids de la politique.

2.4.1 Problématiques induites

L'utilisation de réseaux de neurones au sein de l'apprentissage par renforcement couple les problématiques de stabilité de l'entraînement rencontrées dans les deux domaines. Mais cela crée également de nouvelles difficultés.

Perte de la convergence théorique

Comme expliqué dans la section 2.2.6, l'utilisation d'une fonction d'approximation, et donc d'un réseau de neurones, fait perdre les garanties de convergence théorique. Par exemple, pour le Q-Learning, il existe des contre-exemples connus de divergence lors de l'utilisation d'une fonction d'approximation [37].

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

Deadly triad

Le problème de *Deadly triad* [17] peut apparaître lorsque qu'un algorithme d'apprentissage par renforcement combine trois concepts : le *bootstrapping*, l'apprentissage *Off-policy* et les fonctions d'approximation. Cela peut conduire à une instabilité de l'algorithme l'empêchant de converger vers une politique efficace.

En effet, le *bootstrapping* utilise la fonction de valeurs courante pour l'estimation de la valeur et pas seulement les récompenses, ce qui permet donc une divergence. L'utilisation d'un apprentissage *Off-policy* peut conduire à une politique exploratoire qui ne visite pas assez fréquemment les états entrain de diverger. L'utilisation de fonction d'approximation amplifie le problème car une mise à jour d'un état impacte également les états similaires. Ainsi, la fonction de valeurs peut potentiellement diverger.

Corrélation des transitions

Au sein d'un épisode, les situations rencontrées par l'agent sont organisées de façon séquentielle (car issues d'un MDP) et sont donc fortement corrélées les unes aux autres. Cette forte corrélation entre les transitions est néfaste pour l'entraînement d'un réseau de neurones [26].

Efficacité calculatoire

L'apprentissage par renforcement ainsi que les réseaux de neurones profonds sont deux domaines pouvant demander une forte puissance de calcul.

La génération de trajectoire est hautement séquentielle. Il faut donc généralement utiliser des CPUs pour la simulation d'un environnement, alors que l'utilisation de réseaux de neurones est fortement accélérée par l'utilisation de GPUs.

Cette dualité génère des contraintes sur l'efficacité de l'utilisation des données générées et sur l'architecture globale pour minimiser les temps de calcul. Le choix le plus efficient n'est pas toujours le même et dépend fortement du contexte [38] (taille du réseau, rapidité de l'environnement).

Le temps d'entraînement de l'apprentissage par renforcement est souvent mentionné comme une limite. Il peut aller de quelques heures à plusieurs dizaines de jours pour les modèles les plus complexes [39] [40]. Ainsi, il est fortement dépendant de la puissance de calcul disponible. Les modèles les plus avancés ont souvent eu besoin de cluster de calculs très puissants, couplant CPU et GPU pour réussir à développer des comportements complexes. Par exemple, le *cluster* d'entraînement d'OpenAI, nommé *rapid*, est constitué de 128 000 coeurs CPUs et de 256 GPUs P100 [41].

En outre, dans le cas de l'utilisation d'un modèle à base de réseaux de neurones dans une simulation numérique, le temps d'inférence peut également être un problème [42]. Par exemple, pour une utilisation en temps réel, cela impose un temps maximal d'inférence à ne pas dépasser. Ou, pour un modèle de simulation, il peut être intéressant de considérer un compromis entre performance et temps d'inférence.

2.4.2 Algorithmes

La première réussite notable d'apprentissage par renforcement profond vient de Mnih et al. [26] où les auteurs ont appliqué efficacement le *Deep Q-Learning* (DQN). Ils ont ainsi créé un agent capable de jouer à un ensemble de jeux Atari [43] depuis des images brutes. Cette réussite se base sur l'algorithme présenté dans la section 2.2.7 en utilisant un réseau de neurones

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

convolutionnel pour approximer la fonction de valeur. En outre, plusieurs ajouts algorithmiques ont permis ce succès dont le rejeu d'expérience et l'utilisation d'un réseau cible.

Il existe plusieurs algorithmes de DRL qui utilisent l'approche *Actor-Critic* présentée dans la section 2.2.7. Parmi les algorithmes les plus courant utilisant cette approche, on peut citer :

- *Proximal Policy Optimization* [45] : un algorithme *On-Policy* qui utilise le *Generalized Advantage Estimation* [44] ainsi qu'une amélioration de la politique par zone de proximité.
- *Soft Actor-Critic* [46] : un algorithme *Off-Policy* qui utilise une régularisation entropique, le rejeu d'expérience et l'utilisation de deux fonctions de *Critic*.

Les améliorations algorithmiques du DRL sont souvent des ajouts successifs aux algorithmes DQN ou *Actor-Critic*. Par exemple, l'algorithme *Rainbow DQN* [47] se base sur DQN en ajoutant 6 modifications : une mise à jour *n-step* (voir l'équation 2.14), le *Prioritized Experience Replay* [51], le *Double Q-learning* [48], le *Dueling Net* [50], le *Noisy net* [52] et le *Distributional DQN* [53]. Plus récemment, on peut citer Agent 57 [54], qui est le premier algorithme à avoir réussi à atteindre un niveau sur-humain sur l'ensemble des jeux Atari. Il utilise également l'addition de plusieurs modifications algorithmiques : une politique utilisant des réseaux récurrents, une architecture de parallélisation asynchrone, une optimisation automatique des hyperparamètres et l'ajout d'un terme de récompense intrinsèque à l'exploration.

Dans la suite de cette section, nous détaillerons uniquement les différents algorithmes utilisés pour la mise au point du modèle opérateur dans les chapitres suivants.

Rejeu d'expérience

Le rejeu d'expérience, introduit par [55], est le principe de stocker les transitions expérimentées par un agent dans une mémoire appelée *replay buffer*. La figure 2.7 situe l'ajout du *replay buffer* au sein d'un algorithme de RL (voire figure 2.3). Les phases d'évaluation et d'amélioration de la politique se font sur des transitions tirées depuis le *replay buffer*. Cela induit plusieurs effets :

- une décorrélation des transitions tirées car elles viennent d'un plus grand nombre de trajectoires différentes ;
- une réutilisation des transitions générées par interaction avec l'environnement ;
- un effet *Off-Policy* car les transitions utilisées ne sont pas forcément issues de la politique courante.

Il est souvent irréalisable de stocker toutes les transitions simulées, ainsi, choisir lesquelles stocker, lesquelles tirer et la façon de les utiliser, impacte fortement la qualité de ce mécanisme. Plusieurs paramètres jouent sur son dimensionnement [56] :

- la quantité d'expériences retenue ;
- l'ancienneté des transitions, c.-à-d. le nombre de modifications de la politique depuis que cette transition a été expérimentée ;
- le ratio de rejeu, c.-à-d., le ratio entre le nombre de mises à jour des paramètres et le nombre de transitions générés avec l'environnement.

Plus il y a de données, plus elles sembleront indépendantes et identiquement distribuées. Cependant, s'il y a trop d'expériences les chances d'utiliser des expériences intéressantes (c.-à-d. récentes et/ou qui contiennent encore des informations non apprises) seront moins grandes.

Plus les transitions sont récentes, plus on se rapproche d'un apprentissage *On-Policy* car la politique qui a généré ces expériences est plus proche de la politique actuellement optimisée.

Certaines méthodes biaisent le tirage des expériences du *replay buffer*. La méthode la plus courante est le *Prioritized Experience Replay* (PER) [51] qui augmente la chance de tirer une expérience en utilisant la valeur de l'erreur TD. L'erreur TD est utilisée comme un indicateur de

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

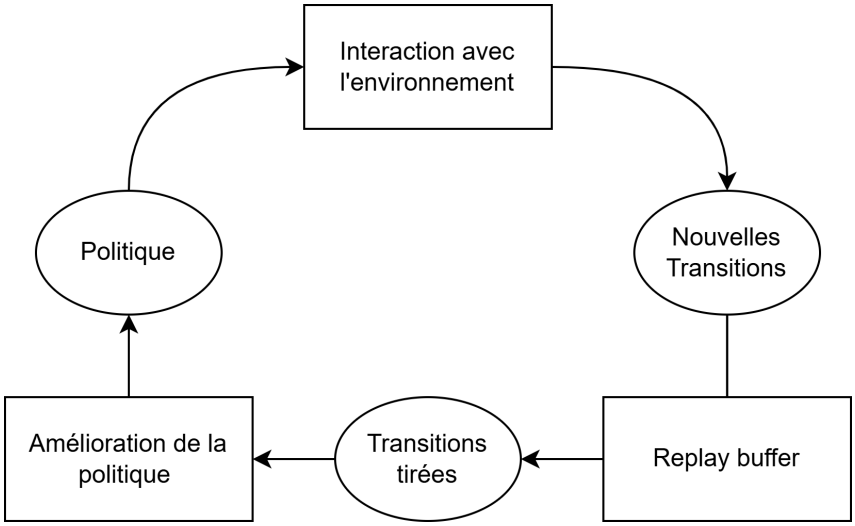


FIGURE 2.7 – Algorithme d'apprentissage par renforcement avec le mécanisme de rejeu d'expérience.

la quantité d'information restante à apprendre de cette transition, il y a donc une augmentation des chances de tirer des transitions avec une forte erreur TD.

Utilisation d'un réseau cible

Comme mentionné dans la section 2.2.7, pour l'évaluation de la fonction de coût $\mathbf{L}$, on utilise le réseau optimisé à la fois pour la fonction à optimiser et sa cible. Cela est problématique car la valeur cible change en même temps qu'on essaie de s'en rapprocher, rendant le problème non stationnaire. En effet, à l'itération i, la descente de gradient va modifier la valeur de $\boldsymbol{\theta}_i$ afin que $Q_{\boldsymbol{\theta}_{i+1}}$ se rapproche de $\hat{Q}_{\boldsymbol{\theta}_i}$. Cependant, comme la cible dépend également des paramètres, à chaque itération, on essaie de s'approcher d'une cible différente. Cette inter-dépendance peut conduire à une instabilité de l'apprentissage.

Afin de réduire ce problème, [26] ont introduit l'ajout d'un réseau cible. Le réseau cible, est un second ensemble de paramètre, noté $\boldsymbol{\theta}'_i$, pour l'évaluation de la cible $\hat{Q}_{\boldsymbol{\theta}'_i}$. L'équation 2.17 devient :

$$\mathbf{L}_i\left(\boldsymbol{\theta}_i\right) = \mathbb{E}_{(s,a,r,s')\sim\pi_{\boldsymbol{\theta}_i}}\left(\left(Q_{\boldsymbol{\theta}_i}(s,a) - \hat{Q}_{\boldsymbol{\theta}'_i}\right)^2\right) \tag{2.33}$$

L'objectif de $\boldsymbol{\theta}'_i$ est d'être modifié plus lentement que $\boldsymbol{\theta}_i$ afin que l'optimisation par descente de gradient soit localement plus proche d'un problème stationnaire. Cependant, ce décalage entre $\boldsymbol{\theta}'_i$ et $\boldsymbol{\theta}_i$ rend la valeur cible légèrement fausse. Donc $\boldsymbol{\theta}'_i$ doit à la fois être stable au cours des mises à jour, pour rendre le problème le plus stationnaire possible, et rester le plus proche du paramétrage de la politique, afin que la cible soit la plus juste possible. Il existe deux approches courantes pour créer $\boldsymbol{\theta}'_i$:

- *Hard Update* : toutes les k mise à jour du réseau, les paramètres $\boldsymbol{\theta}_i$ sont copiés par $\boldsymbol{\theta}'_i$.
- *Soft Update* : à chaque mise à jour de $\boldsymbol{\theta}_i$, $\boldsymbol{\theta}'_i$ suit la règle de mise à jour suivante :

$$\boldsymbol{\theta}'_{i+1} = \boldsymbol{\theta}'_i + \tau(\boldsymbol{\theta}'_{i+1} - \boldsymbol{\theta}'_i) \tag{2.34}$$

  Avec $\tau < 1$, un hyperparamètre modulant la rapidité de prise en compte des changements des paramètres.

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity
PUBLIC

L'approche *Hard Update* est l'approche rendant le problème le plus stationnaire mais, elle crée un décalage potentiellement important entre les paramètres, au contraire de l'approche *Soft Update* garde les paramètres plus proches mais avec un problème moins stationnaire.

La méthode *Soft Update* est la plus courante. Elle est actuellement fortement utilisée pour les approches basées sur le DQN ou de type *Actor-Critic* [46]. Il est possible de la remplacer par d'autres méthodes pour empêcher la divergence comme des fonctions d'activation bornées ou bien un mécanisme de *batch normalization* [57].

Double Q-Learning

Le *Q-Learning* souffre d'un biais de surestimation dans les environnements fortement stochastiques. Ce biais est dû à l'utilisation du maximum sur les actions des valeurs estimées plutôt que d'utiliser le maximum sur les actions des valeurs espérées (voir équation 2.17) [48].

Le *Double Q-Learning* permet d'enlever le biais de sur-estimation. Pour cela, deux fonctions de valeur état-action sont créées, notées Q^A et Q^B. Q^A va alors être utilisé pour l'évaluation de la cible Q^B, et inversement. Cela donnes comme nouvelles règles de mise à jour pour l'algorithme 1 :

$$Q^A[s,a] := Q^A[s,a] + \alpha \left(r + \gamma \max_{a' \in \mathcal{A}} \big(Q^B[s',a']\big) - Q^A[s,a] \right) \tag{2.35}$$

$$Q^B[s,a] := Q^B[s,a] + \alpha \left(r + \gamma \max_{a' \in \mathcal{A}} \big(Q^A[s',a']\big) - Q^B[s,a] \right) \tag{2.36}$$

Afin d'assurer la suppression du biais, chacune des transitions sera utilisé pour la mise à jour d'un seul ensemble de paramètre. Pour le choix des actions par la politique, on utilise la méthode gourmande sur la moyenne de $Q^A[s,a]$ et $Q^B[s,a]$.

Le biais de surestimation du *Q-Learning* se retrouve également dans le DQN [49]. Le concept de *Double Q-Learning* a donc été adapté au DQN. Or comme, le DQN utilise déjà un dédoublement des paramètres (voir paragraphe précédent), [49] a alors utilisé le réseau cible pour évaluer la cible. Avec les notations précédentes, elle s'écrit :

$$\hat{Q}(s,a,r,s') = r + \gamma\, Q_{\boldsymbol{\theta}'}(s', \underset{a' \in \mathcal{A}}{\operatorname{argmax}}(Q_{\boldsymbol{\theta}}(s',a'))) \tag{2.37}$$

Parallélisation des agents

Afin d'augmenter la quantité de transitions obtenues, plusieurs instances de l'environnement sont utilisées en parallèle. La figure 2.8 décrit l'architecture de parallélisation la plus simple : l'architecture synchrone. Plusieurs processus sont instanciés en parallèles, contenant chacun un environnement et la politique courante, ils seront appelés acteur. Chacun des acteurs va faire le même nombre de transitions. Une fois que chacun des acteurs a fini sa tâche, l'ensemble des transitions est utilisé pour la mise à jour de la politique. Puis la nouvelle politique est envoyée à chacun des acteurs pour créer de nouvelles transitions. Cela permet à la fois d'accélérer le processus d'apprentissage en augmentant le nombre de transitions générées par seconde et les transitions rencontrées sont plus décorrélées car elles viennent de simulations différentes.

Nair et al. [58] ont introduit Gorila (*General Reinforcement Learning Architecture*), une architecture où tous les processus sont totalement désynchronisés. Plusieurs acteurs génèrent des transitions en parallèles. Ces transitions alimentent un *replay buffer* commun. Des processus d'apprentissage améliorent continuellement la politique en tirant aléatoirement des expériences du *replay buffer*. Les améliorations sont alors centralisées pour obtenir la politique courante. Les acteurs actualisent périodiquement la politique qu'ils utilisent avec la politique courante. La

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

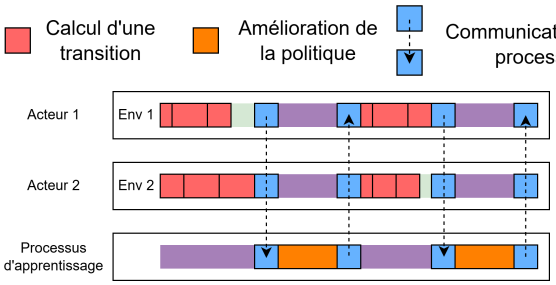


FIGURE 2.8 – Agencement des tâches effectuées par les différents processus avec une architecture synchrone pour un algorithme de DRL.

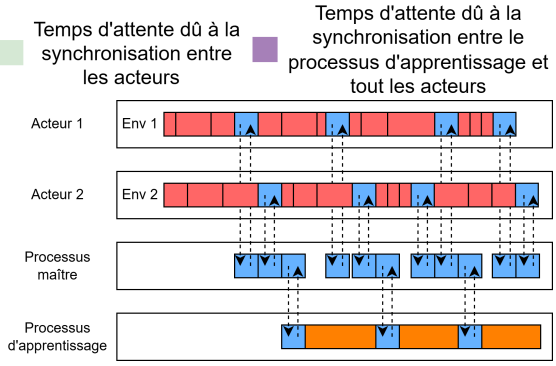


FIGURE 2.9 – Agencement des tâches effectuées par les différents processus avec une architecture asynchrone pour un algorithme de DRL.

figure 2.9 représente l'agencement des tâches pour un apprentissage avec une telle architecture avec 2 acteurs et un processus d'apprentissage. Cette architecture est souvent appelée l'*Actor-Learner Architecture*, où *Actor* fait référence aux acteurs, et *Learner* fait référence aux processus d'apprentissage.

Cette architecture permet d'enlever les temps d'attente créés par la synchronisation entre les processus (zones vertes sur la figure 2.8). En outre, cette architecture est très efficace pour profiter d'une augmentation des ressources calculatoires allouées pour l'apprentissage. Cependant, comme les acteurs n'utilisent pas forcément la dernière version de la politique, cela introduit un effet *Off-Policy*.

Mnih et al. [59] ont également appliqué une architecture *Actor-Learner* à plusieurs algorithmes classiques. Cependant, ils ont choisi une approche n'utilisant pas de rejeu d'expérience en estimant que le nombres d'environnement en parallèle était suffisant pour décorréler les transitions utilisées. En outre, leur implémentation utilise qu'une seule machine afin de réduire les surcoûts de communication entre les processus et utilise l'approche Hogwild [60] pour faire une descente de gradient stochastique asynchrone.

La méthode *IMPortance weighted Actor-Learner Architecture* (IMPALA) utilise également cette architecture [62]. Les auteurs ont notamment ajouté la possibilité d'utiliser plusieurs processus d'apprentissage séparés afin d'augmenter le nombre de mise à jour du réseau possible. De plus, ils utilisent une méthode de pondération des expériences pour réduire l'instabilité due au décalage entre la politique utilisée par les *actors* et celle optimisée par le *learner*.

En utilisant cette approche, le nombre de transitions générées peut être très important. Le PER a donc été étendu à l'architecture asynchrone [61] afin de prioriser les transitions les plus intéressantes parmi ce large ensemble.

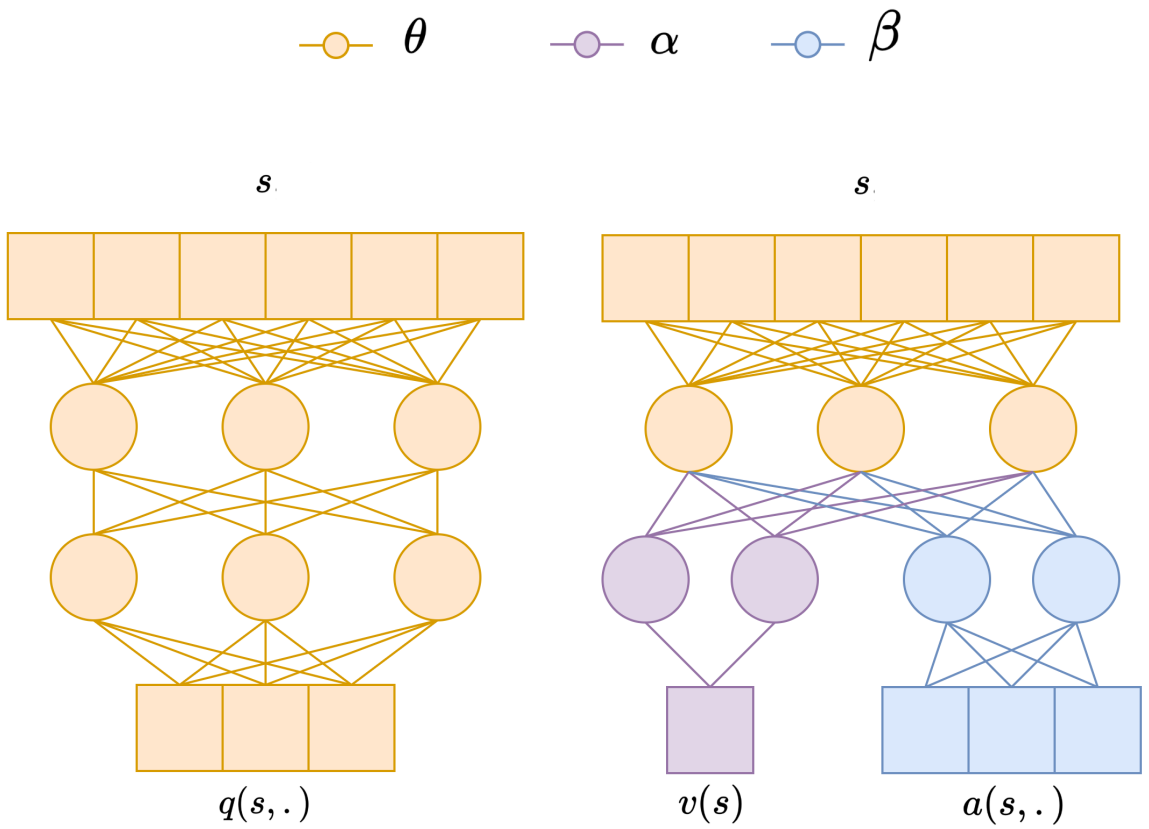
La méthode *SEED* utilise également l'architecture *Actor-Learner* [63]. Cependant, l'inférence, pour obtenir l'action, est déportée des *Actors* aux *Learners*. Cela permet de regrouper les inférences de plusieurs acteurs pour profiter de l'accélération des GPUs pour l'inférence des réseaux de neurones [64]. En outre, cela permet de réduire les coûts de communications interprocessus en ne communiquant que les trajectoires (c.-à-d. pas de transmission des paramètres, ni des gradients).

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

Dueling Net

Le *Dueling Net* est une modification du DQN, qui porte sur le réseau permettant d'estimer la fonction de valeur Q [50]. Dans le DQN, le réseau de neurones a pour but d'approximer la fonction Q, l'architecture la plus simple est donc de l'estimer directement (voir figure 2.10a). L'architecture *Dueling Net* utilise le lien entre la fonction de valeur état, V, la fonction de valeur état-action Q et la fonction d'avantage A (voir équation 2.26). Comme montré dans la figure 2.10b, le *Dueling Net* estime à la fois la fonction V et A. Pour cela, un réseau commun est utilisé, avec comme paramètres θ, puis deux sous-réseaux différents sont utilisés pour chacune des fonctions, avec comme paramètres α pour V et β pour A. Ces deux valeurs vont alors être combinées pour obtenir une estimation de Q :

$$Q_{\boldsymbol{\theta},\boldsymbol{\alpha},\boldsymbol{\beta}}(s,a) = V_{\boldsymbol{\theta},\boldsymbol{\alpha}}(s) + \left(A_{\boldsymbol{\theta},\boldsymbol{\beta}}(s,a) - \frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} A_{\boldsymbol{\theta},\boldsymbol{\beta}}(s,a') \right) \tag{2.38}$$



(a) Architecture d'un réseau de neurones estimant la fonction de valeur état-action Q avec un seul ensemble de paramètres θ.

(b) Architecture d'un réseau de neurones avec l'architecture *Dueling Net* [50],avec un ensemble de paramètres commun, noté θ, un ensemble de paramètres uniquement pour l'estimation de la fonction de valeur état V, noté α et un ensemble de paramètres uniquement pour l'estimation de la fonction d'avantage A, noté β.

FIGURE 2.10 – Deux architectures de réseaux de neurones pour estimer la fonction état-action Q pour l'algorithme DQN.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

L'idée est de découpler l'apprentissage de l'intérêt d'un état et de l'intérêt qu'une action a dans cet état. Cette approche est particulièrement efficace quand il existe des états où aucune des actions n'a d'effet [50].

Noisy Net

Le *Noisy Net* est un mécanisme de gestion du dilemme exploration-utilisation automatique intégré au niveau du réseau de neurones [52]. Cet algorithme ajoute de la stochasticité directement au sein du réseau de neurones.

Dans le cas d'un réseau de neurones classiques, l'inférence se fait par une combinaison affine entre les différentes couches du réseau (voir l'équation 2.28). De façon générale, elle s'écrit donc :

$$y = wx + b \tag{2.39}$$

Où w et b sont les paramètres appris par le réseau.

Pour le *Noisy Net*, l'équation est modifiée pour ajouter un bruit gaussien paramétrique. La formule est :

$$y = (\mu^w + \sigma^w \odot \epsilon^w)x + (\mu^b + \sigma^b \odot \epsilon^b) \tag{2.40}$$

Où μ^w, σ^w, μ^b et σ^b sont les paramètres appris par le réseau. ϵ^w et ϵ^b sont deux variables aléatoires et $\odot$ désigne le produit terme à terme.

Cette modification ajoute naturellement un aspect stochastique à la politique, ce qui permet de remplacer les méthodes classiques d'exploration (ε-*greedy* pour DQN). En outre, cette approche ne nécessite pas d'hyperparamètres car la quantité d'exploration est ajustée automatiquement au cours de l'apprentissage.

Observabilité partielle

L'hypothèse de Markov est une contrainte très forte (voir section 2.2.1) car il est souvent difficile de n'avoir besoin que de l'état courant pour le choix d'une action. Pour décrire un tel problème, le formalisme classique est le processus de décision markovien partiellement observable (POMDP). Pour les définir, on peut ajouter à la définition classique d'un MDP (voir section 2.2.1) les éléments :

— Ω : l'ensemble des observables.
— $\mathbf{O} : \Omega \times \mathcal{S} \to [0; 1]$, un fonction d'observation. Elle définit la probabilité d'observer chacun des observables en étant dans un état.

On définit alors pour une trajectoire dans un POMDP la suite de variable aléatoire $(O_t)_{t\in[\![0,T]\!]} \in \Omega^{T+1}$, comme l'ensemble des observations expérimentées lors de cette trajectoire. Un POMDP implique que le choix d'une action à un instant t ne peut être pris que grâce à l'ensemble des observations expérimentées $(o_t)_{t\in[\![0,T]\!]}$. En effet, dans le cas d'un POMDP, il n'est plus possible d'avoir accès à l'état courant. Ainsi, une politique n'utilise plus un élément de $\mathcal{S}$, mais un ensemble d'éléments de Ω pour la prise de décision. La figure 2.11 illustre le lien entre les différents éléments introduits dans cette section et la section 2.2.1 pour un POMDP.

Les approches principalement utilisées pour définir une politique permettant de résoudre un POMDP sont les suivantes :

— utiliser une politique définie sur une concaténation d'observations passées. Ainsi, si le nombre d'observations concaténées est suffisant, la politique peut alors comprendre les

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

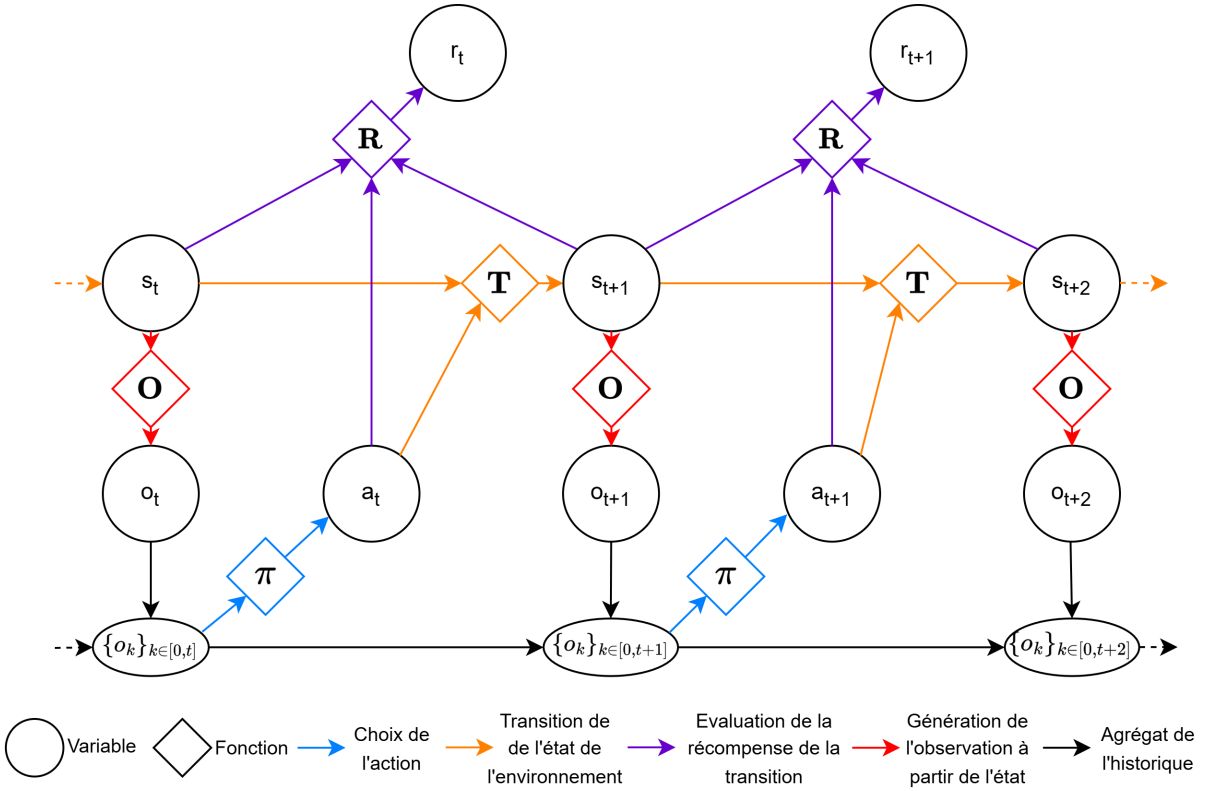


FIGURE 2.11 – Schéma d'un processus de décision markovien partiellement observable.

concepts nécessaires au problème. Par exemple, [26] concatène 4 images afin d'avoir accès à une notion de déplacement des objets affichés dans l'environnement Atari.

— utiliser une politique capable d'interpréter une série temporelle. Par exemple, les réseaux récurrents(présenté dans la section 2.3.2) permettent de garder des informations des observations passées grâce à l'état caché transmis entre chaque pas de temps [40].

L'utilisation des réseaux récurrents avec le rejeu d'expérience peut être instable car l'état caché du réseau (le vecteur qui est transmis entre les couches récurrentes) ne correspond pas à celui généré par la politique actuelle (effet *Off-Policy*). En effet, comme la politique courante est constamment modifiée, cela modifie également les informations transmises dans l'état caché créant ainsi un décalage car l'état caché de la transition dans le *replay buffer* a été générée par une politique précédente [65] [66].

2.5 Exploitation de connaissances extérieures

En pratique, en DRL, il est souvent possible d'avoir des informations a priori sur l'environnement cible. Cela peut prendre plusieurs formes, par exemple, des politiques passées, des démonstrations par des experts métiers ou bien simplement des connaissances métiers a priori sur la façon de résoudre le problème. L'utilisation de ces connaissances peut améliorer l'apprentissage.

Corporate sensitivity
PUBLIC
© Propriété Naval Group SA, (2026) tous droits réservés.

2.5.1 Transfer Learning

Le *Transfer Learning* (TL) désigne l'ensemble des méthodes dont le but est de faire un transfert de connaissances à partir d'un ancien modèle vers un nouveau modèle afin d'accélérer sa convergence ou d'améliorer son efficacité [67].

Ce qui caractérise chacune ces méthodes viennent donc des différences de contextes entre le modèle initial et le modèle final (espace d'états, espace d'action, transitions, récompense, modèle) ainsi que du support de transfert (démonstrations, dynamique du modèle, politique, fonction de valeurs) et comment il est utilisé (biais sur l'exploration, initialisation du modèle, ajout de démonstration).

Il existe plusieurs métriques et concepts pour évaluer l'apport d'une approche TL [67] [68]. Les métriques considérées sont (voir figure 2.12) :

- le *Time to threshold* : il mesure la différence de temps pour que les politiques atteignent un niveau de performance fixé entre deux processus d'apprentissage ;
- l'*Asymptotic performance* : elle mesure la différence de performance après convergence ;
- le *Jumpstart* : elle mesure la différence de performance au début de l'apprentissage ;
- le *Weak transfer* : un *Weak Transfer* ne prend pas en compte le temps passé à générer les connaissances utilisées pour le transfert ;
- le *Strong transfer* : un *Strong Transfer* prend en compte le temps passé à générer les connaissances utilisées pour le transfert.

Plusieurs autres concepts permettent d'évaluer l'intérêt et l'applicabilité d'une approche de TL :

- la *Performance sensitivity* : la sensibilité de la performance aux hyperparamètres ;
- le *Necessary knowledge amount* : la quantité nécessaire de connaissances pour que le TL atteigne un certain niveau d'efficacité ;
- le *Necessary knowledge quality* : la qualité nécessaire des connaissances pour que le TL atteigne un certain niveau d'efficacité.

Le TL permet un développement incrémental de modèles se basant sur des modèles précédents ou des connaissances a priori, afin d'obtenir le meilleur modèle possible. Pour le jeu de Go, Silver et al. [24] ont utilisés de l'apprentissage supervisé à partir de coups d'expert humain puis ont utilisé de l'apprentissage par renforcement à partir de ce modèle.

2.5.2 Utilisation de démonstrations

L'apprentissage en utilisant des démonstrations désigne l'exploitation de comportements générés par un modèle extérieur dans le cadre de la décision séquentielle [69]. C'est donc une approche de TL qui utilise les transitions pour transférer des connaissances d'une politique experte vers une nouvelle politique à apprendre. Ces méthodes offrent plusieurs avantages :

- Permettre de décrire le problème à résoudre par la recherche d'un comportement similaire aux exemples sans avoir à définir de fonction de récompense.
- Utiliser des données déjà existantes permet de profiter des données déjà présentes, réduisant ainsi le coût de l'apprentissage, notamment si l'interaction avec l'environnement est coûteuse ou dangereuse.
- Augmenter la vitesse de convergence en s'inspirant de comportements experts qui répondent déjà partiellement ou totalement à la tâche souhaitée.

Il existe plusieurs approches qui dépendent notamment de différents critères comme :

- La possibilité ou non d'interagir avec l'environnement en plus de profiter des démonstrations.

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

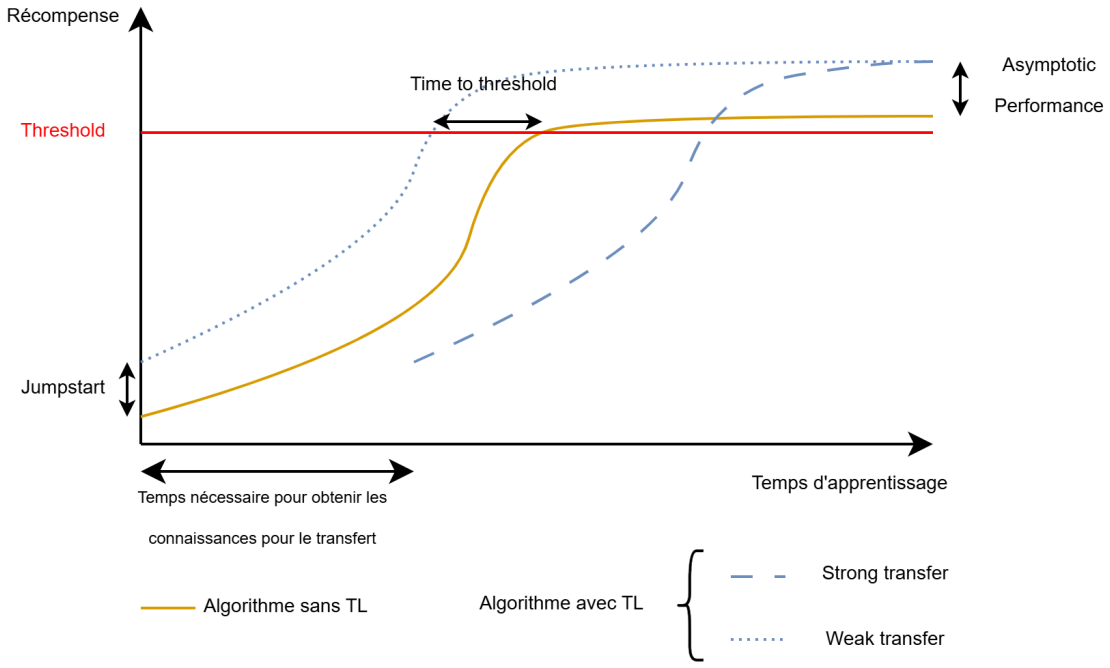


FIGURE 2.12 – Illustration des différentes métriques pour les méthodes de TL [67].

— La possibilité d'interagir avec la politique experte au cours de l'apprentissage.
— L'accès à la fonction de récompense.
— L'accès à un modèle de transition.

Imitation Learning

L'*imitation learning* (IL) définit les approches dont le but est d'apprendre à partir de démonstrations pour obtenir un comportement le plus proche possible de celles-ci [70]. Ainsi, l'IL utilise les démonstrations comme comportement à apprendre et n'a donc pas besoin de fonction de récompense. Les approches se divisent selon deux principes : le *behaviour cloning* et l'*inverse reinforcement learning*.

Le *behaviour cloning* est de l'apprentissage supervisé en utilisant les démonstrations comme un ensemble de données étiquetées. Lors de l'apprentissage, la politique est entraînée à choisir exactement les mêmes actions pour les mêmes observations que celles présentes dans les démonstrations [71]. L'approche est donc de créer un comportement qui copie les démonstrations sans rechercher à maximiser la récompense.

L'*inverse reinforcement learning* infère la fonction de récompense pour laquelle les démonstrations sont optimales. Une méthode classique de RL est alors utilisée à partir de cette fonction de récompense apprise.

L'apprentissage à partir des démonstrations seules est fortement dépendant de la qualité des données de démonstration :

— La quantité.
— La représentativité : la capacité des données à décrire toutes les situations possibles.
— L'optimalité : à quel point les démonstrations reflètent réellement un comportement optimal.

Corporate sensitivity
PUBLIC
© Propriété Naval Group SA, (2026) tous droits réservés.

Apprentissage interactif

S'il est possible d'interagir avec l'environnement pour obtenir des données supplémentaires lors de l'apprentissage, cela permet d'améliorer la généralisation de la politique tout en augmentant la vitesse de convergence grâce aux démonstrations [69].

Par exemple, si l'algorithme utilise un *replay buffer*, il est possible d'ajouter les démonstrations comme des expériences passées. Cela permet donc de traiter les démonstrations comme un ensemble de transitions déjà jouées, et de gérer dynamiquement la place que les démonstrations prennent au sein de la mémoire de l'agent.

Dans [72], Hester et al. ont ainsi combiné la méthode du *Deep Q-Learning* avec l'incorporation de démonstrations dans un *replay buffer* afin d'accélérer la convergence. Ils ont alors montré sur l'environnement ALE [43] une amélioration significative des performances finales et de la vitesse de convergence. Plus récemment, Gulcere et al. [73] ont utilisé l'incorporation de démonstration au sein du *replay buffer* sur l'algorithme proposé par Kapturowski et al. [66], permettant également d'améliorer les performances obtenues pour des environnements où l'exploration est complexe.

QDagger

L'algorithme QDagger [75] transfert des connaissances depuis une politique "enseignante" à laquelle on a accès (c.-à-d. dont on peut avoir les probabilités d'action), lors d'un apprentissage intéragissant directement avec l'environnement. Cet algorithme combine le Deep Q-Learning [26] avec le *n-step* (voir l'équation 2.14) et l'algorithme Dagger [74], un algorithme d'IL interactif.

Il utilise la fonction de coût du DQN avec le *n-step* et ajoute un terme de *distillation* qui impose une proximité entre la politique apprise et la politique "enseignante". La fonction de coût devient donc :

$$\mathbf{L}_{QDagger}(\boldsymbol{\theta}) = \mathbf{L}_{DQN}(\boldsymbol{\theta}) + \lambda_t \mathbb{E}_{(s,a,r,s')\sim\pi_{\boldsymbol{\theta}}}\left(\sum_{a\in\mathcal{A}} \pi_E(a|s)\pi_{\boldsymbol{\theta}}(a|s)\right) \tag{2.41}$$

Avec $\mathbf{L}_{DQN}$ qui correspond à la fonction de coût du DQN décrite par l'équation 2.37 en plus de l'approche *n-step*, λ_t un paramètre décroissant linéairement de λ_0 à 0 au cours de l'entraînement modulant la prise en compte du terme de *distillation* et π_E la politique enseignante.

Pour l'évaluation de $\pi_{\boldsymbol{\theta}}(.|s)$ à partir de $Q_{\boldsymbol{\theta}}(.|s)$, la représentation du *Soft Q-Learning* est utilisée selon [76] :

$$\pi_{\boldsymbol{\theta}}(.|s) = \text{softmax}\left(\frac{Q_{\boldsymbol{\theta}}(.|s)}{\tau}\right) \tag{2.42}$$

Avec τ, un paramètre de température qui permet de moduler la différence de gain espéré nécessaire pour que la politique estime qu'une action est plus probable qu'une autre.

En outre, le *replay buffer* est initialisé grâce à la politique enseignante. Si elle est issue d'un entraînement, le *replay buffer* à la fin de l'entraînement est utilisé, sinon la politique enseignante est utilisée pour générer des expériences.

Une phase de pré-entraînement est faite sur les données initiales du *replay buffer* issues de la politique enseignante. Cela permet d'apprendre à la politique apprise d'imiter la politique enseignante.

Les avantages de cet algorithme sont les suivants :

— une indifférence au type de la politique enseignante. L'algorithme n'impose aucune contrainte sur l'architecture ou l'algorithme utilisé pour générer la politique enseignante.

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

— une indépendance de la politique finale à la politique enseignante. En effet, la politique apprise ne garde aucune dépendance à la politique enseignante utilisée lors de l'apprentissage.
— une efficacité sur l'aspect calculatoire et sur l'utilisation des données. Cet algorithme permet d'accélérer l'apprentissage en utilisant de façon efficace les connaissances de la politique enseignante. En outre, il est également possible que la politique apprise devienne meilleure que la politique enseignante.

2.5.3 Information privilégiée

L'utilisation d'information privilégiée se place dans un POMDP où il est possible d'avoir accès à l'état de l'environnement en plus des observations lors de la phase d'apprentissage. Cela permet d'apprendre une fonction utilisant l'état afin de guider et d'accélérer l'apprentissage d'une politique utilisant les observations. La fonction utilisant l'état est donc inutilisable hors de la phase d'apprentissage car elle est définie sur $\mathcal{S}$, et non sur Ω, elle dispose donc d'informations privilégiées.

Cette approche a permis plusieurs réussites, notamment dans le domaine de la robotique. En effet, il est souvent possible via des capteurs externes ou en simulation d'avoir accès à des informations supplémentaires lors de la phase d'apprentissage. Ces approches peuvent être regroupés en deux catégories [77] :

— *expert distillation* : une politique est apprise à partir de l'état de l'environnement et est utilisée pour guider une politique utilisant les observations. Conceptuellement, cela permet de résoudre d'abord le problème d'apprendre à agir dans un certain état puis d'apprendre à estimer son état dans un cas où les informations sont complexes (bruitées, partiellement observables, déstructurées) [78, 79].
— *asymmetric actor-critic* : la fonction *critic* est apprise en ayant à disposition l'état de l'environnement. Cela permet d'avoir une estimation beaucoup plus fiable du gain espéré sans les difficultés pour estimer son état dans un POMDP [80, 81]. Donc la politique apprise à partir de cette estimation est meilleure.

Pour l'*expert distillation*, une des difficultés notables est que la politique experte doit prendre des décisions compréhensible pour la politique apprenante sachant qu'elle ne connaît que l'historique des observations. Cette difficulté est exprimée par la condition de *deterministic filter* par [77], qui implique qu'il existe une fonction permettant, à partir de l'historique des observations, de déterminer l'état de l'environnement.

2.5.4 Structurer le problème

Lorsque des connaissances sur la structure de l'environnement et/ou le comportement à apprendre sont disponibles, il est alors intéressant de les incorporer pour orienter ou contraindre l'apprentissage.

Machine à états

Le couplage entre l'apprentissage par renforcement et une machine à états (MAE) issue de connaissances a priori permet de diviser l'espace d'état et donc potentiellement d'avoir un traitement algorithmique adapté pour chacun des états [1].

Ainsi, plusieurs approches placent la MAE au sein d'une architecture hiérarchique avec un macro et un microcontrôleur. Cela permet de cibler l'apprentissage sur une partie considérée comme complexe tout en gardant un contrôle et une compréhension sur le modèle final. Par

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

exemple, dans [83], la politique est apprise pour un seul des états de la MAE et garde une définition symbolique pour les fonctions de transition et pour la politique appliquée dans les autres états. Dans [82], la politique apprend directement en utilisant l'état du système mais son espace d'action agit sur une MAE construite à partir d'un contrôleur Proportionnel Dérivé qui transmet des actions bas niveaux aux actionneurs.

Lorsque la fonction peut être représentée par une machine à états, avoir accès à la structure de la fonction de récompense peut faciliter la convergence [84].

Action Masking

L'*Action Masking* rend impossible le choix de certaines actions par la politique en fonction de l'état courant (voir figure 2.13). Les avantages sont de filtrer les actions impossibles ou invalides ce qui empêche l'utilisation d'action invalide et de réduire le besoin d'explorer des actions inutiles [85]. Cette approche est particulièrement efficace dans les environnements où l'espace d'action total est très grand mais dont seulement une petite fraction est valide dans chaque état [27, 86, 87].

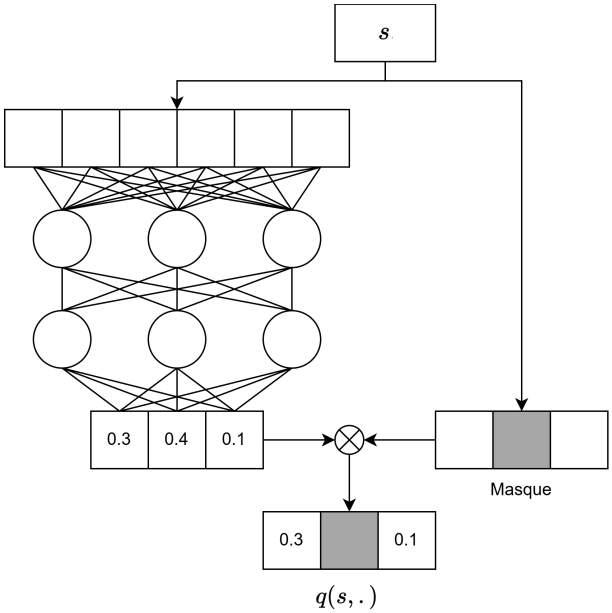


FIGURE 2.13 – Utilisation de l'*Action Masking* pour un réseau estimant la fonction q.

2.5.5 Curriculum Learning

Le *curriculum learning* (CL) consiste en un apprentissage itératif sur des environnements dont la difficulté augmente graduellement. Les connaissances apprises dans les environnements passés doivent être utiles à la résolution des environnements futurs. Les leviers de complexification peuvent être dans la définition du problème (espace d'observation, espace d'action [88], fonction de récompense [89]) ou bien dans la complexité de la tâche à résoudre [90]. Cela peut mener à une convergence plus rapide et à de meilleures performances [91].

L'utilisation du CL implique une méthode de TL entre les différentes versions d'environnement du *curriculum*. Elles peuvent être de différentes natures : exemples, options, politiques, modèles ou fonctions de valeurs. Les métriques d'évaluation du TL sont donc également applicables au CL.

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity
PUBLIC

2.6 Application de l'apprentissage par renforcement

L'apprentissage par renforcement est capable de résoudre des tâches complexes et diverses [26, 78, 39, 25, 40]. Cependant, l'application à des cas réels présente souvent des difficultés liées à la performance, l'évaluation, la fiabilité et/ou l'explicabilité [92, 42].

2.6.1 Benchmarks

Actuellement, la communauté du RL utilise majoritairement 3 ensembles d'environnements comme *benchmarks* afin d'évaluer l'efficacité des algorithmes qu'elle propose [93].

Le plus fréquent est l'*Atari Learning Environment* [43], l'ensemble d'environnement initialement utilisé pour montrer l'efficacité du DQN [26]. C'est un ensemble de 57 jeux Atari, dont l'espace d'observation est généralement l'image, et l'espace d'action, un ensemble discret représentant les commandes à la disposition d'un joueur. Mujoco [94] est un simulateur physique permettant de simuler des robots articulés dont il faut contrôler les actionneurs. Il constitue un ensemble d'environnements riches dont l'espace d'observation est constitué des informations sur les articulations et un espace d'action continue pour les contrôler. CARLA [95] est un simulateur permettant de créer un contexte urbain pour la conduite autonome. Il permet de créer des environnements sur-mesure avec plusieurs types de capteurs, météos, dynamique de trafic et fonction de récompense.

L'apprentissage par renforcement est également appliqué à des environnements plus simples comme Cartpole [96] ou Lunar Lander [97] pour des résultats nécessitant moins de puissance de calcul.

Récemment, le DRL a également été utilisé sur des environnements plus complexes issus de jeux de société (Echecs, Go, Shogi [25]) ou de jeux vidéo (Dota 2 [86], League of Legends [87], Starcraft II [27]).

2.6.2 Librairies

Pour la mise en pratique du DRL, plusieurs librairies existent et répondent à des cas différents. Le tableau 2.1 résume leurs compatibilités avec les frameworks d'apprentissage de réseaux de neurones (PyTorch [98], Tensorflow [99], JAX [100]) et leurs objectifs.

La librairie *Gymnasium* [97] standardise la façon d'interagir avec un environnement. Ainsi, si ce formalisme est adopté, cela permet à n'importe quel environnement de pouvoir utiliser n'importe quelles librairies d'apprentissage.

2.6.3 Création d'un environnement

L'adaptation d'un problème réel à l'apprentissage par renforcement nécessite la définition du MDP à résoudre. Il faut donc notamment définir $\mathcal{S}$, $\mathcal{A}$ et la fonction de récompense.

La fonction de récompense permet de définir le comportement qui sera optimal. Cette approche est très expressive et permet d'apprendre des compétences très complexes [108, 109]. Cependant, il est difficile de définir une fonction de récompense à la fois apprenable et n'induisant pas de biais. En effet, beaucoup de problèmes ont un objectif évaluable qu'à la fin de la trajectoire (victoire dans un jeu, atteinte d'une cible en robotique) ce qui permet de créer une fonction de récompense simple mais éparse. Donner aussi rarement une récompense rend l'apprentissage très complexe car la politique n'a aucun retour sur les actions prises lors de l'exploration. Pour améliorer la capacité à converger, des termes de *reward shaping* sont ajoutés qui

Corporate sensitivity
PUBLIC
© Propriété Naval Group SA, (2026) tous droits réservés.

Librairie	Framework	Objectif	Référence
Stable Baselines 3	PyTorch	Simplicité d'utilisation, Fiabilité, Prototypage	[101]
CleanRL	PyTorch / JAX	Un seul fichier, Lisibilité, Reproductibilité	[102]
TFAgents	TensorFlow	Intégration à l'environnement Google, Production	[103]
Acme	TensorFlow / PyTorch / JAX	Recherche, Calculs distribués	[104]
Gymnax	JAX	Utilisation de GPU/TPU, Rapidité de l'environnement et de la politique	[105]
Ray RlLib	PyTorch / TensorFlow	Calculs distribués, Production, Multi-Agents	[106]
TorchRL	PyTorch	Modulaire, Recherche, Utilisation GPU	[107]

TABLE 2.1 – Comparaison simplifiée des librairies de DRL.

ont pour but de guider la politique vers l'objectif final ou des sous-objectifs similaires [110]. Cependant, plus la fonction de récompense est complexe, plus il est possible de créer un phénomène de *reward hacking* [111], c'est-à-dire l'apprentissage d'un comportement inattendu maximisant la récompense obtenue. Ainsi, la définition de la fonction de récompense (ou *reward engineering*) est une tâche difficile et nécessaire à l'application du RL à un nouveau cas d'application. La qualité de sa définition a un fort impact sur les performances finales et l'alignement du comportement appris par l'agent avec le comportement souhaité [112, 113].

L'espace d'observation et l'espace d'action influe également sur ce qui est compréhensible par l'agent et sa façon d'agir. L'ajout de stochasticité permet de s'assurer que la politique apprend comment réagir à l'environnement et pas une unique séquence d'action optimale [114]. En outre, la notion de *frame skipping* décrit le nombre d'observation (souvent des images) entre chaque action et donc, définit la fréquence d'appel de la politique. Cela a un impact sur les performances de la politique [115] mais également sur la représentativité du modèle. Par exemple, pour la création de OpenAI Five [86], les auteurs défini un ensemble de contraintes afin d'avoir une confrontation juste face à des humains :

- un délai entre la prise de décision de l'action et son application (pour se rapprocher du temps de réaction humain).
- un *frame skipping* permettant de prendre qu'une action toutes les 4 images.
- un bruit sur les actions prises (précision de la souris).

Dans [116], l'auteur a proposé le *framework* W.R.A.P.P.E.R. qui définit un ensemble des considérations nécessaires lorsqu'on souhaite utiliser du RL pour résoudre un problème réel :

- W : *Windows of control*, le périmètre laissé à la politique ;
- R : *Reward and Risk*, les avantages et les risques possibles liés au périmètre ;
- A : *Actions and Actuators*, les leviers d'actions de la politique sur son environnement ;
- P : *Pipeline*, l'interfaçage avec le contexte réel existant ;
- P : *Policies and Priors*, les connaissances, les politiques et les données déjà existantes sur le problème ;
- E : *Evaluation*, la façon dont on peut évaluer la politique ;
- R : *Real-world governance*, l'ensemble des contraintes réelles (métiers, confiance, mainte-

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

nabilité) pour l'intégration du modèle.

2.6.4 Comparaison

La performance d'une politique apprise par un algorithme de DRL est hautement stochastique [117]. C'est la raison pour laquelle, il est nécessaire de réaliser plusieurs entrainements pour une méthode d'apprentissage pour légitimement la comparer à d'autres. En outre, les hyperparamètres d'un algorithme de DRL influent fortement sur ses performances [118], et ces hyperparamètres optimaux peuvent dépendre des choix faits lors de la définition du MDP [119]. Or, le DRL est également connu pour être inefficace dans l'utilisation de ses données [42]. L'ensemble de ces problèmes rend la comparaison entre plusieurs configurations d'apprentissage très couteuses (temps, énergie et puissance de calcul nécessaire).

Plusieurs travaux ont été menés pour réduire la quantité de calcul nécessaire en réduisant le nombre d'environnement utilisé [120, 121] ou pour augmenter la fiabilité des comparaisons entre les algorithmes [117, 122].

Conclusion

L'apprentissage par renforcement permet de résoudre des problèmes de décisions séquentielles. Il se base sur le formalisme des MDP. Son but est de trouver un comportement (politique) maximisant un scalaire (récompense) au cours de ses interactions avec un environnement. Pour cela, les fonctions de valeurs sont utilisées pour estimer les récompenses futures d'une politique à partir d'un état. Les algorithmes de RL utilisent des méthodes itératives alternant entre évaluation de la politique actuelle et amélioration de cette politique (par exemple, le *Q-Learning*). Afin de générer des interactions permettant d'améliorer efficacement la politique, un équilibre entre l'exploration (essai de nouvelles actions) et l'utilisation (utilisation des actions actuellement considérées comme les meilleures) est crucial. Lorsque le nombres d'état devient trop important, il est possible d'utiliser des fonctions d'approximation pour les fonctions de valeurs et de politique. La descente de gradient est une méthode permettant d'utiliser les fonctions d'approximation.

Les réseaux de neurones sont une famille de fonctions paramétrées. Ils permettent d'approximer n'importe quelle fonction en ajustant leurs paramètres grâce à la descente de gradient d'une fonction de coût. L'architecture d'un réseau de neurones est modulaire et permet de s'adapter au type de fonction à approximer, ce qui permet d'accélérer et d'améliorer sa convergence. En outre, ils profitent fortement de l'accélération offerte par les processeurs graphiques.

L'apprentissage par renforcement profond est l'utilisation d'apprentissage par renforcement avec des réseaux de neurones profonds comme fonction d'approximation. L'utilisation de tels réseaux de neurones induit plusieurs problèmes de stabilité et d'efficience calculatoire. Pour cela, plusieurs extensions algorithmiques améliorent notamment l'utilisation des données, la stabilité ou l'utilisation des ressources calculatoires.

L'utilisation de connaissances a priori sur l'environnement peut fortement accélérer et améliorer les résultats obtenus par le DRL. Cela peut prendre la forme d'incorporation de connaissances au cours de l'apprentissage avec des méthodes de *transfer learning* comme l'utilisation de démonstrations. Il est également possible de cibler plus précisément la place laissée à la politique afin de réduire la complexité du problème et/ou de s'assurer que les actions choisies correspondent à ce qui est attendu.

Finalement, la communauté du DRL s'appuient sur des environnements *benchmarks* pour comparer leurs approches. En outre, il existe plusieurs librairies permettant d'appliquer le DRL dont les avantages et objectifs sont diverses. Cependant, l'application de l'apprentissage par

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

renforcement à un nouveau cas d'application implique de nombreux choix pour la définition du MDP. Ces différents choix influent fortement sur la rapidité et la qualité de l'apprentissage de la politique mais également sur son alignement avec ce qui en est attendu. Aussi, les algorithmes de DRL sont hautement stochastiques et dépendent fortement des hyperparamètres ce qui demandent beaucoup de données. A cet effet la comparaison entre différentes configurations d'apprentissage est très coûteuses en temps de calcul.

Pour la suite, le chapitre 3 permet de faire le lien entre la problématique industrielle présentée dans le chapitre 1 et l'ensemble des concepts de l'apprentissage par renforcement profond présentée dans ce chapitre.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

3

Reformulation du sujet de thèse

Ce chapitre décrit la problématique industrielle sous le spectre de l'apprentissage par renforcement profond. Tout d'abord, nous présentons les caractéristiques du problème de décision séquentielle à résoudre par le modèle opérateur. Puis, nous décrivons les choix et les contraintes techniques pour le développement nécessaires à l'application de l'apprentissage par renforcement profond. Finalement, les conditions de représentativité et d'acceptabilité d'un modèle opérateur sont présentées.

Sommaire

3.1 Définition du MDP 54
3.1.1 Espace d'observation 54
3.1.2 Espace d'action 55
3.1.3 Diversité des scénarios 55
3.1.4 Connaissances métiers sur l'environnement 56
3.1.5 Performances de référence 56
3.2 Développement 57
3.2.1 Environnement de développement 58
3.2.2 Ressources allouables 58
3.3 Application du modèle 58
3.3.1 Performances 59
3.3.2 Comportemental 60
3.3.3 Acceptabilité du modèle en production 60

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

3.1 Définition du MDP

La figure 3.1 présente la particularisation du formalisme mathématique introduit dans la section 2.2.1 au rôle du modèle opérateur au sein du simulateur numérique (voir figure 1.9). Le modèle opérateur doit résoudre un problème de décision séquentielle partiellement observable. L'observation est un agrégat formé par les données communiquées par le logiciel tactique à travers la liaison filaire avec la torpille, les modèles des capteurs du lanceur et les conditions initiales (notées o_t). Les actions représentent des arbitrages sur les décisions prises par le logiciel tactique (notées a_t). Le logiciel tactique résout lui-même un problème de décision séquentielle partiellement observable. Il s'appuie sur les données de ses capteurs pour élaborer la consigne de guidage de la torpille. Pour le modèle opérateur, l'état réel de l'environnement (noté s_t) est donc l'ensemble des états des modèles physiques et du logiciel tactique.

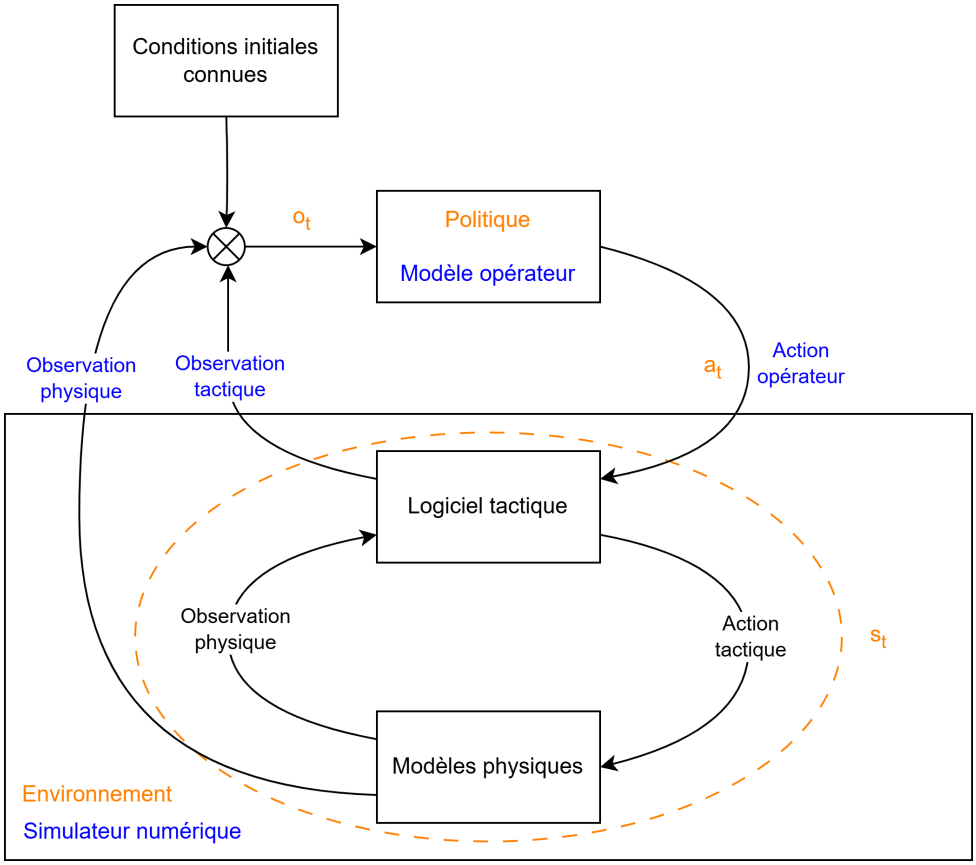


FIGURE 3.1 – Description du problème de décision séquentielle à résoudre par le modèle opérateur dans le cadre de l'apprentissage par renforcement où les termes en bleu représentent la particularisation à la problématique industrielle des termes d'apprentissage en orange.

3.1.1 Espace d'observation

L'opérateur a accès à un agrégat d'information issues de trois sources : les conditions initiales (type de cible, etc.), les observations physiques issues des modèles physiques (c-à-d les informations issues des capteurs du lanceur) et les observations tactiques issues du logiciel tactique (c-à-d les informations communiquées par la liaison filaire).

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

Les conditions initiales sont statiques au cours d'un tir. Il est donc possible de les intégrer à l'observation ou bien de créer des modèles différents pour chacune des conditions initiales possibles.

Le problème de décision séquentielle du logiciel tactique est partiellement observable car les informations issues du sonar sont imparfaites (bruitées), incomplètes (on ne peut estimer l'état complet d'un élément avec une seule utilisation du sonar) et pas toujours disponible (le sonar n'est pas toujours activé ni toujours dirigé sur la cible). Le logiciel tactique intègre donc une notion de mémoire et d'agrégation des informations au cours du temps pour faire ses choix tactiques (par exemple, la création de piste). Comme l'opérateur a accès à des informations du logiciel tactique, il profite donc d'informations agrégeant déjà l'historique des données des capteurs (piste) ce qui réduit l'aspect non markovien.

La quantité de données observées à chaque instant est variable au cours du temps (existence d'une piste, détection ou non d'alarmes). Cela implique donc l'utilisation de réseau de neurones capable de traiter des données de taille variable ou de pré-traiter les données afin qu'il y ait un vecteur de taille fixe (agrégation de données et traitement de données manquantes).

Enfin, puisque le simulateur numérique est développé en boite blanche (produit interne Naval Group), il est possible d'avoir accès à toutes les informations du théâtre opérationnel, ce qui rend l'utilisation d'information privilégiée envisageable.

3.1.2 Espace d'action

Les actions considérées sont des actions discrètes d'arbitrage des décisions prises par le logiciel tactique de la torpille. Le nombre d'actions est d'environ 10^2 mais, en pratique, pour une situation opérationnelle donnée, seule une fraction de ces actions sont pertinentes (notamment à cause de l'existence d'une piste ou non), il est donc intéressant de considérer l'*Action Masking*.

L'opérateur a un rôle de supervision du logiciel tactique, ce qui induit qu'il le laisse la plupart du temps agir sans interférer. Par conséquent, l'action "Rien faire" est particulière car elle doit être choisie la majorité du temps. Les autres actions sont haut niveau avec des effets sur le long terme et des conséquences importantes. Une seule mauvaise action peut conduire à l'échec un tir qui, sans intervention de l'opérateur, aurait été un succès.

En outre, les actions à la disposition de l'opérateur ne lui permettent pas d'avoir un contrôle total du système, cela implique que tous les échecs ne sont pas forcément évitables (voir section 3.1.5).

Contrairement à des actions bas niveau (comme un contrôle directionnel sur ALE [43]), certaines actions peuvent être étudiées séparément, permettant de créer des versions d'environnements plus simples.

3.1.3 Diversité des scénarios

Il existe plusieurs versions de l'environnement en fonction du type de cible, de l'existence d'une réaction et des actions considérées. Il semble donc a priori intéressant d'utiliser l'approche du *Curriculum Learning*. Cependant, l'intérêt d'un transfert de connaissances entre ces différentes versions n'est pas assuré car elles sont très différentes les unes des autres.

Les conditions initiales peuvent donner des informations sur le scénario auquel fait face l'opérateur. Dans notre cas, le type de cible est connu (SM pour sous-marin ou BS pour bâtiment de surface) mais pas l'existence ni le type de réactions de la cible (R avec réactions ou $\bar{\text{R}}$ sans réaction). En effet, il est impossible de savoir dans un cas opérationnel si la cible va réagir en

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

avance. Il est donc possible de créer des modèles opérateurs différents pour les ensembles de tirs SM et BS, mais ils doivent être capable de faire face à des cibles avec ou sans réactions.

3.1.4 Connaissances métiers sur l'environnement

Il existe plusieurs formes de connaissances a priori sur le problème.

Équipes de développement du simulateur et du logiciel tactique

Le modèle opérateur est conçu directement dans l'environnement où développent les ingénieurs en charge du simulateur et du logiciel tactique. Il existe donc de fortes connaissances sur l'environnement d'apprentissage permettant d'orienter a priori le design du problème d'apprentissage. En outre, cela permet d'avoir des avis critiques rapides sur les modèles développés afin de mieux comprendre ce qui a été appris par l'algorithme de RL (utilisation d'imperfection du simulateur, *reward hacking*).

Machine à états

Il existe deux MAE permettant de structurer l'espace d'états de l'environnement :
- La MAE du logiciel tactique (voir section 1.2.3) donne la phase courante et donc l'objectif qui est visé par le logiciel tactique ;
- La MAE du modèle opérateur (voir section 1.4) donne un état censé estimer la confiance de l'opérateur dans les actions du logiciel tactique.

Elles peuvent donc être utilisées pour créer des métriques ou bien améliorer les algorithmes.

Données de tirs

Il existe des données de tirs réels effectués avec des opérateurs réels. Cependant, il est très couteux de générer de tels tirs ce qui induit qu'il y a une faible quantité de données. Par conséquent, ces données ne couvrent pas tout le spectre des situations rencontrées en simulation. Cela rend donc complexe l'utilisation de ces données comme démonstration dans un algorithme de DRL.

3.1.5 Performances de référence

Par construction de l'environnement, il existe une politique à la fois simple et efficace consistant à toujours choisir l'action "Rien Faire". En effet, cette politique revient à évaluer les performances du SATL comme actuellement, c'est-à-dire uniquement celle du logiciel tactique. Ces performances obtenues avec un opérateur inactif seront considérées comme la référence. Tous les modèles opérateurs produits seront donc évalués par rapport à cette référence.

On peut catégoriser chacun des tirs du plan d'expérience sans opérateur (voir figure 3.2) en :
- **Non récupérable** : Tirs en échec, où il n'existe aucune action permettant à un opérateur de modifier le tir en succès.
- **Récupérable** : Tirs en échec, où une des actions disponibles aurait pu permettre à l'opérateur d'agir à un moment du tir de rendre le tir en succès.
- **Décidable** : Tirs en échec, tir Récupérable dont les informations à la disposition de l'opérateur lui permettent de comprendre quand cette action doit être faite.
- **Ratable** : Tirs en succès, où il existe une action opérateur à un moment du tir modifiant le tir en un échec.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

— **Non Ratable** : Tirs en succès, où il n'existe aucune action opérateur pour modifier le tir en un échec.

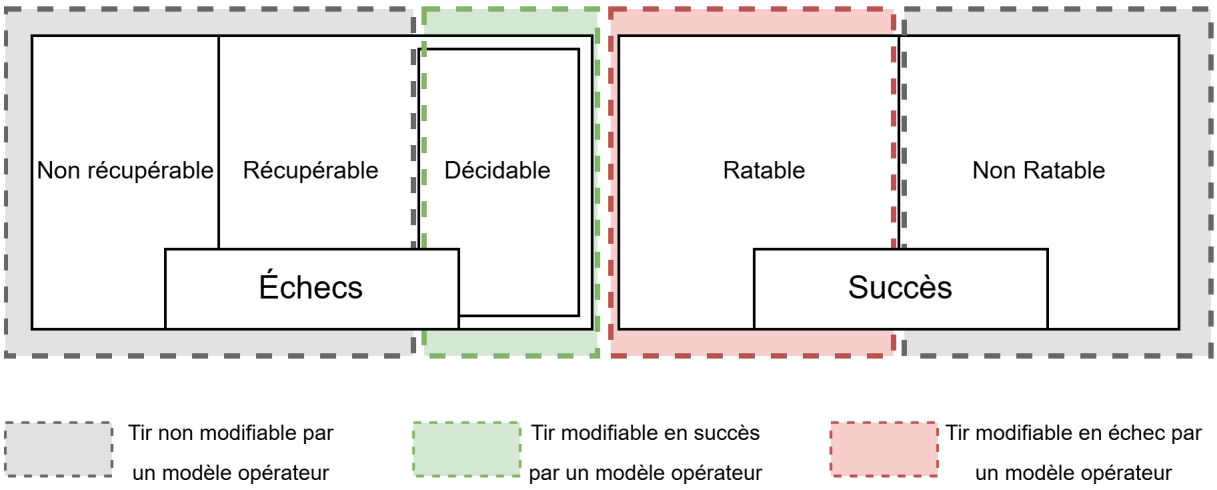


FIGURE 3.2 – Catégorisation des tirs en fonction de leur faisabilité pour le logiciel tactique et pour l'opérateur.

Les tirs "Décidable" constituent le réservoir de gain de performance pour le modèle opérateur. Les tirs "Récupérable" constituent des gains de performance qui peuvent être réalisés lors de l'utilisation d'information privilégiée (position réelle de la cible) mais où les observations à la disposition de l'opérateur ne sont pas suffisantes (voir condition de *deterministic filter* dans la section 2.5.3). Au contraire, les tirs "Ratable" constituent le réservoir de perte de performance pour le modèle opérateur. Ainsi la performance relative finale du modèle correspond à la différence entre les tirs "Ratables" nouvellement en échec et les tirs "Décidable" nouvellement en succès (ou "Récupérable" pour un modèle utilisant des informations privilégiées).

L'utilisation d'une performance relative au lieu d'une performance absolue modifie la complexité des scénarios considérés pour le modèle opérateur. En effet, un scénario opérationnellement simple (comme $\overline{\text{BSR}}$ ou $\overline{\text{SMR}}$) peut être compliqué du point de vue du modèle opérateur car les performances du logiciel tactique seules sont déjà très hautes. Par conséquent, sur ce type de scenarii, le réservoir de gain de performance est très faible tandis que le réservoir de perte de performance est très grand. Au contraire, sur les scénarios considérés comme compliqués (c-à-d avec présence d'une réaction sophistiquée de la cible) où le logiciel tactique a des difficultés, laisse beaucoup plus de place à une amélioration par le modèle opérateur.

3.2 Développement

Le développement lié à l'adaptation du simulateur industriel pour être capable de faire de l'apprentissage par renforcement est primordial. En effet, il faut l'interfacer avec les outils déjà existants en minimisant les surcoûts de temps de calcul et de développement logiciel. En outre, une utilisation optimisée des ressources calculatoires est cruciale pour maximiser le nombre de transitions générées et utilisées. Finalement, les outils développés doivent être hautement modulaire afin de faciliter les itérations sur l'ensemble des éléments du problème d'apprentissage (c-à-d espace d'observation, espace d'action, fonction de récompense) et sur les algorithmes d'apprentissage par renforcement (algorithmes, hyperparamètres).

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity
PUBLIC

3.2.1 Environnement de développement

Le simulateur industriel est :
- mono-processus ;
- programmé en langage C ;
- utilise une sur-couche de Java.

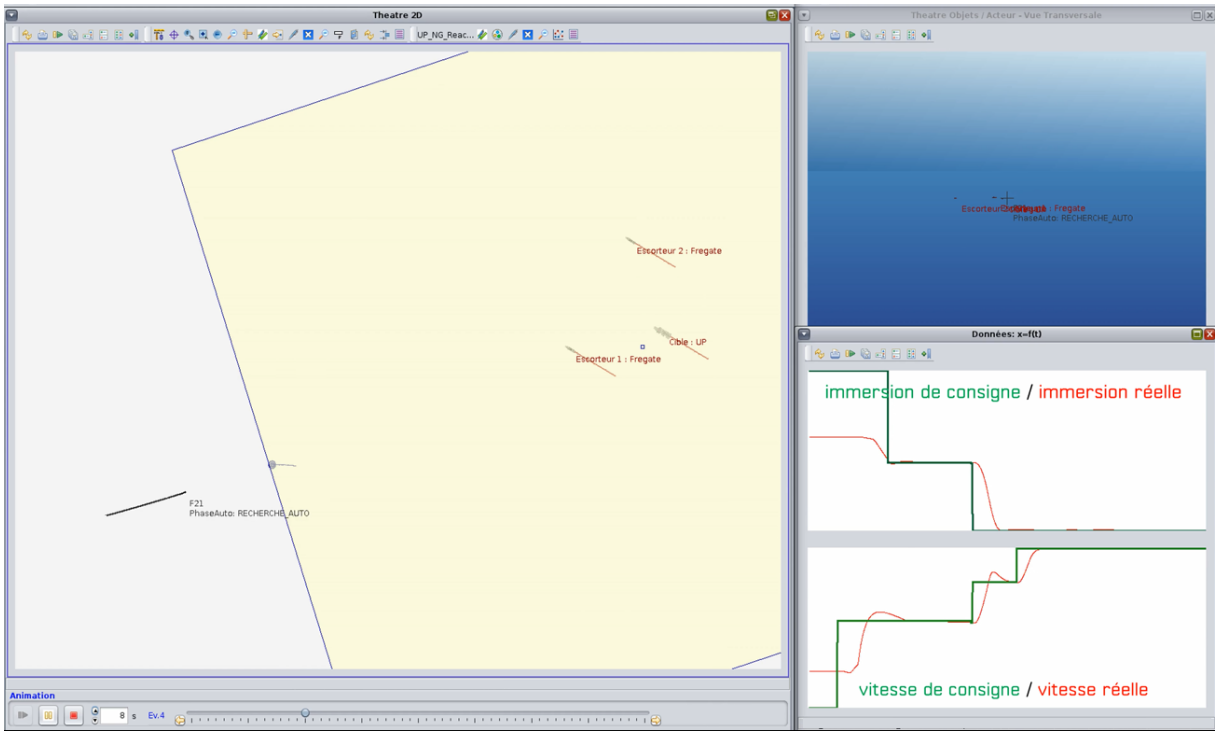


FIGURE 3.3 – Outil de visualisation permettant d'analyser des simulations de tirs.

Des outils de visualisation et d'analyse de simulation disponibles utilisent des données dans un format propre à l'entreprise. La figure 3.3 illustre un outil de visualisation pour analyser la simulation d'un tir. Il existe également de nombreux outils d'analyse statistiques en Matlab et Python se basant sur des données au format tabulaire.

3.2.2 Ressources allouables

Pour l'entraînement, les ressources disponibles sont l'ensemble des fermes de calcul ainsi que le poste de travail avec un GPU (voir section 1.6.2).

Le modèle final doit être utilisable en production indiquant qu'il doit être utilisable pour faire des simulations sur les fermes de calcul sans induire de ralentissement ou presque. Le temps de l'inférence doit donc être faible devant le temps du simulateur en utilisant seulement un CPU. La RAM allouable par simulation est de 1Go (256 coeurs pour 256 Go), le modèle opérateur ainsi que la simulation ne doivent donc pas dépasser cette limite.

3.3 Application du modèle

L'applicabilité du modèle est conditionnée par les performances et la représentativité comportementale du modèle, ainsi que sa facilité de maintien en production.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

3.3.1 Performances

La métrique de performance principale est la probabilité de réussite des tirs. Pour que le modèle améliore cette probabilité rapport à la référence, il faut que la maximisation de la fonction de récompense conduise à une politique dont la performance est haute. Pour cela, le *reward engineering* en collaboration avec les concepteurs du logiciel tactique est crucial.

Les performances obtenues sur chacun des scénarios (BSR, BS$\bar{\text{R}}$, SMR, SM$\bar{\text{R}}$) seront toujours comparées relativement à la performance de référence. Cependant, ces valeurs de performances sont couvertes par le secret de la protection de la défense national. Dans ces travaux, afin de communiquer des performances relatives à une référence sur laquelle on ne peut donner d'informations précises, aucune valeur numérique de performance ne sera communiquée. En effet, il est possible de déduire des informations sur une performance relative chiffrée avec un intervalle de confiance (voir Annexe A).

Les performances des différents modèles opérateur seront donc communiquées grâce à des graphiques blanchis, comme le montre la figure 3.4. Trois zones sont définies basées sur des connaissances métiers pour l'évaluation des performances du modèle opérateur par le simulateur numérique :

— Zone non-acceptable. Tout modèle dont la performance se trouve dans cette zone sera jugé comme inacceptable. En effet, il n'est pas acceptable que l'intégration d'un modèle opérateur puissent conduire à des pertes de performances. Cette zone se trouve en dessous de l'intervalle de confiance à 95% de la performance de référence.

— Zone acceptable. Tout modèle dont la performance se trouve dans cette zone sera jugée comme acceptable d'un point de vue performance. La limite basse correspond à la limite basse de l'intervalle de confiance de la performance de référence. La limite haute a été délimitée par une valeur métier pour chacun des scénarios considérés (voir table 3.1 pour la valeur de chacun des scénarios).

— Zone de gain notable. Tout modèle dont la performance se trouve au-delà de la limite haute de la zone acceptable est considéré comme apportant un gain notable. Cela traduit un gain à partir duquel les concepteurs du logiciel tactique (pour ré-utiliser les termes) estiment avoir réalisé une amélioration considérable du logiciel.

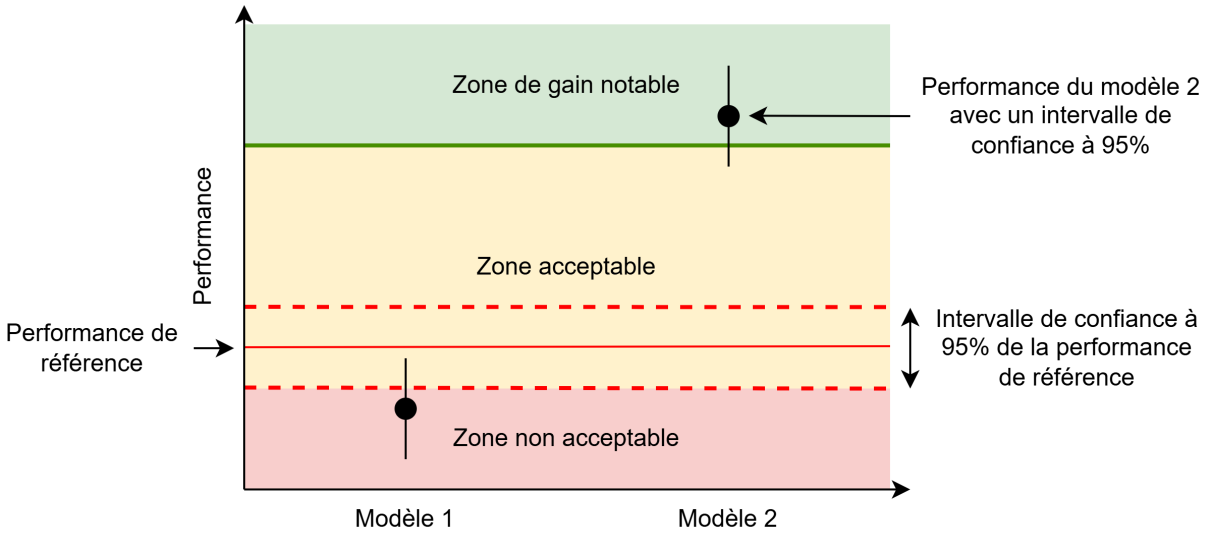


FIGURE 3.4 – Exemple de graphique blanchi présentant les performances de plusieurs modèles.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

TABLE 3.1 – Valeur limite inférieure de la zone de gain notable exprimée en nombre d'intervalle de confiance de la performance de référence en fonction du scénario.

Scénario utilisé pour la référence	$BS\bar{R}$	BSR	$SM\bar{R}$	SMR
Valeur de gain notable en IC à 95% de la référence	2.35	4.54	3.69	3.65

Des métriques de performances secondaires peuvent également être utilisées comme le temps moyen par tir mais celles-ci ne conditionneront pas l'acceptabilité.

3.3.2 Comportemental

L'aspect comportemental est la manière d'agir du modèle opérateur au cours du tir. Les leviers pour modifier une politique peuvent être des contraintes dures ou des contraintes souples.

Les contraintes dures agissent directement sur la fenêtre de contrôle laissée à la politique. De façon similaire à [86], il faut considérer la fréquence d'appel, le délai entre la commande, la prise en compte de l'action et des potentielles erreurs. Il est également possible d'utiliser l'*action masking* pour des actions qui sont en pratique disponibles, mais ne correspondant pas à la doctrine d'emploi de l'arme (c-à-d la façon de mettre en œuvre la torpille en conditions réelles).

Les contraintes souples sont des guides vers le comportement attendu lors de l'apprentissage mais qui n'ont aucune assurance algorithmique d'être strictement suivies. Par exemple, l'ajout de termes négatifs dans la fonction de récompense permet de prendre en compte des contraintes souples.

Pour l'évaluation comportementale, il n'existe pas de référence validée pour le simulateur numérique à laquelle se comparer. Une analyse d'un ensemble de tir réel a donc permis de déterminer le nombre moyen d'actions par minute, le temps minimum entre deux actions différentes et une estimation du temps de réaction. Le temps de réaction et le délai entre deux actions seront implémentés par des contraintes dures ne permettant donc pas d'être brisées. Pour le nombre moyen d'action par minute, nous utiliserons des contraintes souples avec comme référence d'acceptabilité, la moyenne basée sur les tirs analysés. A l'instar des niveaux de performances de la référence, les caractéristiques issues de l'analyse de tirs réels ne peuvent être communiquées. La fréquence d'action sera donc décrit à l'aide d'un graphique blanchi (voir la figure 3.5).

3.3.3 Acceptabilité du modèle en production

Pour utiliser le modèle opérateur en simulation, il faut satisfaire les contraintes liées aux ressources disponibles (voir section 3.2.2). Cependant, il faut également s'assurer que le coût de maintien du modèle dans le temps ne soit pas trop important. En effet, le modèle opérateur n'est qu'un modèle parmi tous les modèles nécessaires au fonctionnement du simulateur numériques. La représentativité du modèle constitue le coeur des travaux de thèse mais sa maintenabilité est également cruciale pour une réelle intégration dans un contexte industriel.

Le logiciel tactique et le simulateur numérique sont des logiciels en constante évolution, ce qui implique des modifications régulières de l'environnement du modèle opérateur. Les travaux de cette thèse ont été mené sur une version figée du logiciel tactique ainsi que du simulateur (sauf

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

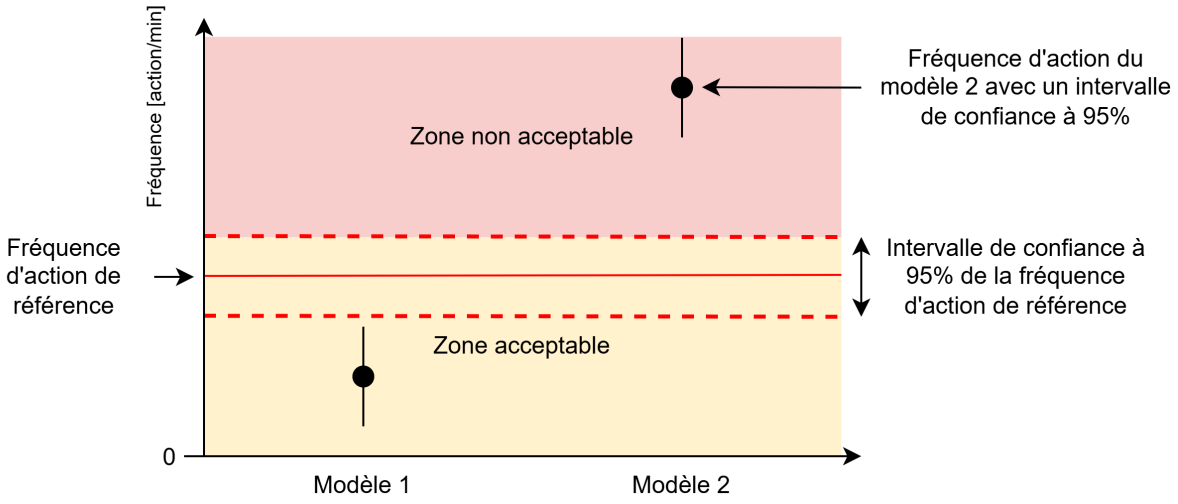


FIGURE 3.5 – Exemple de graphique blanchi présentant la fréquence d'action de plusieurs modèles.

mention contraire). Il se pose donc de la question de la nécessité de modifier le modèle opérateur suite à de nouvelles versions. Plusieurs cas peuvent se présenter :

- Aucune modification nécessaire. Les modifications sont mineures ou n'affecte pas la façon dont le modèle doit agir, permettant de garder le modèle tel quel.
- Entraînement d'un modèle en utilisant le modèle passé. L'ancien modèle n'est plus jugé assez représentatif sur la nouvelle version de l'environnement et nécessite un ajustement. Si les modifications ne changent pas trop le fonctionnement de l'environnement, un réentrainement du modèle ou une approche de *transfer learning* peuvent être envisagés afin de capitaliser sur le modèle précédent.
- Entraînement d'un nouveau modèle. La modification de l'environnement est très importante et rend l'ancien modèle totalement incohérent. Il devient donc nécessaire de reprendre le processus d'apprentissage du modèle élaboré dans ces travaux.
- Modification du processus d'apprentissage. La modification de l'environnement est très importante et rend l'ancien modèle totalement incohérent ainsi que le processus d'apprentissage inapte à produire un modèle cohérent. Il devient donc nécessaire de modifier le processus d'apprentissage en modifiant la définition du problème d'apprentissage (espace d'observation, fonction de récompense, espace d'action, contraintes), des hyperparamètres ou bien les algorithmes.

Le temps et les connaissances en DRL nécessaires pour ces différents cas sont croissants. Ce coût d'adaptation est influencé par :

- le nombre de pas de temps nécessaires pour un apprentissage.
- le nombre de modèles à apprendre. En effet, plusieurs approches présentées impliquent des apprentissages multiples : l'utilisation d'information privilégiée peut impliquer d'apprendre deux modèles successifs (un à partir de l'information privilégiée et le second à partir de l'observation), le *curriculum learning* entre plusieurs environnements et la particularisation du modèle opérateur pour certaines conditions initiales.
- la stabilité de l'apprentissage. En effet, si l'efficacité des politiques apprises est très aléatoire, cela conduit potentiellement à devoir faire plusieurs apprentissages afin d'en obtenir

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity
PUBLIC

un convenable.

— la facilité d'utilisation des outils développés. La mise en place d'apprentissage par renforcement implique d'une version du simulateur adapté à l'apprentissage. Cette version doit être utilisable et maintenable par des ingénieurs qui ne sont potentiellement pas spécialistes du DRL.

Pour obtenir un modèle opérateur maintenable, il faut donc à la fois que les modifications nécessaires suite à un changement de version soient les moins impactantes possible et que ces modifications nécessitent le moins de connaissances spécifiques en DRL possible.

Conclusion

L'élaboration d'un modèle opérateur est un problème de décision séquentielle partiellement observable. L'espace d'observation de ce problème est une combinaison des observations du logiciel tactique, qui doit lui-même résoudre un problème de décision séquentielle partiellement observable, des modèles simulant les capteurs du lanceur et des connaissances a priori sur l'environnement. En outre, il est possible d'avoir accès à des informations privilégiées grâce au simulateur numérique. L'espace d'action est constitué d'une centaine d'actions haut niveau agissant sur le logiciel tactique. Il est attendu que la politique apprise choisisse majoritairement l'action "Rien faire", créant naturellement une asymétrie par rapport aux autres actions. Il existe un ensemble de situations différentes de complexités différentes grâce aux scénarios mis en oeuvre par le simulateur numérique permettant potentiellement d'utiliser le *Curriculum Learning*. De plus, on dispose d'un ensemble de connaissances sur le problème grâce aux concepteurs du simulateur et du logiciel tactique, des machines à états et des données de tirs réels. Finalement, tout modèle opérateur sera comparé à une référence qui est un modèle opérateur ne faisant que l'action "Rien faire".

La création et l'utilisation du modèle opérateur doivent se faire dans la limite des ressources allouables. En effet, le contexte de protection du secret de la Défense nationale ne permet pas d'avoir un accès à des ressources extérieures au système d'information où le logiciel tactique est mis au point.

L'applicabilité du modèle est soumise à plusieurs contraintes : les performances, la représentativité du modèle opérateur et la capacité de maintenir ce modèle dans un environnement industriel. Le modèle doit à la fois être plus performant par rapport à la référence tout en étant vraisemblable d'un comportement d'un opérateur réel. L'évaluation de ces deux aspects est à la fois difficile et sensible. Finalement, la capacité de maintenir cette représentativité du modèle opérateur par les ingénieurs en charge du simulateur est cruciale pour une réelle adoption.

Suite à la compréhension et la définition du placement et des objectifs du modèle opérateur au sein du simulateur industriel, le chapitre suivant présente la partie logicielle développée afin de s'interfacer avec le simulateur industriel pour pouvoir appliquer du DRL.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

4

Architecture logicielle de l'environnement d'apprentissage

Ce chapitre décrit l'architecture logicielle développée pour adapter le simulateur industriel en un environnement d'apprentissage modulaire. Il présente les modifications apportées au simulateur industriel afin de pouvoir l'initialiser et communiquer avec, depuis un processus extérieur. En outre, des blocs fonctionnels supplémentaires sont nécessaires pour l'adaptation du simulateur en un MDP. Finalement, une analyse de performance est faite.

Sommaire

4.1 Interfaçage du *wrapper* avec le simulateur 64
4.1.1 Initialisation du simulateur 64
4.1.2 Communication au cours de la simulation 65
4.2 Création de l'environnement d'apprentissage 65
4.2.1 Communication avec le simulateur 65
4.2.2 Fonction de récompense 66
4.2.3 Transformée d'informations 66
4.2.4 Politique 66
4.2.5 Transformée d'action 67
4.2.6 *Frame Skipping* 67
4.3 Analyse du *wrapper* 67
4.3.1 Librairies choisies 67
4.3.2 Utilisation du *wrapper* 68
4.3.3 Efficacité 68

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Pour transformer le simulateur industriel en un environnement d'apprentissage, il est nécessaire de développer une surcouche logicielle permettant de communiquer avec le simulateur depuis un processus Python. Cette surcouche logicielle permettant l'utilisation du simulateur industriel comme un environnement d'apprentissage depuis un processus Python est appelée *wrapper*.

4.1 Interfaçage du *wrapper* avec le simulateur

L'utilisation du simulateur industriel en tant qu'environnement d'apprentissage nécessite la capacité de créer des simulations à partir de conditions initiales précises, la possibilité de communiquer au cours de la simulation des informations sur son état et des actions à prendre.

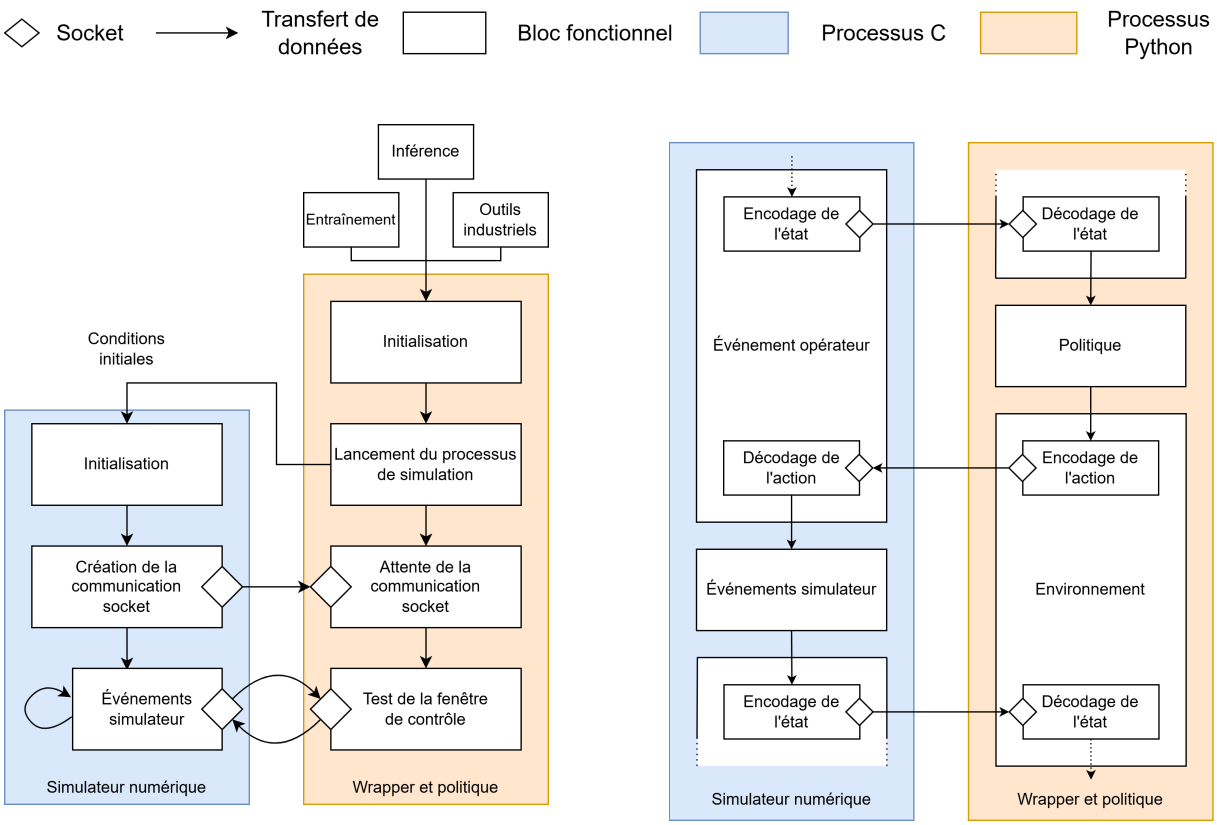


FIGURE 4.1 – Initialisation du processus du *wrapper*, du processus de simulation numérique et de la communication socket entre les deux.

FIGURE 4.2 – Communication entre le processus du *wrapper* et le processus de simulation numérique au cours d'un tir.

4.1.1 Initialisation du simulateur

Afin d'initialiser le *wrapper*, les simulations doivent être lancées depuis le processus Python. Cela est nécessaire lors de l'apprentissage d'un modèle ou lors de l'inférence d'un modèle. La gestion des paramètres initiaux doit donc être effectuée à partir du processus Python pour contrôler les expériences rencontrées par l'agent. La figure 4.1 illustre la méthode développée pour l'initialisation du *wrapper*.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

Afin de générer une trajectoire, le processus Python du *wrapper* initialise à la fois un objet "Politique" et un objet "Environnement". Lors de l'initialisation de l'Environnement, un ensemble de conditions initiales lui est communiqué, ce qui lui permet de lancer une version modifiée du simulateur. Les modifications apportées permettent une maîtrise des conditions initiales directement en ligne de commande (contournement des interfaces des outils industriels) et l'attente de la création d'une communication par *socket*.

Des premières communications sont faites avec l'environnement jusqu'à un état de l'environnement qui valide la fenêtre de contrôle. La fenêtre de contrôle est un ensemble de conditions définissant le périmètre d'action du modèle (démarrage de certains logiciels, proximité de la cible, ...).

4.1.2 Communication au cours de la simulation

Une fois la communication établie et la fenêtre de contrôle valide, le processus du simulateur numérique et celui du *wrapper* sont intriqués grâce à des communications inter-processus par la communication *socket* créée lors de l'initialisation. La figure 4.2 illustre le flux des données entre ces processus.

Au sein de la boucle événementielle du simulateur, le nouvel événement "Évenement opérateur" est ajouté. C'est un événement bloquant utilisant la communication *socket* crée lors de l'initialisation. Il envoie un ensemble d'informations sur l'état interne du simulateur puis attend la réception des informations sur l'action à prendre. Cet état interne est l'ensemble des informations nécessaires aux différents éléments du MDP (espace d'observation, espace d'action, fonction de récompense).

Une fois l'action reçue, l'événement ajoute le futur "Évenement opérateur". Cela induit une durée fixe (en seconde simulée) entre chaque appel, notée t_{op}, donnant la fréquence maximale d'interaction entre une politique et l'environnement. Le choix de cette valeur minimale a été choisie sur la fréquence réelle de rafraîchissement des données communiquées à l'opérateur. Il est important de noter que cette fréquence minimale peut être différente de la fréquence d'appel de la politique avec l'utilisation de *Frame Skipping*.

Du point de vue du processus Python, les communications inter-processus et les événements simulateur entre deux événements opérateurs sont ce qui constitue un pas de temps de l'objet "Environnement".

4.2 Création de l'environnement d'apprentissage

Une fois que la communication inter-processus est établie, il est nécessaire de traiter les données pour créer les éléments nécessaires à un environnement d'apprentissage (action, observation, récompense). À la réception des informations, ces données sont mises en forme puis modifiées en place (afin de minimiser la RAM utilisée) par chacun des blocs fonctionnels. La figure 4.3 présente la chaîne de traitement formée par ces blocs au sein du *wrapper*. Les paragraphes ci-après décrivent le fonctionnement de chacun des blocs.

4.2.1 Communication avec le simulateur

Le bloc "Communication avec le simulateur" est composé de deux sous blocs : le bloc "Décodage de l'information" et le bloc "Encodage de l'action". Pour le processus Python, il y a une attente bloquante pour le "Décodage de l'information" afin que le simulateur exécute le prochain "Événement opérateur".

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

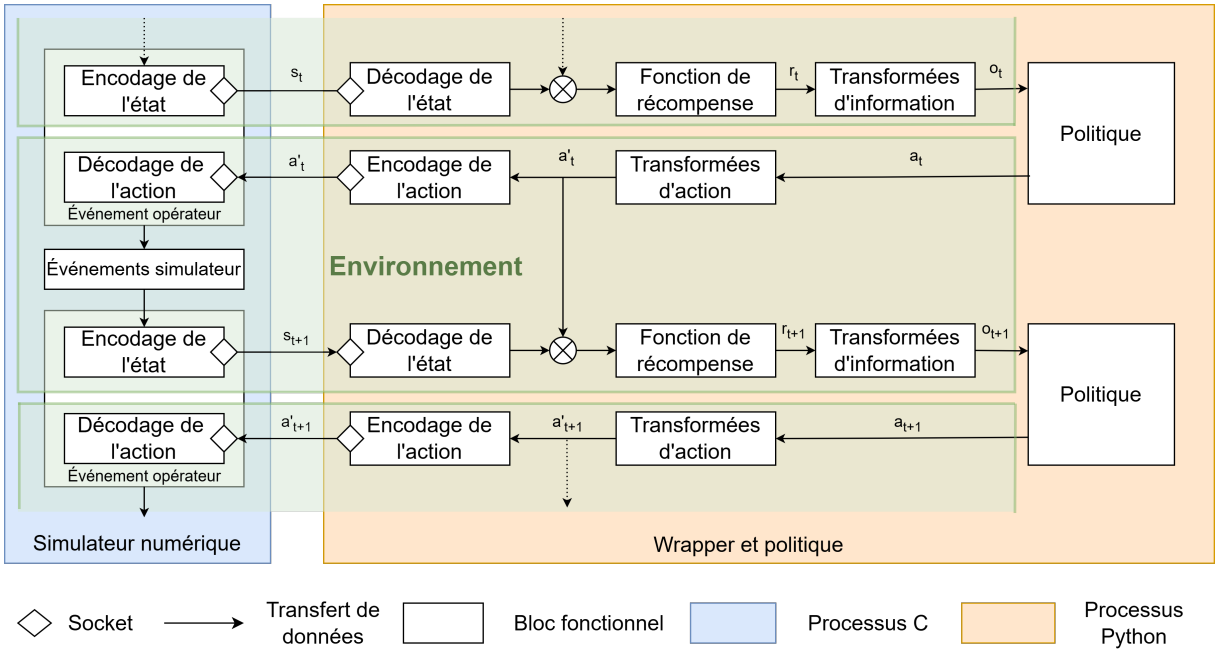


FIGURE 4.3 – Ensemble des blocs fonctionnels composant le *wrapper* afin d'adapter le simulateur numérique en un environnement d'apprentissage par renforcement.

Le sous-bloc "Décodage de l'information" reçoit les informations par la communication *socket*. Un dictionnaire (plus précisément un *tensordict* [107]) est créé avec l'ensemble des informations communiquées sur l'état interne du simulateur (piste, position DO, position cible, alarmes, etc...).

Le sous-bloc "Encodage de l'état" sérialise et communique l'action qui doit être exécutée par l'"Événement opérateur". Cet envoi commence donc le début de l'attente bloquante pour *wrapper* et la réception de cette action par le simulateur indique la fin de l'attente bloquante de l'"Événement opérateur" (voir section 4.1.2).

4.2.2 Fonction de récompense

Le bloc "Fonction de récompense" calcule la valeur de la récompense (r_t) de la transition du simulateur. Nous avons choisi de la définir comme une combinaison linéaire de plusieurs sous-fonctions.

4.2.3 Transformée d'informations

Le bloc "Transformée d'informations" permet de mettre en forme les informations communiquées par le simulateur pour la politique. Cela permet la création d'informations élaborées à partir de l'état du simulateur. Par exemple, l'ajout d'informations données par d'autres politiques et/ou la mise en forme des vecteurs d'observation utilisés.

Ce bloc est composé d'une liste de fonctions appliquées successivement au dictionnaire contenant les informations. Son fonctionnement est similaire à la classe *Transform* de *TorchRL* [107].

4.2.4 Politique

Le bloc "Politique" permet de choisir une action à partir d'une observation. C'est le seul élément du processus Python représenté sur la figure 4.3 qui ne fait pas partie du *wrapper*. Dans

Corporate sensitivity PUBLIC
© Propriété Naval Group SA, (2026) tous droits réservés.

notre cas, une politique peut soit être une fonction utilisant de la programmation symbolique (règles métiers, MAE), soit être construite à partir d'un réseau de neurones.

4.2.5 Transformée d'action

Le bloc "Transformée d'action" permet de modifier l'action choisie par la politique. Cela permet d'ajouter des contraintes dures ou du bruit sur les actions notamment pour les considérations de représentativité comportementale.

Tout comme la "Transformée d'informations", ce bloc est composé d'une liste de fonctions appliquées successivement au dictionnaire contenant les informations. Son fonctionnement est similaire à la classe *Transform* de *TorchRL* [107].

4.2.6 *Frame Skipping*

Le *Frame Skipping* est une boucle au sein de l'environnement qui renvoie une action par défaut et additionne les récompenses obtenues. Cela crée donc un paramètre, noté n_{skip}, qui est le nombre de pas de temps de l'environnement effectué avec l'action par défaut entre chaque appel de la politique. Dans notre cas, l'action par défaut choisie n'est pas la répétition de la dernière action (comme par exemple sur l'environnement *ALE* [114]), mais l'action "Rien faire". En effet, comme les actions sont haut niveau avec une fréquence d'appel assez faible, une répétition de la même action a des effets néfastes. La figure 4.4 montre un exemple pour une valeur de $n_{skip} = 1$.

La fréquence d'appel de la politique est un multiple de t_{op} car la répétition est faite au sein du processus Python.

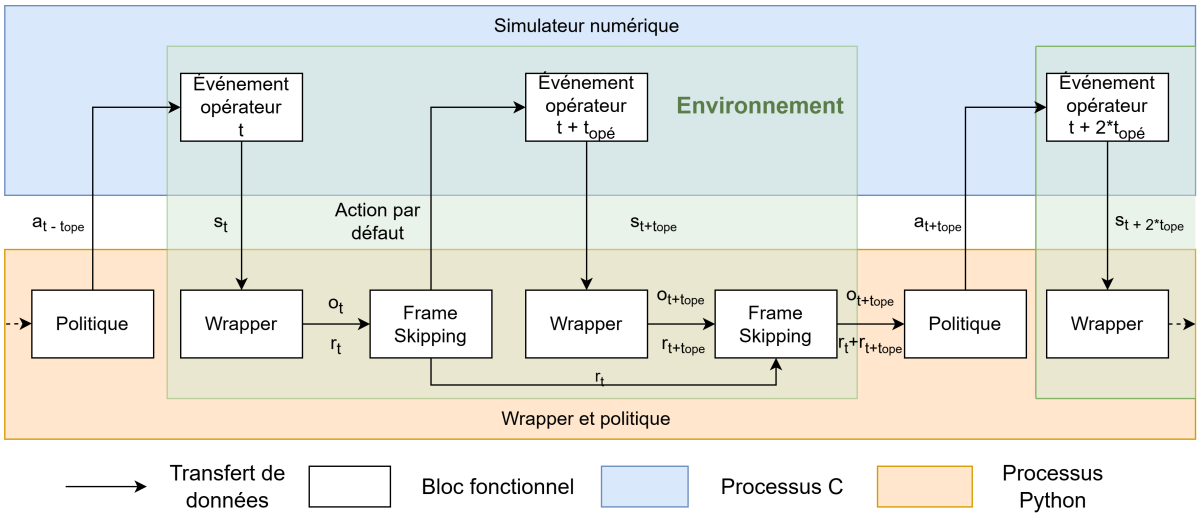


FIGURE 4.4 – Fonctionnement du *Frame Skipping* pour $n_{skip} = 1$.

4.3 Analyse du *wrapper*

4.3.1 Librairies choisies

Pour la création de l'environnement d'apprentissage, l'interfaçage avec simulateur industriel a été fait selon le formalisme de Gymnasium [97] afin d'être facilement adaptable à n'importe quelle librairie de DRL.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Les algorithmes de DRL ont été développés avec TorchRL [107]. La modularité de la librairie se transfère dans les outils développés pour l'environnement d'apprentissage. En effet, TorchRL utilise les *tensordict*, une structure de données sous forme de dictionnaire flexible, efficiente et permettant d'utiliser un GPU. En outre, il est facile de créer, de modifier et/ou de combiner différentes modifications algorithmiques.

Pour l'utilisation en parallèle des différentes machines, la librairie *ray* [123] a été choisie. Elle permet d'utiliser de façon efficace les ressources d'un cluster de machines (dans notre cas les 4 fermes de calcul ainsi que le PC attitré à la thèse).

4.3.2 Utilisation du *wrapper*

Le *wrapper* est instancié par un seul fichier de configuration au format *yaml*. Cela permet de modifier chacun des blocs indépendamment et simplement (voir figure 4.5).

```yaml
Parametres Communication :
  numeroPort : 10000

Parametres Simulation :
  germeAleatoire : 11111
  scenario : SMR
  frameSkipping : 5

Parametres Politique :
  nomPolitique : politiqueLambda
  arg1Politique : valeurArgument1
  arg2Politique : valeurArgument2

Parametres Fonction Recompense :
  listePoids : [0.2, 0.5]
  listeFonctionUnitaire : [
    {
      nomFonction : fonctionOrientation,
      arg1Fonction : valeurArgument1,
    },
    {
      nomFonction : fonctionProximitéPiste,
      arg1Fonction : valeurArgument1,
    }
  ]

Parametres Transformees Information :
  listeFonctionUnitaire : [
    {
      nomFonction : vecteurObservation,
      arg1Fonction : valeurArgument1,
    },
    {
      nomFonction : vecteurObservationPrivilégiée,
      arg1Fonction : valeurArgument1,
    }
  ]

Parametres Transformees Action :
  listeFonctionUnitaire : [
    {
      nomFonction : délai,
      arg1Fonction : valeurArgument1,
    },
    {
      nomFonction : bruit,
      arg1Fonction : valeurArgument1,
    }
  ]
```

FIGURE 4.5 – Fichier de configuration simplifié de l'environnement d'apprentissage

Il est possible d'itérer automatiquement avec des modifications sur n'importe quelle valeur de ce fichier de configuration afin de faciliter la répétition avec des germes aléatoires différents et l'optimisation d'hyperparamètres.

Bien que, in fine, le but de ce wrapper est de permettre l'exploitation du simulateur industriel, il est également possible d'utiliser n'importe quel environnement classique (comme *Cartpole* ou la suite *ALE*) afin de tester les algorithmes sur des environnements dont les résultats sont connus.

4.3.3 Efficacité

Comme les fermes de calcul n'ont pas de GPU, les calculs nécessaires pour le simulateur et le modèle opérateur, devront à terme, tous être faits sur CPU. Afin de pouvoir utiliser les réseaux de neurones sur les fermes sans ralentissement notable du temps par simulation, le nombre de paramètres doit donc être assez faible. Pour la suite des travaux, le nombre de paramètres du réseau de la politique sera de l'ordre 10^6 paramètres.

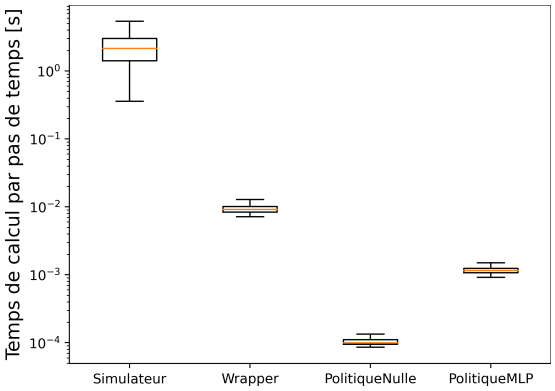
Le temps de calcul ajouté par les communications inter-processus est difficile à évaluer lors de simulation car cela nécessite une synchronisation entre les processus. Des mesures entre le

Corporate sensitivity
PUBLIC

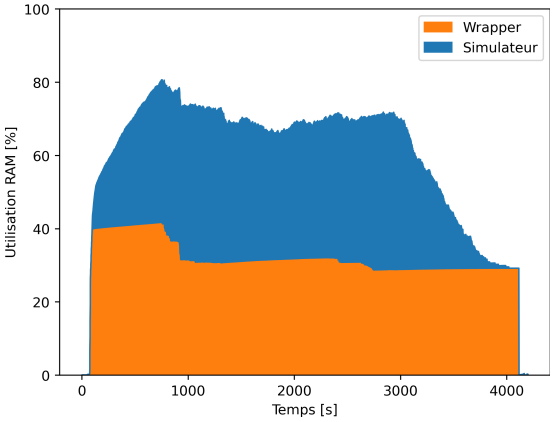
© Propriété Naval Group SA, (2026) tous droits réservés.

simulateur et une version simplifiée du *wrapper* ont montré que le temps nécessaire aux communications inter-processus est négligeable (environ 10^{-3} seconde) devant le temps moyen pour simuler t_{op} par le simulateur numérique. La figure 4.6a montre les temps pris par les différentes parties de l'environnement d'apprentissage industriel sur les scénarios SMR et SM$\bar{\text{R}}$ pour un pas de temps. Trois mesures sont prises au sein du processus Python :

- Simulateur : le temps pris par les blocs situés entre "Encodage de l'action" et "Décodage de l'information". Il mesure donc le temps pris par la communication inter-processus et le simulateur. Comme le temps pris par la communication inter-processus est négligeable, cette mesure est assimilable à celle d'un pas de temps de l'environnement.
- Politique : le temps pris par le bloc "Politique". Il mesure uniquement le temps pris par la politique pour choisir la prochaine action.
- *wrapper* : le temps pris par les blocs situés entre "Décodage de l'information" et "Encodage de l'action" moins le temps mesuré pour le bloc "Politique". Il mesure donc le temps pris par l'ensemble des blocs fonctionnels du processus Python du *wrapper*.



(a) Temps de calcul moyen par pas de temps par épisode de la "Politique", du *wrapper* et du "Simulateur" lors de l'exécution des scénarios SMR et SM$\bar{\text{R}}$ pour une politique envoyant toujours une action "Rien Faire" (PolitqueNulle) et une politique avec un réseau de neurones composé uniquement de couches *fully connected* avec 811 267 paramètres (PolitqueMLP) sur une ferme de calcul.



(b) Utilisation de la RAM lors de la simulation de SMR et SM$\bar{\text{R}}$ avec le *wrapper* sur une ferme de calcul avec 256 environnements en parallèle en utilisant une politique *fully connected* avec 811 267 paramètres

La figure 4.6a montre que le temps pris par le Simulateur est supérieur de plusieurs ordres de grandeur à celui pris par le *wrapper*. On peut également observer une différence notable entre une politique simple et une politique avec un réseau de neurones. Cependant, dans les deux cas, le temps d'inférence reste faible devant le temps pris par le Simulateur. On peut également noter qu'augmenter la valeur de n_{skip} augmente le temps mesuré pour le Simulateur et le *wrapper* mais ne modifie pas celui de la Politique. Globalement, on peut donc dire que l'efficacité calculatoire de l'implémentation du *wrapper* est suffisante pour ne pas induire de ralentissements significatifs.

La figure 4.6b montre l'utilisation de la RAM lors de la simulation de SMR et SM$\bar{\text{R}}$ avec le *wrapper* sur une ferme de calcul avec 256 environnements en parallèle. On peut voir que la RAM supplémentaire utilisée par le *wrapper* est non négligeable (environ autant que la RAM nécessaire pour le simulateur). Cela est dû au fait qu'il y ait un processus Python différent par simulation. Ainsi, l'utilisation de la RAM pour l'implémentation du *wrapper* est suffisante tant que la politique nécessite peu de RAM supplémentaire. Comme le temps moyen du simulateur est

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

grand devant celui du processus Python (Politique + *wrapper*), une perspective d'amélioration est d'instancier plusieurs environnements par processus Python. Cela réduirait le nombre de processus différents et donc réduirait fortement la RAM nécessaire.

Conclusion

Pour transformer le simulateur industriel en environnement d'apprentissage, il a fallu le modifier afin qu'il puisse être lancé de façon maîtrisée et qu'il établisse une communication bidirectionnelle avec un processus Python externe. Pour cela, l'initialisation a été modifiée et un "Événément opérateur" a été ajoutée au sein de la boucle événementielle. Ces modifications utilisent une communication *socket* afin de coupler le processus du simulateur numérique avec un objet Python, appelé *wrapper*.

En outre, plusieurs blocs fonctionnels supplémentaires sont nécessaires au sein du *wrapper* pour adapter le simulateur en un MDP. L'ensemble de ces blocs permettent de créer les données nécessaires (fonction de récompense et vecteur d'observation), de traiter l'action envoyée par la politique et maîtriser la fenêtre de contrôle laissée à la politique.

Finalement, une analyse critique du *wrapper* développé est présentée. Le développement des différents éléments s'appuie majoritairement sur les libraires *ray*, *torchrl* et *tensordict*. La spécification de l'ensemble des éléments du *wrapper* est centralisée dans un seul fichier de configuration. Le surcoût calculatoire est négligeable devant les temps de calcul du simulateur. L'utilisation supplémentaire de la RAM est en revanche assez importante mais reste dans les limites des fermes de calcul pour un réseau de neurones *fully connected* d'environ 10^6 paramètres.

Le chapitre suivant présente les choix pour la définition des différents éléments du MDP implémentés grâce au *wrapper*.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

5

Définition du problème d'apprentissage

Ce chapitre présente les choix effectués pour définir les différents éléments du problème d'apprentissage. Après avoir listé les difficultés inhérentes à la définition d'un problème d'apprentissage, les espaces d'observation et d'action sont définis. Cela permet ensuite de définir la fonction de récompense. Finalement, la fenêtre de contrôle du modèle est explicitée.

Sommaire

5.1 Difficultés inhérentes à la définition d'un problème d'apprentissage 72
5.1.1 Typologie des difficultés rencontrées . . . 72
5.1.2 Impacts des difficultés rencontrées sur la création d'un modèle . . . 74
5.2 Choix de versions . . . 74
5.3 Espace d'observation . . . 75
5.3.1 Construction de l'espace d'observation . . . 75
5.3.2 Représentation des positions dans l'espace d'observation . . . 75
5.4 Espace d'action . . . 76
5.4.1 Analyse de la perte de performance potentielle . . . 76
5.4.2 Construction de l'espace d'action . . . 77
5.5 Fonction de récompense . . . 77
5.5.1 Terme terminal . . . 77
5.5.2 Termes de *shaping* . . . 78
5.5.3 Pondération des termes de *shaping* par rapport au terme terminal . . . 80
5.5.4 Terme comportemental . . . 83
5.6 Fenêtre de contrôle . . . 85

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

5.1 Difficultés inhérentes à la définition d'un problème d'apprentissage

Pour obtenir un modèle, il faut à la fois un problème d'apprentissage à résoudre (ou MDP) et un algorithme de DRL utilisé pour résoudre ce problème. Contrairement au développement de nouveaux algorithmes sur des *benchmarks* où le MDP est déjà défini, dans un nouveau cas d'application, il est nécessaire de créer de façon conjointe ces deux aspects. En effet, évaluer si la définition du problème est correcte ne peut se faire qu'en utilisant un algorithme d'apprentissage puis en observant le modèle appris. Et inversement, il est impossible d'améliorer la capacité du modèle grâce à des modifications de l'algorithme de DRL (choix de méthode, choix d'hyperparamètres) si le MDP ne permet pas d'apprendre une politique adéquate.

Dans la suite du manuscrit, le **processus d'apprentissage** désigne l'ensemble des algorithmes de DRL et la définition du problème d'apprentissage pour créer la politique. La définition du processus d'apprentissage est un processus itératif dont les causes d'échecs potentielles sont multiples.

5.1.1 Typologie des difficultés rencontrées

Cette section décrit l'ensemble des types de difficultés rencontrées au cours de ces travaux lors de la mise en place du processus d'apprentissage. Elles ne sont pas spécifiques à notre cas d'application industriel.

Définition d'un problème d'apprentissage non soluble

L'apprentissage d'un modèle est la recherche d'une fonction capable d'approximer la fonction de politique optimale. Or les performances de cette politique optimale sont dépendantes de la définition du MDP. Ainsi, si même la politique optimale n'est pas satisfaisante, n'importe quelle approximation de cette fonction le sera également. Nous avons expérimenté plusieurs raisons pour lesquelles le MDP peut être non soluble.

L'espace d'observation ne donne pas d'informations suffisantes pour une prise de décision. En effet, si les informations à la disposition de la politique ne sont pas suffisantes, les récompenses expérimentées vont sembler aléatoire au vu de cet espace d'observation et il sera impossible de faire un choix d'action permettant d'augmenter l'espérance de gain futur. Par exemple, cela peut être le cas en définissant une politique n'ayant pas accès à l'historique dans un problème partiellement observable.

Les actions à la disposition de la politique ne permettent pas d'optimiser la fonction de récompense. Si les actions à la disposition du modèle ne permettent pas d'influer sur la récompense obtenue, la politique n'a alors aucun levier à sa disposition

La fenêtre de contrôle de la politique n'est pas adaptée car elle est, soit trop restrictive ce qui ne permet pas à la politique d'avoir assez de possibilités pour optimiser la récompense, soit trop permissive, ce qui donne accès à des transitions où la politique n'a rien à apprendre ce qui rend plus complexe la compréhension du problème en diluant les transitions intéressantes.

Définition d'un problème d'apprentissage non apprenable

Un point crucial lors de la définition du MDP est de faciliter l'apprentissage d'une politique maximisant la récompense. En effet, l'apprentissage d'une politique est un problème difficile qui nécessite de résoudre de façon adéquate le dilemme exploration-exploitation. Ainsi, il se peut que

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

la politique optimale soit effectivement alignée avec ce que l'on souhaite faire mais qu'il soit très difficile d'apprendre une politique proche.

Un premier élément qui peut rendre difficile d'apprendre une politique adéquate est si la fonction de récompense est *sparse*. Une fonction de récompense *sparse* donne une valeur de récompense non nulle que pour très peu de transitions, ce qui rend l'exploration très difficile car il faut une longue suite d'action adéquate avant d'obtenir une récompense positive. Par exemple, pour le jeu d'échec, une fonction de récompense *sparse* est de donner une récompense que lors de la fin de partie (1 pour une victoire, 0 pour un match nul et -1 pour une défaite). L'approche classique est d'ajouter des termes de *reward shaping*, c'est-à-dire des termes plus denses qui orientent vers l'objectif réel. Cependant, la maximisation de la fonction de récompense doit rester alignée avec le comportement souhaité, c'est-à-dire de ne laisser la possibilité de faire du *reward hacking*.

La mise en forme de l'espace d'observation est également cruciale. En effet, comme tout problème d'apprentissage automatique, la façon dont les informations sont représentées numériquement dans le vecteur d'observation peuvent rendre plus ou moins difficile l'extraction de caractéristiques pour la prise de décision. Par exemple, il est possible d'utiliser le *multi-hot encoding* pour les caractéristiques catégorielles ou le *cyclical encoding* pour les caractéristiques cycliques.

L'espace d'action peut avoir une forte influence sur la facilité d'apprendre un modèle. En effet, plus le lien entre les actions laissées à la disposition de la politique et les récompenses obtenues sont complexes, plus il sera difficile d'apprendre une politique efficace.

Mauvais paramétrage du processus d'apprentissage

Les algorithmes de DRL comportent plusieurs hyperparamètres et sont très sensibles à leurs modifications (voir section 2.6.4), c'est-à-dire que le mauvais paramétrage d'un seul hyperparamètre peut à lui seul conduire à la divergence d'un entraînement. En outre, la définition d'un MDP, au même titre que les algorithmes de DRL, nécessitent des choix entre plusieurs éléments (différents espaces d'action, espace d'observation et fonction de récompense) et plusieurs valeurs d'hyperparamètres. Il est donc crucial de trouver un ensemble de valeurs pour les hyperparamètres pour l'ensemble des éléments composant le processus d'apprentissage. Pour l'ensemble du processus d'apprentissage utilisé dans cette section, nous avons 26 hyperparamètres différents, ce décompte exclue les choix faits entre les différentes versions d'éléments du MDP ou d'algorithmes de DRL.

Finalement, il est également nécessaire de s'assurer que la quantité de données générées au cours de l'apprentissage est suffisante. En effet, a priori, il peut être difficile de différencier un modèle qui n'apprend rien d'un autre modèle qui n'est qu'au début de son apprentissage et qui nécessite beaucoup plus de données.

Algorithme de DRL non adapté à l'environnement

Comme montré dans la section 2.4, il existe une multitude de modifications algorithmiques pour le DRL. Cependant, elles ne sont pas toutes efficaces sur tous les environnements et il est a priori difficile de savoir les modifications qui seront efficaces sur un environnement. Par exemple, l'article présentant le *Noisy Linear* [52], Fortunato et al. se comparent à trois algorithmes sur l'ensemble des 57 environnements du *benchmarks* [43]. On peut noter que l'ajout soit globalement bénéfique dans tous les cas. Cependant, pour les trois comparaisons, on peut observer qu'aucune amélioration n'est sur l'ensemble des environnements, et que les environnements sur lesquels il y

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

a une chute de performance sont différents.

Il est donc possible que l'algorithme de DRL ne soit pas adapté aux spécificités de l'environnement.

Erreur d'implémentation

L'implémentation d'un processus d'apprentissage nécessite la mise en place de nombreux éléments (*wrapper*, algorithme de DRL et calculs distribués). Cette multiplicité augmente les chances de faire des erreurs d'implémentation. Une des difficultés notables est la possibilité de faire des erreurs "silencieuses", c'est-à-dire qu'elles ne créent pas de problèmes identifiés par le compilateur ou l'interpréteur du langage, mais elles induisent un comportement mathématique erroné. Lors de la mise au point d'un processus d'apprentissage sans référence à laquelle se comparer, il est alors très difficile de discerner une erreur silencieuse d'une des autres raisons potentielles mentionnées dans les sections précédentes.

5.1.2 Impacts des difficultés rencontrées sur la création d'un modèle

Un seul de ces problèmes peut conduire à l'apprentissage d'une politique totalement inefficace. Initialement, il est possible que plusieurs de ces problèmes soient présents. Alors, une amélioration peut devenir non mesurable. Par exemple, si le processus d'apprentissage ne fonctionne pas car l'espace d'observation est mal défini et que l'algorithme de DRL est inadapté, une meilleure définition de l'espace d'observation peut être indiscernable car l'algorithme reste inadapté.

En outre, la modification du MDP implique souvent une difficulté de comparer des résultats entre eux. Par exemple, il est complexe de comparer des entraînements dont la fonction de récompense est différente. Il faut donc utiliser des métriques métiers ou l'avis d'expert métier, qui sont des mesures extérieures de la performance du comportement appris.

Il y a une interdépendance de tous les éléments du processus d'apprentissage, impliquant qu'un choix peut induire qu'un choix précédent n'est plus optimal. En outre, cette difficulté pour choisir est amplifiée par l'aspect stochastique et les temps de calcul nécessaires (voir section 2.6.4). Un entraînement de $2{*}10^6$ pas de temps sur les scénarios SM prend quatre heures sur deux fermes de calcul, et l'évaluation des performances d'un modèle prend un peu plus de 2 heures sur deux nœuds. Ainsi, tester un processus d'apprentissage en entraînant un modèle en l'évaluant pour 5 germes prend un peu plus de trente heures sur deux fermes de calculs.

La mise au point du processus d'apprentissage est présentée dans les chapitres 5 (définition du MDP), 6 (choix de l'algorithme de DRL) et 7 (utilisation des structures connues). Cependant, les éléments de ces différents chapitres ont été choisis de manière simultanée au cours des travaux de thèses. Pour certains éléments de ce chapitre, des apprentissages ont été nécessaires pour choisir l'élément ou le paramétrage retenu. Ils sont réalisés avec l'algorithme décrit dans le chapitre 6.

5.2 Choix de versions

Lors de l'apprentissage d'un modèle opérateur, un point crucial est de définir l'ensemble des différentes situations simulées par l'environnement. Le simulateur numérique et le logiciel tactique sont en constante évolution avec de nouvelles versions régulières. Nous avons fait le choix d'utiliser un seul couple de version tout au long du développement du processus d'apprentissage. En effet, comme mentionné dans la section 5.1, toute modification sur le logiciel tactique ou le simulateur implique une modification des performances des modèle et de la référence. Afin de

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

garder la capacité de comparer l'ensemble des résultats sur une même version plutôt que d'utiliser la version la plus récente.

Puisque le type de cible est connu dans les conditions initiales, et que l'attaque d'un BS et d'un SM diffère d'un point de vue tactique, nous avons décidé d'apprendre un modèle opérateur différent pour les scénarios SM et BS. Il est important de noter que ce choix a pour but d'augmenter les performances en spécifiant chaque modèle opérateur à un seul type de cible mais cela augmente le nombre de modèles ce qui réduit donc la facilité de mise en œuvre du modèle (plus de modèles à apprendre et à utiliser lors des simulations).

Cependant, les modèles ne pourront pas être spécifiés entre les différentes versions des scénarios R et $\bar{\text{R}}$. En effet, il n'est a priori pas possible de savoir si la cible réagira ou non en condition réelle, chaque modèle doit donc être capable de traiter les versions R et $\bar{\text{R}}$ de son type de cible.

5.3 Espace d'observation

Cette section décrit la construction de l'espace d'observation du MDP qui constitue l'ensemble des informations à la disposition du modèle opérateur pour choisir une action.

5.3.1 Construction de l'espace d'observation

L'espace d'observation est composé de trois types d'informations :

— les informations sur la cible issues des capteurs du lanceur (la position de la DO) et des conditions initiales ;

— les informations sur l'état du logiciel tactique de la torpille comme l'état courant de la MAE (représenté avec un *multi-hot encoding*) ou des informations relatives à une piste (position, vitesse) ;

— les informations issues des capteurs de la torpille.

Une deuxième version de l'espace d'observation est définie disposant d'informations privilégiées. Elle dispose notamment de la position réelle de la cible à la place de la DO. Ainsi, tout modèle utilisant cet espace d'observation est inacceptable car il triche.

Lors des choix faits dans ce chapitre et le chapitre 6, l'espace d'observation privilégié est utilisé car cela simplifie le problème. Ainsi, les gains de performances sont plus importants, ce qui facilite la comparaison des résultats. Cependant, ces performances sont en réalités des bornes majorantes de la performance qu'on peut obtenir avec l'espace d'observation non privilégié. En effet, il est possible que le modèle réussisse des tirs qu'il n'aurait pas pu réussir sans informations privilégiée, c'est-à-dire un tir de l'ensemble Récupérable mais pas de l'ensemble Décidable (voir section 3.1.5).

5.3.2 Représentation des positions dans l'espace d'observation

Toutes les positions sont représentées en coordonnées sphériques dans le référentiel de la torpille. Elles sont donc décrites par trois valeurs (voir la figure 5.1) :

— la distance, notée $d \in \mathbb{R}^+$ en mètre ;

— la circulaire, notée $c \in [-180, 180]$ en degré ;

— l'élévation, notée $e \in [-90, 90]$ en degré.

Dans l'espace d'observation, une position est décrite par 4 valeurs :

— le logarithme de sa distance, afin qu'une modification de quelques mètres soit beaucoup plus importante lorsque la valeur est proche de 0 (c.-à-d. proche de la torpille) que lorsque la valeur est grande ;

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

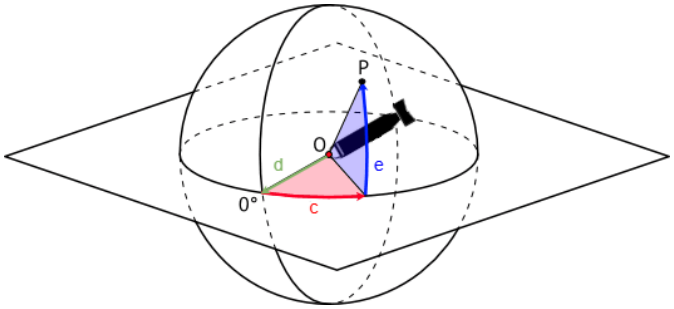


FIGURE 5.1 – Description de la définition de la distance, de la circulaire et de l'élévation d'un objet P dans le repère de la torpille.

— la circulaire est représentée avec un *cyclical encoding*, c'est-à-dire par le cosinus et le sinus de la circulaire. cela permet de ne pas introduire de discontinuité artificielle entre des valeurs de circulaire dont la valeur est proche de 180 degrés et -180 degrés tout en gardant une bijectivité modulo 2π ;
— l'élévation.

Afin de rendre compte de la dynamique de l'objet, la différence entre la dernière et la nouvelle représentation de la position est ajoutée.

5.4 Espace d'action

Comme décrit dans la section 3.1.2, l'espace d'action a une dimension de l'ordre de 10^2 avec une proportion d'actions réellement disponibles en moyenne assez faible au cours d'un tir. Suite à l'analyse des actions faites par les opérateurs réels et des échanges avec des experts métiers, un sous-ensemble de 4 actions a été élaboré. Ces 4 actions incluent de la logique afin de choisir une seule ou une suite d'actions potentiellement intéressante en fonction de la situation. Les 4 actions sont appelées Action$i_{i\in[1,4]}$ car il est impossible de les décrire pour des raisons de sûreté.

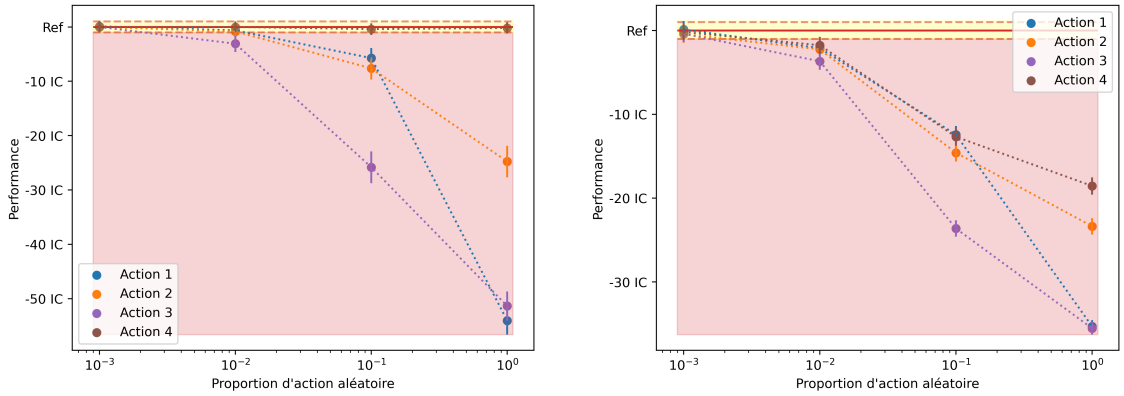
5.4.1 Analyse de la perte de performance potentielle

La figure 5.2 analyse l'impact des actions Action$i_{i\in[1,4]}$ lorsqu'elles sont faites de façon aléatoire sur les performances. Elle montre la performance d'une politique avec un espace d'action de dimension 2 avec l'action "Rien faire" et l'une des actions de l'ensemble Action$i_{i\in[1,4]}$. Le choix entre les deux actions est aléatoire avec une proportion grandissante de choisir l'action non nulle. La figure 5.2a montre une forte perte de performance par rapport à la référence à cause de l'ajout d'une action aléatoire même avec une proportion de 1% pour l'ensemble des actions sur $\overline{\text{SMR}}$ à part pour l'Action 4. La figure 5.2b montre un fort impact négatif sur SMR, même pour l'Action 4. Cela démontre le fort impact des actions opérateurs sur la réussite d'un tir et donc la facilité pour le modèle opérateur d'avoir un fort impact négatif sur les performances par rapport à la référence.

À cause de la méthode de blanchiment des résultats présentée dans la section, il est important de noter que l'échelle de performance entre les différents scénarios n'est pas la même. En effet, elles sont exprimées de façon relative à de la performance de référence en intervalle de confiance à 95% (noté IC) de la performance de référence, or chaque scénario ($\overline{\text{SMR}}$, SMR, $\overline{\text{BSR}}$ et BSR) a une performance et un IC de référence différent. Les résultats sont similaires sur BS et ne sont donc pas montrés.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.



(a) SM$\bar{\text{R}}$ (b) SMR

FIGURE 5.2 – Dégradation des performances sur SM pour une politique choisissant aléatoirement entre l'action "Rien Faire" et une des quatre actions en fonction de la proportion de choix aléatoire.

5.4.2 Construction de l'espace d'action

Pour rappel, les actions sont hauts niveaux et très différentes entre elles. Nous avons donc décidé de faire des apprentissages séparés pour simplifier le problème et observer ce qui est possible d'être fait avec chacune d'entre elles.

Lors de la suite de ce chapitre et du chapitre 6, l'espace d'action considéré est de dimension 2 avec seulement les actions "Rien Faire" et "Action 1". Nous avons choisi cette action car c'est celle qui donne les gains de performance les plus importants, ce qui simplifie la comparaison pour la mise au point du processus d'apprentissage. Cela implique donc une potentielle sous-évaluation des performances des modèles car les possibilités du modèle sont restreintes.

5.5 Fonction de récompense

La fonction de récompense est le seul signal permettant d'orienter l'apprentissage de la politique. La fonction de récompense utilisée est une combinaison linéaire de plusieurs termes. Pour ces termes nous avons considéré trois types :

— terme terminal : c'est un terme utilisant seulement l'état du simulateur, qui peut prendre une valeur non nulle uniquement à la dernière transition de la trajectoire.
— terme de *shaping* : c'est un terme utilisant seulement l'état du simulateur, qui peut prendre une valeur non nulle à toutes les transitions et qui a pour but de densifier la fonction de récompense.
— terme comportemental : c'est un terme pouvant utiliser à la fois l'action choisie et l'état interne du simulateur, qui a pour but d'évaluer la manière d'agir de la politique. Cela permet notamment d'ajouter des contraintes douces pour l'aspect facteur humain.

5.5.1 Terme terminal

Le seul terme terminal considéré est une fonction de la probabilité de destruction du tir. Il est une retranscription directe d'un des deux objectifs de représentativité du modèle opérateur,

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity
PUBLIC

c'est-à-dire la maximisation de la performance de l'arme. Il est effectivement terminal car la probabilité de destruction ne peut être évaluée qu'à la fin du tir.

5.5.2 Termes de *shaping*

Description des termes considérés

Trois concepts de termes de *shaping* ont été développés :

— un terme utilisant la distance torpille-cible, et qui encourage la torpille à se rapprocher de la cible ;

— un terme utilisant la circulaire de la cible, et qui encourage la torpille à s'orienter vers la cible (c.-à-d. une circulaire de la cible proche de 0) ;

— un terme utilisant la distance cible-piste, lorsqu'une piste existe, et qui encourage la piste à être proche de la cible.

Deux types de récompenses ont été implémentées à partir de ces concepts de termes :

— quantitative : la valeur de la récompense est une valeur numérique mesurant à quel point l'événement est réalisé ;

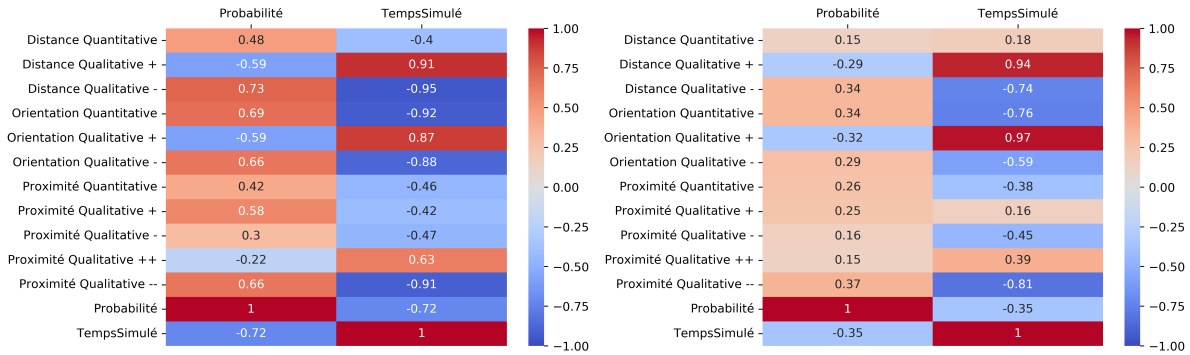
— qualitative : la valeur de la récompense prend un nombre fini de valeurs dépendant de conditions liées à l'événement (valeurs seuils).

Pour une récompense qualitative, il est possible de pénaliser la non-réalisation et/ou de récompenser la réalisation. Un terme qualitatif récompensant la réalisation sera dénoté par un "+" alors qu'un terme pénalisant la non-réalisation sera dénoté par un "-".

Ces différents termes de récompenses sont décrits plus en détails dans l'annexe B.

Choix des termes

Pour choisir les termes de *shaping*, nous avons pu tirer profit de la connaissance d'une politique de référence (c.-à-d. le logiciel tactique seul). En effet, l'évaluation des différentes versions des termes de *shaping* avec la politique de référence permet d'observer son alignement avec les performances pour une bonne politique.



(a) SM$\bar{\text{R}}$ et SMR (b) BS$\bar{\text{R}}$ et BSR

FIGURE 5.3 – Corrélation de l'ensemble des termes de *shaping* considérés avec la probabilité de réussite et la durée d'un tir pour les scénarios BS et SM sans action opérateur.

La figure 5.3 montre la corrélation de chacun des termes avec la probabilité de destruction et la durée d'un tir. Ainsi, un bon terme de *shaping* doit être corrélé positivement avec la probabilité

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

de destruction, mais il doit également être corrélé négativement avec la durée d'un tir afin de privilégier des stratégies plus rapides.

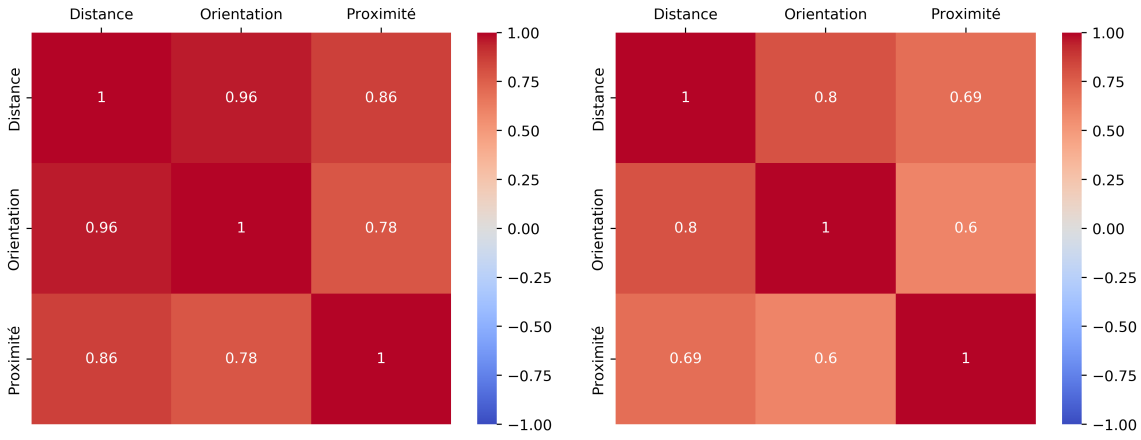
Tout d'abord, la corrélation entre le temps d'un tir et sa probabilité de destruction est plus faible pour BS que SM. En outre, les valeurs de corrélation avec la probabilité de destruction sont globalement plus faibles pour BS que pour SM. Il semble donc que les termes de récompense envisagés sont meilleurs pour l'ensemble du scénario SM et que le temps d'un tir est également un meilleur indicateur de sa réussite pour SM.

On peut observer des valeurs très différentes de corrélations pour l'ensemble des versions de termes de *shaping* d'un seul concept. Par exemple, dans la figure 5.3a, le terme "Distance Qualitative +" est positivement corrélé avec le temps et négativement avec la probabilité de destruction alors que le terme "Distance Qualitative -" est négativement corrélé avec le temps et positivement avec la probabilité de destruction. En outre, une légère modification peut fortement modifier la valeur de corrélation. En effet, le terme "Proximité Qualitative --" ajoute une légère pénalisation lorsqu'il n'existe pas de piste par rapport au terme "Proximité Qualitative -" mais cela amplifie fortement la corrélation positive avec la probabilité de destruction et la corrélation négative avec le temps d'un tir. Ainsi, la façon dont un concept de récompense est implémenté est aussi importante que le concept en lui-même.

En croisant les valeurs de corrélation, l'observation de tirs avec des valeurs extrêmes et l'avis d'experts métier, une version de chacun des termes pour chacun des concepts a été choisie :

— Distance Qualitative - ;
— Orientation Qualitative - ;
— Proximité Qualitative --.

Nous avons également observé la corrélation entre les termes afin de ne pas introduire de redondance. Sur BS et SM, on observe alors une forte corrélation entre le terme d'orientation et de distance. Nous avons donc choisi de garder celui dont la corrélation avec le terme de proximité est le plus faible, c'est-à-dire le terme d'orientation. La corrélation entre le terme d'orientation et de proximité reste très forte, mais ils sont liés à deux concepts différents pouvant être optimisés séparément.



(a) SM$\bar{\text{R}}$ et SMR (b) BS$\bar{\text{R}}$ et BSR

FIGURE 5.4 – Corrélation des termes de *shaping* choisis pour les scénarios BS et SM sans action opérateur.

Finalement, afin de pondérer les termes de *shaping* entre eux, nous avons observé la valeur moyenne des deux récompenses et arbitrairement décidé qu'elles devaient être similaires car les

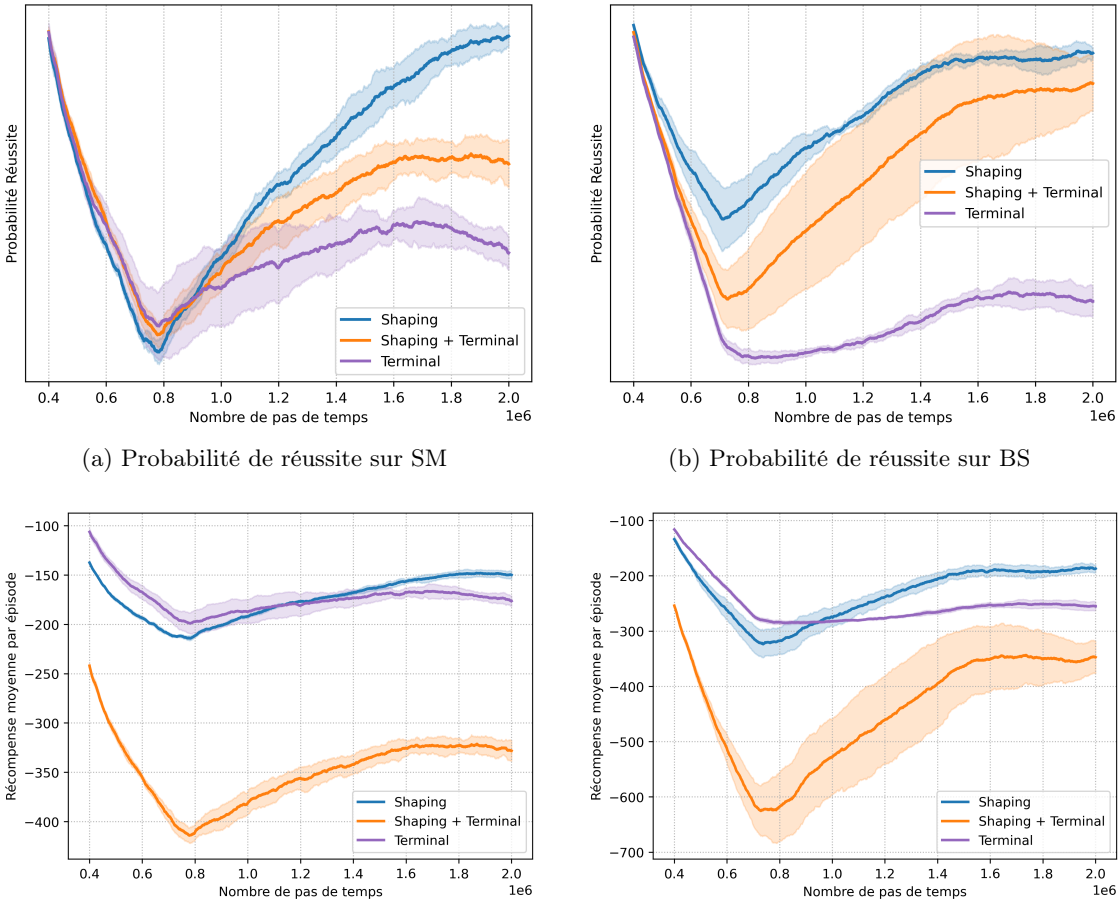
© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity
PUBLIC

deux concepts sont d'importances égales. Cela induit que la fonction de récompense est différente pour SM et BS car les valeurs moyennes sont différentes.

5.5.3 Pondération des termes de *shaping* par rapport au terme terminal

Afin de faciliter la lisibilité de la valeur de la fonction de récompense, nous avons créé un terme de récompense terminal dont la valeur varie linéairement entre -1 et 0 lorsque la probabilité de destruction varie de 0 à 1. Ainsi, peu importe les termes considérés et leur pondération, comme les termes de *shaping* sont dans un paradigme de pénalisation, la récompense accordée à une trajectoire ne peut excéder la valeur plafond de 0 (c.-à-d. aucune pénalisation de *shaping* et une probabilité de destruction de 1).

La figure 5.5 montre l'apprentissage avec 3 fonctions de récompenses différentes en utilisant l'algorithme décrit dans le chapitre 6. Le choix de la pondération entre les deux termes de *shaping* et le terme terminal est fait pour que les valeurs moyennes soient similaires sur la référence.



(a) Probabilité de réussite sur SM

(b) Probabilité de réussite sur BS

(c) Récompense sur SM

(d) Récompense sur BS

FIGURE 5.5 – Courbe d'apprentissage moyennée sur une fenêtre glissante de 400 000 pas de temps moyennée sur 5 germes avec l'écart-type sur les scénarios SM et BS pour différentes fonction de récompense (*shaping*, *shaping* + terminal, terminal).

On peut voir sur les figures 5.5c et 5.5d que la modification de la fonction de récompense

Corporate sensitivity
PUBLIC

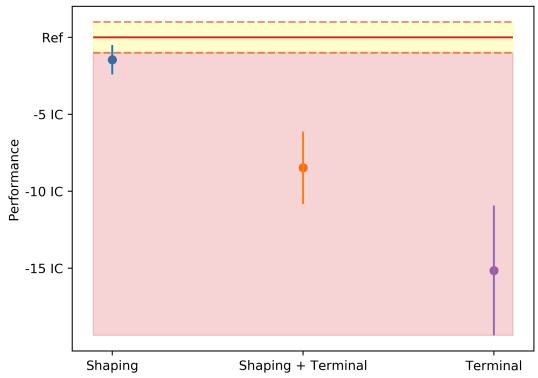
© Propriété Naval Group SA, (2026) tous droits réservés.

rend les courbes d'apprentissage de récompense non comparable à la fois entre scénario et entre les récompenses.

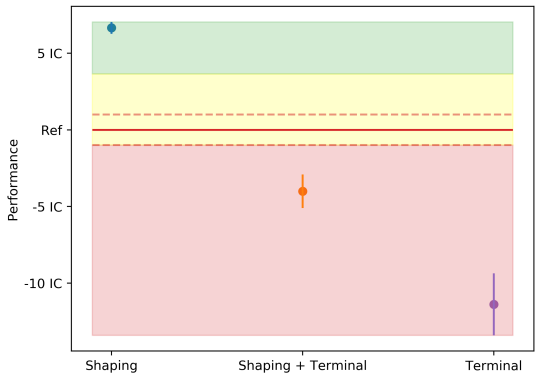
On observe sur les figures 5.5a et 5.5b que les apprentissages utilisant seulement le terme terminal ont les performances les plus faibles, ce qui corrobore l'importance du *reward shaping* pour créer un MDP plus facilement apprenable (comme mentionné dans la section 5.1). En outre, les apprentissages utilisant seulement les termes de *shaping* obtiennent des meilleures probabilités de réussite moyenne par rapport aux apprentissages utilisant les termes de *shaping* et le terme terminal.

Notre interprétation de ce phénomène est que la probabilité de destruction est hautement stochastique pour le modèle opérateur car il n'a qu'un contrôle haut-niveau sur le comportement de la torpille. Autrement dit, l'opérateur n'agit qu'indirectement sur la probabilité de destruction obtenue à la fin du tir. Et cette probabilité de destruction n'est parfois rendue disponible qu'après plusieurs minutes après que l'action ait été réalisée. Il est donc beaucoup plus difficile d'estimer la récompense espérée pour une certaine politique lorsque la probabilité de destruction est présente dans la récompense.

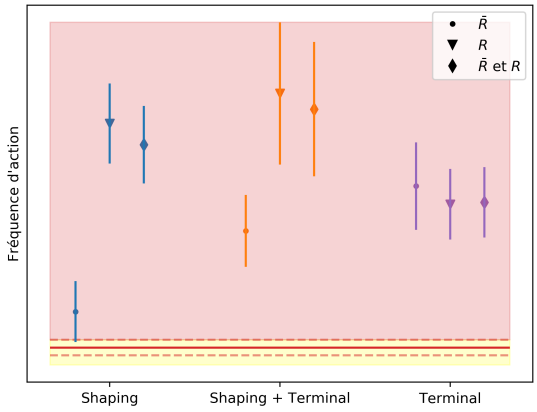
On peut également noter que l'intervalle de confiance est d'autant plus grand que la probabilité de destruction est faible. Cela traduit un plus grand écart-type des 5 modèles montrant une plus grande stochasticité sur les performances des modèles appris.



(a) Probabilité de réussite sur SM$\bar{\text{R}}$



(b) Probabilité de réussite sur SMR



(c) Fréquence d'action sur SM

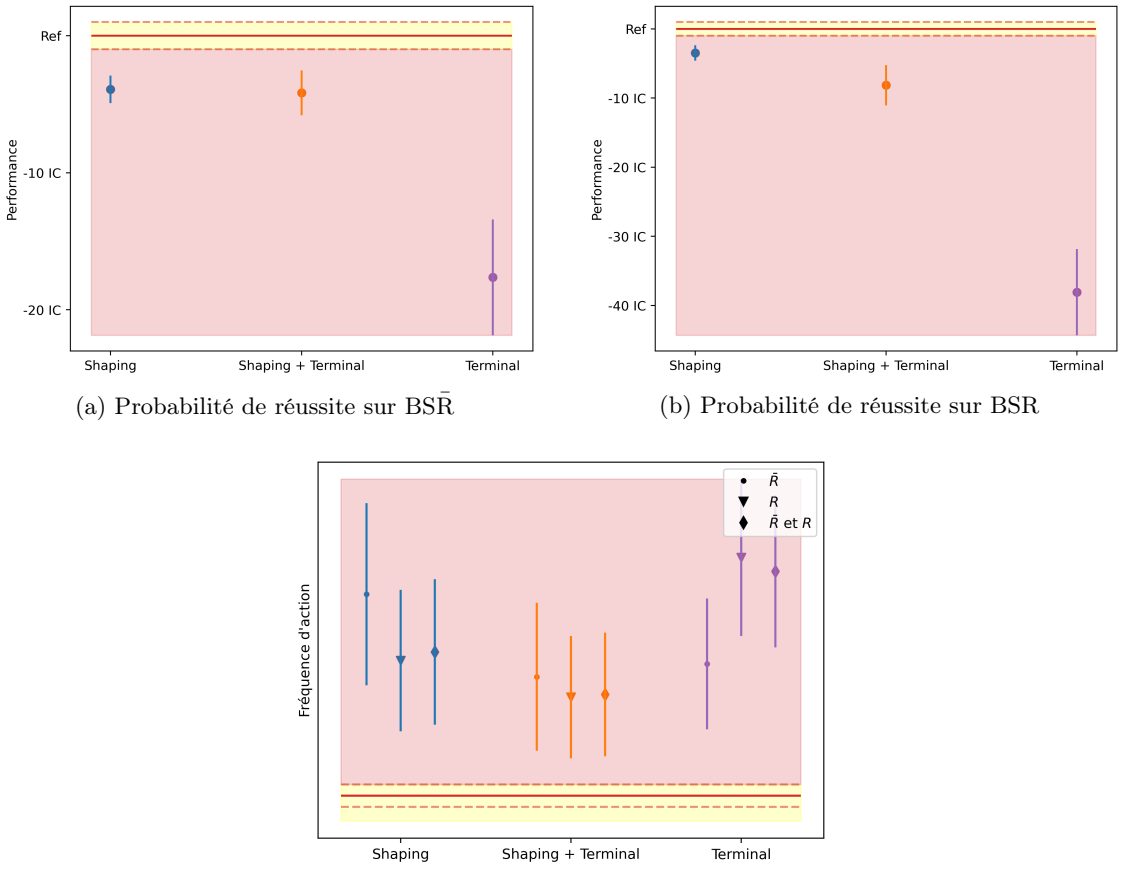
FIGURE 5.6 – Performance et fréquence d'action sur SMR et SM$\bar{\text{R}}$ moyennées sur l'ensemble des modèles appris dans la figure 5.5 avec un intervalle de confiance à 95%.

© *Propriété Naval Group SA, (2026) tous droits réservés.*

Corporate sensitivity
PUBLIC

La figure 5.6 montre l'évaluation des modèles issus des apprentissages sur SM présentés dans la figure 5.5 sur l'ensemble de SM. On peut observer sur la figure 5.6a que les performances gardent l'ordre observé dans la figure 5.5a. De plus, les performances des modèles appris avec seulement le terme de *shaping* sont dans la zone de gain notable pour SMR alors que les deux autres sont dans la zone inacceptable. Pour $\text{SM}\bar{\text{R}}$, on obtient le même ordre de performance entre les différentes fonctions de récompenses. Cependant, toutes les performances sont dans la zone inacceptable, cela montre la difficulté du modèle pour améliorer les performances car la référence est déjà très performante.

La figure 5.6c montre la fréquence d'action sur l'ensemble de SM, on constate qu'elle est largement supérieure à la valeur maximale acceptable pour tous les modèles. On peut également remarquer une fréquence d'action beaucoup plus faible pour $\text{SM}\bar{\text{R}}$ par rapport à SMR. Ainsi, bien que le modèle ne connaisse pas a priori si la cible fait une réaction, la politique apprise est beaucoup moins active sur $\text{SM}\bar{\text{R}}$.



(a) Probabilité de réussite sur $\text{BS}\bar{\text{R}}$

(b) Probabilité de réussite sur BSR

(c) Fréquence d'action sur BS

FIGURE 5.7 – Performance et fréquence d'action sur BS moyennées sur l'ensemble des modèles appris dans la figure 5.5 avec un intervalle de confiance à 95%

La figure 5.7 montre l'évaluation des modèles issus des apprentissages sur BS présentés dans la figure 5.5 sur l'ensemble de BS. On peut observer sur la figure 5.7b que les gains de performances gardent l'ordre observé dans la figure 5.5b. Cependant, contrairement aux modèles sur SM, tous les modèles présentent une performance dans la zone inacceptable. Les hypothèses quant à cette

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

différence sont :

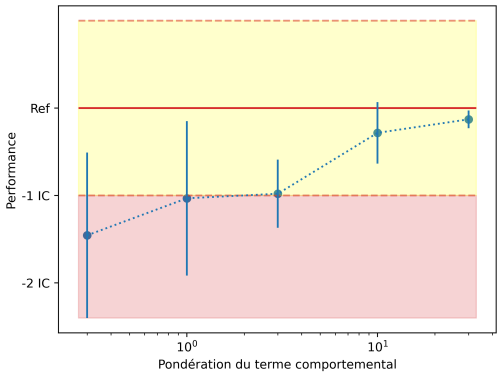
— une fonction de récompense moins adéquate, ce qui est corroborée par la corrélation plus faible avec la probabilité de destruction (voir figure 5.4b) ;
— l'action 1 est moins efficace pour les scénarios BS ;
— améliorer les performances sur l'ensemble BS est plus difficile que sur SM.

La pondération choisie pour la fonction de récompense entre les termes de *shaping* et le terme comportemental est donc de ne pas ajouter le terme terminal pour les scénarios SM et BS.

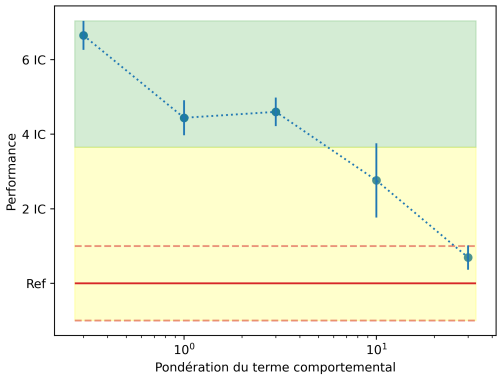
5.5.4 Terme comportemental

Comme l'ensemble des modèles ont des fréquences d'action largement inacceptable, un terme comportemental est ajouté pour pénaliser l'utilisation d'action différentes de "Rien Faire". Le terme est défini par :

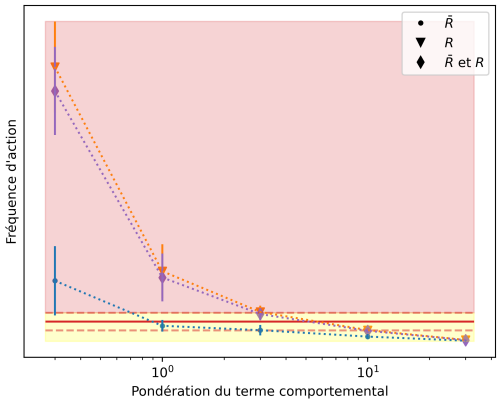
$$r_t(s_t, a_t) = \begin{cases} 0\,, \text{ si } a_t = \text{"Rien Faire"} \\ 1\,, \text{ sinon} \end{cases} \tag{5.1}$$



(a) Probabilité de réussite sur $\text{SM}\bar{\text{R}}$



(b) Probabilité de réussite sur SMR



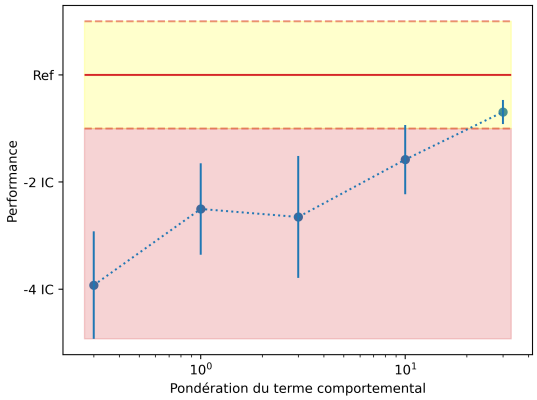
(c) Fréquence d'action sur SM

FIGURE 5.8 – Performance et fréquence d'action sur SM moyennées sur l'ensemble des modèles en fonction de la pondération du terme comportemental avec un intervalle de confiance à 95%. La valeur de pondération de 0 est placée à la valeur 0.3 pour des raisons de lisibilité.

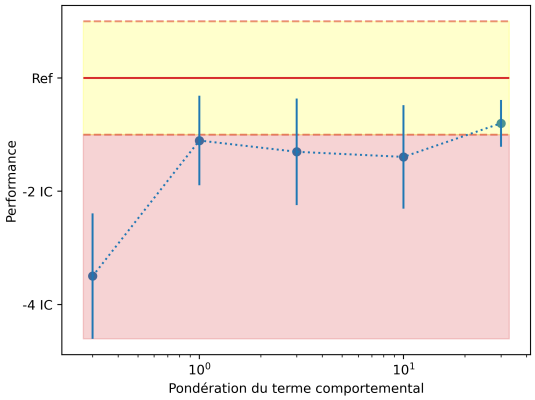
© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

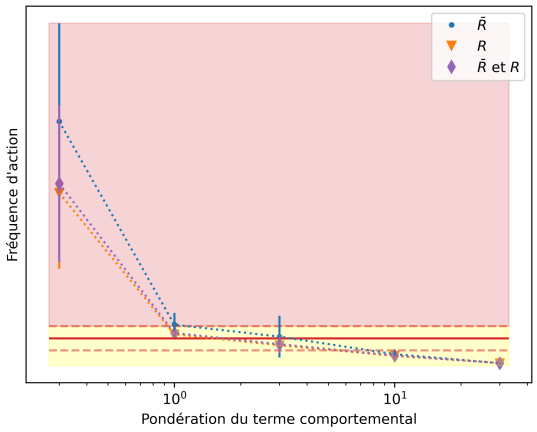
La figure 5.8 montre l'évolution de la performance et de la fréquence d'action en fonction de la pondération du terme comportemental sur l'ensemble des scénarios SM. La figure 5.8c montre une baisse de la fréquence d'action plus la pondération du terme comportemental est forte. Ce terme de pénalisation permet donc bien d'influer sur la fréquence d'action du modèle. On observe également dans la figure 5.8b que la performance sur SMR baisse également avec l'augmentation de la pondération du terme comportemental. En effet, des actions avec des faibles gains de récompenses espérés sur les termes de *shaping* deviennent inefficaces avec la pénalisation du terme comportemental. Néanmoins, la figure 5.8a montre que sur SM$\bar{\text{R}}$, le résultat obtenu est inverse. Cela renforce le constat qu'une amélioration par rapport à la référence sur SM$\bar{\text{R}}$ est très difficile et que, puisque la performance de référence est très haute, alors les modèles doivent apprendre à ne pas agir sur ces tirs.



(a) Probabilité de réussite sur BS$\bar{\text{R}}$



(b) Probabilité de réussite sur BSR



(c) Fréquence d'action sur BS

FIGURE 5.9 – Performance et fréquence d'action sur BS moyennées sur l'ensemble des modèles en fonction de la pondération du terme comportemental avec un intervalle de confiance à 95%. La valeur de pondération de 0 est placée à la valeur 0.3 pour des raisons de lisibilité.

La figure 5.9 montre l'évolution de la performance et de la fréquence d'action en fonction de la pondération du terme comportemental sur l'ensemble des scénarios BS. De façon similaire aux scénarios SM, La figure 5.9c montre que le terme de pénalisation permet d'influer sur la fréquence d'action du modèle. Cependant, comme les modèles sur BS conduisent à une perte

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

de performance, les figures 5.9a et 5.9b montrent que l'augmentation du terme de pénalisation conduit à une augmentation des performances. En effet, comme les modèles sont incapables de choisir des actions bénéfiques, la réduction de la fréquence d'action rapproche les modèles de la performance de référence.

Contrairement à la probabilité de destruction, la valeur de référence de fréquence d'action est la même pour l'ensemble des scénarios. On note donc une fréquence d'action plus faible sur BS que sur SM.

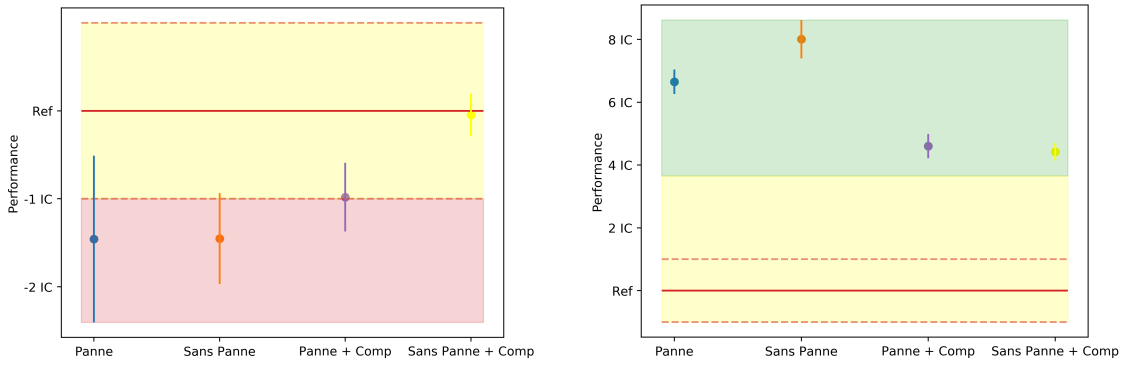
Des versions plus complexes de terme comportementaux ont été expérimentés. L'objectif était d'appliquer une pénalisation croissante en fonction du nombre d'actions effectuées depuis le début du tir tout en ajoutant ce nombre à l'espace d'observation pour rendre le problème observable. Toutes ces versions introduisent de nouveaux hyperparamètres et aucune d'entre elles n'a permis de gain notable par rapport à une version pénalisant de façon constante.

Pour SM, le terme comportemental de pénalisation fixe a donc été choisi avec une pondération de 3 car elle obtient la performance la plus haute sur SMR, tout en ayant une fréquence d'action totale et une performance sur $\text{SM}\bar{\text{R}}$ acceptable. Pour BS, le terme comportemental de pénalisation fixe a été choisi avec une pondération de 3 car il permet de garder un modèle avec une fréquence d'action similaire à celle observée sur les tirs réels.

5.6 Fenêtre de contrôle

La F21 est un système complexe disposant de plusieurs phases et de différents logiciels jouant des rôles différents (voir chapitre 1). Cela divise naturellement le tir en plusieurs parties en fonction du logiciel qui contrôle la torpille. La fenêtre de contrôle a donc été réduite aux moments où le logiciel tactique est en contrôle.

Une autre réduction considérée est de retirer certains types de pannes de la torpille lors de la phase d'entraînement. Afin de se comparer à la référence, ces pannes doivent cependant être ajoutées lors de l'évaluation du modèle.



(a) $\text{SM}\bar{\text{R}}$ (b) SMR

FIGURE 5.10 – Performance sur SM moyennées sur 5 germes aléatoires avec un intervalle de confiance à 95% en fonction de la simulation de panne lors de l'apprentissage.

La figure 5.10 montre l'effet du retrait des pannes lors de la phase d'entraînement pour des apprentissages avec et sans le terme comportemental. Ainsi, la figure 5.10a montre un gain significatif avec le terme comportemental sur $\text{SM}\bar{\text{R}}$. La figure 5.10b montre un gain significatif sans le terme comportemental sur SMR.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Il n'y a pas de changements significatifs sur la fréquence d'action. En outre, aucun gain ou perte significatif n'est observé sur BS.

Ce phénomène est intéressant car un décalage entre les données d'entraînement et d'évaluation est légèrement bénéfique pour les performances. Notre hypothèse est que, puisque le modèle est incapable de réagir à ces pannes, cela augmente l'aspect partiellement observable du problème et donc rend l'apprentissage plus difficile. Puisque le retrait des pannes est soit bénéfique, soit transparent, les pannes ont donc été retirées lors de la phase d'entraînement pour tous les entraînements.

Conclusion

La définition du problème d'apprentissage est un processus itératif étroitement lié aux algorithmes de DRL utilisés. La mise au point conjointe de ces deux éléments est complexe car les sources d'échecs sont multiples et difficiles à identifier lorsque tous les éléments doivent être définis conjointement.

Afin de garder la possibilité de comparer les résultats au cours du développement du processus d'apprentissage, nous avons fait le choix de conserver, à la fois la même version du logiciel tactique et la même version du SN durant l'ensemble des travaux de thèse. La connaissance dans les conditions initiales du type de cible rend possible l'apprentissage séparé d'un modèle pour BS et SM.

L'espace d'observation est composé de trois types d'informations : les informations sur la cible issues du lanceur, les informations sur l'état du logiciel tactique de la torpille et les informations issues des capteurs de la torpille. Une deuxième version de l'espace d'observation est définie avec des informations privilégiées. Pour la définition du problème d'apprentissage et les choix sur les algorithmes d'apprentissage, l'espace d'observation privilégiée est utilisée.

L'espace d'action est constitué est constitué de cinq actions : quatre actions haut niveau et une action "Rien Faire". Ces actions ont été élaborées grâce à des connaissances métiers afin de réduire la combinatoire et d'orienter le modèle vers des actions utiles et vraisemblables. Ces actions peuvent avoir un fort effet négatif sur les performances. Pour la définition du problème d'apprentissage et les choix sur les algorithmes d'apprentissage (dans le chapitre suivant), l'espace d'action utilisé est constitué seulement de l'action "Rien Faire" et de l'action 1.

Pour la fonction de récompense, trois types de termes ont été considérés : les termes terminaux, les termes de *shaping* et les termes comportementaux. Un seul terme terminal a été utilisé basé sur la probabilité de destruction. Pour les termes de *shaping*, plusieurs concepts de termes ainsi que plusieurs versions de ces concepts ont été expérimentés. Grâce à l'expertise métier, l'étude de cas limites et de leurs corrélations avec des métriques métiers et entre eux, nous avons retenu deux termes. Cependant, les apprentissages utilisant seulement les termes de *shaping* sont les plus efficaces, le terme terminal n'a donc pas été retenu. Finalement, un terme comportemental permet de moduler la fréquence d'action des modèles appris.

La fenêtre de contrôle du modèle est restreinte aux instants où le logiciel tactique est en contrôle de la torpille. En outre, le retrait des pannes de la torpille (rendant le problème insoluble par l'opérateur) lors des apprentissages conduit à des gains de performances.

Dans le chapitre suivant, nous présenterons les différents éléments de l'algorithme de DRL utilisé pour résoudre le MDP défini dans ce chapitre.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

6

Choix algorithmiques

Ce chapitre décrit les différents éléments de l'algorithme de DRL utilisé pour apprendre une politique. Tout d'abord, la base algorithmique est justifiée. Par la suite, l'architecture de parallélisation est présentée et étudiée. Finalement, une étude comparative sur l'utilisation de différents algorithmes de l'état de l'art a été réalisée.

Sommaire

6.1 Choix de la base algorithmique 88
6.2 Utilisation des ressources calculatoires 88
6.2.1 Choix de l'architecture de parallélisation 88
6.2.2 Comparaison aux *benchmarks classiques* 89
6.2.3 Méthodologie expérimentale 90
6.2.4 Étude de l'existence d'un biais 91
6.2.5 Existence d'un biais induit par un ralentissement dans une zone sur les données d'entraînement et sur la politique apprise. 92
6.2.6 Influence du nombre d'*actors* en parallèle sur le biais 93
6.2.7 Influence de l'exploration sur le biais 94
6.2.8 Influence de la position de la zone sur le biais 94
6.2.9 Conclusion sur l'étude d'un biais 96
6.3 Ajouts algorithmiques supplémentaires 96
6.3.1 *Prioritized Experience Replay* 97
6.3.2 Initialisation avec la politique de référence 97
6.3.3 *Noisy Net* 100
6.3.4 *Dueling Net* 102

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Afin de réduire les besoins en temps de calcul, l'ensemble de cette section utilise uniquement des apprentissages sur les scénarios SM. Ce choix est motivé par :

— des gains de performances plus significatifs sur les scénarios SM (voir chapitre 5) ;
— l'hypothèse que les scénarios BS sont assez similaires des scénarios SM et donc que les algorithmes se comportent de façon similaire sur les deux.

Comme mentionnée dans le chapitre 5, l'ensemble de cette section utilise un espace d'observation privilégiée ainsi qu'un espace d'action composé uniquement par les actions "Rien faire" et "Action 1".

6.1 Choix de la base algorithmique

Le problème d'apprentissage défini dans le chapitre 5 possède un espace d'action discret. Il est donc possible d'utiliser à la fois un algorithme de DRL utilisant une fonction de valeur ou bien un algorithme de type *Actor-Critic*.

Les algorithmes qui ont été considérés sont :

— PPO [45] car il est populaire, stable et qu'il a été utilisé sur des problèmes complexes [41] [36].
— *Double Deep Q-Learning* (DDQN) [49] car c'est un des algorithmes les plus étudiés de l'état de l'art avec de multiples possibilités d'améliorations (voir chapitre 2).

Lors des premières phases de développement du processus d'apprentissage, de multiples essais ont été faits avec plusieurs hyperparamètres pour les deux algorithmes. DDQN est le premier algorithme dont un ensemble d'hyperparamètres a convergé vers un comportement à peu près cohérent. DDQN a donc rapidement été choisi. En effet, au vu des temps de calculs nécessaires et de l'ensemble des autres éléments à améliorer et des résultats obtenus par DDQN, il a été abandonné d'essayer de trouver un ensemble d'hyperparamètres pour PPO ou d'essayer de faire converger un autre algorithme. En outre, DDQN possède un avantage majeur car c'est un algorithme *Off-Policy*, ce qui facilite l'intégration de connaissances extérieures.

Le problème d'apprentissage défini dans le chapitre 5 a un espace d'observation ne possédant pas de structures notables (contrairement à une image). En outre, pour des raisons de simplicité, l'aspect non-markovien du problème a été partiellement intégré à l'espace d'observation en ajoutant l'évolution des positions pour éviter les difficultés de stabilité liées aux réseaux récurrents. Tous les réseaux de neurones utilisés sont donc des réseaux totalement connectés.

6.2 Utilisation des ressources calculatoires

L'ensemble de cette section est issue de l'article "*Effect of a slowdown correlated to the current state of the environment on an asynchronous learning architecture*" présenté à la conférence *Reinforcement Learning Conference 2025* et publié dans le *Reinforcement Learning Journal* [2].

6.2.1 Choix de l'architecture de parallélisation

Le simulateur numérique propose une variété importante de scénarios comportant des conditions initiales diversifiées et des objets multiples qui ne sont pas tous présents à chaque fois. De plus, la complexité de l'environnement acoustique dépend de l'état courant de l'agent lors de la simulation car les acteurs ne sont pas forcément à portée de détection et peuvent réagir en fonction de la situation sur le théâtre opérationnel. Cela induit des temps de calcul très variables

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

pour chaque pas de temps simulé, dépendant à la fois des scénarios et de l'état courant de la simulation (voir figure 1.8).

Initialement, la première architecture de parallélisation utilisée pour l'algorithme DDQN était l'architecture synchrone. Mais, la haute variabilité des temps de calculs pour un pas de temps couplée au nombre d'environnements en parallèle (généralement 2 noeuds soit 512 environnements) cause des temps d'attente importants dus à la synchronisation entre les acteurs (voir figure 2.8). Par conséquent, l'efficacité d'utilisation des ressources calculatoires avec cette architecture est d'environ 10%, ce qui est très peu efficace pour la génération des transitions.

Nous avons donc choisi d'utiliser une architecture asynchrone (voir figure 2.9) afin de supprimer ces temps d'attentes. L'approche est inspirée de l'architecture *Actor/Learner* avec un seul *Learner* proposée par Espeholt et al. [62]. La figure 6.1 décrit la répartition des différents processus sur les ressources calculatoires disponibles. Ainsi, les *Actors* sont sur les différents noeuds de calculs afin de profiter du grand nombre de cœurs CPU et le *learner* est sur la machine locale afin de profiter de l'accélération GPU. Les communications inter-machines sont assurées par la librairie *ray*.

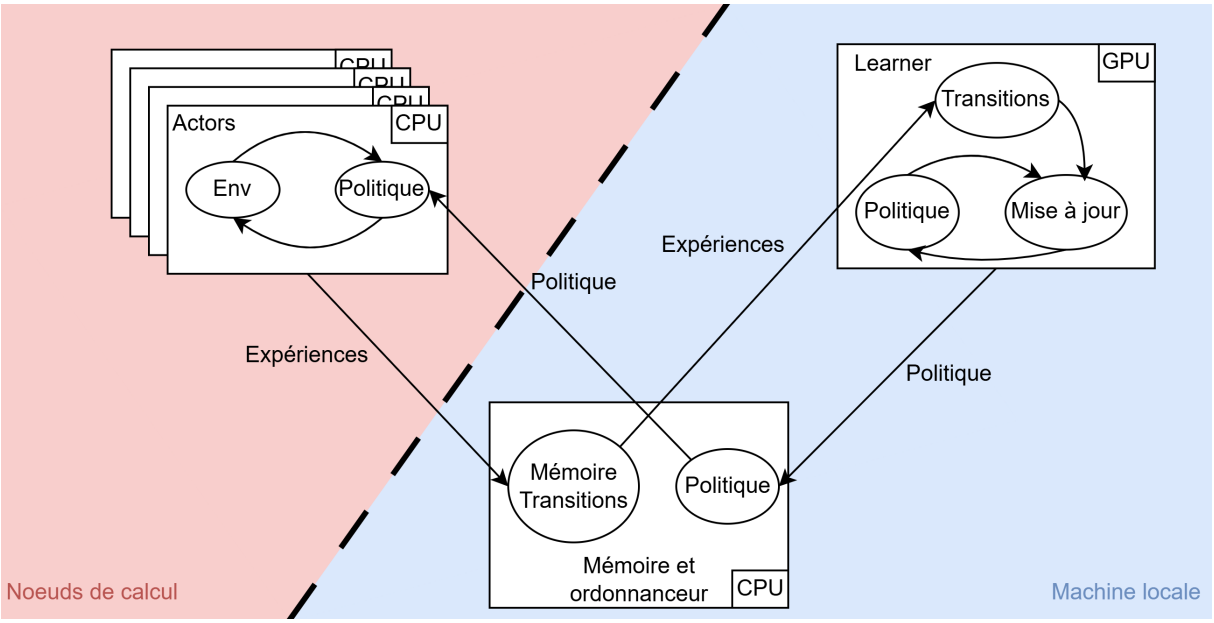


FIGURE 6.1 – Application de l'architecture asynchrone sur les ressources de calculs.

Cependant, l'utilisation d'une architecture asynchrone sur un environnement disposant de forts ralentissements corrélés à l'état courant de l'environnement nous a mené à la question suivante : est-il possible qu'un biais soit induit dans la politique apprise ? En effet, le découplage des acteurs pourrait favoriser les trajectoires dont les temps de calcul sont les plus rapides et donc biaiser la génération des données.

La suite de cette section propose une étude de cette problématique.

6.2.2 Comparaison aux *benchmarks classiques*

L'observation de ralentissements dépendant de l'état courant de l'environnement est classique dans les environnements de type jeu vidéo (à cause du nombre variable d'animation en fonction du contexte) ou bien d'autres simulateurs avec un nombre variable d'objets. Nous avons donc commencé par quantifier les ralentissements présents dans les *benchmarks* classiques du DRL.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Afin d'avoir des métriques de comparaisons entre les distributions des ralentissement observés dans la suite de l'article, nous avons défini les ratios inter quantile à $\alpha\%$ (noté $IQR_{\alpha\%}$) par le ratio entre le quantile $100-\frac{\alpha}{2}\%$ et le quantile $\frac{\alpha}{2}\%$. Cette valeur représente la variabilité maximale de temps de calcul des $\alpha\%$ pas de temps au centre la distribution.

Cette variabilité de temps de calcul n'est pratiquement pas présente dans les environnements classiques de DRL. En effet, comme le montre la figure 6.2, les variabilités de temps de calcul des environnements Mujoco [94] et Atari [43] sont très faibles. En outre, la majorité de la variabilité observée sur Atari peut être attribuée à un *frame skip* aléatoire [114], qui est indépendant de l'état courant de l'environnement. Ainsi, même s'il y a effectivement un temps de calcul corrélé à l'état interne de ces environnements, les ralentissements sont de plusieurs ordres de grandeur en dessous de ceux que nous observons sur notre simulateur (voir les points rouges sur la figure 6.2).

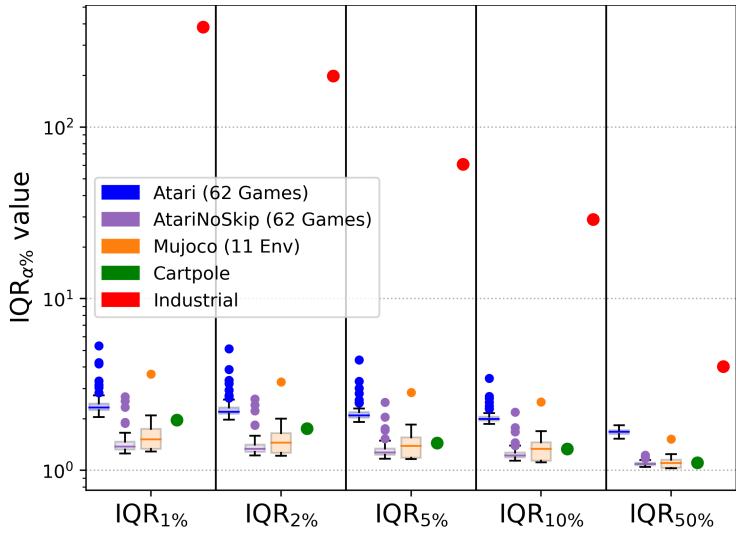


FIGURE 6.2 – Valeurs de plusieurs $IQR_{\alpha\%}$ pour les environnements Atari, Atari sans *frame skip*, Mujoco, Cartpole et le simulateur industriel.

La table 6.1 montre que les temps de calcul des environnements Atari et Mujoco sont plutôt faibles devant les temps d'inférences de la politique et de traitement de l'observation, rendant encore plus négligeable un potentiel ralentissement lié à l'état courant de l'environnement. Dans le cas du simulateur numérique, on peut voir que le temps de l'environnement est largement supérieur à celui de la politique et du traitement de l'observation.

Finalement, du point de vue de l'existence de ralentissements, les *benchmarks* classiques du DRL ne sont pas représentatifs de notre environnement industriel.

6.2.3 Méthodologie expérimentale

Afin d'étudier l'impact d'un tel biais, nous avons créé une version modifiée de l'environnement *Cartpole* [96]. C'est un environnement 1D où un chariot doit contrôler l'équilibre vertical d'un poteau qu'il porte en se déplaçant soit à gauche, soit à droite (voir figure 6.3). L'utilisation de cet

1. Toutes les mesures ont été faites en utilisant l'implémentation et les hyperparamètres par défaut par le github de TorchRL [107]. Ces mesures ont été faites en utilisant un seul environnement utilisant un seul coeur CPU sur un processeur 12th Generation Intel(R) Core(TM) i7-12700H. Pour l'environnement industriel, les mesures sont celles de la figure 4.6a.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

TABLE 6.1 – Ratios entre le temps médian pour l'inférence en utilisant une politique *fully connected* et le temps de traitement de l'observation avec le temps médian d'un pas de temps de l'environnement pour plusieurs environnements [1].

Environnements	Atari (Médiane sur 62 jeux)	Atari No Skip (Médiane sur 62 jeux)	Mujoco (Médiane sur 11 envs)	Cartpole	Environnement industriel
Ratio de temps pris entre la politique et l'environnement	3.36	8.17	4.66	6.62	1.91×10^{-3}
Ratio de temps pris entre le traitement de l'observation et l'environnement	4.70	1.14×10^{1}	5.81	5.37×10^{-4}	4.28×10^{-3}

environnement simple et rapide est nécessaire pour obtenir des résultats statistiquement significatifs. En effet, à cause de la forte stochasticité du DRL, il est difficile de discerner des tendances. En outre, utiliser un environnement avec une dynamique symétrique simplifie l'étude d'un biais potentiel. La facilité pour trouver une politique efficace rend également possible d'essayer de modifier fortement les paramètres tout en restant capable de converger.

L'objectif est donc d'étudier l'effet potentiel d'un ralentissement corrélé à l'observation sur le comportement appris avec un algorithme de DRL utilisant une architecture asynchrone. Cependant, comme montré dans la figure 6.2, il n'y a pratiquement aucun ralentissement sur cet environnement. Nous avons donc ajouté deux zones symétriques définies grâce à la position du chariot, comme illustré par la figure 6.3. Ces zones ont été utilisées pour ajouter des ralentissements artificiels en stoppant le processus pour une durée égale à un multiple du temps réel de calcul pour la transition totale (simulation de l'environnement, traitement de l'observation et choix de la politique). Nous définissons **le coefficient de ralentissement** d'une zone par la valeur par laquelle est multipliée le temps de calcul de la transition totale. Il est donc possible d'observer un biais potentiel en comparant le comportement de la politique dans les deux zones. En effet, comme les deux zones sont symétriques, il ne devrait pas y avoir de différence entre les deux zones.

Les détails techniques pour une implémentation efficace et non biaisée des ralentissements artificiels sont présentées dans l'annexe C.

6.2.4 Étude de l'existence d'un biais

Les valeurs de coefficients de ralentissements vont de 10^0 à 10^4, ce qui permet de couvrir les valeurs expérimentées dans notre environnement industriel. Sauf mention contraire, les hyperparamètres utilisés sont ceux décrits dans l'annexe D. Il est bon de noter que les performances de l'algorithme en termes de nombre de pas de temps sont très mauvaises. En effet, l'environnement est trop simple et rapide pour tirer profit de cette architecture, mais le but de cette étude est de pouvoir mettre en évidence la présence d'un biais induit par des ralentissements corrélés à

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

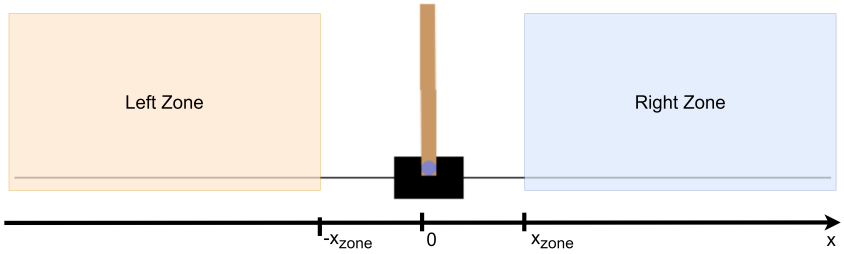


FIGURE 6.3 – Environnement Cartpole avec deux zones symétriques définies par la position du chariot, notée x, telles que la zone gauche est pour $x \in [-\infty; -x_{zone}]$ et la zone droite est pour $x \in [x_{zone}; +\infty]$.

l'observation.

Comme l'ensemble de cette section est basé uniquement sur des résultats empiriques, il était nécessaire d'affiner autant que possible les intervalles de confiance. Par conséquent, tous les points des graphiques sont issus d'une moyenne obtenue sur 1000 entraînements différents avec le même ensemble d'hyperparamètres et des germes aléatoires différents. L'intervalle de confiance est évalué à 95%. La section totalise donc 197 000 entraînements différents, d'où la nécessité d'utiliser un environnement aussi rapide que *CartPole*.

6.2.5 Existence d'un biais induit par un ralentissement dans une zone sur les données d'entraînement et sur la politique apprise.

Dans cette section, nous étudions l'influence du coefficient de ralentissement dans la zone gauche sur les données générées lors de l'apprentissage (voir figure 6.4) et sur la politique apprise (voir figure 6.5). La métrique utilisée est la présence moyenne du chariot par épisode dans chacune des zones. Comme l'environnement est symétrique, s'il n'y a pas de biais induit par les ralentissements, alors la présence mesurée doit être similaire pour les deux zones.

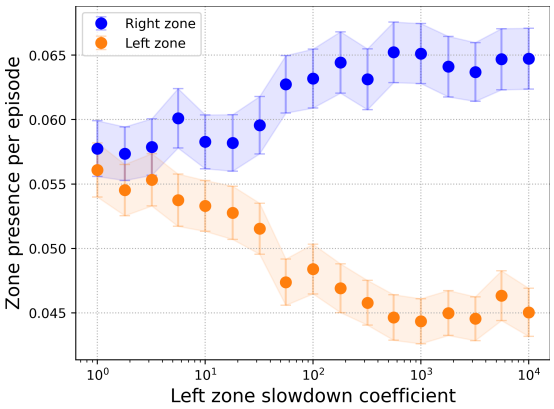


FIGURE 6.4 – Présence moyenne par épisode par zone pendant l'entraînement moyennée sur 1000 entraînements avec un intervalle de confiance à 95% en fonction du coefficient de ralentissement de la zone gauche.

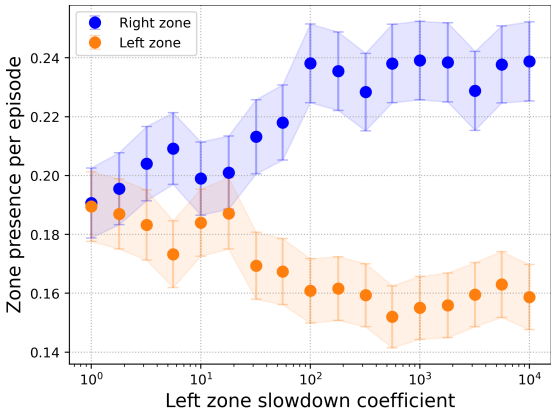


FIGURE 6.5 – Présence moyenne par épisode par zone sur 100 épisodes avec la politique finale moyennée sur 1000 entraînements avec un intervalle de confiance à 95% en fonction du coefficient de ralentissement de la zone gauche.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

Dans la figure 6.4, lorsque qu'il n'y a pas de ralentissement (i.e. un coefficient de ralentissement égale à 1), nous pouvons voir que la présence dans les deux zones est similaires, comme attendue. Cependant, on observe un biais qui croit plus le ralentissement est fort, qui se traduit par une augmentation de la présence dans la zone droite et une diminution dans la zone gauche. En outre, on peut voir que le biais semble se stabiliser pour une valeur de coefficient de ralentissement environ égale à 500.

Notre interprétation est que lors de l'apprentissage, un ralentissement corrélé à l'observation peut créer un biais dans la génération des transitions. Les *actors* créant une trajectoire impliquant des ralentissements vont être sous-représentés car ils vont être en compétition avec les autres qui auront esquivés ces zones de ralentissement.

Puisque la zone gauche (c.-à-d. la zone ralentie) est discrète, il arrive un point où une seule transition dans cette zone rend la trajectoire tellement lente qu'il est pratiquement impossible de prendre autant de temps dans le reste de l'environnement (c.-à-d. la zone centrale et la zone droite). Par conséquent, l'ordre dans lequel les différentes tâches des acteurs sont résolues devient le nombre de transitions passées dans la zone droite, peu importe le coefficient de ralentissement.

La figure 6.5 montre que la politique apprise conserve ce comportement biaisé avec une plus grande présence dans la zone droite que dans la zone gauche. Nous avons donc conclu qu'un ralentissement corrélé à l'observation peut induire un biais dans les données générées lors de l'apprentissage et dans la politique apprise lors de l'utilisation d'une architecture asynchrone.

6.2.6 Influence du nombre d'*actors* en parallèle sur le biais

Cette section étudie l'influence du nombre d'*actors* en parallèle sur le biais.

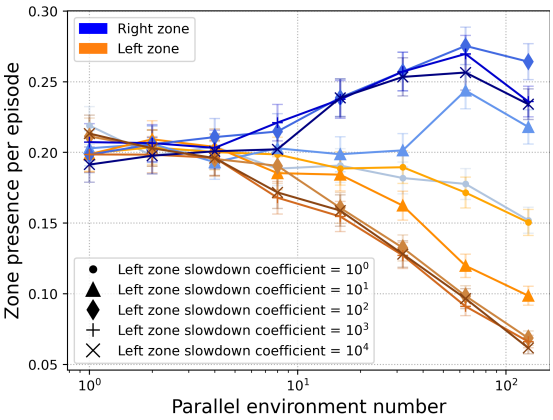


FIGURE 6.6 – Présence moyenne par épisode par zone sur 100 épisodes avec la politique finale moyennée sur 1000 entraînements avec un intervalle de confiance à 95%, en fonction du nombre d'*actors* en parallèle pour plusieurs valeurs de coefficients de ralentissement de la zone gauche.

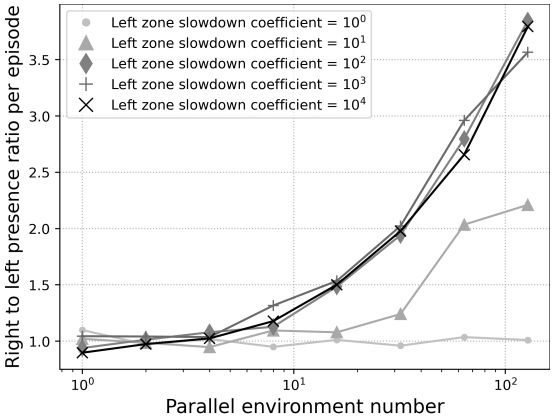


FIGURE 6.7 – Ratio moyen entre la présence dans la zone droite et la zone gauche sur 100 épisodes avec la politique finale moyennée sur 1000 entraînements avec un intervalle de confiance à 95%, en fonction du nombre d'*actors* en parallèle pour plusieurs valeurs de coefficient de ralentissement de la zone gauche.

La figure 6.7 montre le ratio des présences entre les zones de la figure 6.6. Tout d'abord, on constate que même sans ralentissement, le paramètre influe sur la politique apprise par une baisse de la fréquence dans les zones. En outre, elle montre que l'augmentation du nombre d'*actors*

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

amplifie le biais. En effet, on observe aucun biais induit par des ralentissements avec moins de 4 environnements en parallèle. Peu importe le nombre d'environnement en parallèle tant qu'il est supérieur à 4, le biais est amplifié par la parallélisation des environnements.

Nous pouvons donc conclure que l'effet du biais est amplifié par le nombre d'environnements en parallèle. Notre interprétation est que l'augmentation du nombre d'environnements en parallèle augmente le nombre d'*actors* en compétition et donc, cela amplifie les chances de trouver des trajectoires avec des temps de calcul très faible.

6.2.7 Influence de l'exploration sur le biais

Cette section étudie l'influence de l'exploration sur l'ampleur du biais. Nous avons donc modifié la valeur du pas de temps à laquelle la valeur de ε atteint sa valeur finale (pour l'exploration en utilisant la méthode ε-*greedy*, la valeur d'ε est réduite linéairement au cours de l'apprentissage).

De façon similaire à la figure 6.6, on peut voir sur la figure 6.8 que même sans ralentissement, le paramètre influe sur la politique apprise par une baisse de la fréquence dans les zones. En outre, la figure 6.9 montre une augmentation du biais quand l'exploration augmente.

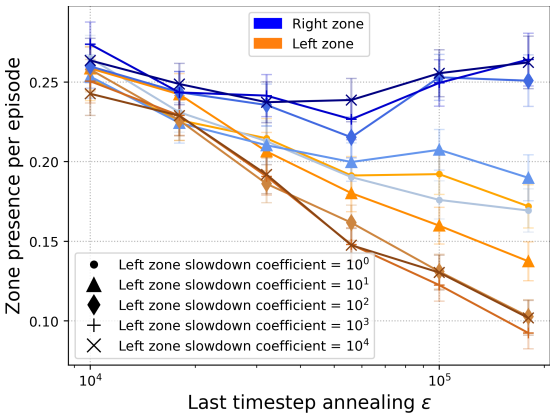


FIGURE 6.8 – Présence moyenne par épisode par zone sur 100 épisodes avec la politique finale moyennée sur 1000 entraînements avec un intervalle de confiance à 95% en fonction du pas de temps auquel ε atteint sa valeur finale pour plusieurs valeurs de coefficients de ralentissement de la zone gauche.

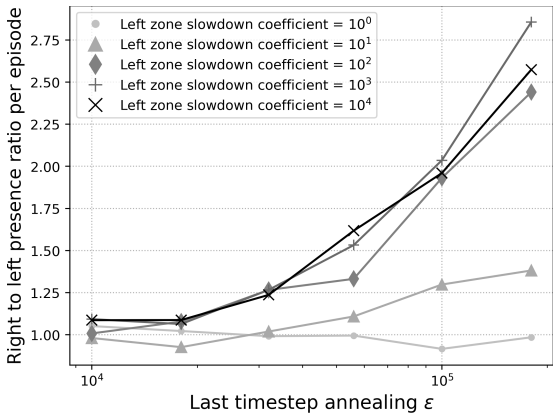


FIGURE 6.9 – Ratio moyen entre la présence dans la zone droite et la zone gauche sur 100 épisodes avec la politique finale moyennée sur 1000 entraînements avec un intervalle de confiance à 95% en fonction du pas de temps auquel ε atteint sa valeur finale pour plusieurs valeurs de coefficient de ralentissement de la zone gauche.

En conclusion, l'exploration a un effet sur l'ampleur du biais induit par un ralentissement corrélé à l'observation. Notre interprétation est qu'une augmentation de l'exploration augmente les probabilités de découvrir des trajectoires évitant la zone ralentie et réduit donc le nombre d'actions choisies purement grâce à la récompense.

6.2.8 Influence de la position de la zone sur le biais

Cette section évalue l'influence de la position de la zone de ralentissement sur le biais. La figure 6.10 montre que la présence dans la zone est fortement reliée à la taille de la zone. Afin d'évaluer et de comparer le biais tout en variant la valeur de x_{zone}, nous avons mesuré la présence

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

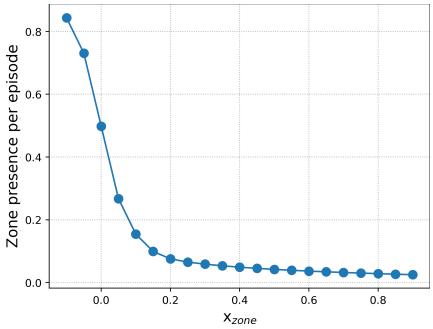


FIGURE 6.10 – Présence moyenne par épisode par zone pendant l'entraînement moyennée sur 1000 entraînements avec un intervalle de confiance à 95% en fonction de la taille de la zone.

dans des zones de taille fixe correspondant à $x_{zone} = 0$ (les deux zones correspondent donc exactement à la moitié gauche et droite).

Les figures 6.11 et 6.12 montrent que le biais induit par le ralentissement de la zone gauche n'est pas directement proportionnel à la taille de cette zone. En effet, on peut observer que pour les zones de ralentissement représentant une large partie de l'espace d'état (voir figure 6.10 entre -0.1 et 0.1), il n'y a pratiquement pas de biais dans la politique apprise. Cependant, le biais s'amplifie pour des valeurs de x_{zone} entre 0.2 et 0.4, même si la zone ralentie représente une partie plus faible des données générées lors de l'apprentissage. Finalement, après ce maximum, le biais réduit d'autant plus que la valeur de x_{zone} est grande.

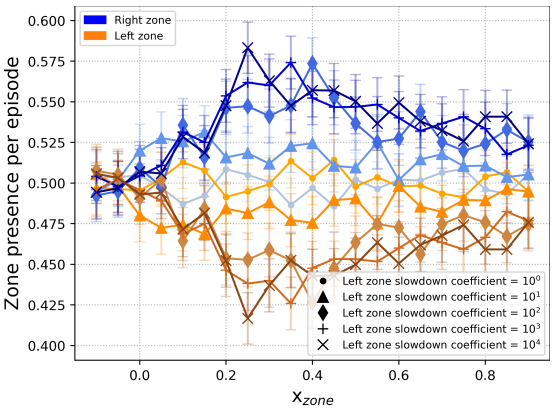


FIGURE 6.11 – Présence moyenne par épisode par zone sur 100 épisodes avec la politique finale moyennée sur 1000 entraînements avec un intervalle de confiance à 95% en fonction de la taille de la zone des ralentissements pour plusieurs valeurs de coefficients de ralentissement de la zone gauche.

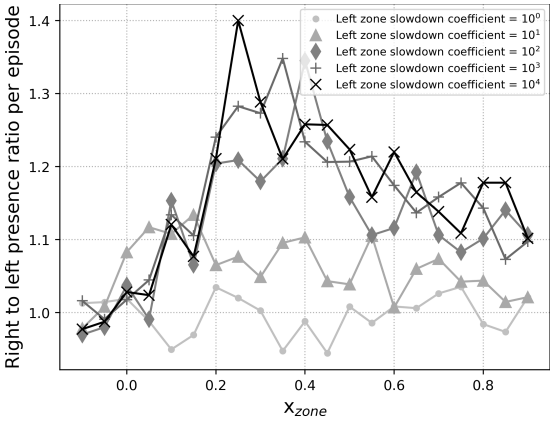


FIGURE 6.12 – Ratio moyen entre la présence dans la zone droite et la zone gauche sur 100 épisodes avec la politique finale moyennée sur 1000 entraînements avec un intervalle de confiance à 95% en fonction de la taille de la zone de ralentissement pour plusieurs valeurs de coefficient de ralentissement de la zone gauche.

Comme les épisodes sont initialisés aléatoirement avec $x \in [-0.05, 0.05]$, il est pratiquement impossible d'éviter une zone de ralentissement trop centrale à cause du fonctionnement de l'environnement. Cela résulte en une zone de ralentissement qui ne peut pas être évitée et qui n'induit donc pratiquement aucun biais sur la politique apprise. Cependant, un ralentissement dans une

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

zone légèrement explorée, où la politique est capable d'apprendre à l'éviter, peut créer un fort biais sur la politique apprise. Finalement, un ralentissement dans une zone quasiment inexplorée n'aura pratiquement aucun impact.

En conclusion, l'existence d'un biais dû à un ralentissement corrélé à l'observation est fortement liée au positionnement des ralentissements au sein de l'espace d'état et de la dynamique de l'environnement.

6.2.9 Conclusion sur l'étude d'un biais

L'utilisation du simulateur industriel en tant qu'environnement d'apprentissage conduit à une variabilité du temps de calcul pour chaque pas de temps qui est corrélée à l'état courant de l'environnement. Afin d'obtenir une utilisation efficace des ressources calculatoires, nous avons donc utilisé une architecture de parallélisation asynchrone. Cependant, les environnements de *benchmarks* classiques du DRL ne sont pas représentatifs de l'ordre de grandeurs des ralentissements et de cette corrélation avec l'état courant de l'environnement.

Nous avons donc commencé par prouver, sur une version modifiée de CartPole, qu'un ralentissement corrélé à l'observation peut induire un biais dans une politique apprise avec une architecture asynchrone.

Par la suite, nous avons étudié l'influence de plusieurs paramètres sur l'ampleur de ce biais. Nos résultats montrent une amplification du biais lorsque le nombre d'environnements en parallèle ou l'exploration est augmenté. En outre, le positionnement des ralentissements au sein de l'espace d'état est crucial pour la possibilité d'avoir un biais. Notre compréhension est que l'ampleur de ce phénomène est fortement liée à la probabilité de découvrir des trajectoires esquivant les ralentissements.

Suite à cette étude, les conclusions que nous avons tirées pour l'environnement industriel sont :

— il est difficile d'étudier l'existence d'un tel biais sur cet environnement car les temps de calcul sont beaucoup plus longs que sur CartPole et il n'existe pas de politique de référence non biaisée à laquelle se comparer ;
— il semble a priori complexe pour le modèle opérateur de trouver des politiques capables d'éviter des trajectoires puisque les actions opérateurs sont très hauts niveaux (c.-à-d. une influence sur les choix du logiciel tactique et pas sur le contrôle direct de la torpille) et que les ralentissements sont induits par des phénomènes difficilement évitables. Il semble donc assez peu probable que l'environnement industriel puisse souffrir d'un tel biais.

Par conséquent, suite à cette étude et les caractéristiques de notre environnement industriel, nous n'avons poursuivi cette étude.

6.3 Ajouts algorithmiques supplémentaires

Comme mentionnée dans la section 2.4, l'algorithme DQN possède plusieurs versions avec de multiples ajouts. L'algorithme sur lequel nous nous sommes basé est le *Double Deep Q-Learning* présenté par Hasselt et al. [49] avec une architecture de parallélisation asynchrone (voir section 6.2). L'ensemble des éléments composant cette base est donc :

— l'algorithme du Q-Learning ;
— un réseau de neurones pour approximer la fonction de valeur Q ;
— l'approche *n-step* pour l'évaluation de la fonction de coût ;
— le rejeu d'expérience (sans le PER) ;
— l'utilisation d'un réseau cible avec une mise à jour de type *soft update* ;

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

— le *double Q-Learning* avec l'utilisation du réseau cible comme deuxième réseau ;
— une architecture de parallélisation asynchrone.

Dans cette section, nous présentons l'impact d'ajouts algorithmiques supplémentaires sur cet ensemble. Il est important de noter que les ajouts algorithmiques choisis étaient déjà présents dans l'algorithme utilisé pour les apprentissages du chapitre 5. Ainsi, si l'ajout a été retenu, la sous-section présente l'impact du retrait de cette modification, si l'ajout n'a pas été retenu, la sous-section présente l'impact de l'ajout de cette modification.

Cette revue non homogène est une conséquence directe des difficultés d'appliquer le DRL mentionnées dans la section 5.1. En effet, les choix de la base algorithmique antidatent les derniers choix sur la définition du problème d'apprentissage présentée dans le chapitre 5.

6.3.1 *Prioritized Experience Replay*

Le *Prioritized Experience Replay* a été implémenté mais finalement pas retenu, cette section présente donc l'effet de son ajout.

Le *Prioritized Experience Replay* [51] est un ajout algorithmique utilisé dans plusieurs algorithmes de DRL [61] [47]. Il est notamment mentionné par Horgan et al.[61] et Hesselt et al. [47] comme central pour la performance de l'apprentissage.

Cependant, dans notre cas, tous les essais d'ajout du PER montrent une instabilité conduisant à une divergence de l'apprentissage. Il est possible que cette instabilité soit due à un mauvais choix d'hyperparamètres mais aucune combinaison n'a été trouvée dans le temps alloué pour cette modification algorithmique.

En outre, bien qu'une architecture asynchrone soit utilisée, il y a en réalité assez peu de transitions générées par seconde. En effet, comme l'environnement est beaucoup plus lent que les environnements de *benchmarks*, la quantité de données générées est relativement faible. Par exemple, pour 256 *actors* en parallèle, Horgan et al. [61] génère environ 9,000 transitions par seconde alors que pour le même nombre d'*actors*, nous générons environ 70 transitions par seconde (pour le scénario SM). Par conséquent, le gain de prioriser les expériences tirées est donc plus faible car la quantité de données générées est plus faible.

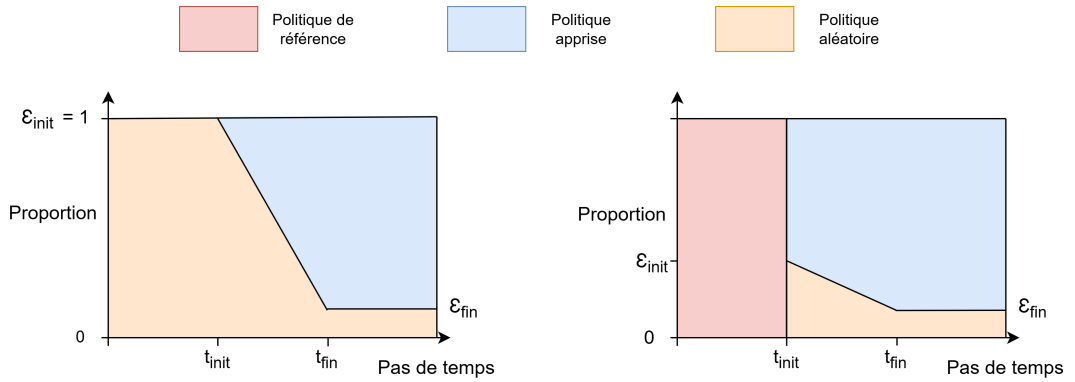
6.3.2 Initialisation avec la politique de référence

L'initialisation avec la politique de référence a été retenue, cette section présente donc l'effet de son retrait.

Lors de l'utilisation de l'algorithme *ε-greedy* pour l'exploration, la proportion allouée à l'exploration change dynamiquement au cours de l'apprentissage. La figure 6.13b illustre cette évolution qui se divise en trois phases :

— une phase d'initialisation : entre le début de l'entraînement et t_{init}, la politique utilisée garde un taux constant d'exploration aléatoire ε_{init} (généralement égal à 1). Cela permet d'initialiser le *replay buffer* avec des expériences avant de commencer à mettre à jour la politique pour avoir un minimum de diversité dans les expériences utilisées ;
— une phase de décroissance : entre t_{init} et t_{fin}, la proportion d'exploration aléatoire ε décroit linéairement de ε_{init} à ε_{fin}. Cela permet de passer dynamiquement d'une politique avec seulement de l'exploration à une politique avec une forte exploitation et un peu d'exploration.
— une phase d'exploration faible : de t_{fin} à la fin de l'apprentissage, la proportion d'exploration aléatoire reste constante. Ainsi, la politique peut profiter d'une exploration faible afin de capitaliser sur la politique déjà apprise.

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC



(b) Exploration avec l'approche ε-*greedy*

(c) Exploration avec une initialisation avec la référence

FIGURE 6.13 – Schéma de l'évolution des proportions des différentes politiques utilisées au cours d'un entraînement pour différentes méthodes d'exploration.

Suite aux spécificités de l'environnement, nous avons créé une modification de l'exploration tirant profit qu'il y ait une politique de référence simple et que la politique apprise doit faire très peu d'action. La figure 6.13c montre les proportions des différentes politiques dans les trois phases :

- une phase d'initialisation : entre le début de l'entraînement et t_{init}, la politique utilise uniquement l'action "Rien Faire". Cela permet d'initialiser le *replay buffer* avec des expériences issues de la politique de référence avec donc un comportement déjà très efficace ;
- une phase de décroissance : entre t_{init} et t_{fin}, une exploration aléatoire où ε décroit linéairement de ε_{init} à ε_{fin}. La politique de référence n'est plus forcée et de l'exploration aléatoire est ajoutée. Cependant, la valeur de ε_{init} est assez faible (égal à 0.1), impliquant une proportion d'exploration plus faible que pour l'approche ε-*greedy*. Comme les actions aléatoires ont un fort impact négatif sur les performances (voir figure 5.2), une exploration plus faible permet d'utiliser plus rapidement une politique qui fait très peu d'action ;
- une phase d'exploration faible : de t_{fin} à la fin de l'apprentissage, la proportion d'exploration aléatoire reste constante. De façon similaire à l'approche ε-*greedy*, la politique peut profiter d'une exploration faible afin de capitaliser sur la politique déjà apprise.

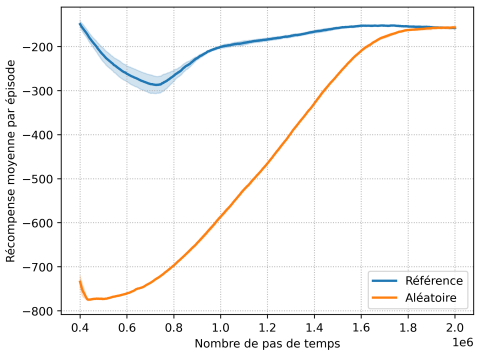


FIGURE 6.14 – Récompense moyenne par épisode au cours de l'apprentissage (sur une fenêtre de 400 000 pas de temps) avec une initialisation du *replay buffer* avec une politique aléatoire ou une politique de référence moyennée sur 5 apprentissages et l'écart-type.

Corporate sensitivity
PUBLIC

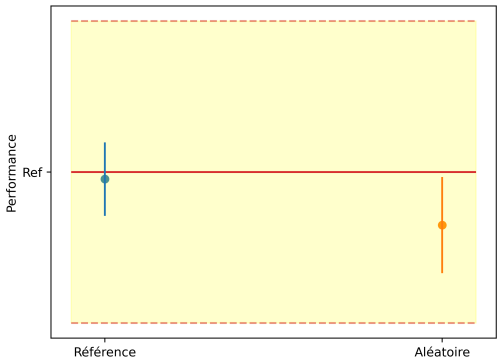
© Propriété Naval Group SA, (2026) tous droits réservés.

La figure 6.14 montre la récompense moyenne obtenue au cours de l'apprentissage avec les deux approches d'exploration.

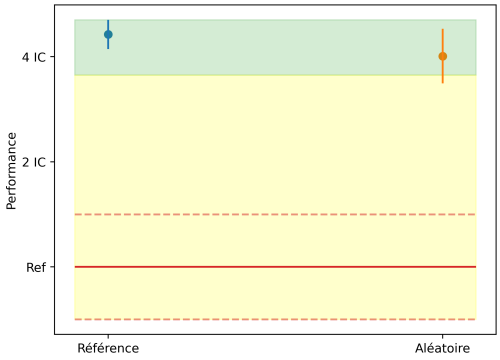
Pour l'apprentissage utilisant une politique aléatoire pour l'initialisation, on observe une valeur de récompense très basse au début. Cela est dû à la politique majoritairement aléatoire qui rend difficile la réussite d'un tir (voir figure 5.2). On observe ensuite une augmentation continue de la valeur moyenne de récompense au fur et à mesure que la part d'exploration réduit et que la politique converge.

Pour l'apprentissage utilisant la politique de référence pour l'initialisation, on observe une valeur de récompense très haute au début. Cela est dû à la politique de référence qui est efficace et que les premiers épisodes qui se finissent sont des tirs réussis car ils ont été rapides (en temps simulé) et/ou issus de SM$\bar{R}$. La chute de récompense correspond à l'ajout de l'exploration aléatoire dans la phase de décroissance et à l'obtention des premiers épisodes correspondant à des tirs en échecs (donc avec un temps simulé plus important). Finalement, on observe une augmentation puis une convergence de la valeur de récompense.

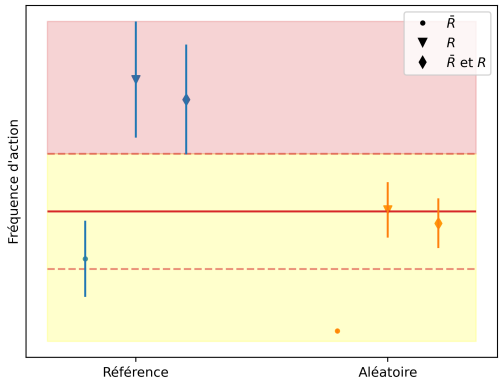
La différence la plus notable entre les deux approches est la valeur de récompense au début de l'apprentissage. Cela permet à l'approche utilisant la politique de référence pour l'initialisation de converger plus vite. Cependant, on observe qu'à la fin de l'apprentissage, il n'y a pas de réels gains entre les deux approches.



(a) Performances sur SM$\bar{R}$



(b) Performances sur SMR



(c) Fréquence d'action sur SM

FIGURE 6.15 – Évaluation des modèles avec l'initialisation aléatoire ou utilisant la politique de référence issus des entraînements présentés dans la figure 6.14 sur les scénarios SM.

Les figures 6.15a et 6.15b montrent que l'approche utilisant l'initialisation avec la politique

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

de référence induit des gains de performance sur les ensembles $\bar{R}$ et R mais qui sont non statistiquement significatifs. En revanche, l'initialisation avec la politique aléatoire conduit à des politiques dont la fréquence d'action est plus faible et dans la zone acceptable (voir figure 6.15c).

L'initialisation avec la politique de référence a été retenue car au moment où elle a été essayée, les hyperparamètres et la définition du problème d'apprentissage étaient moins efficaces. Le gain de temps pour faire converger la politique était alors non négligeable et conduisait à une réelle différence de performance dans les politiques apprises. Il semble cependant, que ce gain de performance a un effet néfaste sur la fréquence d'action.

6.3.3 *Noisy Net*

Le *Noisy Net* [52] n'a pas été retenu, cette section présente donc l'effet de son ajout. Le *Noisy Net* est un mécanisme ajustant automatiquement la proportion d'exploration au cours de l'apprentissage. La comparaison se fait donc entre une version avec l'approche ε-*greedy* et une version avec le *Noisy Net*. Nous avons utilisé l'initialisation du *replay buffer* par la politique de référence dans les deux cas.

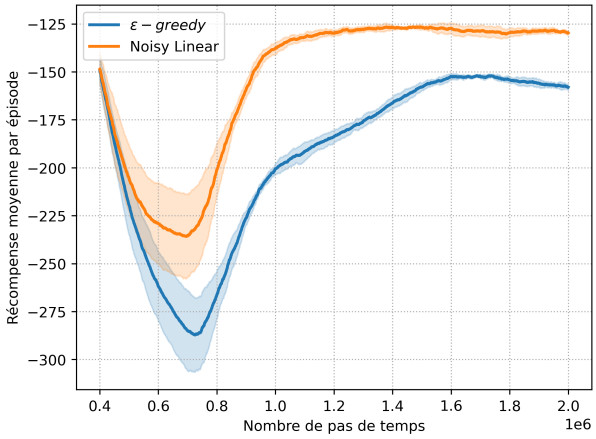


FIGURE 6.16 – Récompense moyenne par épisode au cours de l'apprentissage (sur une fenêtre de 400 000 pas de temps) avec une exploration utilisant la méthode ε-*greedy* ou la méthode *Noisy Net* [52] moyennée sur 5 apprentissages et l'écart-type.

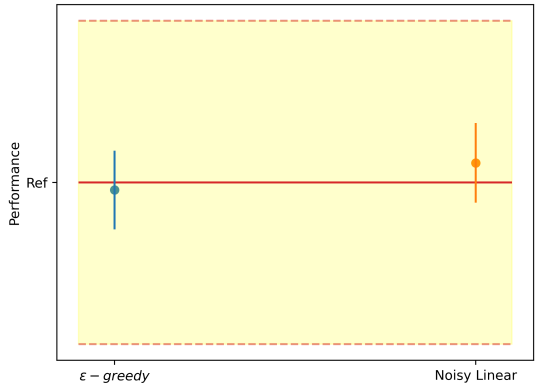
La figure 6.16 montre la récompense obtenue au cours de l'apprentissage pour ces deux approches d'exploration. On observe que l'utilisation d'un *Noisy Net* permet de converger à la fois plus rapidement et vers une valeur de récompense supérieure.

La différence de vitesse de convergence entre les deux approches peut s'expliquer par deux phénomènes : l'algorithme d'exploration est plus efficace sur ce problème et/ou il est mieux paramétré. En effet, l'approche ε-*greedy* nécessite 3 hyperparamètres (t_{fin}, ε_{init}, ε_{fin}, voir figure 6.13c) alors que le *Noisy Net* n'en nécessite aucun et ajuste automatiquement la quantité d'exploration.

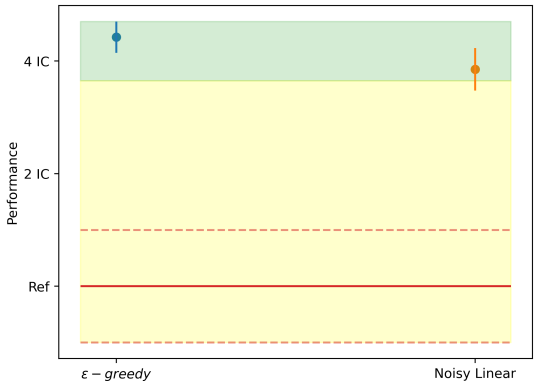
La différence de valeur de récompense finale après convergence peut s'expliquer par deux phénomènes : une meilleure performance finale et/ou la baisse de performance due à la part d'action aléatoire minimale (la valeur d'ε_{fin}). En effet, pour rappel,la figure 5.2 montre une chute importante de la performance même avec une très faible valeur d'action aléatoire.

La figure 6.17a montre que l'utilisation d'un *Noisy Net* conduit à un gain de performance non significatif alors que la figure 6.17b montre une perte de performance significative. La figure 6.17c

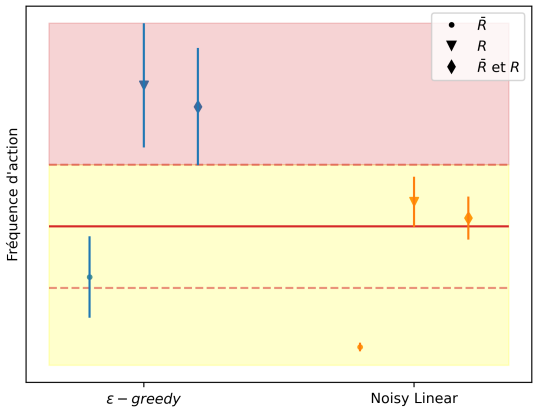
Corporate sensitivity
PUBLIC
© Propriété Naval Group SA, (2026) tous droits réservés.



(a) Performances sur SM$\bar{R}$



(b) Performances sur SMR



(c) Fréquence d'action sur SM

FIGURE 6.17 – Évaluation des modèles avec et sans le *Noisy Net* issus des entraînements présentés dans la figure 6.16 sur les scénarios SM.

indique que l'utilisation du *Noisy Net* mène à des politiques dont la fréquence d'action est plus faible, ce qui leur permet d'être dans la zone acceptable.

Enfin, la figure 6.18 montre que l'utilisation d'un *Noisy Net* augmente fortement la quantité de RAM utilisée. Cette augmentation dans notre cas crée des forts ralentissements voir des dépassements des limites de RAM conduisant à l'arrêt des nœuds de calcul. Cette augmentation est induite par le mécanisme de bruit intégré aux poids du réseau de neurones. Le retrait de ce bruit lors de l'évaluation conduit à une inefficacité totale de la politique. Même une forte réduction de la taille du réseau ne permettait pas de limiter suffisamment l'utilisation de la RAM pour empêcher le dépassement de la capacité des nœuds de calcul.

L'approche *Noisy Net* est donc plus simple d'utilisation et donne des résultats avec une fréquence d'action plus acceptable que l'exploration avec ε-*greedy* pour une légère perte de performance. Cependant, nous n'avons pas trouvé de solution rapide pour gérer le surplus de RAM nécessaire et nous n'avons donc pas retenu cet ajout algorithmique.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

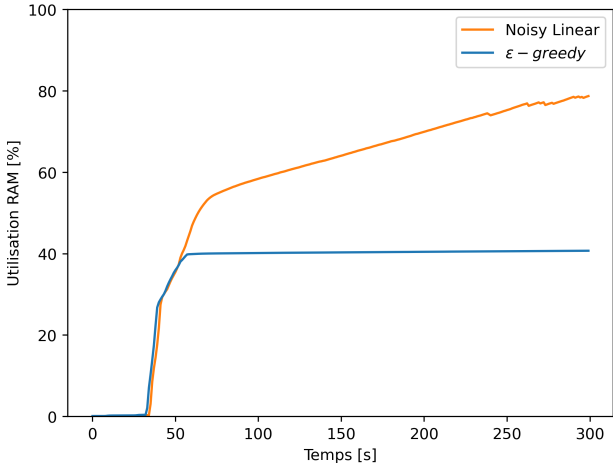


FIGURE 6.18 – Utilisation de la RAM par les processus Python du *wrapper* pour l'évaluation de SM sur un noeud de calcul avec 256 environnements en parallèle pour une politique utilisant un MLP (entraîné par l'approche ε-*greedy*) ou un *Noisy Network* en fonction du temps.

6.3.4 *Dueling Net*

Le *Dueling Net* [50] a été retenu, cette section présente donc l'effet de son retrait. C'est une modification du réseau permettant d'estimer la fonction de valeur état-action Q en estimant séparément la fonction d'avantage A et la fonction de valeur état V (voir section 2.2.1). La comparaison se fait donc avec un réseau estimant directement la fonction Q.

La figure 6.19 compare la récompense moyenne par épisode au cours de l'apprentissage. Elle ne montre aucune différence notable entre les deux approches lors de l'entraînement.

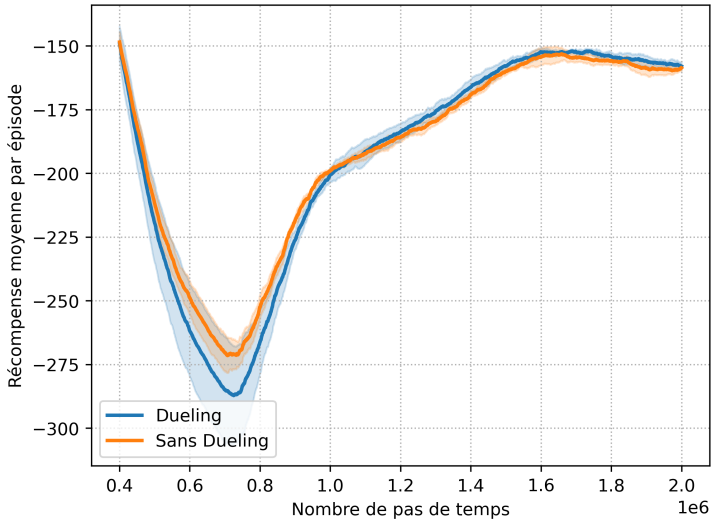


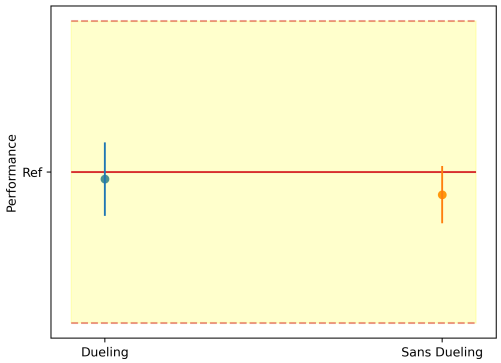
FIGURE 6.19 – Récompense moyenne par épisode au cours de l'apprentissage (sur une fenêtre de 400 000 pas de temps) avec et sans le *Dueling Net* [52] moyennée sur 5 apprentissages et l'écart-type.

Les figures 6.20a et 6.20b ne montrent aucune différence de performance notable avec et sans le *Dueling Net*. Cependant, la figure 6.20c montre que l'utilisation du *Dueling Net* conduit à une

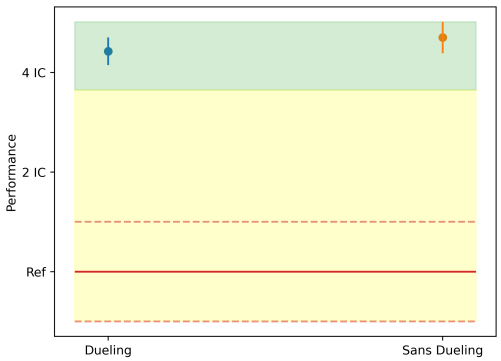
Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

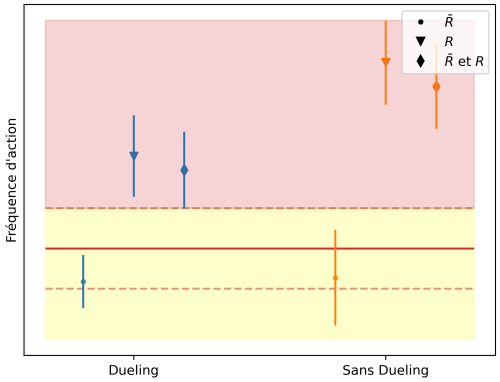
baisse significative du nombre d'action sur SMR.



(a) Performances sur SM$\bar{R}$



(b) Performances sur SMR



(c) Fréquence d'action sur SM

FIGURE 6.20 – Évaluation des modèles avec et sans le *Dueling Net* issus des entraînements présentés dans la figure 6.19 sur les scénarios SM.

Nous interprétons cette baisse comme une conséquence logique du *Dueling Net* qui est particulièrement efficace quand il existe des états où aucune des actions n'a d'effet [50]. Cela implique ici une meilleure compréhension quand une action a un réel effet positif et donc une baisse de la fréquence d'action tout en gardant la même performance.

Le *Dueling Net* est donc bénéfique et permet de réduire fortement la fréquence d'action du modèle appris.

Conclusion

Nous avons choisi l'algorithme *Dueling Double Q-Learning* comme base pour l'algorithme de DRL utilisé sur notre environnement d'apprentissage. Ce choix a été motivé par plusieurs raisons. Tout d'abord, cet algorithme est fortement étudié dans l'état de l'art, il correspond aux caractéristiques de notre environnement et c'est le premier algorithme dont on a trouvé un ensemble de paramètres permettant de converger vers une politique cohérente.

Les forts ralentissements de l'environnement et le nombre d'environnement en parallèle nous ont conduit à utiliser une architecture de parallélisation asynchrone afin d'améliorer fortement l'efficacité de l'utilisation des ressources de calcul. Nous avons alors étudié l'impact d'un ralen-

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

tissement corrélé à l'observation sur une architecture asynchrone. Cette étude prouve, sur une version modifiée de CartPole, l'existence potentielle d'un biais dans la politique apprise. En outre, cette étude nous a permis de comprendre certains leviers amplifiant ce phénomène. À la lumière de ces résultats, nous avons conjecturé l'absence de biais sur notre environnement industriel.

Finalement, nous avons étudié l'ajout de certains algorithmes afin d'améliorer la convergence des apprentissages. Nous n'avons pas retenu le *Prioritized Experience Replay* car il déstabilise l'apprentissage. L'initialisation du *replay buffer* avec la politique de référence permet quant à lui d'accélérer la convergence, il a donc été ajouté. Le *Noisy Net* améliore la convergence et réduit la fréquence d'action, mais il conduit à une utilisation de RAM supplémentaire trop importante, il n'a donc pas été retenu. Enfin, le *Dueling Net* permet d'améliorer le modèle en réduisant la fréquence d'action sans contrepartie, il a donc été retenu.

Ce chapitre présente les choix conduisant à l'algorithme de DRL utilisée pour résoudre le MDP correspondant à l'environnement industriel. Suite à ces choix, le prochain chapitre présente l'apprentissage d'un modèle capable d'utiliser toutes les actions à partir de l'espace d'observation non privilégiée. En outre, plusieurs modifications algorithmiques sont présentées dont le but est de tirer profit des structures connues de l'environnement.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

7

Utilisation des spécificités de l'environnement

Ce chapitre montre des résultats d'algorithmes tirant profit de spécificités de l'environnement industriel. Nous décrivons tout d'abord les possibilités de couplage avec une machine à états. Nous étudions alors chacune des actions de façon séparée. L'impact de l'*action masking* est présenté pour les actions qui peuvent en bénéficier. Finalement, les utilisations de l'espace d'action complet puis de l'espace d'observation non privilégié sont étudiées.

Sommaire

7.1 Utilisations possibles de la structure de machines à états 106
7.1.1 Définitions 106
7.1.2 Couplages possibles avec une machine à états 106
7.1.3 Reconsidération de l'état de l'art sur le couplage MAE et DRL 107
7.1.4 Nouvelles possibilités de couplages identifiées 108
7.1.5 Conclusion partielle sur le couplage entre une MAE et un algorithme de DRL 110
7.2 Apprentissages séparés des actions 111
7.2.1 Analyse sur le scénario bâtiment de surface 111
7.2.2 Analyse sur le scénario sous-marin 114
7.3 Ajout de l'*action masking* 117
7.4 Apprentissage commun 117
7.4.1 Apprentissage de toutes les actions 118
7.4.2 Utilisation de QDagger à partir des modèles à une action 120
7.5 Information privilégiée ou non 124

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

7.1 Utilisations possibles de la structure de machines à états

Il existe deux machines à états (MAE) sur le problème industriel : la MAE du logiciel tactique (recherche, poursuite, attaque) et la MAE d'un modèle opérateur (voir section 1.4.2). Cette section étudie la possibilité de coupler un algorithme d'apprentissage par renforcement avec une MAE. L'ensemble de cette section est issu de l'article "*Couplage entre un apprentissage par renforcement profond et une machine à états : approche théorique*" [1] que nous avons présenté et publié à la conférence *Conference on Artificial Intelligence for Defense 2024*.

7.1.1 Définitions

Dans la suite, nous nous plaçons dans le cadre d'une machine à états fini (MAE) définie par le tuple $\langle \mathcal{Q}, \mathcal{U}, \delta, q_0 \rangle$ par un ensemble d'états possibles, $\mathcal{Q}$, un ensemble d'observable $\mathcal{U}$, une fonction de transition $\delta : \mathcal{Q} \times \mathcal{U} \to \mathcal{Q}$ ainsi qu'un état initial $q_0 \in \mathcal{Q}$.

Nous définissons une machine à états politique (MAE-π) par une machine de Mealy définie par un tuple $\langle \mathcal{Q}, \mathcal{U}, \delta, q_0, \mathcal{A}, \pi \rangle$ avec $\langle \mathcal{Q}, \mathcal{U}, \delta, q_0 \rangle$ définissant la MAE, $\mathcal{A}$ l'ensemble des actions possibles et $\pi : \mathcal{Q} \times \mathcal{U} \to \mathcal{A}$, une fonction de sortie, qui est donc une politique, dépendant de l'état courant de la MAE et de l'observation. Par exemple, le modèle opérateur utilisant une MAE (voir section 1.4.2) est une MAE-π définie avec $U = \Omega$.

7.1.2 Couplages possibles avec une machine à états

Pour cette section, nous décrirons un algorithme de DRL comme étant la succession de deux phases. Une première phase exploratoire de récolte d'expériences et une seconde phase d'apprentissage utilisant ces expériences afin de mettre à jour la politique en vue de créer une nouvelle politique exploratoire pour une nouvelle phase exploratoire. Cette représentation est illustrée par la figure 7.1. Avec ce découpage, nous étudions plusieurs possibilités de couplages avec une MAE.

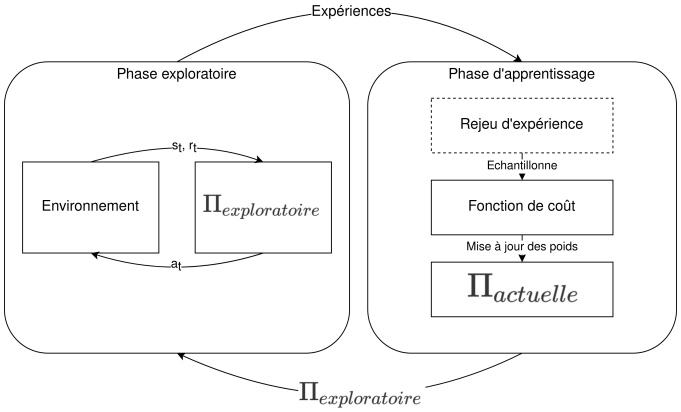


FIGURE 7.1 – Algorithme d'apprentissage par renforcement découpé en une phase exploratoire et une phase d'apprentissage.

Définition du couplage de la MAE au sein de l'algorithme de DRL

Tout d'abord, il faut définir si la fonction de transition de la MAE est utilisée lors de la phase d'exploration ou bien lors de la phase d'apprentissage. En effet, les deux possibilités ne donnent

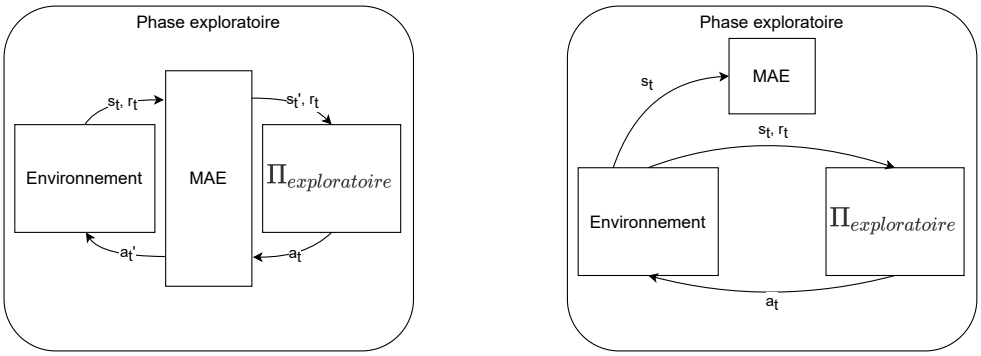
Corporate sensitivity
PUBLIC
© Propriété Naval Group SA, (2026) tous droits réservés.

pas accès aux mêmes informations.

En utilisant la MAE lors de phase exploratoire, la fonction de transition est appelée à chaque interaction avec l'environnement. Nous définissons alors deux types de placements :

— le placement en **série**, c'est-à-dire qu'elle permet de définir l'espace d'observation et/ou l'espace d'action de la politique du MDP résolu par l'algorithme de DRL comme décrit par la figure 7.2a.

— le placement en **parallèle**, c'est-à-dire qu'elle utilise le même vecteur d'observation que la politique comme décrit par la figure 7.2b.



(a) MAE placée en série

(b) MAE placée en parallèle

FIGURE 7.2 – Placements possibles d'une MAE en phase exploratoire.

En utilisant la MAE lors de phase d'apprentissage, les informations à sa disposition sont l'ensemble des transitions expérimentées lors des phases exploratoires précédentes ainsi que les métriques d'apprentissage (par exemple l'erreur TD).

Modifications algorithmiques permises par ces différents usages de la MAE

Pour les modifications en phase exploratoire, celles qui modifient l'espace d'observation de la politique et/ou le choix des actions doivent être gardées pour le modèle final car elles modifient le MDP. Dans le cas d'une MAE utilisée dans la phase exploratoire, le placement en série implique naturellement ce type de modifications. Pour une MAE placée en parallèle, cela dépend de si elle est utilisée pour augmenter l'espace d'observation et/ou pour changer activement l'action choisie.

Il est important de noter qu'une MAE utilisant $\mathcal{S}$ (qui utilise donc des informations privilégiées) ne peut pas être utilisée pour modifier le MDP car tout modèle appris utilisera alors des informations privilégiées.

Pour les modifications en phase d'apprentissage, l'utilisation de la MAE a une influence sur les modifications de la politique lors de la phase exploratoire. Contrairement aux modifications en phase exploratoire, cette approche n'induit pas de modifications directes sur le MDP et donc sur les espaces d'observation et d'action du modèle.

7.1.3 Reconsidération de l'état de l'art sur le couplage MAE et DRL

Le tableau 7.1 repositionne différentes contributions de l'état de l'art grâce aux définitions de la section 7.1.1.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Fonctionnement et modification en phase exploratoire

Plusieurs approches placent la MAE en série avec une architecture utilisant un macrocontrôleur et un microcontrôleur afin de cibler l'apprentissage sur une partie considérée comme complexe tout en gardant un contrôle et une compréhension sur le modèle final. Par exemple, dans [83], qui aborde l'insertion d'une voiture dans une voie, la fonction de politique est apprise pour un seul état de la MAE et garde une définition symbolique pour les fonctions de transition et pour la politique appliquée dans les autres états. Dans [82], qui propose un algorithme de contrôle pour de la marche bipède, la politique apprend directement en utilisant l'état du système mais son espace d'action agit sur une MAE construite à partir d'un contrôleur Proportionnel Dérivé qui transmet des actions plus bas niveau aux actionneurs.

L'utilisation d'une MAE en parallèle est proposée dans [84] qui introduit le concept de *Reward Machine*. Tout d'abord, la fonction de récompense est une *Reward Machine*, c'est-à-dire une fonction de récompense qui est une MAE, et l'observation est augmentée par l'état courant de cette MAE. Ainsi, elle est utilisée en parallèle des interactions de la politique afin d'obtenir la récompense. Ces travaux montrent que cette utilisation peut rendre markovien un problème initialement non markovien grâce aux informations fournies par l'état courant de la MAE.

Fonctionnement en phase exploratoire et modification de la phase d'apprentissage

Ce type d'approche utilise une MAE lors de la phase d'exploration et tire profit des informations données par la MAE sur les transitions afin d'améliorer l'apprentissage.

Les algorithmes les plus notables sont le *Q-Learning for Reward Machines* QRM et le *Hierarchical Learning for Reward Machines* (HRM) issus de [84] qui profitent tous les deux de la structure de MAE pour augmenter artificiellement le nombre de transitions rencontrées en rejouant chaque transition avec tous les états possibles de la MAE. De plus, le HRM maintient une politique différente pour chacun des états. Ces deux approches impliquent donc une modification du MDP en augmentant l'espace d'observation avec l'état courant de la MAE.

Plusieurs travaux se sont penchés sur le fait d'apprendre automatiquement une MAE ou un automate dans ce contexte [124] [125] mais ces approches perdent alors l'intérêt d'utiliser une MAE créée par un humain qui est plus facilement interprétable.

Fonctionnement en phase d'apprentissage et modification de la phase exploratoire

Les méthodes de l'état de l'art se rapprochant le plus de ce concept sont issues du *Curriculum Learning* mais n'utilisent pas la structure d'une MAE, mais plutôt celle d'un graphe [91]. En effet, plusieurs approches mettent en place un ensemble de tâches et font évoluer la tâche courante à partir de laquelle les interactions sont générées en phase exploratoire en fonction de l'état d'avancement de la politique dans son processus d'apprentissage.

7.1.4 Nouvelles possibilités de couplages identifiées

Métriques d'évaluation de l'état courant de la politique

Plusieurs approches se basent sur des métriques de l'état courant de la politique afin de pouvoir mettre en place des rétroactions dans le processus d'apprentissage, par exemple, l'utilisation de l'entropie pour [45]. Ainsi, ces métriques permettent d'ajuster le mécanisme d'apprentissage en prenant en compte l'état courant de la politique afin d'ajuster la rétroaction.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

Placement			Modification	Article
Exploratoire	Parallèle	Ω	Exploratoire	[84] [125]
			Apprentissage	[84] [125]
		$\mathcal{S}$	Exploratoire	
			Apprentissage	
	Série	Ω	Exploratoire	[82] [83]
			Apprentissage	
Apprentissage			Exploratoire	[91]
			Apprentissage	

TABLE 7.1 – Positionnement des méthodes de l'état de l'art présentées au regard de l'approche proposée.

L'utilisation d'une MAE en phase exploratoire placée en parallèle peut permettre d'attribuer à chacune des transitions un état issu de la fonction de transition de la MAE. Cet ajout d'information peut permettre de créer de nouvelles métriques permettant d'évaluer l'état courant de la politique.

Un premier type de métrique est défini par la proportion de chacun des états rencontrés au cours d'un épisode. Cela permet d'avoir des éléments de compréhension sur l'état courant de la politique au cours de l'apprentissage en observant des modifications de proportions entre les différents états au cours de l'apprentissage de la politique.

Un second type de métrique peut se baser sur des métriques déjà existantes (comme l'entropie ou l'erreur TD), mais en les évaluant uniquement par état afin d'avoir une évaluation de la politique pour chacun des états.

Utilisation d'une MAE-π

Dans un cas où une MAE-π existe déjà pour répondre au problème de décision séquentielle cible, comme dans notre cas industriel, il peut être intéressant d'étudier la capacité de tirer profit de la structure de MAE tout en ayant accès à une politique pour chacun de ces états. Cependant, il est bon de noter, qu'on se place dans un contexte où la politique fournie par la MAE est sous-optimale car sinon, l'intérêt d'appliquer une méthode d'apprentissage par renforcement est inexistant.

Puisqu'une MAE-π est une politique, il est possible d'utiliser les approches de *transfer learning* basées sur l'utilisation d'une politique extérieure, bien qu'elles ne tirent pas profit de la structure de la MAE et qu'elles doivent être capable de tirer profit d'une politique imparfaite. Il peut donc être intéressant d'explorer l'utilisation de ces approches, mais en tirant profit de la structure de MAE, ce qui permettrait d'avoir un traitement différent pour chacun des états.

Les MAE-π semblent intéressantes pour une utilisation en parallèle avec des modifications dans la phase exploratoire, car cela permet d'introduire une action alternative à celle fournie par la politique optimisée, ce qui peut permettre de guider l'exploration lors de l'apprentissage grâce à une heuristique métier si certains états sont complexes. De plus, il est également possible de restreindre l'utilisation de la politique issue du DRL à seulement certains états de façon similaire à [83].

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

MAE utilisant des informations privilégiées

L'applicabilité des algorithmes couplant une MAE et du DRL est fortement liée à la possibilité de pouvoir créer une MAE adéquate au MDP. Cependant, lorsque le traitement de l'espace d'observation est complexe (images, observabilité partielle, bruit, etc ...) il devient difficile de définir une MAE capable d'ajouter une réelle plus-value à un algorithme de DRL. Pour pallier cette difficulté, nous avons notamment identifié la possibilité d'utiliser des MAE explicitant des informations privilégiées ($U = \mathcal{S}$). Cela simplifie la création de la MAE car la fonction de transition est définie sur un espace dont le traitement est plus facile, cependant, il devient impossible d'utiliser un couplage qui conduit à une modification du MDP.

Il est intéressant de noter qu'il est possible d'utiliser une MAE exploitant des informations privilégiées à la fois pour la création de nouvelles métriques et pour la création de MAE-π.

7.1.5 Conclusion partielle sur le couplage entre une MAE et un algorithme de DRL

L'apprentissage par renforcement profond est une approche très intéressante pour la résolution de problèmes de décision séquentielle comme celui de la mise en place d'un modèle opérateur en simulation numérique. Cependant, cette approche présente des lacunes sur l'efficience de l'utilisation des données et l'explicabilité du modèle final. Dans un contexte industriel, il est nécessaire d'obtenir un résultat le plus explicable possible pour un coût calculatoire raisonnable.

Les MAE sont déjà fortement présentes pour résoudre ce type de problème, car elles montrent de bonnes performances tout en restant simples et explicables. En outre, l'utilisation couplée d'une MAE avec le DRL est une approche efficace et déjà proposée dans la littérature pour utiliser des connaissances extérieures sur le MDP afin d'aider le processus d'apprentissage. En effet, cela permet d'accélérer la convergence et/ou d'augmenter l'interprétabilité du modèle final soit par des restrictions de l'espace d'action ou bien par des processus d'apprentissage plus interprétables.

Cette section présente donc un cadre théorique pour l'incorporation d'une MAE au sein d'un processus d'apprentissage par renforcement. Ainsi, ces définitions permettent d'identifier les contraintes et les gains potentiels en fonction du placement du couplage. Elles nous ont permis de catégoriser plusieurs approches de l'état de l'art et d'identifier des couplages algorithmiques non explorés jusqu'à présent. Tout d'abord, nous proposons la mise en place de nouvelles métriques utilisant la structure d'une MAE afin d'avoir plus d'informations sur l'état courant de la politique au cours de l'apprentissage. Ensuite, l'utilisation d'une MAE politique semble également être une approche intéressante pour tirer profit de connaissances a priori sur le comportement souhaité. Finalement, l'utilisation d'information privilégiée peut permettre de simplifier et d'améliorer la mise en place de MAE et donc d'améliorer les gains d'interprétabilité et de performance d'un couplage.

Cette étude a été faite en amont des développements logiciels permettant d'appliquer le DRL à la problématique industrielle. Finalement, nous n'avons utilisé la structure de MAE du logiciel tactique que pour analyser le comportement des modèles opérateurs. Nous n'avons pas eu le temps d'appliquer l'ensemble des approches proposées et cela constitue une perspective de nos travaux.

Corporate sensitivity
PUBLIC

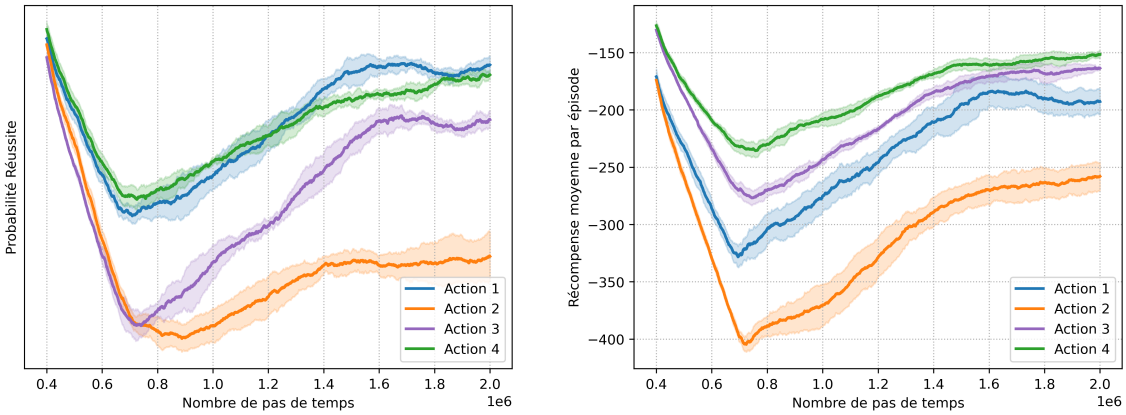
© Propriété Naval Group SA, (2026) tous droits réservés.

7.2 Apprentissages séparés des actions

Nous notons l'espace d'action complet $\mathcal{A} = \{\text{Ne Rien Faire, Action 1, Action 2, Action 3, Action 4}\}$ et les espaces d'action partielles $\forall i \in [1, 4]$, $\mathcal{A}_i = \{\text{Rien Faire, Action i}\}$.

Pour rappel, jusqu'ici (voir chapitres 5 et 6), tous les entraînements ont été faits en utilisant seulement $\mathcal{A}_1$ afin de simplifier le problème d'apprentissage. Cette section étudie l'apprentissage séparé de chacune des actions pour les scénarios BS et SM, c'est-à-dire les apprentissages sur chacun des $\mathcal{A}_i$.

7.2.1 Analyse sur le scénario bâtiment de surface



(a) Performances sur BS

(b) Récompense sur BS

FIGURE 7.3 – Courbes d'apprentissage moyennée sur une fenêtre glissante de 400 000 pas de temps moyenné sur 5 germes avec l'écart-type sur le scénario BS pour les différentes actions considérées.

La figure 7.3 montre les courbes d'apprentissage pour des entraînements avec les 4 actions considérées. On peut observer sur les figures 7.3a et 7.3b que le changement d'action modifie fortement la probabilité de réussite et la récompense reçue au cours de l'entraînement. Cela s'explique par des différences dans :

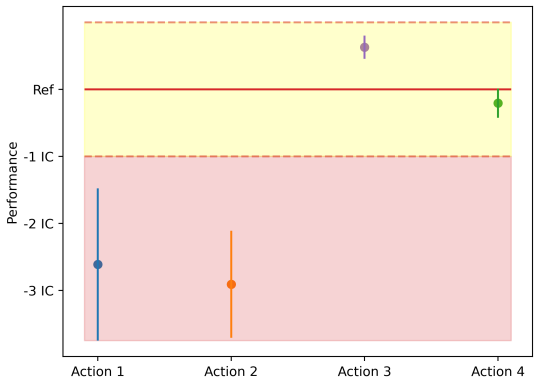
— la capacité de l'action à améliorer la situation tactique pour le scénario ;
— la difficulté pour la politique de comprendre l'intérêt de l'action à partir des informations contenues dans l'espace d'observation ;
— l'impact de la proportion d'action aléatoire dû à l'exploration (voir la figure 5.2).

En outre, on peut observer que l'ordre des valeurs après convergence est différent pour la récompense (Action 3 > Action 4 > Action 1 > Action 2) et la probabilité de réussite (Action 1 > Action 4 > Action 3 > Action 2). Cette différence montre notamment que la fonction de récompense n'est pas totalement alignée avec la probabilité de réussite car il est possible d'améliorer la récompense obtenue sans améliorer la probabilité de réussite. Cela est explicable par le choix de n'utiliser que des termes de *shaping* lors de la définition du problème d'apprentissage.

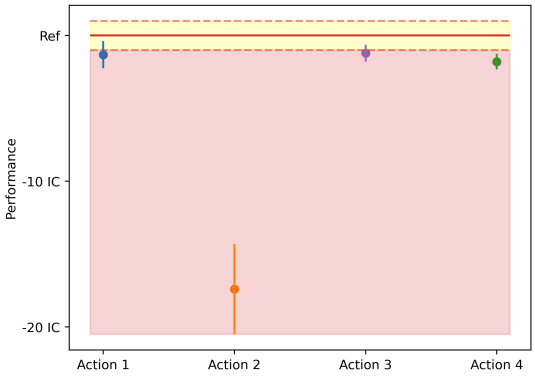
Les figures 7.4a et 7.4b montrent que les performances des modèles sont toutes inacceptables. En outre, on observe que les différences de performances observées dans la figure 7.3 ne se retrouvent pas pour les Actions 1, 3 et 4. En revanche, l'action 2 reste celle avec des performances notoirement plus faibles. D'après la figure 7.4c, la fréquence d'action est quant à elle acceptable

© Propriété Naval Group SA, (2026) tous droits réservés.

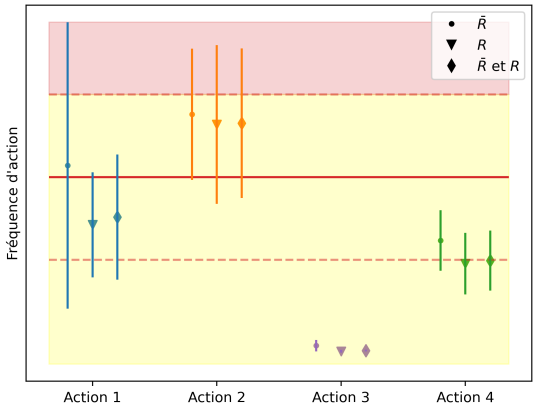
Corporate sensitivity
PUBLIC



(a) Performances sur $\text{BS}\bar{\text{R}}$



(b) Performances sur BSR



(c) Fréquence d'action sur BS

FIGURE 7.4 – Performances et fréquence d'action sur l'ensemble des scénarios BS moyennés des modèles appris dans la figure 7.3 avec un intervalle de confiance à 95%.

pour toutes les actions.

Cette tendance globale semble montrer que le problème d'apprentissage sur BS est mal défini car aucune des actions ne conduit à un gain de performance par rapport à la référence.

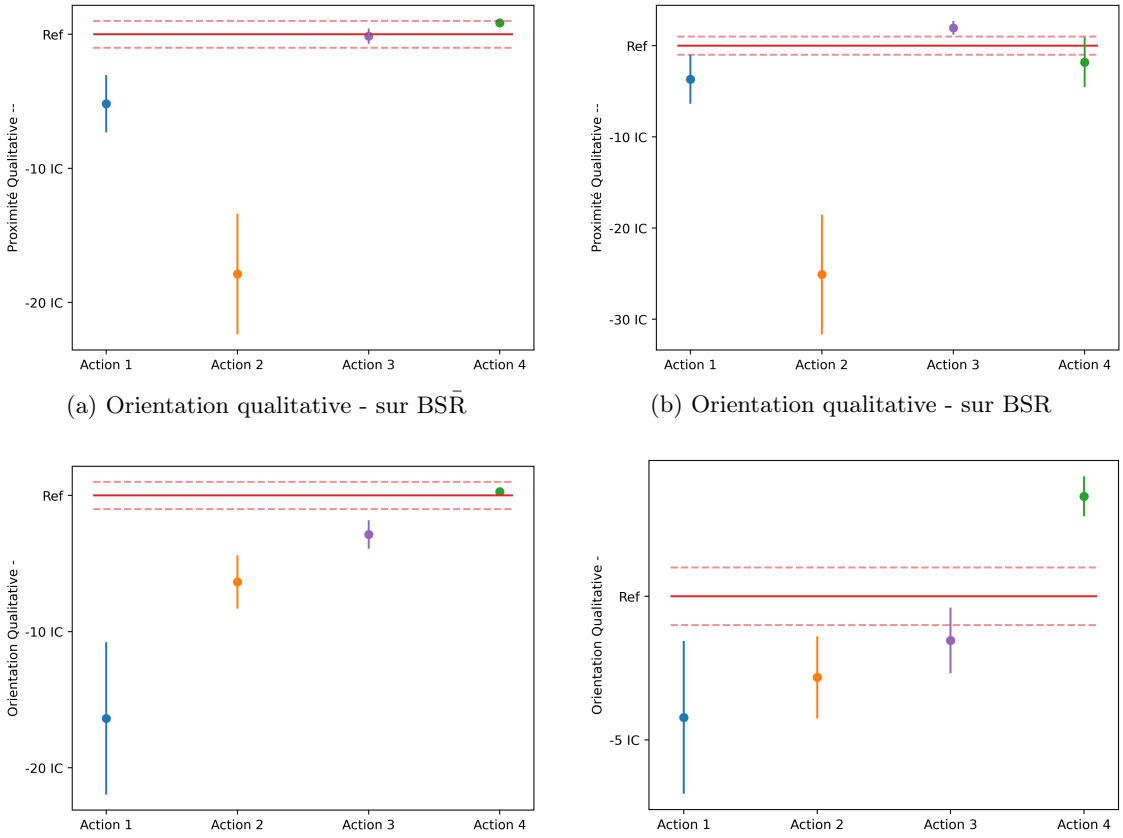
La figure 7.5 montre que l'ensemble des actions semblent inefficaces pour améliorer les termes de la fonction de récompense par rapport à la valeur obtenue par la référence. En outre, on observe que mis à part l'Action 4, l'ensemble des modèles ont mêmes des effets négatifs par rapport à la référence sur BSR.

La figure 7.6 montre même que les modèles opérateur utilisant l'Action 2 sur BS ont appris à créer un arrêt non nominal du simulateur. On peut voir que la proportion de fin d'épisode à cause d'un "Crash" de l'environnement augmente au cours de l'apprentissage jusqu'à une valeur bien plus haute que pour les autres actions. Nous attribuons ce phénomène à du *reward hacking* car comme les récompenses ne peuvent être que nulles ou négatives, réussir à arrêter la simulation de façon prématurée permet d'optimiser la récompense. Il est également intéressant de noter que ce comportement n'existe pas pour les modèles utilisant l'Action 2 sur SM, ce qui montre encore une fois la différence des problèmes à résoudre pour le modèle opérateur entre les scénarios SM et BS.

L'ensemble des apprentissages semblent montrer que le processus d'apprentissage développé

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.



(a) Orientation qualitative - sur $\text{BS}\bar{\text{R}}$

(b) Orientation qualitative - sur BSR

(c) Proximité qualitative – sur $\text{BS}\bar{\text{R}}$

(d) Proximité qualitative – sur BSR

FIGURE 7.5 – Termes de *shaping* d'orientation qualitative - et de proximité qualitative – sur l'ensemble des scénarios BS moyennés des modèles appris dans la figure 7.3 avec un intervalle de confiance à 95%.

pour BS ne permet pas d'apprendre un modèle opérateur obtenant des performances acceptables. Les deux hypothèses qui nous semblent les plus probables sont qu'il est plus difficile d'améliorer les performances par rapport à la référence que pour SM car la définition du MDP est moins bonne. En effet, la fonction de récompense est moins efficace car elle est à la fois moins alignée avec l'objectif et plus difficile à optimiser avec les actions considérées. Il est également possible que les choix d'hyperparamètres pour les algorithmes de DRL soient moins efficaces pour BS que pour SM.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

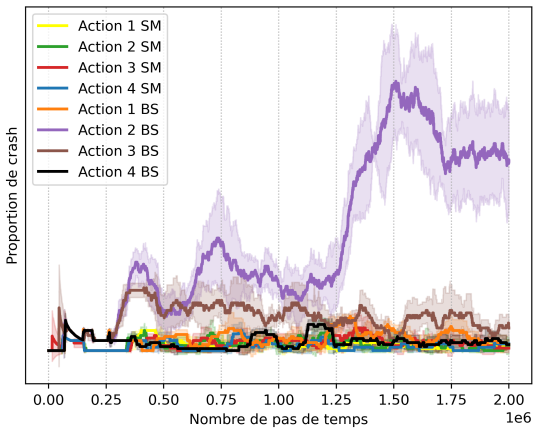
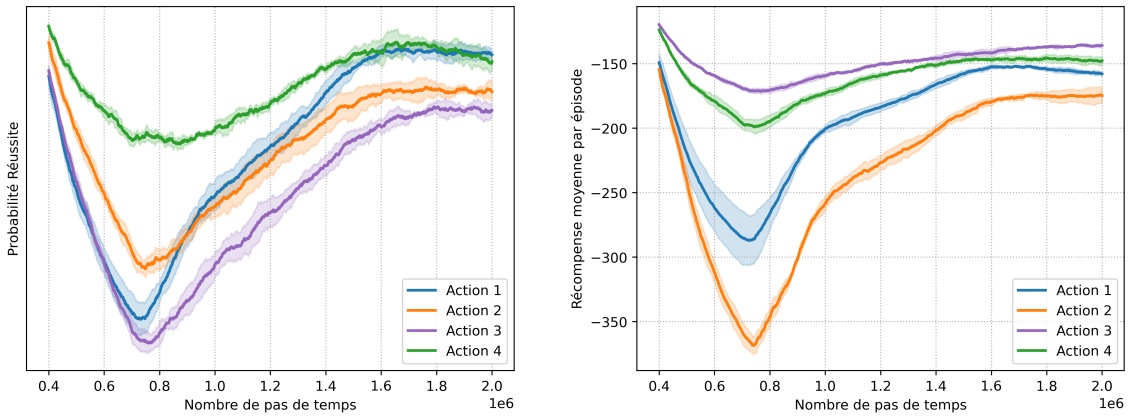


FIGURE 7.6 – Proportion d'épisode qui se finissent par un "crash" de l'environnement au cours de l'apprentissage en fonction du scénario et de l'action à la disposition du modèle.

7.2.2 Analyse sur le scénario sous-marin



(a) Performances sur SM

(b) Récompense sur SM

FIGURE 7.7 – Courbes d'apprentissage moyenné sur une fenêtre glissante de 400 000 pas de temps moyenné sur 5 germes avec l'écart-type sur le scénario SM pour différentes actions.

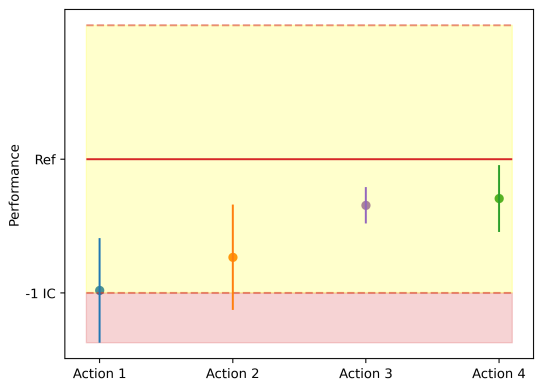
De façon similaire aux entraînements sur BS, la figure 7.7 montre que l'action à la disposition du modèle lors de l'apprentissage influe fortement les performances obtenues.

On observe également que l'ordre des valeurs après convergence est différent pour la récompense (Action 4 > Action 1 > Action 2 > Action 3) et la probabilité de réussite (Action 4 > Action 3 > Action 1 > Action 2). Cela montre que la fonction de récompense n'est pas totalement alignée non plus avec la probabilité de réussite bien que les corrélations observées avec les termes de *shaping* soient plus hautes sur SM que sur BS (voir section 5.5.2).

La figure 7.8a montre qu'il n'y a aucun gain sur SMR. Cependant, on observe sur la figure 7.8b que toutes les actions mènent à un gain de performance sur SMR par rapport à la référence, bien que le seul gain notable soit obtenu avec l'action 1. Cela renforce le constat qu'il est plus difficile d'améliorer les performances sur un scénario sans réaction car le logiciel tactique est déjà très performant.

Corporate sensitivity
PUBLIC

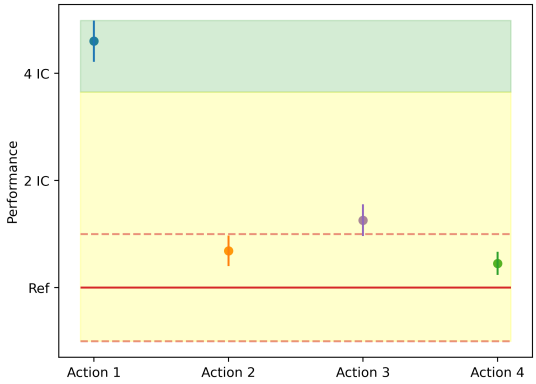
© Propriété Naval Group SA, (2026) tous droits réservés.



(a) Performances sur $\overline{\text{SMR}}$



(b) Performances sur SMR



(c) Fréquence d'action sur SM

FIGURE 7.8 – Performances sur l'ensemble des scénarios SM moyennés des modèles appris dans la figure 7.7 avec un intervalle de confiance à 95%.

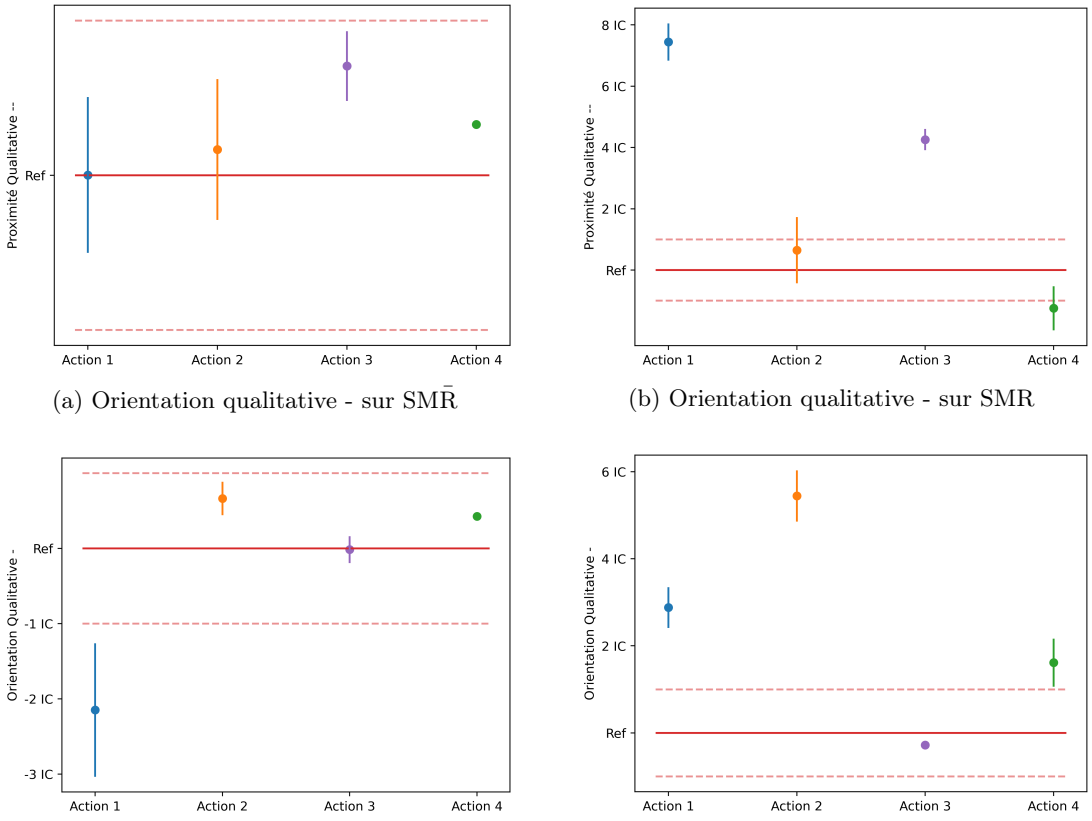
On observe sur la figure 7.8c que la fréquence d'action est fortement dépendante de l'action, avec notamment une valeur globale sur SM inacceptable pour l'action 2. On peut également voir que le fait d'ajouter une réaction (différence entre $\overline{\text{SMR}}$ et SMR) a un impact pour toutes les actions. Pour les actions 1, 2 et 3 cela conduit à une augmentation notable de la fréquence d'action contrairement à l'action 4 qui conduit à une baisse.

Les figures 7.9a et 7.9c ne montrent aucune amélioration des termes de récompense par rapport à la référence pour $\overline{\text{SMR}}$. Cela corrobore l'incapacité des modèles appris à améliorer les deux termes de récompense sur $\overline{\text{SMR}}$. Cependant, on observe sur les figures 7.9b et 7.9d des gains par rapport à la référence sur au moins un des termes pour toutes les actions. On peut noter que les modèles utilisant les actions 2 et 4 ne mènent à aucun gain sur le terme d'orientation qualitative - et que les modèles utilisant l'action 3 ne font aucun gain sur le terme de proximité qualitative --. Seulement l'action 1 permet un gain conséquent sur les deux termes. Cela est une conséquence des différences opérationnelles entre les actions considérées.

La figure 7.10 montre la fréquence d'action en fonction de l'état de la MAE du logiciel tactique (présentée dans la section 1.2.3). On peut notamment observer que tous les modèles sont le moins actifs dans la phase de recherche. Les modèles utilisant l'action 1 et 2 sont beaucoup plus actifs que les modèles utilisant les action 3 et 4. Globalement, tous les modèles sont très actifs dans la phase d'attaque. Notre interprétation est que la situation tactique est beaucoup plus simple

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC



(a) Orientation qualitative - sur $\text{SM}\bar{\text{R}}$

(b) Orientation qualitative - sur SMR

(c) Proximité qualitative -- sur $\text{SM}\bar{\text{R}}$

(d) Proximité qualitative -- sur SMR

FIGURE 7.9 – Évaluation des termes de *shaping* d'orientation qualitative - et de proximité qualitative -- sur l'ensemble des scénarios SM moyennés des modèles appris dans la figure 7.7 avec un intervalle de confiance à 95%.

dans la phase de recherche, elle ne nécessite donc pas d'arbitrage de la part de l'opérateur. Cependant, les phases de poursuite et d'attaque induisent beaucoup plus de données à traiter et de choix tactiques à faire, ce qui crée plus de difficultés au logiciel tactique et donc potentiellement augmente la nécessité d'une action par le modèle opérateur.

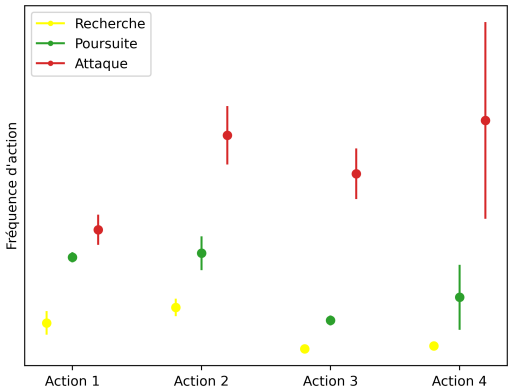


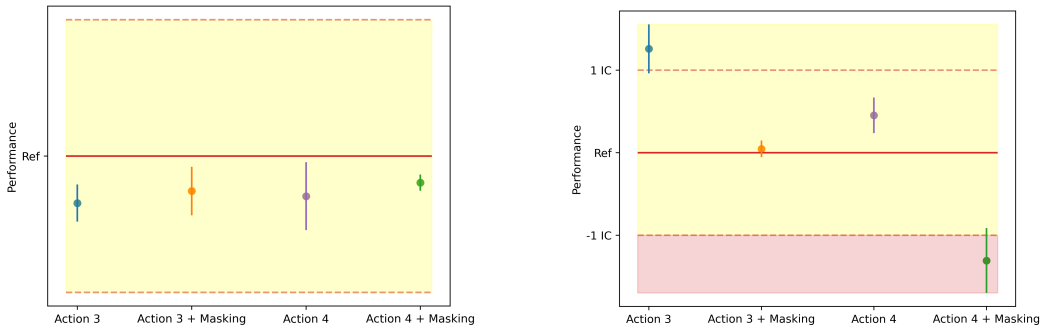
FIGURE 7.10 – Fréquence d'action par phase en fonction de l'action à la disposition du modèle.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

7.3 Ajout de l'*action masking*

L'*action masking*, présenté dans la section 2.5.4, est le fait de retirer la possibilité de réaliser une action invalide pour le modèle plutôt que de lui laisser apprendre quand elle est efficace. Parmi les 4 actions considérées, seulement les actions 3 et 4 peuvent devenir invalides au cours d'un tir.



(a) Performances sur SM$\bar{R}$

(b) Performances sur SMR

(c) Fréquence d'action sur SM

FIGURE 7.11 – Performances sur l'ensemble des scénarios SM moyennés des modèles appris dans la figure 7.7 avec un intervalle de confiance à 95%.

La figure 7.11a montre que l'*action masking* n'a aucun effet sur SM$\bar{R}$. En revanche, la figure 7.11b révèle une perte de performances pour l'action 3 et 4.

La figure 7.11c montre que l'*action masking* conduit à des politiques avec des fréquences d'action très faible pour l'action 3. Il semble donc que l'*action masking* conduit à l'apprentissage d'une politique qui choisit uniquement l'action "Rien Faire". Cela explique les performances très proches de la référence observée sur les figures 7.11a et 7.11b.

En conclusion, l'*action masking* est néfaste pour les deux actions pouvant en bénéficier, nous ne l'avons donc pas conservé pour la suite des travaux. Nous n'avons pas d'hypothèse permettant d'expliquer cet effet néfaste sur les performances.

7.4 Apprentissage commun

Suite à l'étude de modèles capables avec un espace d'action partiel, cette section présente l'apprentissage de modèles capables d'utiliser l'ensemble des actions.

© Propriété Naval Group SA, (2026) tous droits réservés.

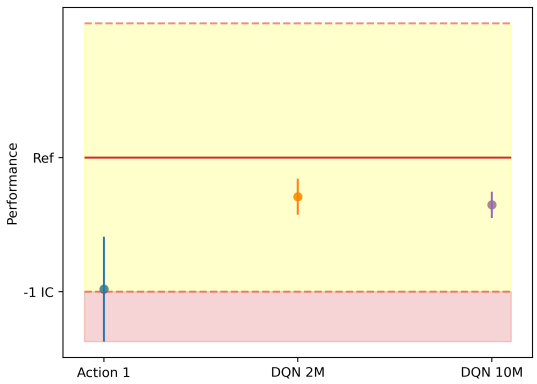
Corporate sensitivity
PUBLIC

7.4.1 Apprentissage de toutes les actions

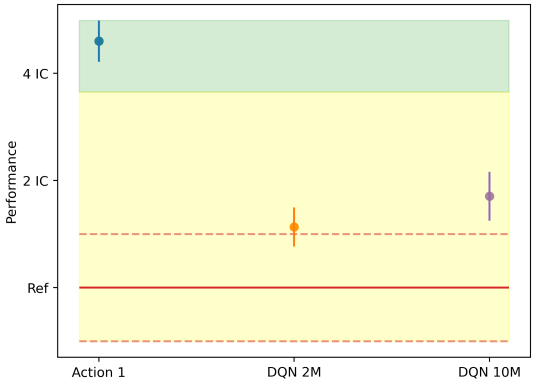
Dans cette section, les modèles utilisent l'espace d'observation privilégié ($\mathcal{S}$) et sont appris avec la version de DQN décrite dans le chapitre 6. Les différents modèles mentionnés sont :

- **Action 1** : modèle appris avec comme espace d'action $\mathcal{A}_1$ pendant 2 millions de pas de temps (issu de la section 7.2.2).
- **DQN 2M** : modèle appris avec comme espace d'action $\mathcal{A}$ pendant 2 millions de pas de temps.
- **DQN 10M** : modèle appris avec comme espace d'action $\mathcal{A}$ pendant 10 millions de pas de temps.

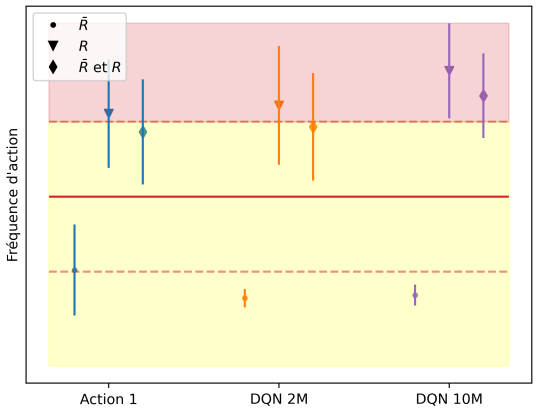
La figure 7.12a montre que la possibilité d'utiliser toutes les actions semblent n'avoir aucun effet bénéfique sur $\overline{\text{SMR}}$. La figure 7.12b montre que les performances obtenues par le modèle utilisant seulement l'espace d'action $\mathcal{A}_1$ sont supérieures sur SMR à celles des modèles ayant accès à l'ensemble de l'espace d'action. En outre, on peut voir que l'augmentation du temps d'entraînement de 2 à 10 millions de pas de temps est bénéfique pour un apprentissage utilisant l'ensemble de l'espace d'action. La figure 7.12c montre que la fréquence d'action est similaire sur l'ensemble des modèles considérés.



(a) Performances sur $\overline{\text{SMR}}$



(b) Performances sur SMR



(c) Fréquence d'action sur SM

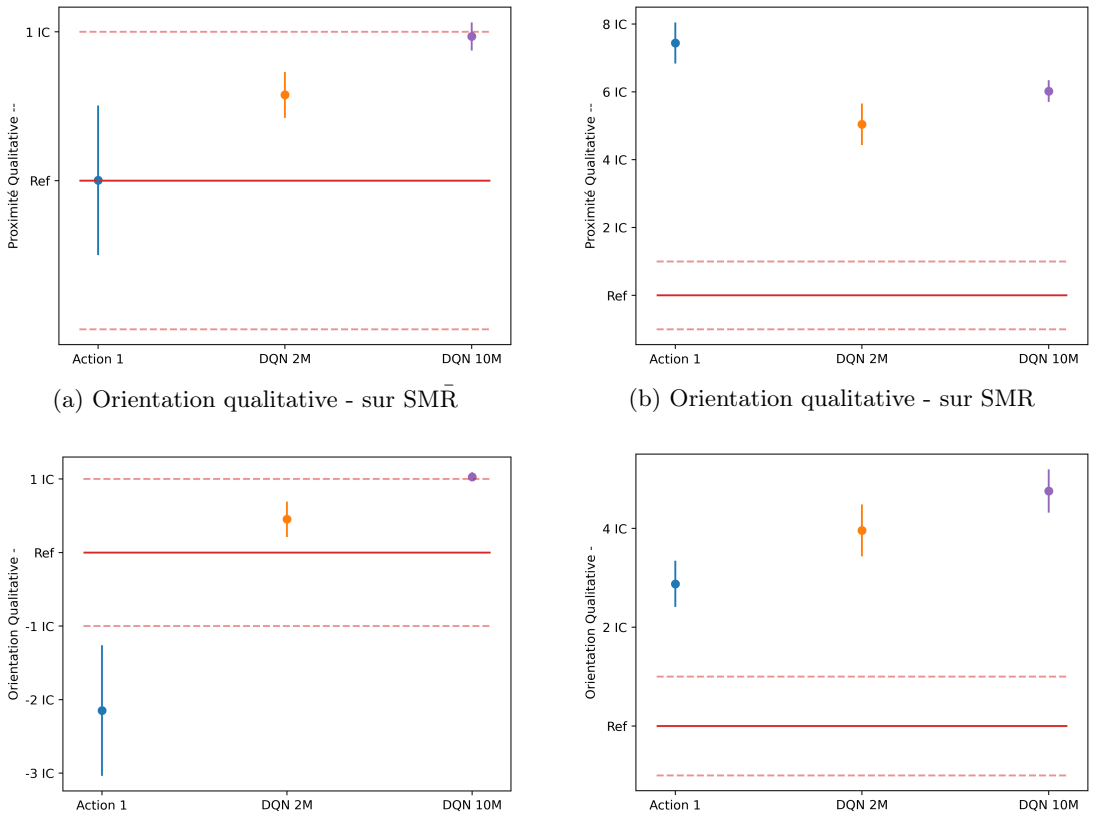
FIGURE 7.12 – Performances et fréquence d'action sur SM avec un intervalle de confiance à 95% pour les modèles **Action 1**, **DQN 2M** et **DQN 10M**

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

On observe sur les figures 7.13a et 7.13c que les modèles ayant accès à l'ensemble des actions obtiennent une meilleure récompense sur $\overline{\text{SMR}}$. La figure 7.13b montre que sur SMR, les modèles utilisant seulement l'Action 1 obtiennent une meilleure valeur pour le terme d'Orientation Qualitative -. En revanche, on constate sur la figure 7.13d que les modèles utilisant l'espace d'action complet ont une meilleure valeur moyenne pour le terme de Proximité Qualitative --. Globalement, les modèles définis sur $\mathcal{A}$ ont une meilleure valeur de récompense que les modèles utilisant seulement l'Action 1.

Ainsi, bien que les modèles utilisant l'Action 1 soient moins performants selon le critère de réussite métier, ils sont meilleurs selon le critère de la récompense, qui est celui réellement optimisé par les algorithmes. Comme montré dans la section 7.2.2, les Actions 2, 3 et 4 sont capables d'optimiser la récompense tout en apportant des gains faibles de performances, cela est sûrement la cause de cette baisse de performance lors de l'utilisation de l'espace d'action $\mathcal{A}$.



(a) Orientation qualitative - sur $\overline{\text{SMR}}$

(b) Orientation qualitative - sur SMR

(c) Proximité qualitative -- sur $\overline{\text{SMR}}$

(d) Proximité qualitative -- sur SMR

FIGURE 7.13 – Termes de *shaping* d'orientation qualitative - et de proximité qualitative -- sur SM avec un intervalle de confiance à 95% pour les modèles **Action 1**, **DQN 2M** et **DQN 10M**.

La figure 7.14 montre la fréquence d'action moyenne des modèles **DQN 10M** par phase. On peut observer que l'action 1 est la plus utilisée dans l'ensemble des 3 phases. L'Action 2, qui est celle qui permettait le plus de gain sur le terme de proximité qualitative --, est également fortement utilisée dans les phases de Recherche et de Poursuite. Cela corrobore donc les observations faites à partir de la figure 7.13 : l'accès à d'autres actions permet d'optimiser la récompense d'une façon qui maximise moins la performance.

En conclusion, l'accès à l'espace d'action complet a conduit à un modèle qui nécessite plus

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

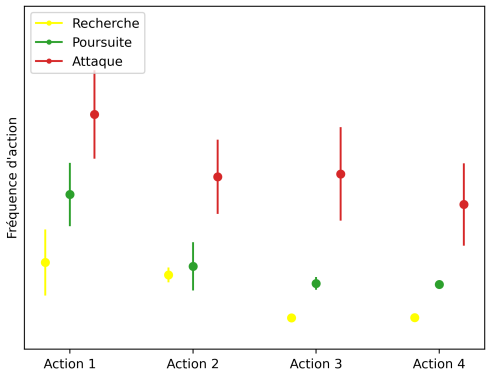


FIGURE 7.14 – Fréquence d'action par action et par phase pour les modèles **DQN 10M** avec un intervalle de confiance à 95%.

de pas de temps pour converger. Cela est sûrement dû à la complexification du MDP dû à l'augmentation des possibilités d'actions. En outre, les performances du modèle utilisant toutes les actions sont plus faibles que celui qui a seulement accès à l'Action 1. Ce résultat s'explique par la possibilité de maximiser la récompense avec d'autres actions sans augmenter la probabilité de réussite, notamment l'Action 2.

7.4.2 Utilisation de QDagger à partir des modèles à une action

Au vu des performances des modèles utilisant chacune des actions séparées par rapport aux modèles utilisant toutes les actions, nous avons souhaité utiliser une approche de *transfer learning*. L'objectif est donc de profiter des connaissances apprises par les modèles utilisant chacune des actions séparées pour guider un modèle les utilisant toutes à la fois.

Motivation du choix de l'algorithme *QDagger*

Nous avons choisi l'algorithme *QDagger* [75] au vu de ces avantages :

— une indifférence au type de la politique enseignante. L'algorithme n'impose aucune contrainte sur l'architecture ou l'algorithme utilisé pour générer la politique enseignante.
— une indépendance de la politique finale à la politique enseignante. En effet, la politique apprise ne garde aucune dépendance à la politique enseignante utilisée lors de l'apprentissage.
— une efficacité sur l'aspect calculatoire et sur l'utilisation des données. Cet algorithme permet d'accélérer l'apprentissage en utilisant de façon efficace les connaissances de la politique enseignante. En outre, il est également possible que la politique apprise devienne meilleure que la politique enseignante.

Définition des objets

Nous notons $\pi^A : \mathcal{S} \times \mathcal{A} \to [0,1]$ la politique apprise et $\forall i \in [1,4]$, $\pi_i^E : \mathcal{S} \times \mathcal{A}_i \to [0,1]$ les politiques enseignantes utilisant une seule action (elles correspondent aux politiques apprises dans la section 7.2).

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

Implémentation de l'algorithme *QDagger*

Nous avons donc appliqué les modifications suivantes issues de *QDagger* à l'algorithme présenté dans le chapitre 6 :

— utilisation des politiques enseignantes pour initialiser le *replay buffer* ;
— ajout d'un terme à la fonction de coût qui encourage la proximité entre la probabilité de la politique apprise et de la politique enseignante.

Pour l'implémentation, on peut noter deux différences majeures entre le cas d'utilisation des auteurs de l'article [75] et notre contexte industriel :

— nous considérons 4 politiques enseignantes (l'ensemble $\{\pi_i^E\}_{i\in[1,4]}$) au lieu d'une seule ;
— l'espace d'action de la politique apprise est différent de ceux des politiques enseignantes ($\forall i \in [1,4]$, $\mathcal{A} \neq \mathcal{A}_i$).

Dans l'article, le *replay buffer* est initialisé avec les transitions présentes à la fin de l'entraînement de la politique enseignante. Cette approche permet de réduire les temps de calcul en utilisant des transitions issues d'entraînements passés. Cependant, nous ne l'avons pas utilisé car elle crée plusieurs difficultés dans notre cas d'application :

— comme il y a plusieurs enseignants, les transitions issues d'un entraînement d'une des politique enseignantes n'ont pas l'évaluation de la probabilité des autres politiques enseignantes ;
— comme la définition du MDP change (modification de l'espace d'observation, de la fonction de récompense, de l'espace d'action), les anciennes transitions deviennent régulièrement obsolètes.

Nous avons donc adopté une approche similaire à celle présentée par Agarwal et al. [75] dans l'annexe A.5, en utilisant les politiques enseignantes avec l'environnement pour générer au début de l'entraînement les transitions pour initialiser le *replay buffer*. Comme il y a quatre politiques enseignantes et non pas une, il faut donc utiliser quatre politiques pour générer les transitions. Nous avons fait le choix de tirer une politique enseignante aléatoirement au début de chaque épisode et de garder la même politique jusqu'à sa fin. Ainsi, les politiques apprises ne sont pas utilisées simultanément dans un même épisode. En effet, cela peut conduire les politiques à se trouver dans des situations non explorées lors de leurs apprentissages dû à l'utilisation des autres actions et donc à prendre de mauvaises décisions.

Agarwal et al. [75] ont défini le terme de coût qui encourage la proximité entre la probabilité de la politique apprise et de la politique enseignante à partir d'un état par :

$$\mathbf{L}(s) = \sum_{a\in\mathcal{A}} \pi^E(a|s)\pi^A_{\boldsymbol{\theta}}(a|s) \tag{7.1}$$

Afin d'adapter cette fonction de coût pour plusieurs politiques enseignantes, nous avons utilisé la moyenne sur l'ensemble des politiques enseignantes :

$$\mathbf{L}(s) = \frac{1}{4}\sum_{i\in[1,4]}\sum_{a\in\mathcal{A}} \pi^E_i(a|s)\pi^A_{\boldsymbol{\theta}}(a|s) \tag{7.2}$$

Comme les politiques enseignantes et la politique apprise ne sont pas définies avec les mêmes espaces d'action, la fonction définie dans l'équation 7.2 n'est pas correctement définie également car les politiques enseignantes donnent une probabilité que pour un sous-ensemble de l'espace d'action $\mathcal{A}$.

Nous avons alors développé deux approches afin de définir les politiques enseignantes sur $\mathcal{A}$. La première approche est d'utiliser un paramètre ε afin de donner une valeur par défaut pour les

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

actions non définies. Avec $i \in [1,4]$ l'indice de la politique enseignante, $\varepsilon \in [0, \frac{1}{3}]$ la probabilité par défaut des actions non définies, nous définissons la fonction $\pi^{E\to\mathcal{A}}_{i,\text{défaut}_\varepsilon} : \mathcal{S} \times \mathcal{A} \to [0,1]$:

$$\forall s, a \in \mathcal{S} \times \mathcal{A},\ \pi^{E\to\mathcal{A}}_{i,\text{défaut}_\varepsilon}(a|s) = \begin{cases} (1-3\varepsilon) * \pi^E_i(a|s), \text{ si } a \in \{\text{Rien Faire}, \text{Action } i\} \\ \varepsilon \text{ , sinon} \end{cases} \tag{7.3}$$

La seconde approche utilise la valeur de politique apprise. Avec $i \in [1,4]$ l'indice de la politique enseignante, nous définissons la fonction $\pi^{E\to\mathcal{A}}_{i,\text{auto}} : \mathcal{S} \times \mathcal{A} \to [0,1]$:

$$\forall s, a \in \mathcal{S} \times \mathcal{A},\ \pi^{E\to\mathcal{A}}_{i,\text{auto}}(a|s) = \begin{cases} \left(1 - \sum_{a' \in \mathcal{A}\setminus\mathcal{A}_i} \pi^A_i(a'|s)\right) * \pi^E_i(a|s), \text{ si } a \in \{\text{Rien Faire}, \text{Action } i\} \\ \pi^A_i(a|s) \text{ , sinon} \end{cases} \tag{7.4}$$

Cette approche a pour but de ne pas introduire d'hyperparamètres supplémentaires ni de valeur artificielle dont la politique apprise doit s'approcher (contrairement à l'approche précédente où on introduit ε). Ainsi, la valeur de la fonction de coût ne sera pas minimale seulement pour les actions appartenant à $\mathcal{A}_i$.

En notant, $\left\{\pi^{E\to\mathcal{A}}_i\right\}_{i\in[1,4]}$ les politiques enseignantes adaptées à l'espace d'action $\mathcal{A}$, le fonction de coût utilisée est donc :

$$\mathbf{L}(s) = \frac{1}{4} \sum_{i\in[1,4]} \sum_{a\in\mathcal{A}} \pi^{E\to\mathcal{A}}_i(a|s) \pi^A_{\boldsymbol{\theta}}(a|s) \tag{7.5}$$

Résultats

Dans cette section, l'ensemble des modèles utilisent l'espace d'observation privilégiée ($\mathcal{S}$) et l'espace d'action ($\mathcal{A}$). Les modèles utilisant *QDagger* utilisent 4 politiques enseignantes apprises à partir de l'espace d'observation privilégiée ($\mathcal{S}$) et d'un espace d'action partielle ($\mathcal{A}_i$) pendant 2 millions de pas de temps (issu de la section 7.2.2). Les différents modèles mentionnés sont :

- **DQN 10M** : modèle appris avec l'algorithme décrit dans le chapitre 6 pendant 10 millions de pas de temps.
- **QDag Défaut** : modèle appris avec l'algorithme *QDagger* avec les modifications présentées dans la section 7.4.2 et la formule d'adaptation des politiques enseignantes présentée dans l'équation 7.3, pendant 2 millions de pas de temps.
- **QDag Auto** : modèle appris avec l'algorithme *QDagger* avec les modifications présentées dans la section 7.4.2 et la formule d'adaptation des politiques enseignantes présentée dans l'équation 7.4, pendant 2 millions de pas de temps.
- **QDag + Anti Crash** : modèle appris avec l'algorithme *QDagger* avec les modifications présentées dans la section 7.4.2, la formule d'adaptation des politiques enseignantes présentée dans l'équation 7.4 et une récompense pénalisant un épisode se finissant par un *crash*, pendant 2 millions de pas de temps.

Afin de faire une comparaison de ces modèles en *strong transfer* (c'est-à-dire en prenant en compte les pas de temps nécessaires pour générer les connaissances précédentes voir section 7.15), il faut comparer les modèles obtenus par *QDagger* en 2 millions de pas de temps à un modèle appris avec 10 millions de pas de temps avec DQN. En effet, chacun des 4 modèles enseignants a été appris avec 2 millions de pas de temps. Avec les 2 millions de pas de temps générés lors de

Corporate sensitivity PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

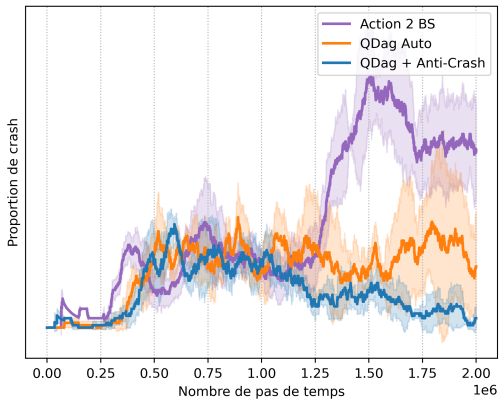
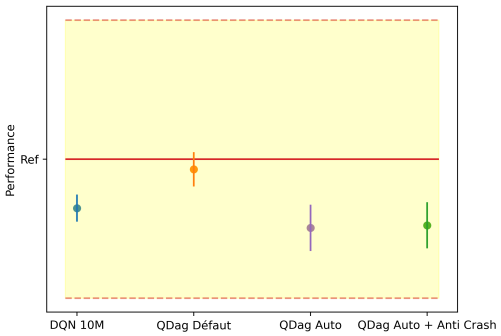
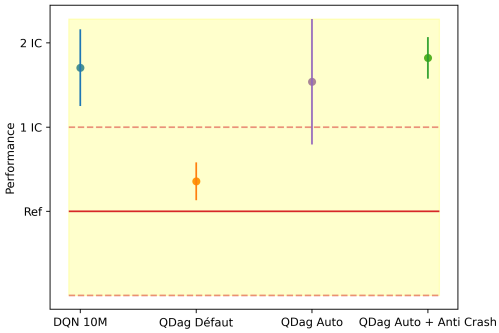


FIGURE 7.15 – Proportion d'épisode se terminant par un *crash* au cours de l'entraînement pour l'apprentissage utilisant DQN et seulement l'action 2 sur BS, *QDagger* avec l'adaptation de l'équation 7.4 et *QDagger* avec l'adaptation de l'équation 7.4 ainsi qu'un terme terminal de pénalisation.

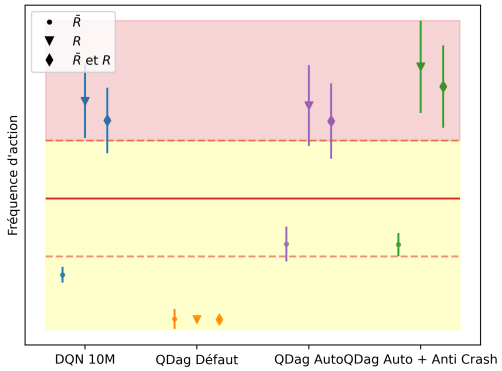
l'apprentissage avec l'algorithme *QDagger*, chacun de ces modèles a donc nécessité 10 millions de pas de temps également.



(a) Performances sur SM$\bar{\text{R}}$



(b) Performances sur SMR



(c) Fréquence d'action sur SM

FIGURE 7.16 – Performances et fréquence d'action sur SM avec un intervalle de confiance à 95% pour les modèles **DQN 10M**, **QDag Défaut**, **QDag Auto** et **QDag + Anti Crash**.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

De façon similaire à l'action 2 sur BS (voir section 2.5.1), l'ajout de l'algorithme *QDagger* conduit à l'apprentissage d'un comportement qui tire profit d'une possibilité de créer un *crash* du simulateur. On peut voir sur la figure 7.15 que l'apprentissage utilisant **QDag Auto** a une proportion de *crash* qui est non nominale. Nous avons donc ajouté un terme terminal de pénalisation lors d'un *crash* (**QDag Auto** + **Anti Crash**) qui conduit à la réduction du nombre de *crash* au cours de l'apprentissage.

On peut observer dans les figures 7.16a et 7.16b que les modèles **QDag Défaut** ne conduisent à aucune amélioration de performance par rapport à la référence. On observe sur la figure 7.16c que ces modèles ont une fréquence d'action très faible. Notre interprétation de ce phénomène est que l'utilisation d'une valeur par défaut dans l'adaptation de la politique enseignante conduit à une pénalisation artificielle d'une action différente de l'action "Rien Faire".

En revanche, les modèles **QDag Auto** ont des gains de performances significatifs par rapport à la référence. On peut également voir que l'ajout du terme de récompense pénalisant les *crashs* (**QDag Auto** + **Anti Crash**) conduit à un léger gain de performance mais surtout une forte réduction de la variabilité des performances des modèles appris. On peut voir sur la figure 7.16c que la fréquence d'action est similaire pour **DQN 10M**, **QDag Défaut**, **QDag Auto** et **QDag** + **Anti Crash**.

En conclusion, pour l'adaptation de la politique enseignante à l'espace d'action de la politique apprise, l'approche **QDag Auto** + **Anti Crash** semble être plus efficace car elle a de meilleures performances et un hyperparamètre de moins. Par rapport à un apprentissage utilisant seulement DQN, les modèles **QDag Auto** + **Anti Crash** ont un léger de gain de performance et une baisse de la variabilité des performances des modèles appris. Ces résultats sont à nuancer à cause de la faible significativité statistique due au manque de germes aléatoires différents. Cependant, on peut noter que les modèles **QDag Auto** + **Anti Crash** sont meilleurs que ceux de **DQN 2M**, ce qui montre que l'algorithme permet de tirer profits des politiques enseignantes pour améliorer la convergence.

7.5 Information privilégiée ou non

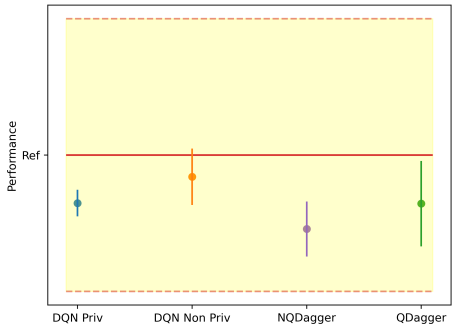
Cette section présente l'apprentissage de modèles capables d'utiliser l'ensemble des actions à partir de l'espace d'observation non privilégié (Ω), contrairement à tous les modèles présentés jusqu'à maintenant. Pour rappel, la différence fondamentale est que l'espace d'observation privilégiée a directement accès à la position de la cible contrairement à l'espace d'observation non privilégiée. Les différents modèles étudiés sont :

- **DQN Priv** : modèle appris avec l'algorithme DQN décrit dans le chapitre 6 pendant 10 millions de pas de temps en utilisant l'espace d'observation privilégiée $\mathcal{S}$. Ce sont les mêmes résultats que **DQN 10M** dans la section 7.4.1 et permet d'avoir une référence des modèles utilisant l'espace d'observation privilégiée.
- **DQN Non Priv** : modèle appris avec l'algorithme DQN décrit dans le chapitre 6 pendant 10 millions de pas de temps en utilisant l'espace d'observation non privilégiée (Ω).
- **NQDagger** : modèle appris avec l'algorithme *QDagger* avec les modifications présentées dans la section 7.4.2 et la formule d'adaptation des politiques enseignantes présentée dans l'équation 7.4 pendant 2 millions de pas de temps en utilisant l'espace d'observation non privilégiée (Ω). Les politiques enseignantes sont issues de la section 7.2.2 et utilisent donc l'espace d'observation privilégiée ($\mathcal{S}$).
- **QDagger** : modèle appris avec l'algorithme *QDagger* avec comme politique enseignante un modèle de **QDag** + **Anti Crash** pendant 2 millions de pas de temps en utilisant

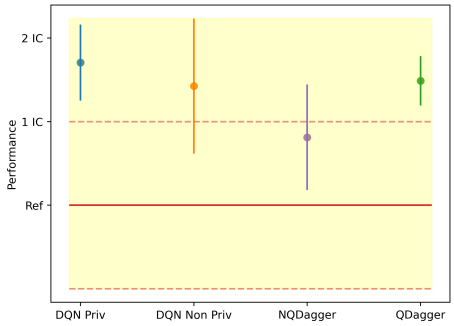
Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

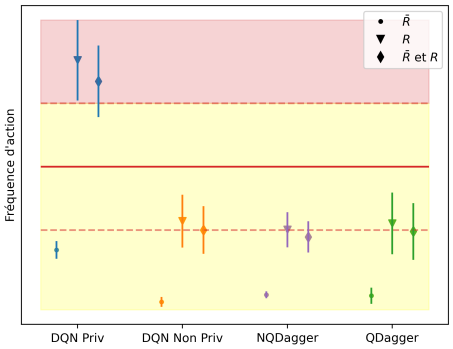
l'espace d'observation non privilégiée (Ω).



(a) Performances sur SM$\bar{R}$



(b) Performances sur SMR



(c) Fréquence d'action sur SM

FIGURE 7.17 – Performances et fréquence d'action sur SM avec un intervalle de confiance à 95% pour les modèles **DQN Priv**, **DQN Non Priv**, **NQDagger** et **QDagger**.

La figure7.17a montre aucune différence significative entre les 4 ensembles de modèles proposés sur SM$\bar{R}$. On observe sur la figure 7.17b que parmi les approches utilisant des modèles passés, **QDagger** est celle qui a les meilleures performances. En outre, de façon similaire à la section 7.4.2, on observe aucun gain de performance significatif entre un apprentissage d'un nouveau modèle *from scratch* (**DQN Non Priv**) et d'un modèle utilisant des politiques privilégiées comme enseignant (**QDagger**). La seule différence notable est que la performance semble moins variable entre les modèles pour l'approche **QDagger**.

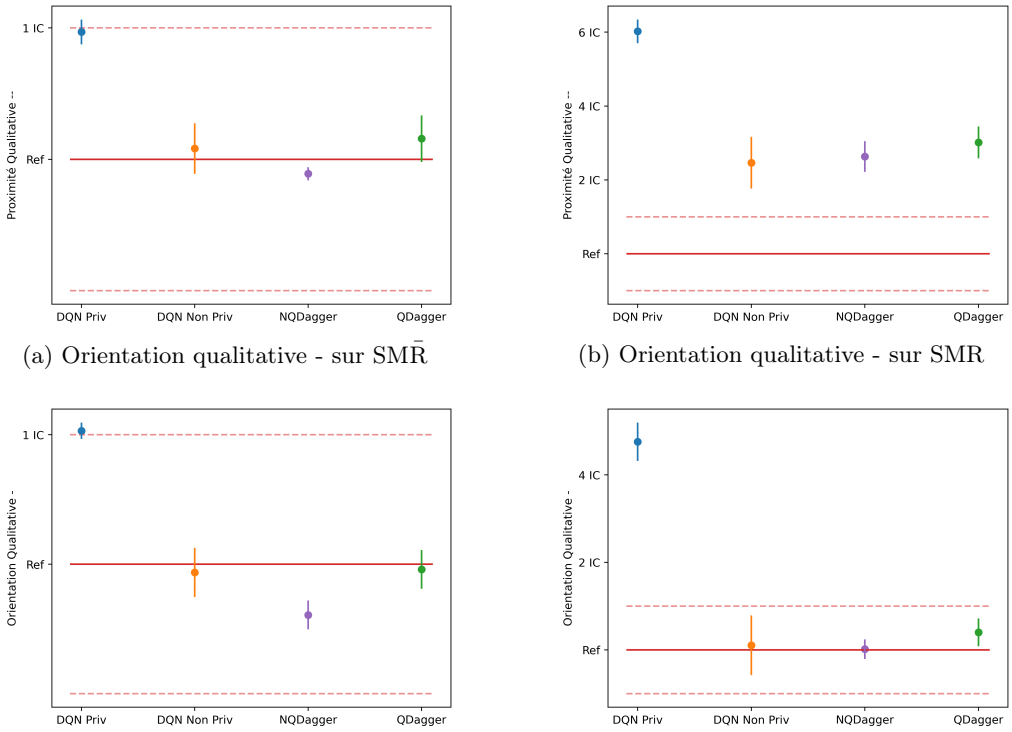
La figure 7.17c montre que les modèles utilisant l'espace d'observation non privilégié ont une fréquence d'action beaucoup plus faible que ceux utilisant l'espace d'observation privilégié (**DQN Priv**). Cet effet rend notamment tous les modèles acceptables sur le critère de la fréquence d'action. En revanche, il n'y a aucune différence notable entre les différents modèles utilisant l'espace d'observation non privilégiée.

Bien qu'il n'y ait qu'une faible différence de performance entre les modèles, on observe sur les figures 7.18a, 7.18b, 7.18c et 7.18d que la récompense des modèles utilisant l'espace d'observation non privilégiée est plus faible. On note que ces modèles ne sont capables de faire un gain significatif de récompense que sur SMR et pour le terme d'orientation qualitative -. Cela explique notamment la faible fréquence d'action observée dans la figure 7.17c, car pour une même pondération du terme comportemental, le modèle trouve moins de situation où il est intéressant de faire une action.

En outre, les figures 7.19a et 7.19b montre une très forte différence entre la durée des tirs d'un

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC



(a) Orientation qualitative - sur SM$\bar{R}$

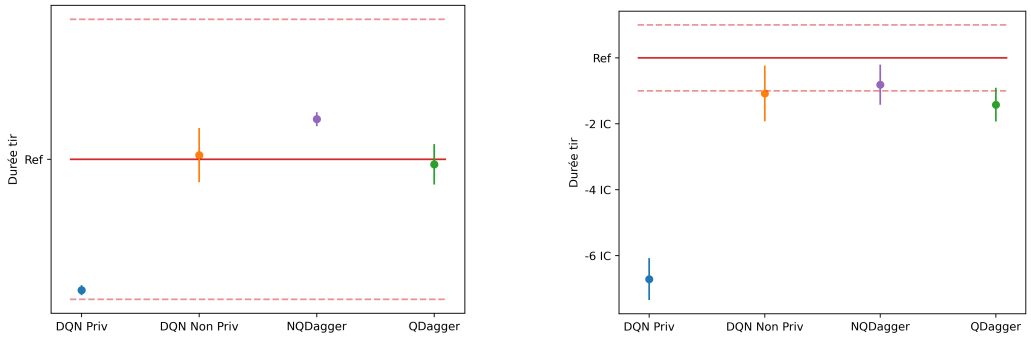
(b) Orientation qualitative - sur SMR

(c) Proximité qualitative -- sur SM$\bar{R}$

(d) Proximité qualitative -- sur SMR

FIGURE 7.18 – Termes de *shaping* d'orientation qualitative - et de proximité qualitative -- sur SM avec un intervalle de confiance à 95% pour les modèles **DQN Priv**, **DQN Non Priv**, **NQDagger** et **QDagger**.

modèle utilisant l'espace d'observation privilégié et un modèle utilisant l'espace d'observation non privilégié. En effet, on observe un gain de temps significatif par rapport à la référence que pour les modèles **QDagger**, et qui reste largement inférieur à celui des modèles **DQN Priv**. Cela corrobore notamment les faibles gains de récompenses de *shaping*.



(a) Durée d'un tir sur SM$\bar{R}$

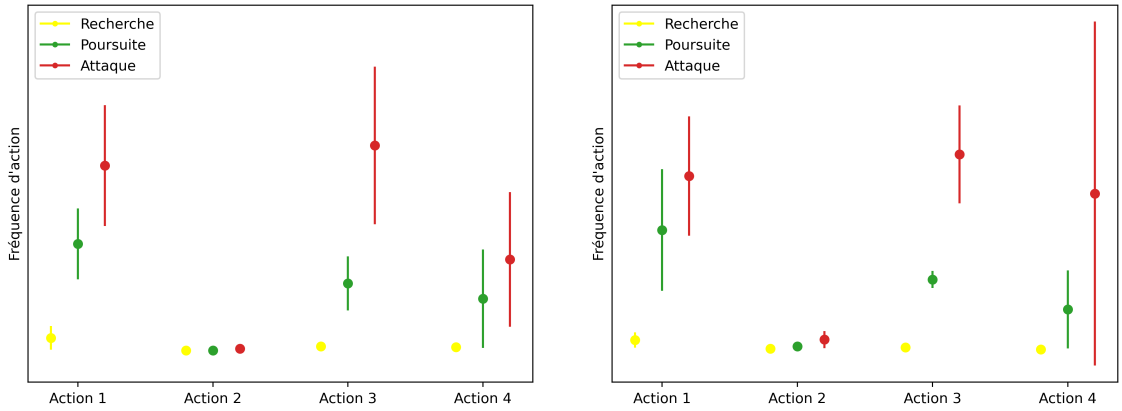
(b) Durée d'un tir sur SMR

FIGURE 7.19 – Durée d'un tir sur SM avec un intervalle de confiance à 95% pour les modèles **DQN Priv**, **DQN Non Priv**, **NQDagger** et **QDagger**.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

Globalement, sur l'ensemble de la figure 7.18, on observe que les modèles **QDagger** sont ceux obtenant des meilleurs résultats, bien que le gain soit faible et pas toujours statistiquement significatif.



(a) DQN Non Priv (b) QDagger

FIGURE 7.20 – Fréquence d'action par action par phase pour les modèles **DQN Non Priv** et **QDagger** avec un intervalle de confiance à 95%.

La figure 7.20 montre aucune différence notable entre les modèles *DQN Non Priv* et *QDagger* sur le choix des actions. Cependant, contrairement au modèle présenté dans la figure 7.14, on voit que l'Action 2 n'est pratiquement pas utilisée par rapport aux autres actions. Le changement d'espace d'observation semble donc conduire à l'incapacité des modèles d'utiliser l'Action 2, ce qui peut notamment expliquer l'absence de gain sur le terme de proximité qualitative -- car c'est l'action qui conduit aux plus grands gains sur ce terme comme le montre la figure 7.9.

Conclusion

Il existe deux machines à états (MAE) sur notre problème industriel : la MAE du logiciel tactique et la MAE d'un modèle opérateur. Nous avons donc étudié les possibilités de les coupler à un algorithme d'apprentissage par renforcement [1]. Suite à cette étude, nous avons catégorisé plusieurs approches de l'état de l'art et proposé de nouvelles.

Jusqu'à ce chapitre, l'ensemble des apprentissages n'avaient utilisé que l'Action 1. Nous avons donc créé des modèles utilisant chacune des quatre actions séparément pour les scénarios BS et SM. Les résultats sur BS sont des modèles avec des performances inacceptables, ce qui a conduit à l'abandon de l'étude ce scénario pour la fin du chapitre. En revanche, pour SM, on observe des gains de performance pour l'ensemble des actions, bien qu'ils soient plus faibles que pour l'Action 1. En outre, on observe que les gains de récompense et la fréquence d'action changent en fonction de l'action considérée, ce qui corrobore le fait que les actions influent sur des aspects différents de la tactique et donc que les modèles apprennent des comportements différents.

L'*action masking* a été essayé sur les Actions 3 et 4 et a conduit à des modèles moins performants a été abandonné.

Suite à ces résultats sur ces apprentissages séparés, des modèles capables d'utiliser toutes les actions ont été étudiés. Plusieurs approches ont été testées : l'apprentissage d'un nouveau modèle *from scratch* et le *transfer learning* à partir des modèles n'utilisant qu'une action. Pour l'approche

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

de *transfer learning*, nous avons adapté l'algorithme *QDagger* pour utiliser plusieurs politiques enseignantes et des espaces d'actions différents. Nos résultats montrent que les approches donnent des résultats similaires mais que l'approche de *transfer learning* semble conduire à une plus faible variabilité des performances des modèles. En outre, les modèles appris conduisent à une meilleure récompense que les modèles n'utilisant qu'une action, cependant, les performances des modèles avec l'Action 1 restent largement supérieures.

Finalement, l'impact de l'utilisation de l'espace d'observation non privilégié est présenté. Plusieurs approches ont été testés : l'apprentissage d'un nouveau modèle *from scratch* et le *transfer learning* à partir des modèles utilisant l'espace d'action privilégiée. Pour cette étude également, nos résultats montrent que ces différentes approches conduisent à des performances similaires mais que celles issues du *transfer learning* ont une plus faible variabilité. Finalement, l'utilisation de l'espace d'observation non privilégié réduit assez faiblement les gains de performances mais réduit fortement les gains de récompense et la fréquence d'action des modèles.

Le chapitre suivant montre les résultats de l'ensemble des chapitres en présentant les meilleurs modèles obtenus au cours de ces travaux de recherche.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

8

Élaboration finale du modèle opérateur

Ce chapitre présente les meilleurs modèles opérateurs élaborés. Tout d'abord, nous exposons des différences conceptuelles et pratiques entre la mise au point du processus d'apprentissage et la création du meilleur modèle possible. Par la suite, nous décrivons les meilleurs modèles d'opérateur obtenus au cours de ces travaux de thèse. Finalement, nous faisons une analyse de sensibilité par rapport à l'ajout de délai entre le choix et l'application de l'action suivi par étude de l'impact d'un changement de scénario.

Sommaire

8.1 Élaboration d'un modèle 130
8.1.1 Différence entre la mise au point d'un processus d'apprentissage et l'élaboration d'un modèle 130
8.1.2 Intérêt de QDagger 130
8.2 Analyse des modèles 131
8.3 Robustesse aux changements du MDP 133
8.3.1 Ajout de délai 133
8.3.2 Changement de scénario 134

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

8.1 Élaboration d'un modèle

8.1.1 Différence entre la mise au point d'un processus d'apprentissage et l'élaboration d'un modèle

Jusqu'à présent dans ces travaux, nous avons présenté la mise au point du processus d'apprentissage des modèles, c'est-à-dire l'ensemble constitué du MDP à résoudre et des algorithmes de DRL utilisés. Cependant, la problématique industrielle de Naval Group est d'obtenir le meilleur modèle possible, peu importe les moyens utilisés. Dans cette optique, il n'y a pas besoin de reproductibilité, ni de prouver que l'approche utilisée est la meilleure mais seulement d'obtenir le meilleur modèle acceptable.

Ainsi, utiliser des méthodes plus incrémentales, c'est-à-dire de repartir de modèles précédents qui n'ont pas forcément été entraînés dans des considérations similaires ou reproductibles, n'est pas un problème car l'on ne cherche pas à comparer de façon équitable des approches. En effet, l'application de l'apprentissage par renforcement implique beaucoup de choix (choix sur la ou les politiques enseignantes utilisées) ce qui démultiplie la difficulté à faire des comparaisons statistiquement significatives.

De même, il est plus facile d'augmenter le nombre de pas de temps alloués à un entraînement car il n'est pas forcément nécessaire de le reproduire plusieurs fois afin d'avoir des résultats statistiquement significatifs.

En outre, utiliser une architecture asynchrone rend également le résultat dépendant du nombre de ressources disponibles car cela modifie le ratio entre le nombre de pas de temps collectés et le nombre de mise à jour. Dans notre cas, un entrainement avec une ferme de calcul pendant quatre heures aura autant de mise à jour du réseau qu'un entraînement utilisant trois fermes de calcul, par contre il aura accès à trois fois moins de transitions différentes au cours de son apprentissage. En effet, les mises à jour sont effectuées sur la même machine peu importe le nombre de fermes. Dans un contexte de ressources partagées, il est donc couteux de s'assurer de la comparabilité des résultats car les ressources calculatoires doivent être les mêmes pour toutes les itérations, ce qui conduit à des ressources potentiellement inutilisées.

Ainsi, bien qu'il soit nécessaire de faire des comparaisons statistiquement significatives sur des choix algorithmiques pour améliorer le processus d'apprentissage et donc les chances d'obtenir de meilleurs modèles, il nous a semblé que, par moment, cela rentrait en conflit avec la volonté d'obtenir le meilleur modèle. Il y a donc énormément de choix qui ont été fait sur des hypothèses métiers et pour lesquels nous n'avons pas pris le temps de tester l'influence sur les résultats.

8.1.2 Intérêt de QDagger

Dans un cadre industriel, il est naturel de vouloir profiter des modèles passés pour accélérer la convergence et tirer profit des calculs passés, surtout quand ils sont couteux et longs. Ainsi, nous avons commencé par initialiser les poids des modèles avec des modèles passés. Cependant, dans un contexte où nous modifions le MDP, beaucoup de modifications rendent le réseau invalide (changement de l'espace d'observation, changement de l'espace d'action, changement de l'architecture du réseau).

Dans ce contexte, l'algorithme *QDagger* [75] nous a permis de s'affranchir de toutes ses contraintes sur la politique enseignante a prouvé son efficacité sur notre cas d'application industrielle. C'est dans cette optique que nous avons proposé l'extension de l'algorithme dans la section 7.4.2 afin d'élargir les cas d'usage possibles.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

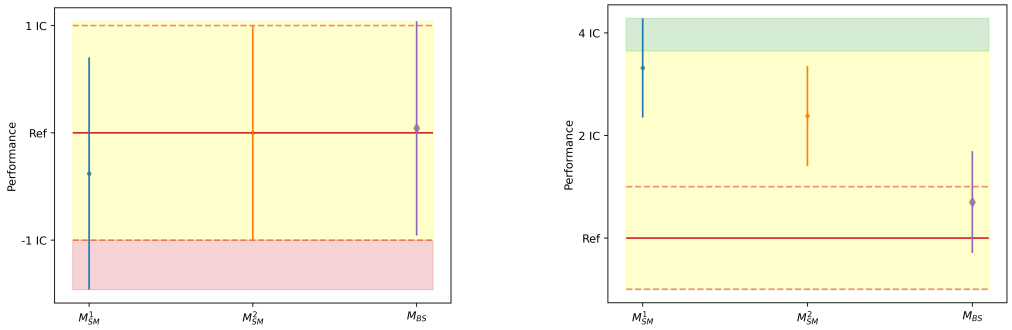
8.2 Analyse des modèles

Au cours de ces travaux, les meilleurs modèles que nous avons réussi à produire sont :

— M^1_{SM} : un modèle entraîné pour le scénario SM capable d'utiliser seulement l'action 1 ($\mathcal{A}_1$). Pour l'obtenir, nous avons commencé par créer un modèle utilisant seulement l'action 1 et l'espace d'observation privilégiée qui a été entraîné pendant 2 millions de pas de temps avec la version de l'algorithme DQN présentée dans le chapitre 6. Puis, nous avons utilisé ce modèle comme politique enseignante avec l'algorithme *QDagger* pendant 2 millions de pas de temps pour créer un modèle qui n'utilise pas d'information privilégiée.

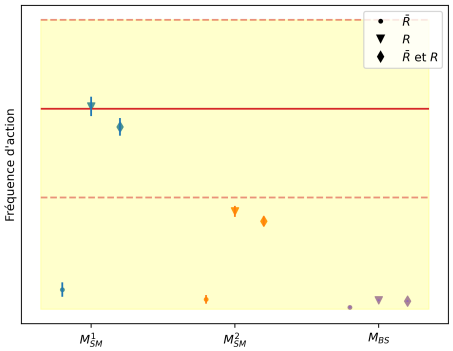
— M^2_{SM} : un modèle entraîné pour le scénario SM capable d'utiliser l'ensemble des actions ($\mathcal{A}$). Pour obtenir ce modèle, nous avons utilisé l'algorithme DQN présentée dans le chapitre 6 pendant 10 millions de pas de temps directement avec l'ensemble des actions et l'espace d'observation non privilégiée.

— M_{BS} : un modèle entraîné pour le scénario BS capable d'utiliser les actions 1, 3 et 4. Comme présenté dans le chapitre 7, l'accès à l'action 2 sur les scénarios BS conduit à du *reward hacking* que nous n'avons pas réussi à résoudre. Pour l'obtenir, nous avons commencé par créer un modèle utilisant les action 1, 3 et 4 et l'espace d'observation privilégiée qui a été entraîné pendant 18 millions de pas de temps avec la version de l'algorithme DQN présentée dans le chapitre 6. Puis, nous avons utilisé ce modèle comme politique enseignante avec l'algorithme *QDagger* pendant 3,2 millions de pas de temps pour créer un modèle qui n'utilise pas d'information privilégiée.



(a) Performance sur $\bar{\text{R}}$

(b) Performance sur R



(c) Fréquence d'action

FIGURE 8.1 – Performances et fréquence d'action des modèles M^1_{SM}, M^2_{SM} et M_{BS} respectivement sur les scénarios SM, SM et BS avec un intervalle de confiance à 95%

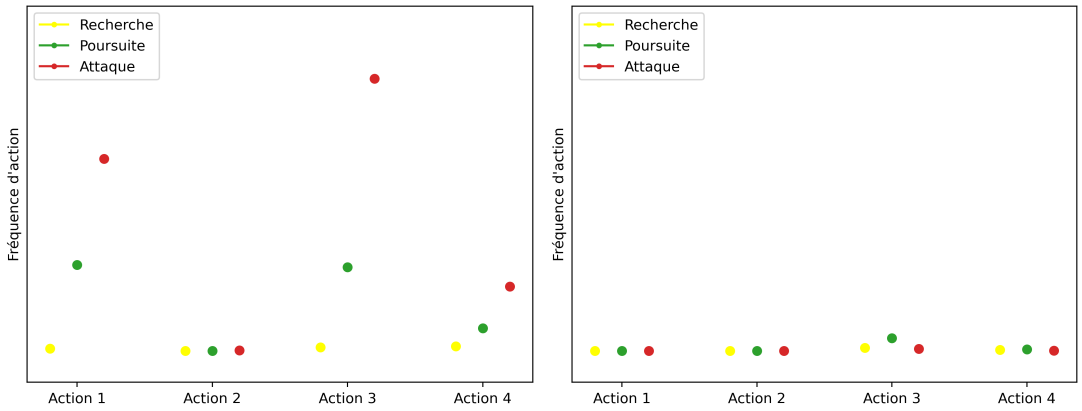
© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

La figure 8.1a montre que l'ensemble des modèles proposés ne font aucun gain de performance par rapport à la référence pour des tirs sans réaction ($\bar{\text{R}}$). Notre hypothèse est qu'il est très difficile d'améliorer les performances de l'arme sur ce type de scénario et donc que les modèles n'ont pratiquement aucune possibilité d'amélioration. En revanche, la figure 8.1b montre que les modèles sont capables de faire des gains de performances sur les scénarios avec réaction. On peut observer sur la figure 8.1b que sur SM, le modèle utilisant seulement l'action 1 obtient des meilleures performances que le modèle avec toutes les actions. Cela est contre-intuitif car le modèle M^2_{SM} a plus de possibilités que le modèle M^1_{SM} puisque $\mathcal{A}_1 \in \mathcal{A}$. Cependant, cette observation est similaire à celle de la section 7.2. Il semble donc que la fonction de récompense est plus facile à maximiser sans gain de performance pour les actions 2, 3 et 4.

La figure 8.1c montre que M^1_{SM} a une fréquence d'action plus faible que M^2_{SM} bien qu'il ait moins d'actions à sa disposition. Cela semble encore une fois contre-intuitif car le fait d'avoir plus d'actions à sa disposition devrait mener à plus de situations différentes où il est intéressant de faire une action. Notre hypothèse est qu'il est plus facile d'apprendre un modèle avec l'espace d'action $\mathcal{A}_1$ que $\mathcal{A}$. On observe également que le modèle M_{BS} a la fréquence d'action la plus faible. Notre interprétation de ce phénomène est que, comme on a pu le voir dans les chapitres 5, 6 et 7, il semble plus difficile de créer un modèle avec des performances supérieures à la référence sur les scénarios BS. Il n'est cependant pas aisé de déterminer si cela vient d'une mauvaise définition du MDP ou si elle est inhérente au scénario BS. Une des perspectives de ces travaux sera d'essayer d'améliorer la définition du MDP pour les scénarios BS.

Pour la suite du chapitre, nous étudierons le modèle M^2_{SM} seulement plutôt que le modèle M^1_{SM} car comme il utilise plusieurs actions, il a donc été jugé plus représentatif d'un opérateur réel.



(a) M^2_{SM} (b) M_{BS}

FIGURE 8.2 – Évaluation de la fréquence d'action par action et par phase du logiciel tactique pour les modèles M^2_{SM} et M_{BS}.

La figure 8.2 illustre les fréquence d'action par action et par phase des deux modèles retenus. On peut observer que pour le modèle M_{BS}, la fréquence d'action la plus forte est dans la phase de Poursuite contrairement au modèle M^2_{SM}, où c'est la phase d'Attaque. En outre, le modèle M^2_{SM} n'utilise pas l'Action 2 et le modèle M_{BS} utilise pratiquement que l'Action 3, un peu l'Action 4 et pas du tout l'Action 1. Ces observations illustrent, une fois encore, les différences conceptuelles et / ou opérationnelles entre les scénarios de type BS et les scénarios de type SM.

En conclusion, les modèles M^1_{SM}, M^2_{SM} et M_{BS} respectent à la fois les contraintes de per-

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

formances et de représentativité sur la fréquence d'action fixées dans le chapitre 3 par Naval Group.

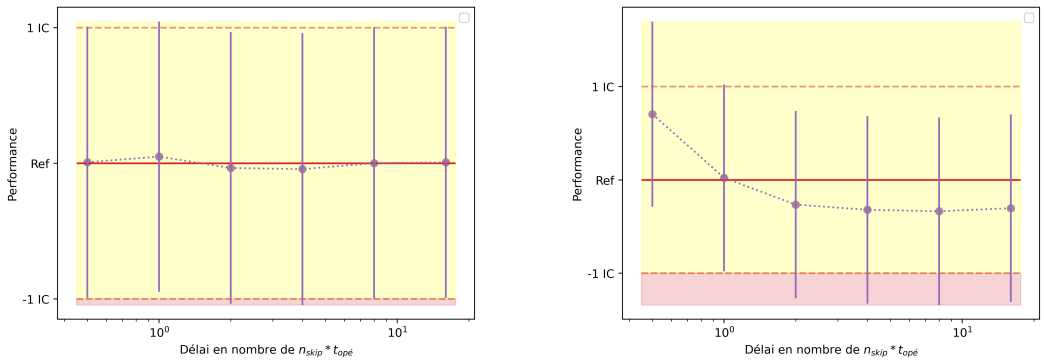
8.3 Robustesse aux changements du MDP

Cette section étudie l'impact de changement du MDP sur les performances et la fréquence d'action des modèles M^2_{SM} et M_{BS}.

8.3.1 Ajout de délai

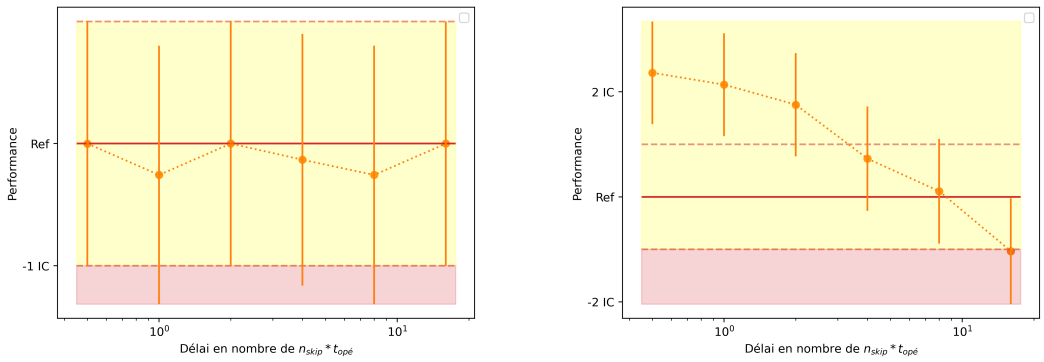
Nos modèles actuels agissent instantanément après l'observation, il n'y a donc actuellement aucune notion de temps de réaction et de compréhension de l'information. Nous avons donc souhaité étudier la sensibilité des modèles créés à un délai entre le choix et l'application de l'action.

Pour implémenter ce délai, nous avons utilisé une file qui prend en mémoire l'action choisie par la modèle et l'applique réellement qu'un certain nombre d'interaction plus tard. Le délai ne peut donc prendre que des valeurs qui sont un multiple de $n_{skip} * t_{op}$ (respectivement la valeur de *frame skip* et le temps d'appel entre deux événements opérateur dans le simulateur). Comme ordre de grandeur, $16 * n_{skip} * t_{op}$ secondes est de l'ordre de plusieurs dizaines de secondes.



(a) M_{BS} sur BS$\bar{\text{R}}$

(b) M_{BS} sur BSR



(c) M^2_{SM} sur SM$\bar{\text{R}}$

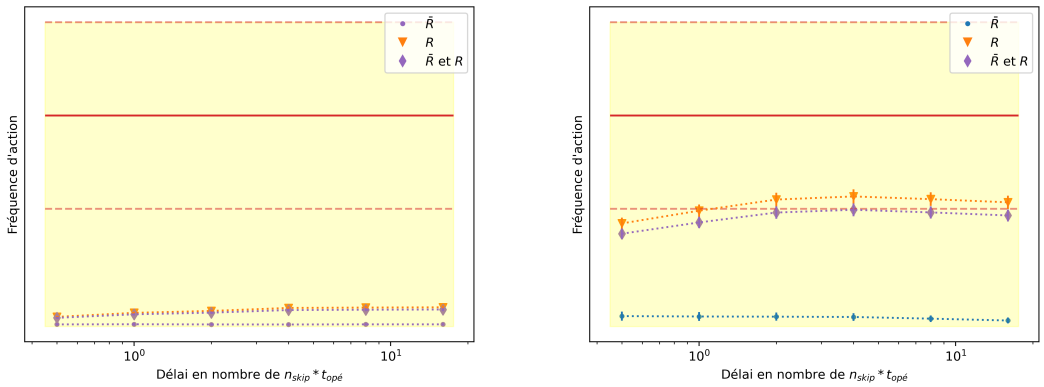
(d) M^2_{SM} sur SMR

FIGURE 8.3 – Performances des modèles M^2_{SM} et M_{BS} respectivement sur les scénarios SM, SM et BS avec un intervalle de confiance à 95% en fonction du délai d'application de l'action. La valeur sans délai est affichée pour la valeur d'abscisse 0.5 pour des raisons d'affichage.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

On observe sur les figures 8.3a et 8.3c aucun effet significatif du délai. Cela semble logique car, comme le montre la figure 8.1c, les modèles M^2_{SM} et M_{BS} n'effectuent pratiquement aucune action quand il n'y a pas de réactions. Cependant, les figures 8.3b et 8.3d montrent une baisse de performance sur les tirs avec une réaction. Comme ces modèles ont une fréquence d'action lorsqu'il y a une réaction, il semble donc cohérent que l'impact du délai soit plus fort. En revanche, la chute de performance reste assez faible ce qui semble penser que faire une bonne action avec du délai est finalement assez peu impactant par rapport à faire une mauvaise action (comme on a pu le voir avec l'ajout d'action aléatoire dans la figure 5.2). Notre interprétation est que comme les actions sont haut niveau et que la majorité des actions est Rien Faire, l'ajout de délai semble moins impactant que pour un problème de contrôle bas niveau où chaque action est une commande distincte avec un effet direct sur le système.



(a) M_{BS} sur BS

(b) M^2_{SM} sur SM

FIGURE 8.4 – Fréquence d'action des modèles M^2_{SM} et M_{BS} respectivement sur les scénarios SM, SM et BS avec un intervalle de confiance à 95% en fonction du délai d'application de l'action. La valeur sans délai est affichée pour la valeur d'abscisse 0.5 pour des raisons d'affichage.

La figure 8.4 montre une augmentation de la fréquence d'action sur les tirs avec réaction en fonction du délai. On peut observer un effet de seuil pour ce phénomène à partir de $4 * n_{skip} * t_{op}$. Notre interprétation de ce phénomène est que le délai ne permet pas au modèle d'agir aussi rapidement que lors de son apprentissage sur le système. Donc si la situation reste similaire, il essaie de nouveau de faire cette même action. Notre interprétation de cet effet de seuil est qu'au bout d'un certain délai, le modèle arrête d'essayer de répéter l'action car son observation du théâtre opérationnel a eu le temps de fortement changer.

8.3.2 Changement de scénario

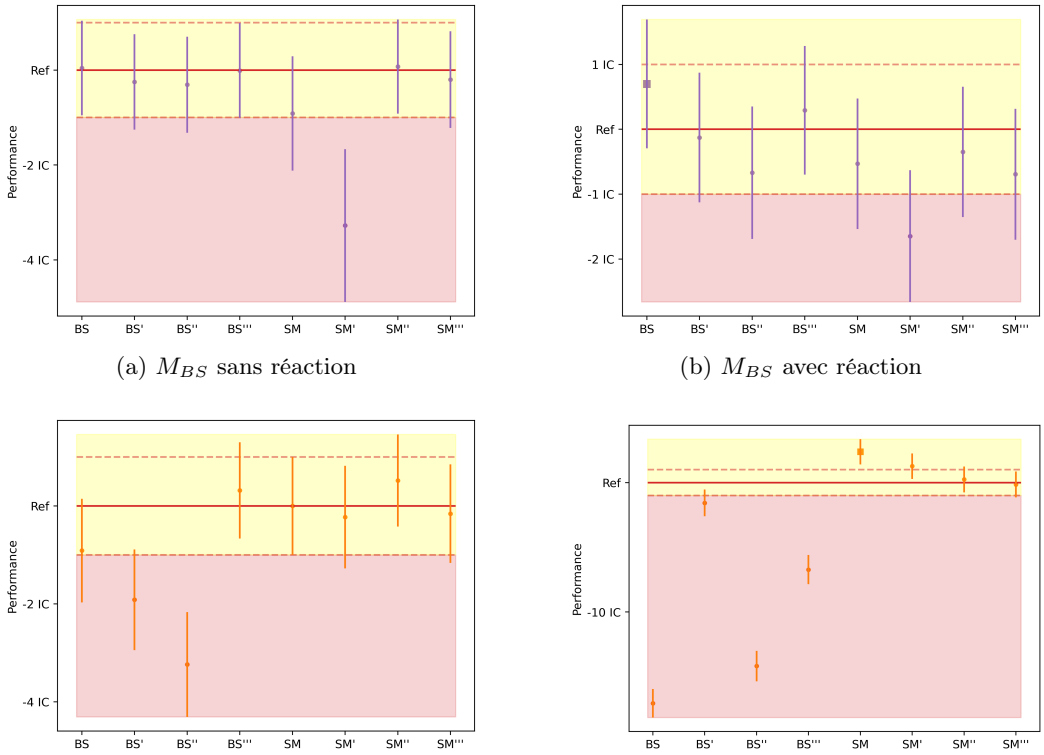
Les scénarios BS et SM considérés pour ces travaux jusqu'à présent sont un sous-ensemble de tous les scénarios utilisés par Naval Group dans le plan d'expérience permettant l'évaluation des performances (comme vu dans la section 1.3). Nous avons donc souhaité observer l'impact d'un changement de scénario sur les modèles M^2_{SM} et M_{BS} en les testant sur un ensemble de scénarios plus large que ceux considérés lors de leurs entraînements respectifs.

Nous avons donc considéré six autres scénarios en plus de SM et BS. Il y a donc trois scénarios dont la cible est un sous-marin (notés SM', SM" et SM"') et trois scénarios dont la cible est un bâtiment de surface (notés BS', BS" et BS"'). Les modifications sont à la fois sur la configuration du théâtre opérationnel, le type de cible et les réactions mises en œuvre par la cible. Chacun

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

de ces scénarios existe avec et sans réaction. Il est important de noter que les performances et intervalle de confiance de référence de ces différents scénarios sont différents.



(a) M_{BS} sans réaction

(b) M_{BS} avec réaction

(c) M^2_{SM} sans réaction

(d) M^2_{SM} avec réaction

FIGURE 8.5 – Performances des modèles M^2_{SM} et M_{BS} respectivement sur les scénarios SM, SM et BS avec un intervalle de confiance à 95% en fonction du scénario.

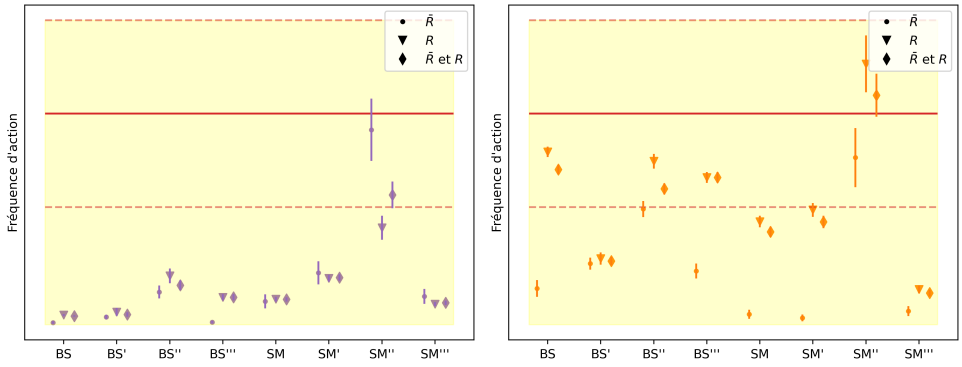
En comparant les figures 8.5a et 8.5b et les figures 8.5c et 8.5d, on note que les variations de performances sont plus importantes pour le modèle M^2_{SM} que pour le modèle M_{BS}. Notre interprétation de ce résultat est que le modèle M_{BS} a globalement appris à faire moins d'action que le modèle M^2_{SM}, peu importe la situation, comme le montre les figures 8.6a et 8.6b.

On peut également observer sur le figure 8.5 que le modèle M^2_{SM} est globalement meilleur sur les scénarios de type SM (SM, SM', SM" et SM"') que sur les scénarios de type BS (BS, BS', BS" et BS"') et inversement pour le modèle M_{BS}. Cela semble donc que les problèmes que doit résoudre le modèle opérateur ont assez de similarités entre différents scénarios et notamment si le type de cible est le même. On peut notamment remarquer que le modèle M^2_{SM} effectue des gains de performance au-delà de l'intervalle de confiance de la référence sur SM' sans aucun réentraînement. En outre, cela confirme le choix d'avoir fait des modèles différents pour les scénarios BS et SM.

Cette analyse est donc intéressante pour des perspectives futures pour Naval Group car il semble donc cohérent de considérer du *transfer learning* entre les différents scénarios et que la définition du MDP présentée dans le chapitre 6 semble assez générique pour être appliquée à d'autres scénarios.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC



(a) M_{BS} (b) M^2_{SM}

FIGURE 8.6 – Fréquence d'action des modèles M^2_{SM} et M_{BS} respectivement sur les scénarios SM, SM et BS avec un intervalle de confiance à 95% en fonction du scénario.

Conclusion

Tout d'abord, nous présentons la différence conceptuelle entre essayer de créer le meilleur processus d'apprentissage et de créer le meilleur modèle. Dans cette optique, l'algorithme *QDagger* nous a paru particulièrement adéquate pour obtenir le meilleur modèle opérateur possible.

Nous présentons les trois meilleurs modèles que nous avons réussi à créer au cours de ces travaux. Ces trois modèles sont acceptables selon les critères de Naval Group. Cependant, les deux modèles sur SM apportent un plus grand gain de performance par rapport à celui sur BS. En outre, pour SM, le modèle qui n'a accès qu'à une partie réduite de l'espace d'action a de meilleures performances, ce qui, avant tout entrainement, n'est pas intuitif. Nous analysons les comportements de deux modèles en fonction des états de la MAE du logiciel. L'ensemble de ces observations mènent à penser qu'il est encore possible d'améliorer les modèles proposés.

Finalement, les modèles générés sont assez peu sensibles à l'ajout de délai entre le choix et l'application d'une action. Nous pensons que cela est dû à l'aspect haut niveau des actions. De plus, les modèles sont assez sensibles au scénario sur lequel ils sont joués et notamment au type de cible. En effet, nous avons pu observer que le modèle entraîné sur le scénario SM se comportait de façon assez performante sur les autres scénarios dont la cible est un sous-marin. En revanche, il a des performances largement inacceptables sur les scénarios dont la cible est un bâtiment de surface. Il semble donc adéquat d'avoir au moins deux modèles opérateurs distincts pour ces deux types de cible.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

9

Conclusion et perspectives

9.1 Conclusion générale

Ce travail de recherche a porté sur la mise en oeuvre d'apprentissage par renforcement profond à une problématique industrielle : la modélisation d'un opérateur humain pour le simulateur industriel de la torpille F21. La première difficulté a été de comprendre et de retranscrire la problématique industrielle en un problème d'apprentissage par renforcement profond. Suite à cela, la deuxième difficulté a été de s'interfacer de façon efficace au simulateur industriel. La troisième difficulté a été de définir puis d'ajuster le problème d'apprentissage et l'algorithme de DRL afin de créer des modèles opérateurs acceptables. Finalement, ces travaux ont été réalisé dans un contexte fortement contraint par la protection du secret de la Défense nationale ce qui a impacté les échanges avec le LORIA et nos publications.

Le chapitre 3 a fait le lien entre le chapitre 1 et le chapitre 2 en décrivant les concepts métiers liés à la modélisation de l'opérateur sous le spectre de l'apprentissage par renforcement. Ainsi, nous avons décrit le problème de décision séquentielle à résoudre par le modèle opérateur. Nous avons donc retranscrit les objectifs de performances et de représentativité attendu par Naval Group en des métriques quantifiables pour définir qu'un modèle soit acceptable tout en respectant les contraintes de communications liées à la sensibilité des données manipulées.

Dans le chapitre 4, nous avons présenté le *wrapper*, c'est-à-dire la partie logicielle développée qui permet d'utiliser le simulateur industriel comme environnement d'apprentissage. Nous avons également montré que le *wrapper* est à la fois modulaire, simple d'utilisation et n'impacte pas les performances calculatoires de façon significative.

Le chapitre 5 a montré les différents développements faits pour définir le MDP à résoudre. Nous avons commencé par expliciter l'espace d'observation utilisé ainsi que sa version privilégiée. Par la suite, nous avons décrit et étudié l'impact des actions haut niveau utilisées. Nous avons proposé plusieurs concepts et formulations de fonctions de récompense, ce qui a nous conduit à une version n'utilisant que des termes de *shaping* et un terme de pénalisation comportemental permettant de moduler la fréquence d'action. Finalement, grâce à des connaissances métiers, nous avons réduit la fenêtre de contrôle laissée au modèle opérateur à un sous-ensemble du tir.

Le chapitre 6 a expliqué comment nous avons choisi de façon empirique le *Deep Q-Learning* (ou DQN) comme base algorithmique. En outre, à cause de la forte variabilité des temps de calcul, nous avons choisi une architecture de parallélisation asynchrone, ce qui a augmenté d'un facteur 10 le nombre de transitions générées. Nous avons alors démontré que des ralentissements corrélés à l'observation peuvent induire un biais sur la politique apprise. Notre étude a également permis de comprendre certains facteurs pouvant amplifier ce biais, ce qui nous a conduit à la

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

conjecture que notre problématique industrielle n'était pas soumis à ce phénomène. Finalement, nous avons essayé quatre modifications algorithmiques à l'algorithme DQN et nous en avons retenu deux.

Nous avons présenté dans le chapitre 7 plusieurs approches pour tirer profit de spécificités du MDP créé. Une étude séparée de chacune des actions a montré que l'impact des quatre actions considérées diffère fortement. Nous avons alors observé que l'ensemble des actions semblent inefficaces sur les scénarios BS alors qu'elles ont toutes conduit à des gains de performance sur SM. L'*action masking* s'est montré inefficace sur les actions qui pouvaient en bénéficier. En outre, nous avons créé des modèles avec un accès à l'ensemble de l'espace d'action et nous avons proposé des modifiées de l'algorithme *QDagger* pour appliquer du *transfer learning* à partir de plusieurs modèles disposant d'un espace d'action différent de la politique apprise. Cet algorithme permet de tirer profit des politiques enseignantes et assez stable mais n'apporte pas de gains significatifs de performance par rapport à un apprentissage utilisant directement l'espace d'action complet. Finalement, nous avons également proposé d'utiliser des politiques enseignantes utilisant l'espace d'observation privilégiée pour guider l'apprentissage mais cela mais n'apporte pas de gains significatifs de performance.

Finalement, le chapitre 8 a présenté les meilleurs modèles d'opérateur et qui sont acceptables d'après les critères définis dans le chapitre 3. Pour les scénarios SM, deux modèles sont présentés avec des gains de performances significatif par rapport à la référence. Cependant, celui qui fait le plus de gain de performances n'utilise qu'une seule action alors que le second modèle utilise toutes les actions mais avec des performances plus faibles, ce qui prouve qu'il est encore possible d'améliorer les modèles obtenus. Pour les scénarios BS, le meilleur modèle créé ne fait qu'un léger gain de performances non significatif mais reste acceptable. Finalement, une étude de sensibilité a montré que ces modèles sont assez peu sensibles à l'ajout de délai mais qu'ils sont fortement impactés par une modification du scénario et plus particulièrement, d'un changement entre une cible de type bâtiment de surface et sous-marin.

Globalement, la mise en œuvre d'apprentissage par renforcement profond au problème industriel était une tâche complexe à cause de la multiplicité des choix à faire (définition du MDP, choix des algorithmes de DRL, choix des hyperparamètres, diversité des scénarios, ...) et de l'absence de référence à laquelle se comparer. Ces difficultés ont été amplifiées par les temps de calcul et la forte stochatisticité des résultats. En effet, il est important de noter que, dans ces travaux, même en s'appuyant sur l'espace d'observation privilégié (i.e. en ayant accès à l'ensemble des observables sans aucun bruit) il n'est déjà pas aisé d'obtenir un modèle opérateur conduisant à une augmentation des performances de l'arme. Il a donc été primordial de pouvoir communiquer et travailler avec des ingénieurs connaissant le contexte industriel (le simulateur et le logiciel tactique) afin de pouvoir s'interfacer correctement avec le simulateur, analyser les résultats obtenus par les modèles et pouvoir faire des choix basés sur des heuristiques métiers (qui ne nécessite donc pas de temps de calcul). Malgré ces difficultés, nous avons tout de même réussi à élaborer des modèles opérateurs acceptables pour Naval Group, avec notamment des gains de performances significatifs sur les tirs contre sous-marin.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

9.2 Perspectives

9.2.1 Amélioration des modèles actuels

Une perspective directe de ces travaux est l'amélioration des modèles actuels autant sur le plan des performances que sur l'aspect de la représentativité humaine.

Amélioration des performances

Une perspective directe est de continuer à itérer sur les modèles actuels pour les améliorer progressivement grâce notamment aux différentes versions de *QDagger* présentée. En effet, le temps nécessaire au développement du processus d'apprentissage et des algorithmes a été très important et, finalement, force est de constater que relativement peu de temps a été consacré à l'apprentissage des modèles. Il semble donc possible d'augmenter les performances en augmentant les temps de calcul tout en réutilisant les meilleurs modèles actuels.

Dans un second temps, il semble également possible d'apporter des améliorations grâce à des modifications dans la définition du MDP ou bien des modifications algorithmiques. Par exemple, la fonction de récompense semble beaucoup moins efficace sur BS que sur SM, il semble donc intéressant d'explorer de nouveaux concepts de termes de *shaping*. Pour les améliorations algorithmiques, il nous semble que l'ajout de réseau récurrent est prometteur car le problème est partiellement observable. En effet, nous n'utilisons qu'une concaténation des deux dernières positions pour y remédier.

Amélioration de la représentativité d'un opérateur humain

Une limite de notre travail est la modélisation de l'aspect humain dans nos modèles. Nous cherchons un comportement qui maximise les performances et dont la fréquence d'action est inférieure ou proche de celle d'un opérateur réel tout en contraignant le temps minimal entre deux actions. Ainsi, nous apprenons un comportement qui maximise le même objectif que l'opérateur et pas à imiter un comportement d'opérateur réel. Cependant, la quantité de données d'opérateur réel rend difficile d'avoir une approche d'apprentissage plus supervisée pour l'aspect humain. Il pourrait être intéressant d'avoir un retour sur les actions prises par les modèles par des opérateurs réels pour valider ou invalider les comportements appris. En outre, il serait intéressant de considérer le *reinforcement learning from human feedback* [127]. En effet, cette méthode utilise l'avis d'un humain afin d'apprendre une fonction de récompense, ainsi, il pourrait être possible de créer une fonction de récompense alignée avec un comportement humain réel grâce à l'aide d'ingénieurs spécialiste en autoguidage ou d'opérateurs réel.

9.2.2 Extension à des problèmes similaires

Application à de nouveaux scénarios

Une extension naturelle est l'apprentissage de modèles sur d'autres scénarios (notamment ceux présentés dans la section 8.3.2). Au vu des résultats, il semble profitable d'utiliser le *transfer learning* entre certains scénarios car les performances des modèles BS et SM ne sont pas très éloignées de la référence sur des scénarios avec le même type de cible que le scénario utilisé lors de l'entraînement. Il est cependant possible que le MDP doive être changé et plus précisément la fonction de récompense. En effet, en fonction du scénario considéré et du type de réaction, l'objectif de l'opérateur peut être différent et il peut donc être intéressant de modifier ou d'ajouter des termes de récompense.

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

Application à de nouvelles problématiques

Sur une perspective plus long terme, le coût de développement d'outils dans un environnement aussi sécurisé est fort, il semble donc intéressant de capitaliser sur les développements faits au cours de ces travaux et les compétences acquises en appliquant le DRL à d'autres problématiques de décisions séquentielles. Par exemple, de façon similaire à l'état de l'art, il peut être possible d'étudier l'application de DRL au contrôle de la torpille afin d'optimiser la trajectoire pour certaine phase. Il peut également être intéressant de considérer les différents bâtiments présents dans le théâtre opérationnel. Par exemple, d'observer s'il est possible d'apprendre un comportement de la cible efficace pour échapper à la torpille F21.

9.2.3 Aide à la décision

La création d'un modèle opérateur efficace peut permettre d'envisager le développement de modules d'aide à la décision pour un opérateur en condition réelle. Cependant, les modèles sont entraînés sur des scénarios bien précis et sur des données simulées, il semble donc nécessaire d'évaluer l'impact d'un changement de contexte entre le simulateur et un contexte réel. En outre, les modèles appris sont réactifs, c'est à dire qu'il ne prévoit pas en avance l'action qui doit être faite car la politique indique l'action qui doit être faite à l'instant. Bien que les modèles semblent relativement résilient à l'ajout d'un délai (voir résultat de la section 8.3.1), il y a tout de même une dégradation des performances ce qui indique qu'il serait intéressant de modifier la façon dont les modèles sont appris. Ainsi, ils pourraient suggérer des actions qui sont efficaces même avec un certain délai car il faut également le temps de réaction et de réflexion d'un opérateur réel face à un module d'aide à la décision.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

A

Informations déductibles d'une performance blanchie par rapport à une valeur de référence

Une approche naturelle pour blanchir des informations est d'utiliser une référence afin d'obtenir des informations relatives et non plus absolues. Cette annexe présente des déductions mathématiques possibles sur des informations chiffrées de performance par rapport à une référence. Elle n'a pas pour but d'être exhaustive sur l'ensemble des déductions possibles mais surtout d'appuyer la nécessité de ne communiquer aucune valeur chiffrée sensible, même relative à une référence.

On note $x \in [0,1]$ la performance réelle et $x_{ref} \in [0,1]$ la valeur de référence utilisée. On considère que les valeurs x et x_{ref} ne doivent pas être connues.

A.1 Majorant / Minorant de performances par rapport à une référence

Cette section décrit les informations données par la communication d'une information relative à une référence pour blanchir des performances afin de ne pas connaître la valeur réelle.

A.1.1 Ratio d'amélioration

Proposition 1. *Soit $x \in [0,1]$ et $x_{ref} \in [0,1]$, on définit $r = \frac{x}{x_{ref}}$. La connaissance de r permet d'assurer que :*

$$x \leq r \tag{A.1}$$

$$x_{ref} \leq \frac{1}{r} \tag{A.2}$$

Démonstration.

$$x = r * x_{ref} \tag{A.3}$$

$$\leq r \tag{A.4}$$

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Car $r > 0$ et $0 \leq x_{ref} \leq 1$.

$$x_{ref} = \frac{x}{r} \tag{A.5}$$
$$\leq \frac{1}{r} \tag{A.6}$$

Car $r > 0$ et $0 \leq x \leq 1$. □

A.1.2 Écart de performance

Proposition 2. *Soit $x \in [0,1]$ et $x_{ref} \in [0,1]$, on définit $d = x - x_{ref}$. La connaissance de d permet d'assurer que :*

$$d \leq x \leq d + 1 \tag{A.7}$$
$$-d \leq x_{ref} \leq 1 - d \tag{A.8}$$

Démonstration.

$$x = d + x_{ref} \tag{A.9}$$
$$d \leq x \leq d + 1 \tag{A.10}$$

Car $0 \leq x_{ref} \leq 1$.

$$x_{ref} = x - d \tag{A.11}$$
$$-d \leq x_{ref} \leq 1 - d \tag{A.12}$$

Car $0 \leq x \leq 1$. □

A.1.3 Écart relatif de performance

Proposition 3. *Soit $x \in [0,1]$ et $x_{ref} \in [0,1]$, on définit $e = \frac{x - x_{ref}}{x_{ref}}$. La connaissance de d permet d'assurer que :*

$$d \leq x \leq d + 1 \tag{A.13}$$
$$-d \leq x_{ref} \leq 1 - d \tag{A.14}$$

Démonstration.

$$x = (e + 1) * x_{ref} \tag{A.15}$$
$$\leq (e + 1) \tag{A.16}$$

Car $e + 1 \geq 0$ et $0 \leq x_{ref} \leq 1$.

$$x_{ref} = \frac{x}{e + 1} \tag{A.17}$$
$$x_{ref} \leq \frac{1}{e + 1} \tag{A.18}$$

Car $e + 1 \geq 0$ et $0 \leq x \leq 1$. □

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

A.2 Informations données par l'intervalle de confiance

On considère un ensemble de $n \in \mathbb{N}$ tirs dont les probabilité de réussite des tirs sont un ensemble de variable aléatoire notées $\{x_k\}_{k\in[1,n]} \in [0,1]^n$. La performance moyenne est notée $\bar{x}$ et l'écart-type de la performance est noté σ.

Le théorème central limite permet de déduire que $\bar{x}$ suit une loi gaussienne. La formule de l'intervalle de confiance à 95 % pour une distribution gaussienne : $IC_{95\%} = 1.96 * \frac{\sigma}{\sqrt{n}}$.

Cette section décrit les informations données par la communication de l'intervalle de confiance d'une performance réelle.

A.2.1 Cas limite avec seulement des échecs/réussite

Pour cette section, on suppose que $\{x_k\}_{k\in[1,n]} \in \{0,1\}^n$, c'est à dire qu'il n'y a que des échecs et des réussites assurées.

On note $n_1 = |\{k, x_k = 1, k \in [0,n]\}|$, le nombre de tir dont la probabilité de réussite est 1 et $n_0 = |\{k, x_k = 0, k \in [0,n]\}|$, le nombre de tir dont la probabilité de réussite est 0.

Proposition 4. *Si* $\{x_k\}_{k\in[1,n]} \in \{0,1\}^n$ *alors* $\bar{x} = \frac{1\mp\sqrt{1-4\sigma^2}}{2}$

Les deux valeurs (à cause du $\mp$) sont symétriques de 0.5. Une connaissance de σ implique donc une connaissance de $\bar{x}$ avec un doute entre une valeur inférieure à 0.5 et une supérieure à 0.5.

Démonstration. On note $n_1 = |\{k, x_k = 1, k \in [0,n]\}|$, le nombre de tir dont la probabilité de réussite est 1 et $n_0 = |\{k, x_k = 0, k \in [0,n]\}|$, le nombre de tir dont la probabilité de réussite est 0.

$$\sigma^2 = \frac{1}{n}\sum_{k=1}^{n}(x_k - \bar{x})^2 \tag{A.19}$$

$$= \frac{1}{n}\left(\sum_{k=1}^{n_1}(1-\bar{x})^2 + \sum_{k=1}^{n_0}\bar{x}^2\right) \tag{A.20}$$

$$= \frac{1}{n}\left(n_1 * (1-\bar{x})^2 + n_0(\bar{x})^2\right) \tag{A.21}$$

Or $n = n_1 + n_0$ et $n_1 = n\bar{x}$.

$$= \frac{1}{n}\left(n * \bar{x} * (1-\bar{x})^2 + (n - n*\bar{x})\bar{x}^2\right) \tag{A.22}$$

$$= \bar{x}(1-\bar{x})^2 + (1-\bar{x})\bar{x}^2 \tag{A.23}$$

$$= \bar{x}(1-\bar{x}) \tag{A.24}$$

Donc $\bar{x}$ est de solution de l'équation, $X^2 - X + \sigma^2 = 0$.

Donc

$$\bar{x} = \frac{1 \mp \sqrt{1-4\sigma^2}}{2} \tag{A.25}$$

□

© Propriété Naval Group SA, (2026) tous droits réservés.

Proposition 5. *Si* $\{x_k\}_{k\in[1,n]} \in \{0,1\}^n$ *alors* $\bar{x} = \frac{1\pm\sqrt{1-\frac{4*n*IC_{95\%}{}^2}{1.96^2}}}{2}$.

Cela implique que la connaissance de l'intervalle de confiance et du nombre de tir permet de connaître la performance pour un ensemble de tir composé seulement de réussites et d'échecs.

Démonstration. Utilisation de la définition de l'$IC_{95\%}$ dans le cas gaussien et de la proposition 4. □

Proposition 6. *Soit deux performances* $\bar{x}_a \in [0,1]$ *et* $\bar{x}_b \in [0,1]$ *différentes et évaluées sur deux ensembles de tirs avec seulement des échecs et des réussites, et le même nombre* n *tir. La connaissance de* Δx^{b-a}_{perf}*, de* $IC^a_{95\%}$ *et de* $IC^b_{95\%}$ *permet de connaître* n*.* n *est l'unique solution de l'équation :*

$$2 * \Delta x^{b-a}_{perf} = \sqrt{1 - \frac{4 * n * IC^b_{95\%}{}^2}{1.96^2}} - \sqrt{1 - \frac{4 * n * IC^a_{95\%}{}^2}{1.96^2}} \tag{A.26}$$

Si pour un même nombre de tir, deux valeurs différentes sont données avec leurs intervalles de confiance, notés $IC^a_{95\%}$ et $IC^b_{95\%}$ et qu'il est possible d'avoir accès à la différence réelle des performances entres les valeurs, notée Δx^{a-b}_{perf} (comme une méthode la section A.1.2), il est alors possible de déduire n d'une combinaison de performances relatives et de leurs intervalles de confiance. Il est donc possible de déduire la performance absolue avec la proposition 4.

Démonstration. L'utilisation de la définition de l'$IC_{95\%}$ dans le cas gaussien et de la proposition 4 pour $\bar{x}_a$ et $\bar{x}_b$ permet de déduire que :

$$2 * \Delta x^{b-a}_{perf} = \sqrt{1 - \frac{4 * n * \mathrm{IC}^b_{95\%}{}^2}{1.96^2}} - \sqrt{1 - \frac{4 * n * \mathrm{IC}^a_{95\%}{}^2}{1.96^2}} \tag{A.27}$$

On pose la fonction $f_{a,b} : x \mapsto \sqrt{1-ax} - \sqrt{1-bx}$, définie pour $x \in]0, \frac{1}{\max(a,b)}]$ avec $a, b \in \mathbb{R}^{+*2}$. Soit $x \in 0, \frac{1}{\max(a,b)}]$.

$$f'_{a,b}(x) = \frac{b}{2\sqrt{1-bx}} - \frac{a}{2\sqrt{1-ax}} \tag{A.28}$$

$$= \frac{b\sqrt{1-ax} - a\sqrt{1-bx}}{2\sqrt{1-bx}\sqrt{1-ax}} \tag{A.29}$$

$$= \frac{\left(b\sqrt{1-ax} - a\sqrt{1-bx}\right)\left(b\sqrt{1-ax} + a\sqrt{1-bx}\right)}{2\sqrt{1-bx}\sqrt{1-ax}\left(b\sqrt{1-ax} + a\sqrt{1-bx}\right)} \tag{A.30}$$

$$= \frac{(a-b)\left(a^2+b^2+abx\right)}{2\sqrt{1-bx}\sqrt{1-ax}\left(b\sqrt{1-ax} + a\sqrt{1-bx}\right)} \tag{A.31}$$

$$\tag{A.32}$$

Comme $a > 0$, $b > 0$ et $x > 0$, on a $\left(a^2+b^2+abx\right) > 0$.
Donc $f'_{a,b}$ est soit strictement positive, soit strictement négative en fonction du signe de $a-b$.
Donc $f_{a,b}$ est injective.
Avec $a = \frac{4*\mathrm{IC}^a_{95\%}{}^2}{1.96^2}$ et $b = \frac{4*\mathrm{IC}^b_{95\%}{}^2}{1.96^2}$, $2*\Delta x^{b-a}_{perf}$ admet donc un unique antécédent par la fonction $f_{a,b}$ qui est n. □

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

A.2.2 Cas général

Proposition 7. *Soit $\bar{x}$ et σ, une probabilité moyenne et son écart-type associé à l'ensemble de variables aléatoires $\{x_k\}_{k\in[1,n]} \in [0,1]^n$.*

$$|\bar{x} - \frac{1}{2}| \geq \frac{\sqrt{1-4\sigma^2}}{2} \tag{A.33}$$

Avec égalité seulement dans le cas où $\{x_k\}_{k\in[1,n]} \in \{0,1\}^n$

La connaissance de σ permet donc d'avoir une connaissance sur la valeur.

Démonstration. Pour le cas de l'égalité, voir la proposition 4.

Soit $\bar{x}$ et σ, une probabilité moyenne et son écart-type associé à l'ensemble de variables aléatoires $\{x_k\}_{k\in[1,n]} \in [0,1]^n$.

Si $\{x_k\}_{k\in[1,n]}$ n'est pas à valeur dans $\{0,1\}^n$, alors il existe un ensemble $E = \{k \in [1,n] \; | x_k \neq 0 et x \neq 1\}$ non vide. On note n_E le cardinal de cet ensemble. On note n_0 et n_1 le nombre d'éléments dont la probabilité vaut respectivement 0 et 1. Cela induit que :

$$n = n_0 + n_1 + n_E \tag{A.34}$$

$$\bar{x} = \frac{1}{n}\left(n_1 + \sum_{k\in E} x_k\right) \tag{A.35}$$

$$\iff \quad |\bar{x} - \frac{1}{2}| > \frac{\sqrt{1-4\sigma^2}}{2} \tag{A.36}$$

$$\iff \quad \bar{x}(1-\bar{x}) < \sigma^2 \tag{A.37}$$

$$\iff \frac{1}{n}\left(n\bar{x}(1-\bar{x})^2 + (n-n\bar{x})\bar{x}^2\right) < \frac{1}{n}\sum_{k\in[0,n]} (\bar{x} - x_k)^2 \tag{A.38}$$

$$\iff \quad n\bar{x}(1-\bar{x})^2 + (n-n\bar{x})\bar{x}^2 < n_1(1-\bar{x})^2 + n_0\bar{x}^2 + \sum_{k\in E} (\bar{x} - x_k)^2 \tag{A.39}$$

$$\tag{A.40}$$

En utilisant, les équations A.34 et A.35, on obtient :

$$\iff \sum_{k\in E} x_k(1-\bar{x})^2 + \left(n_E - \sum_{k\in E} x_k\right)\bar{x}^2 < \sum_{k\in E} (\bar{x} - x_k)^2 \tag{A.41}$$

$$\iff \quad \sum_{k\in E} x_k < \sum_{k\in E} x_k^2 \tag{A.42}$$

Or,

$$\forall k \in E,\; 0 < x_k < 1 \tag{A.43}$$

$$\forall k \in E,\; x_k^2 < x_k \tag{A.44}$$

Comme E est non vide, $\sum_{k\in E} x_k < \sum_{k\in E} x_k^2$. □

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

Proposition 8. *Si* $\{x_k\}_{k\in[1,n]} \in [0,1]^n$ *alors* $|\bar{x} - \frac{1}{2}| \geq \frac{\sqrt{1-\frac{4*n*IC_{95\%}{}^2}{1.96^2}}}{2}$.

Cela implique que la connaissance de l'intervalle de confiance et du nombre de tir permet de connaître des informations sur la valeur moyenne dans le cas général.

Démonstration. Utilisation de la définition de l'$IC_{95\%}$ dans le cas gaussien et de la proposition 7. □

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

B

Définition des différents termes de *shaping*

B.1 Distance à la cible

Cet ensemble de termes a pour objectif d'encourager la torpille à se rapprocher de la cible. On note d_t la distance de la cible à l'instant t.

B.1.1 DistQuant

La fonction **DistQuant** est une fonction quantitative utilisant l'évolution de la distance à la cible. Elle récompense de façon proportionnelle à l'évolution de la distance de la cible entre l'instant $t-1$ et t. Elle est définie par :

$$r_t(s_{t-1}, s_t) = d_{t-1} - d_t \tag{B.1}$$

B.1.2 DistQual+

La fonction **DistQual+** est une fonction qualitative utilisant l'évolution de la distance à la cible. Elle récompense positivement une réduction de la distance à la cible. Elle est définie par :

$$r_t(s_{t-1}, s_t) = \begin{cases} 1\,, \text{ si } d_{t-1} - d_t > 0 \\ 0\,, \text{ sinon} \end{cases} \tag{B.2}$$

B.1.3 DistQual-

La fonction **DistQual-** est une fonction qualitative utilisant l'évolution de la distance à la cible. Elle récompense négativement une augmentation de la distance à la cible. Elle est définie par :

$$r_t(s_{t-1}, s_t) = \begin{cases} 0\,, \text{ si } d_{t-1} - d_t > 0 \\ -1\,, \text{ sinon} \end{cases} \tag{B.3}$$

B.2 Orientation à la cible

Cet ensemble de termes a pour objectif d'encourager la torpille à s'orienter vers la cible. On note c_t la circulaire de la cible à l'instant t.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

B.2.1 OriQuant

La fonction **OriQuant** est une fonction quantitative utilisant l'orientation de la torpille. Elle récompense de façon proportionnelle la valeur absolue de la circulaire de la cible, ce qui encourage donc d'avoir une valeur faible de circulaire et donc une torpille alignée avec la cible. Elle est définie par :

$$r_t(s_t) = -|c_t| \tag{B.4}$$

B.2.2 OriQual+

La fonction **OriQual+** est une fonction qualitative utilisant l'orientation de la torpille. Elle récompense positivement le fait que la cible soit devant la torpille, c'est-à-dire que la circulaire se trouve entre -90 et 90 degrés. Elle est définie par :

$$r_t(s_t) = \begin{cases} 1\text{ , si } |c_t| < 90 \\ 0\text{ , sinon} \end{cases} \tag{B.5}$$

B.2.3 OriQual-

La fonction **OriQual-** est une fonction qualitative utilisant l'orientation de la torpille. Elle récompense négativement le fait que la cible soit derrière la torpille, c'est-à-dire que la valeur absolue de la circulaire soit supérieure à 90 degrés. Elle est définie par :

$$r_t(s_t) = \begin{cases} 0\text{ , si } |c_t| < 90 \\ -1\text{ , sinon} \end{cases} \tag{B.6}$$

B.3 Proximité de piste à la cible

Cet ensemble de termes a pour objectif d'encourager le logiciel tactique à avoir une piste proche de la cible. On note p_t la distance entre la piste et la cible à l'instant t.

B.3.1 ProxQuant

La fonction **ProxQuant** est une fonction quantitative utilisant la distance de la piste à la cible. Une valeur de seuil (fixée par expertise métier) permet de délimiter la distance seuil, noté d_{seuil}, à partir de laquelle il est mieux de ne pas avoir de piste plutôt qu'une piste erronée.

$$r_t(s_t) = \begin{cases} d_{seuil} - p_t\text{ , s'il existe une piste} \\ 0\text{ , sinon} \end{cases} \tag{B.7}$$

B.3.2 ProxQual+

La fonction **ProxQual+** est une fonction qualitative utilisant la distance de la piste à la cible. Une valeur de seuil (fixée par expertise métier) permet de délimiter la distance maximum entre la piste et à la cible à partir de laquelle la récompense est positive.

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

$$r_t(s_t) = \begin{cases} 1\,, \text{ s'il existe une piste et } p_t < d_{seuil} \\ 0\,, \text{ sinon} \end{cases} \tag{B.8}$$

B.3.3 ProxQual-

La fonction **ProxQual-** est une fonction qualitative utilisant la distance de la piste à la cible. Une valeur de seuil (fixée par expertise métier) permet de délimiter la distance minimum entre la piste et à la cible à partir de laquelle la récompense est négative.

$$r_t(s_t) = \begin{cases} -1\,, \text{ s'il existe une piste et } p_t > d_{seuil} \\ 0\,, \text{ sinon} \end{cases} \tag{B.9}$$

B.3.4 ProxQual++

La fonction **ProxQual++** est une fonction qualitative utilisant la distance de la piste à la cible. Une valeur de seuil (fixée par expertise métier) permet de délimiter la distance maximum entre la piste et à la cible à partir de laquelle la récompense est positive. En outre, le fait de ne pas avoir de piste est légèrement récompensée afin d'encourager de ne pas avoir de piste plutôt qu'une piste à une distance supérieure à d_{seuil}.

$$r_t(s_t) = \begin{cases} 1\,, \text{ s'il existe une piste et } p_t < d_{seuil} \\ 0\,, \text{ s'il existe une piste et } p_t > d_{seuil} \\ 0.1\,, \text{ s'il n'existe pas de piste} \end{cases} \tag{B.10}$$

B.3.5 ProxQual--

La fonction **ProxQual--** est une fonction qualitative utilisant la distance de la piste à la cible. Une valeur de seuil (fixée par expertise métier) permet de délimiter la distance minimum entre la piste et à la cible à partir de laquelle la récompense est négative. En outre, le fait de ne pas avoir de piste est légèrement pénalisée afin d'encourager d'avoir une piste à une distance inférieure à d_{seuil} plutôt que de ne pas avoir de piste.

$$r_t(s_t) = \begin{cases} 0\,, \text{ s'il existe une piste et } p_t < d_{seuil} \\ -1\,, \text{ s'il existe une piste et } p_t > d_{seuil} \\ -0.1\,, \text{ s'il n'existe pas de piste} \end{cases} \tag{B.11}$$

© Propriété Naval Group SA, (2026) tous droits réservés.
Corporate sensitivity PUBLIC

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

C

Implémentation du ralentissement

C.1 Planification des mises à jour

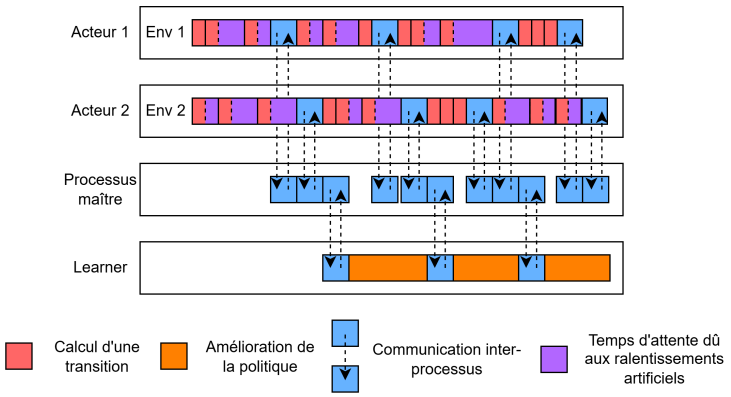


FIGURE C.1 – Agencement des tâches effectuées par les différents processus avec une architecture asynchrone pour un algorithme de DRL avec des ralentissements artificiels.

La figure C.1 illustre l'ajout de ralentissement artificiel sur une architecture asynchrone. Ces temps supplémentaires induisent que le *learner* fait plus de mise à jour pour le même nombre de transition générées par les *actors*.

La figure C.2 montre l'impact de cette modification sur les performances de l'apprentissage. Cet effet est déjà connu [62] et n'affecte pas la distribution des données. Nous avons donc créé un planificateur de mise à jour (ou *update scheduler*) qui planifie les mises à jour au cour de l'apprentissage en fonction du nombre de transitions générées. Cette méthode élimine l'influence des ralentissements sur le nombre de mises à jour effectuées. Ainsi, cela nous permet de n'étudier que l'effet des ralentissements sur la distribution des données générées. Nous avons donc utilisé un planificateur de mise à jour qui planifie les mises à jour comme un apprentissage qui n'a aucun ralentissement artificiel.

C.2 Architecture asynchrone simulée

L'utilisation des ralentissements décrits dans la section C.1 induit une non-représentativité car les temps de communication et d'apprentissage par rapport à notre environnement d'apprentissage. En outre, l'ajout de ralentissements réels ralentit fortement le temps d'un apprentissage,

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

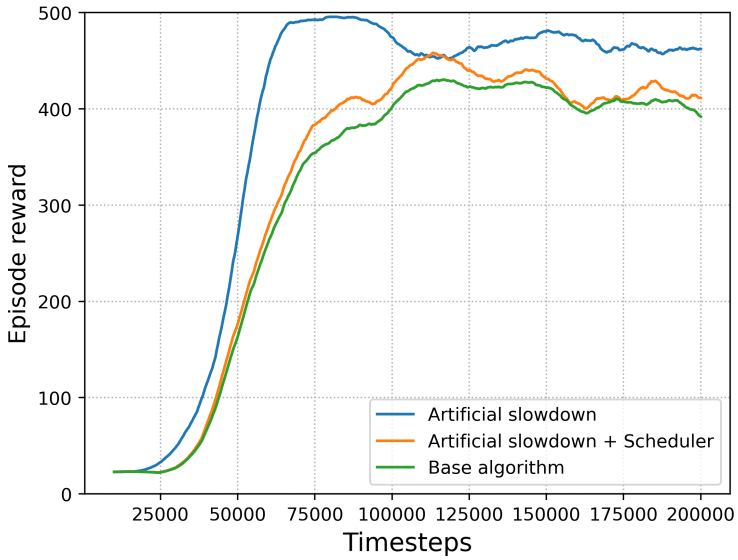


FIGURE C.2 – Apprentissage avec l'algorithme DQN et une architecture asynchrone sur l'environnement Cartpole avec un coefficient de ralentissement de 10 sur toutes les transitions et un planificateur de mise à jour moyenné sur 25 germes.

ce qui limite fortement les valeurs de coefficient de ralentissement utilisable.

Nous avons donc créé une version simulée de l'architecture asynchrone qui se base sur le mécanisme de simulation à événement discret. L'idée principale est de ne plus attendre réellement les ralentissements artificiels en utilisant un temps simulé. Pour cela, à partir du coefficient de ralentissement, du temps réel pour calculer la transition et de la trajectoire, il est possible d'évaluer le temps simulé (c.-à-d. qui prend en compte les ralentissements artificiel) auquel chacun des acteurs aurait dû finir sa tâche. Ainsi, à chaque fois, l'acteur dont le temps simulé est le plus proche du temps actuel ajoute ses transitions générées au *replay buffer*, et sa prochaine tâche est simulée afin d'ajouter le temps de résolution de celle-ci à la boucle d'événement.

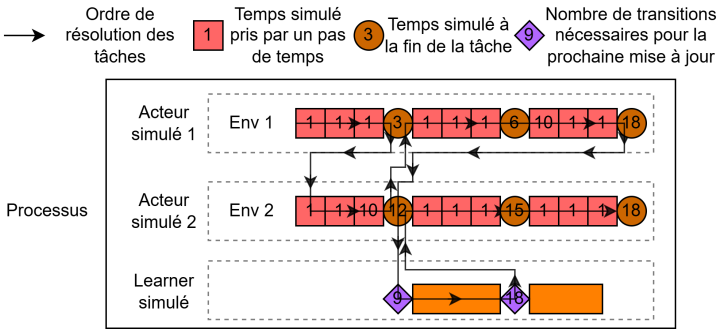


FIGURE C.3 – Agencement des tâches effectuées par les différents processus avec une architecture asynchrone simulée pour un algorithme de DRL avec des ralentissements artificiels.

Les figures C.3 et C.4 illustrent l'ordre de résolution des tâches pour un cas avec 2 *actors* où chaque acteur simule 3 transitions où certaines d'entre elles subissent un coefficient de ralentissement de 10, et le *learner* est planifié pour une mise à jour toutes les 9 transitions ajoutées au *replay buffer*.

Les avantages de cette version de l'algorithme est qu'elle permet de négliger les temps de com-

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

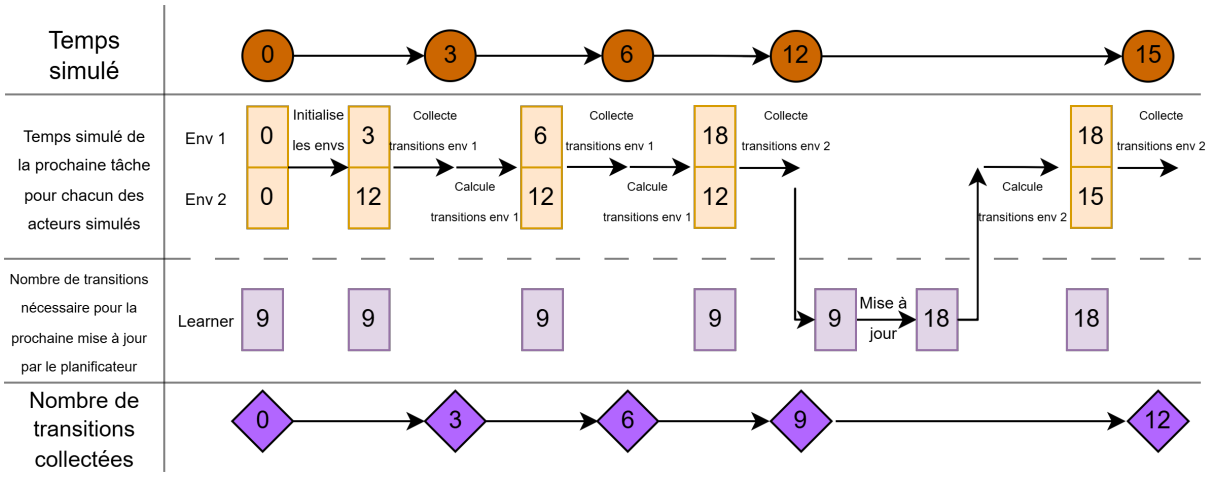


FIGURE C.4 – Gestion des tâches par la boucle événementielle pour l'architecture asynchrone simulée de la figure C.3.

munication et d'apprentissage et d'utiliser une valeur arbitraire de coefficient de ralentissement tout en gardant un temps de calcul constant. En outre, cette version est hautement parallélisable car elle est mono-processus.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

D

Paramètres utilisés pour Cartpole

Paramètre	Valeur
Architecture du réseau	[64, 64, 32]
Taux d'apprentissage	0.001
Optimiseur	Adam
Nombre de pas de temps	200000
Nombre d'*actors*	16
Nombre de *learner*	1
Pas de temps par tâche par *actor*	200
Mise à jour par tâche pour le *learner*	32
Pas de temps par mise à jour pour le *learner*	1024
γ	0.99
n-step	5
Valeur initiale pour ε	1
Valeur finale pour ε	0.005
Valeur finale de réduction pour ε	50000
Nombre de pas de temps pour initialiser le *replay buffer*	10000
Taille du *replay buffer*	50000
x_{zone}	0.3
Coefficient de ralentissement pour la zone droite	1
Coefficient de ralentissement pour la zone gauche	1

TABLE D.1 – Paramètres par défaut utilisés pour *Cartpole*

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

Bibliographie

[1] Abdallah I., Ciarletta L., Hénaff P., Champagne J. et Bonavent M., "Couplage entre un apprentissage par renforcement profond et une machine à états : approche théorique", *CAID - Conference on Artificial Intelligence for Defense, DGA, Nov 2024, Rennes, France.*

[2] Abdallah I., Ciarletta L., Hénaff P., Champagne J. et Bonavent M., "Effect of a slow-down correlated to the current state of the environment on an asynchronous learning architecture", *Reinforcement Learning Journal, 2025, pages 384-398.*

[3] `https://fr.wikipedia.org/wiki/Naval_Group`

[4] `https://fr.wikipedia.org/wiki/Torpille_F21`

[5] `https://cyber.gouv.fr/reglementation/cybersecurite-systemes-dinformation/protection-du-secret/instruction-generale-interministerielle-n1300/`

[6] `https://fr.wikipedia.org/wiki/Simulation_%C3%A0_%C3%A9v%C3%A9nements_discrets`

[7] `https://fr.wikipedia.org/wiki/P%C3%A9gase_(1975)`

[8] Hong D., Kim M. et Park S. "Study on Reinforcement Learning-Based Missile Guidance Law.", *Applied Sciences 10*, 2020.

[9] Tang J., Bai Z., Liang Y., Zheng F. et Li K., "An Exoatmospheric Homing Guidance Law Based on Deep Q Network", *International Journal of Aerospace Engineering*, 2022.

[10] Hong D., Sungsu P., "Avoiding Obstacles via Missile Real-Time Inference by Reinforcement Learning", *Applied Sciences 12*, 2022.

[11] Liang Y., Tang J., Bai Z. et Li K., "Homing Guidance Law Design against Maneuvering Targets Based on DDPG", *International Journal of Aerospace Engineering*, 2023.

[12] Wang X., Deng Y., Cai Y. et Jiang, H., "Deep Recurrent Reinforcement Learning for Intercept Guidance Law under Partial Observability" *Applied Artificial Intelligence*, 2024.

[13] Xu W., Yifan D., Yuanli C. et Haonan J., "An event-triggered deep reinforcement learning guidance algorithm for intercepting maneuvering target of speed advantage.", *ISA Transactions*, 2025, pages 9-22.

[14] Hwang K. et Kim J., "Wake Homing Torpedo Guidance Using a Hierarchical Deep Reinforcement Learning Framework.", *IEEE Access, 2025*, pages 72938-72952.

[15] Elywn E., "Man and Machine - Systems for Safety". *Outlook on Safety : Proceedings of the Thirteenth Annual Technical Symposium.*, 1972

[16] Shteingart H. et Loewenstein Y., "Reinforcement learning and human behavior", *Current Opinion in Neurobiology*, 2014, pages 93-98.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

[17] Sutton R. et Barto A., "Reinforcement Learning : An Introduction", *Second Edition, MIT Press, 2018, isbn : 978-0-262-03924-6.*

[18] Ziebart B. D., Maas A. L., Bagnell J. A. et Dey A. K., "Human Behavior Modeling with Maximum Entropy Inverse Optimal Control.", *AAAI Spring Symposium : Human Behavior Modeling*, 2009.

[19] Ruan J., "Reinforcement Learning-based Human Operator Decision Support Agent for Highly Transient Industrial Processes", *Purdue University Graduate School*, 2024.

[20] Haberland S., Ruge H. et Frimmel H., "Advances in deep reinforcement learning enable better predictions of human behavior in time-continuous tasks.", *PLoS One*, 2025.

[21] Human-like Decision Making for autonomous vehicles at the intersection using inverse reinforcement learning

[22] Chen D., Zhou B., Koltun V. et Krähenbühl P., "Learning by Cheating.", *Proceedings of Machine Learning Research, 2019, pages 66-75.*

[23] Khan R. A., Serpiva V., Aschalew D., Fedoseev A. et Tsetserukou D., "AgilePilot : DRL-Based Drone Agent for Real-Time Motion Planning in Dynamic Environments by Leveraging Object Detection.", *International Conference on Unmanned Aircraft Systems, 2025.*

[24] Silver D., Huang A., Maddison C. J., Guez A., Sifre L., van den Driessche G., Schrittwieser J., Antonoglou I., Panneershelvam V., Lanctot M., Dieleman S., Grewe D., Nham J., Kalchbrenner N., Sutskever I., Lillicrap T., Leach M., Kavukcuoglu K., Graepel T. et Hassabis D., "Mastering the game of Go with deep neural networks and tree search", *Nature, 2016.*

[25] Silver D., Hubert T., Schrittwieser J., Antonoglou I., Lai M., Guez A., Lanctot M., Sifre L., Kumaran D., Graepel T., Lillicrap T. P., Simonyan K. et Hassabis D., "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm.", *Computing Research Repository, 2017.*

[26] Mnih V., Kavukcuoglu K., Silver D., Graves A., Antonoglou I., Wierstra D., et Riedmiller M., "Playing Atari with Deep Reinforcement Learning", *Présenté dans le Deep Learning and Representation Learning Workshop à NIPS, 2013.*

[27] Vinyals O., Ewalds T., Bartunov S., Georgiev P., Vezhnevets A. S., Yeo M., Makhzani A., Küttler, H., Agapiou J., Schrittwieser J., Quan J., Gaffney S., Petersen S., Simonyan K., Schaul T., van Hasselt H., Silver D., Lillicrap T. P., Calderone K., Keet P., Brunasso A., Lawrence D., Ekermo A., Repp J. et Tsing R., "StarCraft II : A New Challenge for Reinforcement Learning.", *Computing Reasearch Repository, 2017.*

[28] Sutton R., "Learning to predict by the methods of temporal differences", *Machine Learning, Volume 3, 1988, pages 9-44.*

[29] Watkins C., et Dayan P., "Q-learning", *Machine Learning, Volume 8, 1992, pages 279-292.*

[30] Goodfellow I., Bengio Y., et Courville A., "Deep Learning", *MIT Press, 2016, isbn : 978-0-262-03561-3.*

[31] Cybenko G., "Approximation by Superpositions of a Sigmoidal Function", *Mathematics of Control, Signals, and Systems, 1989, pages 303-314.*

[32] Kingma D., et Ba J., "Adam : A Method for Stochastic Optimization", *The 3rd International Conference for Learning Representations, 2014.*

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

[33] Hochreiter S., et Schmidhuber J., "Long Short-Term Memory", *Neural Computation, 1997, pages 1732-1780.*

[34] Chung J., Gulcehre C., ,KyungHyun C., et Bengio Y., "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling", *Présenté dans le Deep Learning and Representation Learning Workshop à NIPS, 2014.*

[35] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A., Kaiser L., et Polosukhin I., "Attention is All you Need", *The Thirtieth Annual Conference on Neural Information Processing Systems, 2017.*

[36] OpenAI, "GPT-4 Technical Report", *arXiv : 2303.08774, 2023.*

[37] Baird L., "Residual Algorithms : Reinforcement Learning with Function Approximation", *The 12th International Conference on Machine Learning, 1995, pages 30-37.*

[38] `https://nusit.nus.edu.sg/technus/minimizing-time-training-deep-reinforcement/`

[39] Vinyals O., Babuschkin I., Czarnecki W., Mathieu M., Dudzik A., Chung J., Choi D., Powell R., Ewalds T., Georgiev P., Oh J., Horgan D., Kroiss M., Danihelka I., Huang A., Sifre L., Cai T., Agapiou J., Jaderberg M., Vezhnevets A., Leblond R., Pohlen T., Dalibard V., Budden D., Sulsky Y., Molloy J., Paine T., Gülçehre C., Z. Wang, Pfaff T., Wu Y., Ring R., Yogatama D., Wünsch D., McKinney K., Smith O., Schaul T., LillicrapT., Kavukcuoglu K., Hassabis D., Apps C., et Silver D., "Grandmaster level in StarCraft II using multi-agent reinforcement learning.", *Nature, 350-354, 2019.*

[40] Baker B., Kanitscheider I., Markov T., Wu Y., Powell G., McGrew B., et Mordatch I., "Emergent Tool Use From Multi-Agent Autocurricula.", *The 9th International Conference for Learning Representations, 2020.*

[41] `https://openai.com/research/openai-five`

[42] Dulac-Arnold G., Levine N., Mankowitz D., Li J., Paduraru C., Gowal S. et Hester T., "An empirical investigation of the challenges of real-world reinforcement learning.", *Computing Research Repository, 2020.*

[43] Bellemare M. G., Naddaf Y., Veness J. et Bowling M., "The Arcade Learning Environment : An Evaluation Platform for General Agents", *Computing Research Repository, 2012.*

[44] Schulman J., Moritz P., Levine S., Jordan M. et Abbeel P., "High-Dimensional Continuous Control Using Generalized Advantage Estimation", *The 5th International Conference for Learning Representations, 2016.*

[45] Schulman J., Wolski F., Dhariwal P., Radford A. et Klimov O., "Proximal Policy Optimization Algorithms.", *Computing Research Repository, 2017.*

[46] Haarnoja T., Zhou A., Abbeel P. et Levine S., "Soft Actor-Critic : Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor", *The 35th International Conference on Machine Learning, 2018, pages 1861-1870.*

[47] Hessel M., Modayil J., van Hasselt H., Schaul T., Ostrovski G., Dabney W., Horgan D., Piot B., Azar M., et Silver D., "Rainbow : Combining Improvements in Deep Reinforcement Learning.", *The 32nd Annual AAAI Conference on Artificial Intelligence, AAAI Press, 2018, pages 3215-3222.*

[48] van Hasselt H., "Double Q-learning", *The 23rd Neural Information Processing Systems, 2010, pages 2613–2621.*

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

[49] van Hasselt H, Guez A. et Silver D., "Deep Reinforcement Learning with Double Q-Learning.", *The 30th Annual AAAI Conference on Artificial Intelligence, AAAI Press, 2016, pages 2094-2100.*

[50] Wang Z., Schaul T., Hessel M., van Hasselt H., Lanctot M. et de Freitas N., "Dueling Network Architectures for Deep Reinforcement Learning", *The 33rd International Conference on Machine Learning, 2016.*

[51] Schaul T., Quan J., Antonoglou I. et Silver D., "Prioritized experience replay.", *The 4th International Conference for Learning Representations, 2015.*

[52] Fortunato M., Azar M. G., Piot B., Menick J., Osband I., Graves A., Mnih, V., Munos R., Hassabis D., Pietquin O., Blundell C., et Legg S., "Noisy networks for exploration". *Computing Research Repository, 2017.*

[53] Bellemare M. G., Dabney, W., et Munos R., "A distributional perspective on reinforcement learning.", *Proceedings of Machine Learning Research, 2017, pages 449-458.*

[54] Badia A. P., Piot B., Kapturowski S., Sprechmann P., Vitvitskyi A., Guo Z. D. et Blundell C., "Agent57 : Outperforming the Atari Human Benchmark.", *The 37th International Conference on Machine Learning, 2020, pages 507-517.*

[55] Lin L. J., "Self-Improving Reactive Agents Based on Reinforcement Learning, Planning, and Teaching", *Machine Learning, 1992, pages 293-321.*

[56] Fedus W., Ramachandran P., Agarwal R., Bengio Y., Larochelle H., Rowland M. et Dabney W., "Revisiting Fundamentals of Experience Replay.", *Computing Research Repository, 2020.*

[57] Bhatt A., Palenicek D., Belousov B., Argus M., Amiranashvili A., Brox T. et Peters J., "CrossQ : Batch Normalization in Deep Reinforcement Learning for Greater Sample Efficiency and Simplicity.", *The 13th International Conference for Learning Representations, 2024.*

[58] Nair A., Srinivasan P., Blackwell S., Alcicek C., Fearon R., Maria A., Panneershelva, V., Suleyman M., Beattie C., Petersen S., Legg S., Mnih V., Kavukcuoglu K. et Silver D., "Massively Parallel Methods for Deep Reinforcement Learning.", *Computing Research Repository, 2015.*

[59] Mnih V., Badia A. P., Mirza M., Graves A., Lillicrap T. P., Harley T., Silver D. et Kavukcuoglu K., "Asynchronous Methods for Deep Reinforcement Learning.", *The 33rd International Conference on Machine Learning, 2016, pages 1928-1937.*

[60] Recht B., Ré C., Wright S. J. et Niu F., "Hogwild : A Lock-Free Approach to Parallelizing Stochastic Gradient Descent.", *The 24th Neural Information Processing Systems, 2011, pages 693-701.*

[61] Horgan D., Quan J., Budden D., Barth-Maron G., Hessel M., van Hasselt H. et Silver D., "Distributed Prioritized Experience Replay.", *Computing Research Repository, 2018.*

[62] Espeholt L., Soyer H., Munos R., Simonyan K., Mnih V., Ward T., Doron Y., Firoiu V., Harley T., Dunning I., Legg S., et Kavukcuoglu K., "IMPALA : Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures.", *Proceeding of Machine Learning Research, 2018, pages 1406-1415.*

[63] Espeholt L., Marinier R., Stanczyk P., Wang K. et Michalski M., "SEED RL : Scalable and Efficient Deep-RL with Accelerated Central Inference.", *The 9th International Conference for Learning Representations, 2020.*

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

[64] Raina R., Madhavan A. et Ng A. Y., "Large-scale deep unsupervised learning using graphics processors.", *The 26th International Conference on Machine Learning, 2009, pages 873-880.*

[65] Hausknecht M. J. et Stone P., "Deep Recurrent Q-Learning for Partially Observable MDPs.", *Computing Research Repository, 2015.*

[66] Kapturowski S., Ostrovski G., Quan J., Munos R. et Dabney W., "Recurrent Experience Replay in Distributed Reinforcement Learning.", *The 8th International Conference for Learning Representations, 2019.*

[67] Taylor M. E. et Stone P., "Transfer Learning for Reinforcement Learning Domains : A Survey", *Journal of Machine Learning Research, 2009, pages 1633-1685.*

[68] Zhu Z., Lin K., Jain A. et Zhou J., "Transfer Learning in Deep Reinforcement Learning : A Survey.", *IEEE Transaction on Pattern Analysis and Machine Intelligence, 2023, pages 13344-13362.*

[69] Mu T. et Su H., "Boosting Reinforcement Learning and Planning with Demonstrations : A Survey", *arXiv :2303.13489, 2023.*

[70] Osa T., Pajarinen J., Neumann G., Bagnell J., Abbeel P. et Peters J., "An Algorithmic Perspective on Imitation Learning", *Foundations and Trends in Robotics, 2018, pages 1-179.*

[71] Pomerleau D., "ALVINN : An Autonomous Land Vehicle In a Neural Network", *Advances in Neural Information Processing Systems, 1989.*

[72] Hester T., Vecerík M., Pietquin O., Lanctot M., Schaul T., Piot B., Horgan D., Quan J., Sendonaris A., Osband I., Dulac-Arnold G., Agapiou J. P., Leibo J. Z. et Gruslys A., "Deep Q-learning From Demonstrations.", *The 32nd Annual AAAI Conference on Artificial Intelligence, 2018, pages 3223-3230.*

[73] Gülçehre Ç., Paine T. L., Shahriari B., Denil M., Hoffman M., Soyer H., Tanburn R., Kapturowski S., Rabinowitz N. C., Williams D., Barth-Maron G., Wang Z., de Freitas N. et Team Worlds, "Making Efficient Use of Demonstrations to Solve Hard Exploration Problems.", *The 9th International Conference for Learning Representations, 2020.*

[74] Ross S., Gordon G. et Bagnell D. "A reduction of imitation learning and structured prediction to no-regret online learning", *Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pages 627–635.*

[75] Agarwal R., Schwarzer M., Castro P., Courville A. et Bellemare M., "Reincarnating Reinforcement Learning : Reusing Prior Computation to Accelerate Progress.", *The 35th Annual Conference on Neural Information Processing Systems, 2022.*

[76] Haarnoja T., Tang H., Abbeel P. et Levine S., "Reinforcement Learning with Deep Energy-Based Policies.", *Proceedings of Machine Learning Research, 2017, pages 1352-1361.*

[77] Cai Y., Liu X., Oikonomou A. et Zhang K., "Provable Partially Observable Reinforcement Learning with Privileged Information.", *The 37th Annual Conference on Neural Information Processing Systems, 2024.*

[78] Chen D., Zhou B., Koltun V. et Krähenbühl P., "Learning by Cheating.", *Proceedings of Machine Learning Research, 2019, pages 66-75.*

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

[79] Nguyen H., Baisero A., Wang D., Amato C. et Platt R., "Leveraging Fully Observable Policies for Learning under Partial Observability.", *Proceedings of Machine Learning Research, 2022, pages 1673-1683.*

[80] Wang J., Yu Z., Zhou D., Shi J. et Deng R., "Vision-Based Deep Reinforcement Learning of UAV Autonomous Navigation Using Privileged Information.", *Computing Research Repository, 2024.*

[81] Li J., Zhao E., Wei T., Xing J. et Xiang S., "Leveraging Privileged Information for Partially Observable Reinforcement Learning.", *IEEE Trans. Games, 2025, pages 765-776.*

[82] Kang G.-C. et Lee Y., "Finite State Machine-Based Motion-Free Learning of Biped Walking", *IEEE Access, t. 9, p. 20 662-20 672, 2021, issn : 2169-3536.*

[83] Hwang S., Lee K., Jeon H. et Kum D., "Autonomous Vehicle Cut-In Algorithm for Lane-Merging Scenarios via Policy-Based Reinforcement Learning Nested Within Finite-State Machine", *IEEE Transactions on Intelligent Transportation Systems, 2022, pages 17 594-17 606.*

[84] Icarte R. T., Klassen T. Q., Valenzano R. A. et McIlraith S. A., "Reward Machines : Exploiting Reward Function Structure in Reinforcement Learning.", *Journal of Artificial Intelligence Research, 2022, pages 173-208.*

[85] Huang S. et Ontañón S., "A Closer Look at Invalid Action Masking in Policy Gradient Algorithms.", *Florida Online Journal, 2022.*

[86] Berner C., Brockman G., Chan B., Cheung V., Debiak P., Dennison C., Farhi D., Fischer Q., Hashme S., Hesse C., Józefowicz R., Gray S., Olsson C., Pachocki J., Petrov M., de Oliveira P., Henrique P., Raiman J., Salimans T., Schlatter J., Schneider J., Sidor S., Sutskever I., Tang J., Wolski F. et Zhang S., "Dota 2 with Large Scale Deep Reinforcement Learning.", *Computing Reasearch Repository, 2019.*

[87] Ye D., Liu Z., Sun M., Shi B., Zhao P., Wu H., Yu H., Yang S., Wu X., Guo Q., Chen Q., Yin Y., Zhang H., Shi T., Wang L., Fu Q., Yang W. et Huang L., "Mastering Complex Control in MOBA Games with Deep Reinforcement Learning.", *The 34th Annual AAAI Conference on Artificial Intelligence, 2020, pages 6672-6679.*

[88] Zimmer M., Boniface Y. et Dutech A., "Developmental Reinforcement Learning through Sensorimotor Space Enlargement.", *IEEE International Conference on Development and Learning and Epigenetic Robotics, pages 33–38, 2018.*

[89] Wu Y. et Tian Y., "Training Agent for First-Person Shooter Game with Actor-Critic Curriculum Learning.", *The 6th International Conference for Learning Representations (Poster session), 2017.*

[90] Chaffre T., Moras J., Chan-Hon-Tong A. et Marzat J., "Sim-to-Real Transfer with Incremental Environment Complexity for Reinforcement Learning of Depth-based Robot Navigation.", *ICINCO, 2020, pages 314-323.*

[91] Narvekar S., Peng B., Leonetti M., Sinapov J., Taylor M. et Stone P., "Curriculum Learning for Reinforcement Learning Domains : A Framework and Survey.", *Journal of Machine Learning Resarch, 2020, pages 181 :1-181 :50.*

[92] `https://rl4rs.github.io/RL4RS/`, "Workshop on Practical Insights into RL for Real Systems", *Reinforcement Learning Conference, 2025.*

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

[93] Delfosse Q., Blüml J., Gregori B., Sztwiertnia S. et Kersting K., "OCAtari : Object-Centric Atari 2600 Reinforcement Learning Environments.", *Reinforcement Learning Journal, 2024, pages 400-449.*

[94] Todorov E., Erez T. et Tassa Y., "Mujoco : Aphysics engine for model-based control", *Intelligent Robot and Systems, 2012, pages 5026-5033.*

[95] Dosovitskiy A., Ros G., Codevilla F., Lopez A. et Koltun V., "CARLA : An Open Urban Driving Simulator", *Proceeding of the 1st Annual Conference on Robot Learning, 2017, pages 1-16.*

[96] Barto A. G., Sutton R. S. et Anderson C. W., "Neuronlike adaptive elments that can solve difficult learning control problems.", *IEEE Transactions on Systems, Man, and Cybernetics, 1983.*

[97] Brockman G., Cheung V., Petterson L., Schneider J., Schulman J., Tang J. et Zaremba W., "Openai gym", *arXiv :1606 ;01540, 2016.*

[98] Paszke A., Gross S., Massa F., Lerer A., Bradburry J., Chanan G., Kileen T., Lin Z., Gimelshein N., Antiga L., Desmaison A., Köpf A., Yang E. Z., DeVito Z., Raison M., Tejani A., Chilamkurthy S., Steiner B., Fang Lu., Bai J. et Chintala S., "PyTorch : An Imperative Style, High-Performance Deep Learning Theory.", *The 32nd Annual Conference on Neural Information Processing Systems, 2019, pages 8024-8035.*

[99] Abadi M., Barham P., Chen J., Chen Z., Davis A., Dean J., Devin M., Ghemawat S., Irving G., Isard M., Kudlur M., Levenberg J., Monga R., Moore S., Murray D. G., Steiner B., Tucker P. A., Vasudevan V., Warden P., Wicke M., Yu Y. et Zhan X., "TensorFlow : A sytem for large-scale machine learning.", *Computing research repository, 2016.*

[100] Bradbury J., Frostig R., Hawkins P., Johnson M. J., Leary C., Maclaurin D., Necula G., paszke A., Vanderplas J., Wanderman-Milne S. et Zhang Q., "JAX : composable transformation of Python+Numpy programs.", `http://github.com/jax-ml/jax`, *2018.*

[101] Raffin A., Hill A., Gleave A., Kanervisto A., Ernestus M. et Dormann N., "Stable-Baselines3 : Reliable Reinforcement Learning Implementations.", *Journal of Machine Learning Research, 2021, pages 268 :1-268 :8.*

[102] Huang S., Dossa R. F. J., Ye C., Braga J., Chakraborty D., Mehta K. et Araùjo J. G. M., "CleanRL : High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms.", *Journal of Machine Learning Research, 2022, pages 274 :1-274 :18.*

[103] Guadarram S., Korattikara A., Ramirez O., Castro P., Holly E., Fishman S., Wang K., Gonina E., Wu N., Kokiopoulou E., Sbaiz L., Smith J., Bartok G., Berent J., Harris C., Vanhoucke V. et Brevdo E., "TF-Agents : A library for Reinforcement Learning in TensorFlow", `https://github.com/tensorflow/agents`, *2018.*

[104] Hoffman M. W., Shahriari B., Slanides J., Barth-Maron G., Momchev N., Sinopalnikov D., Stanczyk P., Ramos S., Raichuk A., Vincent D., Hussenot L., Dadashi R., Dulac-Arnold G., Orsini M., Jacq A., Ferret J., Vieillard N., Ghasemipour S. K. S., Girgin S., Pietquin O., Behbahani F., Norman T., Abdomaleki A., Cassirer A., Yang F., Baumli K., Henderson S., Friesen A., Haroun R., Novikov A., Colmenarejo S. G., Cabi S., Gulcehre C, Le Paine T., Sinicasan S., Cowie A., Wang Z., Piot B. et de Freitas N., "Acme : A Research Framework for Distributed Reinforcement Learning", *arXiv :2006.00979, 2020.*

© Propriété Naval Group SA, (2026) tous droits réservés.

[105] Lange R. T., "gymnax : A JAX-based Reinforcement Learning Environment Library", `https://github.com/RoberTLange/gymnax`, *2022.*

[106] Liang E., Liwa R., Nishihara R., Moritz P., Fox R., Goldberg K., Gonzalez J., Jordan M. I. et Stoica I., "RLlib : Abstractions for Distributed Reinforcement Learning.", *The 35th International Conference on Machine Learning, 2018, pages 3059-3068.*

[107] Bou A., Bettini M., Dittert S., Kumar V., Sodhani S., Yang X., Fabritis G. D. et Moens V., "TorchRL : A data-driven decision-making library for PyTorch.", *The 13th International Conference for Learning Representations, 2024.*

[108] Silver D., Singh S., Precup D. et Sutton R. S., "Reward is Enough.", *Artificial Intelligence, 2021, 103535.*

[109] Bowling M., Martin J. D., Abel D. et Dabney W., "Settling the Reward Hypothesis.", *The 40th International Conference on Machine Learning, 2023, pages 3003-3020.*

[110] Gupta A., Pacchiano A., Zhai Y., Kakade S. M. et Levine S., *Unpacking Reward Shaping : Understanding the Benefits of Reward Engineering on Sample Complexity.*, *The 35th Annual Conference on Neural Information Processing Systems, 2022.*

[111] Amodei D., Olah C., Steinhardt J, Christiano P., Schulman J. et Mané D., "Concrete Problems in AI Safety", *arXiv :1606..06565, 2016.*

[112] Dewey D., "Reinforcement Learning and the Reward Engineering Principle.", *AAAI Spring Symposia, 2014.*

[113] Ibrahim S., Mostafa M., Jnadi A. et Osinenko P., "Comprehensive Overview of Reward Enfineering and Shaping in Advancing Reinforcement Learning Applications.", *Computing Research Repository, 2024.*

[114] Machado M. C., Bellemare M. G., Talvitie E., Veness J., Hausknechtt M. J. et Bowling M., "Revisiting the Arcade Learning Environment : Evaluation Protocols and Open Problems for General Agents", *Journal of Artificial Intelligence, 2018, pages 523-562.*

[115] Kalyanakrishnan S., Aravindan S., Bagdawat V., Bhatt V., Goka H., Gupta A., Krishna K. et Piratla V., "An Analysis of Frame-skipping in Reinforcement Learning", *arXiv :2102.03718, 2021.*

[116] Bacon P.-L., Présentation à "Workshop on Practical Insights into RL for Real Systems", *Reinforcement Learning Conference, 2025.*

[117] Agarwal R., Schwarzer M., Castro P., Courville A. et Bellemare M., "Deep Reinforcement Learning at the Edge of the Statistical Precipice.", *The 34th Annual Conference on Neural Information Processing Systems, 2021, pages 29304-29320.*

[118] Becktepe J., Dierkes J., Benjamins C., Mohan A., Salinas D., Rajan R., Hutter F., Hoos H., Lindauer M. et Eimer T., "ARLBench : Flexible and Efficient Benchmarking for Hyperparameter Optimization in Reinforcement Learning.", *17th European Workshop on Reinforcement Learning, 2024.*

[119] Dierkes J., Cramer E., Hoos H. H. et Trimpe S., "Combining Automated Optimisation for Hyperparameters and Reward Shape", *Reinforcement Learning Journal, 2024, pages 1441-1466.*

[120] Aitchison M., Sweetser P. et Hutter M., "Atari-5 : Distilling the Arcade Learning Environment down to Five Games.", *The 40th International Conference on Machine Learning, 2023, pages 421-438.*

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

[121] Obando-Ceron J. S. et Castro P. S., "Revisiting Rainbow : Promoting insightful and inclusive deep reinforcement learning research", *The 38th International Conference on Machine Learning, 2021, pages 1373-1383.*

[122] Mathieu T., Meideros Centa M., Della Vecchia R., Kohler H., Shilova A., Maillard O. A. et Preux P., "AdaStop : sequential testing for efficient and reliable comparisons of Deep RL Agents", *16th European Workshop on Reinforcement Learning, 2023.*

[123] Moritz P., Nishihara R., Wang S., Tumanov Alexey, Liaw R., Liang E., Paul W., Jordan M. I. et Stoica I., "Ray : A Distributed Framework for Emerging AI Applications", *Computing Reasearch Repository, 2017.*

[124] Icarte R., Klassen T., Valenzano R., Castro M., Waldie E. et McIlraith, "Learning reward machines : A study in partially observable reinforcement learning.", *The 32nd Annual Conference on Neural Information Processing Systems, 2019.*

[125] Furelos-Blanco D., Law M., Russo A., Broda K. et Jonsson A. , "Induction of Subgoal Automata for Reinforcement Learning", *The 34th Annual AAAI Conference on Artificial Intelligence, 2020, pages 3890-3897.*

[126] Rusu A. A., Colmenarejo S. G., Gulcehre C., Desjardins G., Kirkpatrick J., Pascanu R., Mnih V., Kavukcuoglu K., et Hadsell R., "Policy distillation", *arXiv preprint arXiv :1511.06295*, 2015.

[127] Christiano P. F., Leike J., Brown T. B., Martic M., Legg S. et Amodei D., "Deep Reinforcement Learning from Human Preferences.", *The 34th Annual Conference on Neural Information Processing Systems, 2017, pages 4299-4307.*

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

NAVAL GROUP

Résumé

Ces travaux de recherche s'inscrivent dans le contexte industriel du développement de la torpille lourde F21, une arme sous-marine. Une fois tirée, l'arme est autonome et s'appuie sur une intelligence embarquée (dit logiciel tactique). Cette intelligence embarquée est elle-même mis au point à l'aide d'un simulateur de combat naval capable de modéliser une multitude de théâtres opérationnels. Pour être le plus représentatif possible, Naval Group modélise l'ensemble des phénomènes physiques (par exemple : acoustique sous-marine) et les différents acteurs présents sur le théâtre opérationnel, comme l'opérateur à bord du lanceur supervisant le tir torpille. Comme tous autres éléments du théâtre opérationnel, il est donc nécessaire de modéliser l'opérateur. Cependant, les modèles d'opérateur développés jusqu'à présent ne sont pas satisfaisants.

L'objectif de cette thèse est donc d'élaborer un modèle réactif d'opérateur humain pour augmenter la représentativité du simulateur en utilisant l'apprentissage automatique.

L'apprentissage par renforcement profond (DRL) a été choisi car la problématique industrielle est un problème de décisions séquentielles pour lequel on a accès à un simulateur. En outre, cette méthode a montré sa capacité à résoudre des problèmes complexes issus de multiples domaines (jeux vidéo, jeux de société, contrôle robotique). Cette thèse concerne donc une application de l'apprentissage par renforcement profond à un simulateur industriel (le simulateur industriel de combat naval).

Cependant, les algorithmes de DRL sont connus pour être difficile à appliquer à des problèmes réels. En effet, ils nécessitent une quantité de données importantes ce qui rend les temps de calculs importants. En outre, ils sont à la fois stochastiques et instables ce qui rend les comparaisons coûteuses pour être statistiquement significatives. Finalement, les hyperparamètres sont nombreux et ont un fort impact sur les performances, ce qui amplifie une fois encore la difficulté à régler le processus d'apprentissage par renforcement.

Tout d'abord, nous présentons l'adaptation du simulateur industriel en un environnement d'apprentissage permettant d'utiliser le DRL. C'est une tâche complexe et hautement dimensionnante pour les résultats finaux. Elle mêle une utilisation de connaissances métiers et de connaissances sur l'apprentissage par renforcement afin de créer un environnement permettant d'apprendre des modèles répondant à la problématique industrielle.

Par la suite, nous étudions les résultats de plusieurs algorithmes de DRL. Nous avons choisi l'algorithme Dueling Deep Q-Learning avec une architecture de parallélisation asynchrone. Nous présentons plusieurs approches pour tirer profits des spécificités de notre environnement. Nous étudions donc la capacité de coupler une MAE au DRL, l'impact d'un Curriculum learning au niveau de l'espace d'action, l'utilisation d'action masking et l'utilisation d'information privilégiée.

Finalement, nous présentons les performances des meilleurs modèles obtenus et dans quelle mesure ils répondent au besoin industriel. En outre, nous faisons une étude de sensibilité en perturbant les modèles avec l'ajout de délai et en modifiant les scénarios. Ces travaux ont abouti à un modèle opérateur à la fois performant et représentatif selon les critères de l'industriel Naval Group.

Mots-clés: Apprentissage par renforcement profond, Application industrielle, Modélisation d'un humain, Défense, Militaire, Transfer Learning, Information privilégiée

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC

Abstract

This reasearch work takes place in the industrial context of the development of the F221 heavy torpedo, a submarine weapon. Once fired, the weapon is autonomous and relies on embedded intelligence (known as tactical software). This embedded intelligence is itself developed using a naval combat simulator capable of modeling a multitude of operational theaters. To be as representative as possible, Naval Group models all the physical phenomena (for example : underwater acoustics) and the various actors present in the operational theater ; it is therefor necessary to model the operator. However, the operator models developped so far ar not satisfactory.

The objective of this thesis is therefore to create a reactive human operator model to increase the representativeness of the simulator using machine learning.

Deep reinforcement learning (DRL) was chosen because the industrial problem is a sequential decision problem for which we have access to a simulator. In addition, this method has shown its ability to solve complex problems from multiple fields (video games, board games, robotic control). This thesis concerns an application of deep reinforcement learning to an industrial simulator (the naval combat simulator).

However, DRL algorithms are known to be difficult to apply to real problems. Indeed, they require a large amount of data, which makes the computation time significant. Furthermore, they are both stochastic and unstable which makes comparisons costly to be statistically significant. Finally, the hyperparameters are numerous and have a strong impact on performance, which once again amplifies the difficulty of adjusting the reinforcement learning process.

First, we present the adaptation of the industrial simulator into a learning environment for using DRL. It is a complex and highly dimensioning task for the final results. It combines knowledge on the use case and knowledge on reinforcement learning in order to create an environment for learning models that fulfil the industrial needs.

Next, we study the outcomes of several DRL algorithms. We chose the Dueling Deep Q-Learning algorithm with an asynchronous parallelization architecture. We present several approaches to take advantage of the specificities of our environment. We therefore study the ability to couple a FSM to the DRL, the impact of applying curriculum learning on the action space, the use of masking actions, and the use of privileged information.

Finally, we present the performance of the best models obtained and to what extent they meet industrial needs. In addition, we conduct a sensitivity study by disrupting the models with addition of delays and modifying the scenarios. This work concerns operator models that are both efficient and representative according to the criteria of the industrial company Naval Group.

Keywords: Deep reinforcement learning, Industrial case, Human modeling, Defence, Military, Transfer learning, Privileged information

Corporate sensitivity
PUBLIC

© Propriété Naval Group SA, (2026) tous droits réservés.

© Propriété Naval Group SA, (2026) tous droits réservés.

Corporate sensitivity
PUBLIC