Universitat
de les Illes Balears

TREBALL DE FI DE GRAU

APRENENTATGE PER REFORÇ APLICAT A ENTORNS 3D: IMPLEMENTACIÓ I ANÀLISI D'AGENTS AUTÒNOMS A ESCENARIS VIZDOOM

Jordi Florit Ensenyat

Grau d'Enginyeria Informàtica

Escola Politècnica Superior

Any acadèmic 2024-25

APRENENTATGE PER REFORÇ APLICAT A ENTORNS 3D: IMPLEMENTACIÓ I ANÀLISI D'AGENTS AUTÒNOMS A ESCENARIS VIZDOOM

Jordi Florit Ensenyat

Treball de Fi de Grau

Escola Politècnica Superior

Universitat de les Illes Balears

Any acadèmic 2024-25

Paraules clau del treball: Deep Reinforcement Learning, VizDoom, DQN, PPO

Tutor: José Maria Buades Rubio

Autoritz la Universitat a incloure aquest treball en el repositori institucional per consultar-lo en accés obert i difondre'l en línia, amb finalitats exclusivament acadèmiques i d'investigació	Autor/a		Tutor/a	
	Sí	No	Sí	No
	☑	☐	☑	☐

Voldria expressar el meu agraïment a totes aquelles persones que han fet possible la realització d'aquest treball. Primerament, al meu tutor, José María Buades, per la seva orientació i consells imprescindibles. També als meus companys de carrera, per les hores compartides, suport constant i pels bons moments. Finalment, agraeixo especialment a la meva família, per haver estat sempre al meu costat amb paciència i suport incondicional.

SUMARI

ACRÒNIMS

AR Aprenentatge per Reforç

RL Reinforcement Learning

DRL Deep Reinforcement Learning

MDP Processos de Decisió de Markov

ANN Artificial Neural Network

DNN Deep Neural Network

CNN Convolutional Neural Network

FC Fully Connected

DQN Deep Q-Network

DDQN Double Deep Q-Network

PPO Proximal Policy Optimization

A2C Advantage Actor–Critic

TRPO Trust Region Policy Optimization

TD Temporal Difference

TD3 Twin Delayed DDPG

SAC Soft Actor–Critic

API Application Programming Interface

SB3 Stable-Baselines3

IQR Interquartile Range

MWU Mann–Whitney U

FPS Frames Per Second

WAD *Where's All the Data* (fitxer de mapa Doom)

ACS Action Code Script

GPU Graphics Processing Unit

CUDA Compute Unified Device Architecture

RGB Red–Green–Blue

RESUM

El Deep Reinforcement Learning (DRL) s'ha convertit en un instrument clau per dotar d'autonomia agents capaços de prendre decisions en entorns complexos i parcialment observables. Aquest treball de fi de grau explora fins a quin punt dos dels algorismes més emprats —DQN i PPO— poden adquirir habilitats de navegació i combat en tres escenaris tridimensionals de *VizDoom* amb una escalada progressiva de dificultat (Basic, Defend the Center i Deadly Corridor).

S'han implementat amb *Gymnasium* i Stable-Baselines3 (SB3) agents completament funcionals, i s'ha dissenyat un *pipeline* experimental amb 50 execucions independents per configuració, més de 50 milions de timesteps en total i un enregistrament detallat de mètriques a *TensorBoard*. Els anàlisis combinen indicadors descriptius robustos (mediana + Interquartile Range (IQR)), proves estadístiques no paramètriques (Mann–Whitney U) i inspecció qualitativa de polítiques.

Els resultats confirmen que DQN sobresurt quan l'espai d'acció és reduït i la recompensa és densa (cas Defend the Center), aprofitant la *replay memory* per accelerar l'aprenentatge. En canvi, PPO mostra una convergència més suau, estable i homogènia en entorns visuals estocàstics o amb recompenses més variades (Basic i, sobretot, Deadly Corridor), gràcies al seu mecanisme de *clipping* i a la sinergia actor–critic. Les diferències són estadísticament significatives ($p < 0{,}01$) i recolzen la teoria subjacent de cada mètode.

A partir d'aquestes evidències es proposen *bones pràctiques* per a futures aplicacions de DRL en entorns 3D: selecció d'hiperparàmetres sensibles al tipus d'escenari, estratègies d'exploració progressiva i principis de *reward shaping*. Tanmateix, es reconeixen limitacions clau —cost computacional elevat, només tres mapes i absència d'algoritmes contínus com Soft Actor–Critic (SAC) o Twin Delayed DDPG (TD3)— i es plantegen línies futures que inclouen transferència d'aprenentatge entre mapes, entorns amb accions contínues i integració de mètodes de *Safe Reinforcement Learning (RL)*.

En conjunt, l'estudi intenta demostrar que l'elecció de l'algoritme de DRL ha de respondre a la naturalesa de l'entorn: PPO ofereix un compromís sòlid quan l'estabilitat i la gestió de recompenses esparses són crucials, mentre que DQN continua sent competitiu en tasques discretes amb retroalimentació freqüent. Tot plegat aporta evidència valuosa i una guia metodològica per a treballs futurs que aspiren a entrenar agents robustos en entorns tridimensionals.

CAPÍTOL 1

INTRODUCCIÓ

1.1 Context i motivació

L'Aprenentatge per Reforç (AR) s'ha convertit en un dels pilars de la intel·ligència artificial moderna gràcies a la seva capacitat per aprendre comportaments òptims mitjançant prova i error. Quan aquest paradigma es combina amb xarxes neuronals profundes —el DRL—, els agents poden processar observacions d'alta dimensionalitat (imatges, trànsits sensorials, etc.) i actuar de manera autònoma en entorns complexos.

Els videojocs, i en particular els motors tridimensionals, presenten un camp de proves ideal per aquesta disciplina per tres motius principals:

(i) Proporcionen entorns controlables i repetibles.

(ii) Ofereixen reptes progressius de dificultat (des de tasques d'orientació bàsica fins a escenaris de combat complexos).

(iii) Permeten obtenir, gairebé en temps real, grans quantitats de dades d'interacció imprescindibles per entrenar xarxes profundes.

VizDoom reuneix aquestes característiques: simula escenaris 3D visuals rics, exposa una Application Programming Interface (API) lleugera basada en OpenAI *Gymnasium* i facilita la creació de mapes personalitzats. Treballar amb entorns visuals tridimensionals comporta reptes d'escalabilitat (gran volum d'imatges), de diferències de distribució (cada escenari té dinàmiques úniques) i d'estabilitat d'entrenament (explosió de gradients o *overfitting*).

1.2 Plantejament del problema

Aquest projecte aborda la pregunta següent: *quin dels dos algoritmes de DRL de referència —DQN o PPO— ofereix el millor compromís entre estabilitat, rapidesa de convergèn-*

cia i qualitat de la política quan l'agent opera únicament amb observacions visuals en entorns 3D amb recompenses parcialsment esparses?

Per donar-hi resposta, s'ha concebut una sèrie d'experiments que obliguen l'agent a:

- Processar imatges en directe (120×160 píxels en escala de grisos).
- Gestionar recompenses.
- Actuar tant en espais d'acció discrets (gir, disparar) com en configuracions amb múltiples botons simultanis (nivells de dificultat escalables).

1.3 Objectius

Per estructurar la investigació, es defineixen els objectius següents:

(O1) **Implementar** agents DQN i PPO amb `Gymnasium` i SB3, integrant-los a un codi modular i reutilitzable.

(O2) **Dissenyar** tres escenaris de VizDoom ("Basic", "Defend the Center" i "Deadly Corridor") que avaluïn precisió de tir, gestió de recursos i navegació sota foc enemic, respectivament.

(O3) **Afinar** l'enginyeria de recompenses a entorns més complexos (*reward shaping*) i els hiperparàmetres clau per garantir convergència robusta.

(O4) **Avaluar quantitativament** el rendiment amb corbes d'aprenentatge (mediana + IQR), boxplots finals i contrastos de significació (Mann–Whitney U (MWU)).

(O5) **Analitzar qualitativament** les polítiques apreses (vídeos, heatmaps, comportaments emergents) i detectar situacions de fallada.

1.4 Metodologia i eines

El flux de treball adoptat es pot resumir en cinc fases:

1. **Revisió bibliogràfica** de DQN, PPO i comparatives recents en escenaris 3D.
2. **Disseny dels escenaris** VizDoom (`.cfg` i `.wad`), incloent recompenses personalitzades i nivells de dificultat graduats.
3. **Desenvolupament de la interfície** `BaseVizDoomEnv` (`Gymnasium`) i integració de `Monitor` per registrar mètriques a TensorBoard; creació d'un pipeline reproductible (`experiment_pipeline.py`) amb *seeds* controlades.
4. **Entrenament massiu** dels agents (50 execucions independents per configuració), monitorització de corbes i checkpoints, i explotació dels logs amb Matplotlib/-TensorBoard.
5. **Anàlisi de resultats** —quantitatiu i qualitatiu— i redacció d'un informe complet amb gràfics, taules i annexos de configuració per afavorir la replicació.

1.5 Estructura del document

El text es divideix en cinc capítols principals i tres annexos:

- **Capítol 1.** Introducció: context, objectius, metodologia i esquema del treball.
- **Capítol 2.** Fonaments teòrics: repàs de RL clàssic, DRL, DQN i PPO.
- **Capítol 3.** Desenvolupament experimental: implementació dels agents, configuració d'entorns, hiperparàmetres i pipeline d'experiments.
- **Capítol 4.** Resultats: comparativa detallada de rendiment (corbes, boxplots, estadística) i discussió escenari per escenari.
- **Capítol 5.** Conclusions, limitacions i línies futures.
- **Annex A.** Fitxers de configuració dels escenaris VizDoom (`.cfg`).
- **Annex B.** Detalls arquitectures dels agents SB3.
- **Annex C.** Gràfiques de TensorBoard i anàlisi addicional de mètriques internes.

CAPÍTOL 2

FONAMENTS TEÒRICS

Aquest capítol presenta els conceptes bàsics i el marc teòric de l'aprenentatge per reforç. S'hi defineixen formalment desde elements bàsics per entendre l'aprenentatge per reforç com son els processos de decisió de Markov, la funció de valor i les equacions de Bellman, així com les categories d'algoritmes d'aprenentatge per reforç (clàssics i profunds).

2.1 Introducció a l'Aprenentatge per Reforç

L'Aprenentatge per Reforç (RL, per les seves sigles en anglès, *Reinforcement Learning*) és un paradigma de l'aprenentatge automàtic que se centra en com un agent intel·ligent ha de prendre decisions seqüencials en un entorn per maximitzar una mesura de recompensa acumulada [1]. A diferència de l'aprenentatge supervisat, que es basa en dades etiquetades, o de l'aprenentatge no supervisat, que cerca patrons en dades no etiquetades, el RL opera mitjançant la interacció directa de l'agent amb el seu entorn. L'agent no rep instruccions sobre quines accions dur a terme, sinó que ha de descobrir les accions òptimes a través d'un procés de prova i error, en què les accions "bones" es reforcen positivament i les "dolentes" es penalitzen [2].

Aquest procés d'aprenentatge interactiu i de maximització de recompenses a llarg termini és fonamental per a la resolució de problemes complexos de presa de decisions seqüencials en diversos dominis, des de la robòtica i els jocs fins a les finances i els sistemes d'energia intel·ligents [2].

2.1.1 El Problema d'Aprenentatge per Reforç

El problema d'Aprenentatge per Reforç es formalitza mitjançant la interacció contínua entre un agent i el seu entorn. En cada pas de temps t, l'agent observa un estat de l'entorn, $S_t \in \mathcal{S}$, on $\mathcal{S}$ és el conjunt de tots els estats possibles. Basant-se en aquest estat, l'agent selecciona una acció, $A_t \in \mathcal{A}(S_t)$, on $\mathcal{A}(S_t)$ és el conjunt d'accions disponibles

en l'estat S_t. Com a resultat de l'acció de l'agent, l'entorn transita a un nou estat, S_{t+1}, i emet una recompensa numèrica, $R_{t+1} \in \mathbb{R}$ [1].

L'objectiu de l'agent és maximitzar la quantitat total de recompensa que rep al llarg del temps. Aquest objectiu es formalitza sovint com la maximització del valor esperat de la suma acumulada de recompenses futures, descompteades per un factor $\gamma \in [0,1]$, conegut com el factor de descompte. Una recompensa futura es valora menys que una recompensa immediata [1].

2.1.2 Interfície Agent-Entorn

La interacció entre l'agent i l'entorn es descriu mitjançant una interfície clara:

- **Agent**: El prenedor de decisions i l'"aprenent".
- **Entorn**: Tot allò que està fora de l'agent i amb què interactua.

Aquesta interacció es desenvolupa en una seqüència discreta de passos de temps. En cada t:

1. L'agent rep una observació de l'estat S_t.
2. L'agent selecciona una acció A_t.
3. L'entorn transita al nou estat S_{t+1} i emet la recompensa R_{t+1}.

En definitiva, aquesta formulació resumeix de la manera més senzilla què significa percebre un entorn, actuar i perseguir un objectiu [1].

2.1.3 Components Fonamentals

Els elements clau que defineixen un sistema d'Aprenentatge per Reforç són:

- **Estats (S_t)**: Representen la situació actual de l'entorn que l'agent percep. Poden ser sensacions de baix nivell (per exemple, lectures de sensors) o abstraccions d'alt nivell.[1].
- **Accions (A_t)**: Són les decisions que l'agent pot prendre en un estat donat. Poden ser controls de baix nivell o decisions d'alt nivell [1].
- **Recompenses (R_t)**: És un senyal numèric escalar que l'entorn envia a l'agent en cada pas de temps. Defineix com de "bones" o "dolentes" són les accions de l'agent. L'objectiu de l'agent és maximitzar la recompensa total rebuda a llarg termini [1].
- **Política (π)**: És el "cervell" de l'agent, un mapeig des dels estats a les probabilitats de seleccionar cada possible acció. Defineix el comportament de l'agent en un moment donat. Una política pot ser determinista o estocàstica, i és el que l'agent aprèn i ajusta [1].
- **Funcions de Valor (V i Q)**: Estimen com de "bo" és per a l'agent estar en un estat donat (o realitzar una acció donada en un estat donat), en termes de la recompensa esperada acumulada a partir d'aquest punt.

- **Funció de Valor d'Estat ($v_\pi(s)$)**: El retorn esperat en iniciar en l'estat s i seguir la política π a partir de llavors.
- **Funció de Valor d'Acció ($q_\pi(s,a)$)**: El retorn esperat en iniciar en l'estat s, prendre l'acció a, i seguir la política π a partir de llavors.

Les funcions de valor són prediccions de recompenses futures i són crucials per prendre decisions òptimes [1].

Finalment, la figura 2.1 il·lustra gràficament aquest procés de percepció, decisió i recompensa en un entorn.[1].

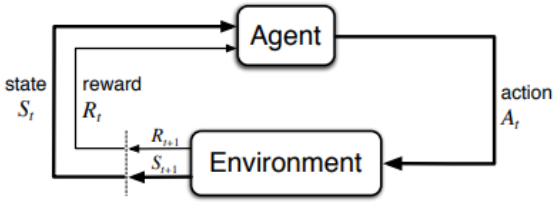


Figura 2.1: Interacció agent–entorn. *Font:* Richard S. Sutton & Andrew G. Barto, *Reinforcement Learning: An Introduction*, 2a ed. (MIT Press, 2018), Fig. 3.1.

2.1.4 Equacions de Bellman

Les funcions de valor satisfan relacions recursives fonamentals conegudes com a Equacions de Bellman. Aquestes equacions expressen una condició de consistència entre el valor d'un estat (o parell estat-acció) i els valors dels seus possibles estats successors. L'equació de Bellman per a v_π i q_π descriu la relació entre el valor d'un estat i els valors dels estats als quals es pot transitar sota la política π.

Per a una política òptima π^*, les funcions de valor òptimes $v^*(s)$ i $q^*(s,a)$ satisfan les Equacions de Bellman d'Optimalitat:

$$v^*(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v^*(s')]$$

$$q^*(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} q^*(s',a')]$$

Aquestes equacions formen la base per a molts algorismes d'RL, buscant trobar les funcions de valor òptimes a partir de les quals es pot derivar una política òptima [1].

2.1.5 Processos de Decisió de Markov (MDP)

Un problema d'Aprenentatge per Reforç que satisfà la propietat de Markov es coneix com a Procés de Decisió de Markov (MDP). La **Propietat de Markov** estableix que el futur és independent del passat donat l'estat present. Un MDP es defineix formalment per un conjunt d'estats ($\mathcal{S}$), un conjunt d'accions ($\mathcal{A}$), una funció de probabilitat de transició $p(s',r|s,a)$ que especifica la probabilitat de transitar a l'estat s' i rebre la recompensa r en prendre l'acció a en l'estat s, i un factor de descompte γ. Els MDPs

proporcionen un marc matemàtic robust per modelar problemes de presa de decisions seqüencials en entorns estocàstics [1, 3].

La figura 2.2 mostra un exemple sencill d'un MDP amb tres estats: S_0, S_1 i S_2. Els cercles verds representen aquests estats possibles. Les fletxes indiquen les transicions produïdes quan l'agent tria una acció concreta (marcada com a_0, a_1, etc.). A cada fletxa s'indica la probabilitat de transició i, quan aplica, amb la recompensa que rep l'agent a la vora de la fletxa (per exemple, "+5" o "−1").

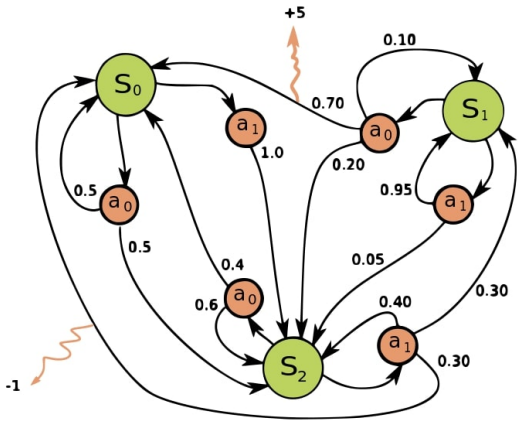


Figura 2.2: Processos de Decisió de Markov (MDP). *Font:* waldoalvarez, *Markov Decision Process* (Wikimedia Commons, 2017)

2.1.6 Mètodes de Monte Carlo i Diferència Temporal (TD)

Existeixen dues categories principals de mètodes per estimar funcions de valor i aprendre polítiques:

- **Mètodes de Monte Carlo**: Aquests mètodes aprenen a partir de l'experiència completa, mitjançant la mitjana dels retorns observats d'episodis acabats. Requereixen que les tasques siguin episòdiques, ja que les actualitzacions es realitzen només al final de cada episodi. No «bootstrappen», és a dir, l'estimació d'un estat no es basa en l'estimació de cap altre estat [1].

- **Aprenentatge per Diferència Temporal (TD)**: Els mètodes TD combinen idees de Monte Carlo i programació dinàmica. Aprenen directament de l'experiència sense un model de l'entorn, igual que Monte Carlo, però actualitzen les seves estimacions basant-se en una recompensa immediata més l'estimació del següent estat (bootstrapping) sense esperar el resultat final d'un episodi. L'actualització TD(0) per a una funció de valor V és:

$$V(S_t) \leftarrow V(S_t) + \alpha\left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]$$

  [1].

La figura 2.3 mostra, a l'esquerra, un arbre d'experiència en un mètode de Monte Carlo, on la trajectòria completa des de l'estat inicial S_t fins a l'estat terminal (marcat amb "T") s'usa per calcular el retorn total G_t. En contrast, a la dreta es veu el mètode més senzill de Diferència Temporal, on només es pren la recompensa immediata R_{t+1} i es fa "bootstrap" amb l'estimació de valor de l'estat següent $V(S_{t+1})$. Les línies i nodes blancs i negres representen possibles estats i transicions, mentre que el traçat en vermell ressalta el camí efectiu que s'usa per a l'actualització de la funció de valor.

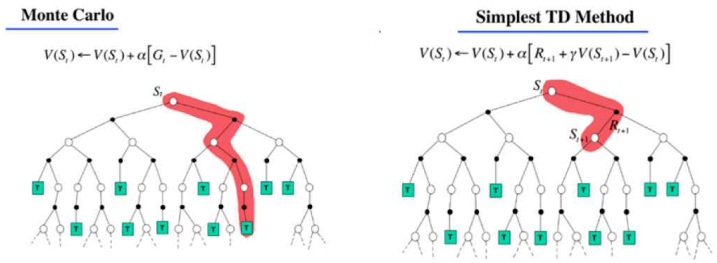


Figura 2.3: Comparació entre l'enfocament complet d'un mètode de Monte Carlo (esquerra) i l'actualització d'un mètode de Diferència Temporal (TD) (dreta). *Font:* Baihan Lin, "Reinforcement Learning and Bandits for Speech and Language Processing: Tutorial, Review and Outlook" (arXiv preprint, 2022), Fig. 7.

2.2 Aprenentatge per Reforç Profund (Deep Reinforcement Learning)

L'Aprenentatge per Reforç Profund (DRL) sorgeix de la unió de l'Aprenentatge per Reforç amb les Xarxes Neuronals Profundes (Deep Learning). Aquesta combinació ha permès als agents d'RL abordar tasques de presa de decisions complexes que abans estaven fora de l'abast de les màquines, especialment aquelles amb espais d'estat d'alta dimensionalitat [2].

La motivació principal per al DRL és superar les limitacions dels mètodes tradicionals d'RL en la gestió d'entrades sensorials complexes, com ara imatges o seqüències de temps. Les xarxes neuronals profundes actuen com aproximadors de funcions, permetent que els agents aprenguin representacions abstractes directament de dades brutes, sense la necessitat d'una enginyeria de característiques manual [2].

2.2.1 Xarxes Neuronals com a Aproximadors de Funcions

En el context del DRL, les Xarxes Neuronals Artificials (Artificial Neural Network (ANN)) i, en particular, les Xarxes Neuronals Convolucionals (CNN), s'utilitzen com a aproximadors de funcions per a la política i/o les funcions de valor. Una ANN consisteix en capes de neurones interconnectades, on cada connexió té un pes que determina el nivell d'activació de la neurona. L'"aprenentatge profund" es refereix a ANN amb múltiples capes ocultes, capaces de realitzar transformacions no lineals successives per aprendre diferents nivells d'abstracció [3].

Les CNN són especialment adequades per processar dades visuals, com els píxels d'una imatge o vídeo, gràcies a les seves capes convolucionals, de no linealitat i de

pooling, que extreuen característiques rellevants de manera jeràrquica [3, 2]. La figura 2.4 mostra un esquema general d'una CNN: primer, l'entrada (per exemple, un fotograma del joc) passa per diverses capes de convolució, on els filtres aprenen a detectar patrons locals (arestes, textures, regions de contrast), generant múltiples mapes de característiques. A continuació, les capes de pooling redueixen la resolució espacial dels mapes, conservant les característiques més importants. Finalment, els mapes resultants s'aplanen i serveixen com el input a una xarxa de capes totalment connectades, que combinen la informació global per produir l'output, que en el context del DRL pot ser la distribució de probabilitats d'accions o una estimació de valor. Aquesta separació entre «extracció de característiques» i «classificació» facilita que l'agent aprengui representacions complexes directament de les dades brutes.

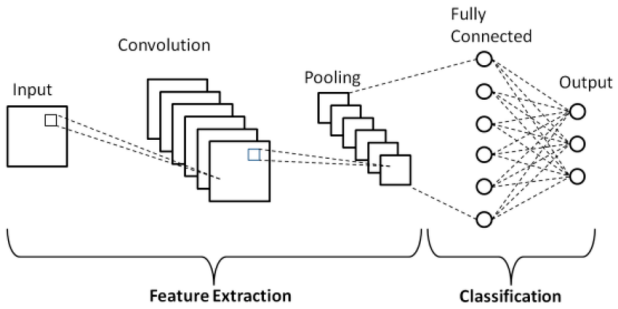


Figura 2.4: Arquitectura general d'una xarxa neuronal convolucional (CNN). *Font:* Van Hiep Phung & Eun Joo Rhee, "A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets", *Applied Sciences* 9(21):4500 (2019), Fig. 1. CC BY 4.0.

Els avantatges d'usar xarxes neuronals profundes en RL inclouen:

- **Maneig d'entrades d'alta dimensionalitat**: Són intrínsecament bones per processar entrades com ara sèries de temps i fotogrames, escalant eficientment amb dimensions addicionals en l'espai d'estat o acció.
- **Entrenament incremental i utilització de mostres**: Poden ser entrenades incrementalment i utilitzar eficientment mostres addicionals a mesura que l'aprenentatge progressa, la qual cosa és crucial en entorns d'aprenentatge en línia [2].

2.2.2 Algorismes On-policy vs. Off-policy

Els algorismes de DRL es classifiquen comunament en on-policy i off-policy:

- **Algorismes On-policy**: Requereixen que les dades d'entrenament es recullin seguint la mateixa política que s'està millorant. Això significa que la política de comportament (la que genera les mostres) i la política objectiu (la que s'està aprenent) són la mateixa. Exemples inclouen PPO i Advantage Actor–Critic (A2C). Tenen alta complexitat de mostreig ja que les dades d'una política anterior no es poden reutilitzar directament, la qual cosa sovint requereix noves interaccions amb l'entorn [3].

- **Algorismes Off-policy**: Permeten que les dades d'entrenament es recullin seguint una política diferent de la que s'està millorant. Això facilita la reutilització de dades d'experiències passades, la qual cosa millora l'eficiència de la mostra. DQN és un exemple prominent d'algorisme off-policy. Encara que són més eficients en l'ús de dades, poden ser més difícils d'optimitzar a causa del desalineament entre les polítiques [3].

2.3 Deep Q-Networks (DQN)

Deep Q-Networks (DQN) és un dels algorismes pioners en DRL que va demostrar la capacitat d'un agent per aprendre a jugar jocs d'Atari directament des de l'entrada de píxels, aconseguint un rendiment comparable o superior a l'humà [4]. DQN és una extensió de l'algorisme de Q-learning, adaptat per funcionar amb xarxes neuronals profundes.

2.3.1 Fonaments de Q-learning

El Q-learning és un algorisme d'aprenentatge per reforç basat en valors que busca aprendre la funció de valor d'acció òptima, $Q^*(s,a)$, que representa el retorn esperat màxim en prendre l'acció a en l'estat s i seguir una política òptima a partir de llavors. En la seva forma tabular, $Q(s,a)$ s'actualitza utilitzant l'Equació de Bellman d'optimalitat:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(s,a)]$$

On α és la taxa d'aprenentatge. Per a entorns amb un nombre elevat d'estats i accions, una taula de consulta esdevé inviable [2].

2.3.2 Innovacions de DQN

DQN aborda el problema de l'alta dimensionalitat utilitzant una xarxa neuronal profunda per aproximar la funció $Q(s,a;\theta)$, on θ són els paràmetres de la xarxa. No obstant això, l'ús directe de xarxes neuronals amb Q-learning bàsic introdueix inestabilitats. DQN introdueix dues innovacions clau per mitigar això [4, 2]:

1. **Xarxa Q Objectiu (Target Q-network)**: En lloc d'utilitzar la mateixa xarxa neuronal per calcular el valor Q actual ($Q(s,a;\theta_k)$) i el valor Q objectiu ($Y_{Q_k} = r + \gamma \max_{a'} Q(s',a';\theta_k)$), DQN utilitza una segona xarxa, la xarxa objectiu, amb paràmetres θ^- que s'actualitzen periòdicament (cada C iteracions) amb els paràmetres de la xarxa principal ($\theta^- \leftarrow \theta$). Això estabilitza l'objectiu d'aprenentatge, reduint la correlació entre l'estimació i l'objectiu.

2. **Memòria de Replay (Replay Memory)**: Les experiències (transicions $< s,a,r,s' >$) s'emmagatzemen en un amortidor de replay. Durant l'entrenament, es mostregen mini-lots aleatoris d'aquestes experiències per actualitzar la xarxa. Això trenca la correlació temporal de les seqüències d'experiències, la qual cosa és crucial per a l'estabilitat de l'aprenentatge amb xarxes neuronals. A més, permet reutilitzar experiències passades, millorant l'eficiència de la mostra [5, 2].

Altres heurístiques utilitzades en DQN inclouen el preprocessament dels estats visuals (per exemple, reducció de RGB a escala de grisos i redimensionament) [4, 2].

Finalment s'ilustra la figura 2.5 on es mostra un esquema simplificat del cicle d'interacció i aprenentatge d'un DQN. El bloc **state** s representa l'estat actual de l'entorn, aquesta entrada es passa a la **Deep Neural Network (DNN)** parametritzada per θ, que aprèn a aproximar la funció de valor d'acció $Q(s,a;\theta)$. Els **nodes de sortida** (indicats com a política $\pi_\theta(s,a)$) corresponen als valors estimats per a cadascuna de les accions possibles, o bé a les probabilitats de selecció de cada acció en una versió estocàstica. A partir d'aquests valors, l'agent tria una **acció** a («Take action a»), que s'envia al bloc **Environment**. L'entorn respon amb una **recompensa** r i el nou estat següent, que torna a alimentar el DQN com a nova entrada. Aquest bucle tanca el procés d'aprenentatge per reforç, on el paràmetre θ s'actualitza a través de les diferències temporals dels torns de recompensa per millorar la precisió de $Q(s,a;\theta)$.

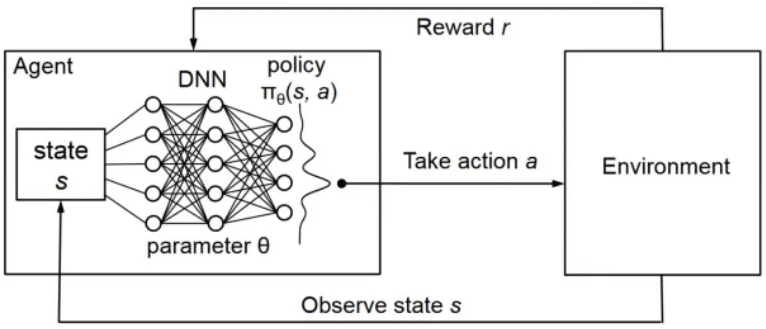


Figura 2.5: Arquitectura d'un Deep Q-Network. *Font:* Dilith Jayakody, "Deep Q-Networks (DQN) – A Quick Introduction" (blog, 18 des. 2022).

2.3.3 Millores en DQN

Des de la seva introducció, DQN ha estat objecte de diverses millores per abordar les seves limitacions i augmentar el seu rendiment:

- **Double DQN (Double Deep Q-Network (DDQN))**: L'operació max en el Q-learning tradicional pot portar a una sobreestimació dels valors Q, la qual cosa pot perjudicar la convergència. DDQN [6] aborda això desacoblament la selecció de l'acció de l'avaluació del seu valor. L'acció es selecciona utilitzant la xarxa Q principal, mentre que el seu valor s'avalua utilitzant la xarxa Q objectiu: $Y_{DDQN_k} = r + \gamma Q(s', \text{argmax}_{a'} Q(s', a'; \theta_k); \theta_k^-)$. Això resulta en estimacions de valor menys optimistes i un millor rendiment.

- **Dueling Network Architectures**: En Dueling DQN [7], la xarxa neuronal es divideix en dos fluxos separats: un que estima la funció de valor de l'estat $V(s)$ i un altre que estima la funció d'avantatge de l'acció $A(s,a)$. La funció Q es reconstrueix a partir d'aquests dos components. Aquesta arquitectura permet que la xarxa

aprengui la importància dels estats de forma més eficient, ja que el valor d'un estat pot ser independent de les accions disponibles [2].

- **Distributional DQN**: En lloc d'aproximar el valor esperat dels retorns, aquesta extensió [8] cerca aprendre la distribució completa dels retorns. En proporcionar una representació més rica de la incertesa en els retorns, pot conduir a un aprenentatge més robust i permetre comportaments sensibles al risc.

- **Multi-step Learning**: Les actualitzacions de DQN es basen en un sol pas de la recompensa i el següent estat. L'aprenentatge multi-pas estén això per considerar n passos de recompensa i després bootstrapear des de l'estat n-èsim. Això pot propagar la informació de la recompensa més ràpidament i reduir la variància de les estimacions [2].

Malgrat aquestes millores, els mètodes basats en DQN són menys adequats per a espais d'acció grans o continus i no poden aprendre polítiques estocàstiques explícitament [2].

2.4 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) és un algorisme de gradient de política relativament recent, proposat per OpenAI el 2017, que ha guanyat popularitat a causa de la seva simplicitat d'implementació, el seu rendiment i la seva bona eficiència de mostra [9, 3]. PPO es basa en la família de mètodes Actor-Critic.

2.4.1 Concepte de Gradient de Política

Els mètodes de gradient de política optimitzen directament una funció de rendiment (generalment el retorn esperat) trobant la derivada d'aquesta funció respecte als paràmetres de la política (θ). La política és típicament una xarxa neuronal paramètrica. L'objectiu és ajustar els paràmetres de la política en la direcció que maximitzi la recompensa esperada. El teorema del gradient de política proporciona una forma d'estimar aquest gradient a partir de les experiències [1, 2].

Una manera senzilla d'entendre això és que els mètodes de gradient de política augmenten la probabilitat de les accions que resulten en altes recompenses i disminueixen la probabilitat de les accions que condueixen a baixes recompenses.

2.4.2 Arquitectura Actor–Critic

Els mètodes Actor–Critic són una classe d'algorismes d'aprenentatge per reforç que combinen els beneficis dels enfocaments basats en polítiques (*actor*) amb els enfocaments basats en valors (*critic*). La figura 2.6 mostra de manera esquemàtica l'estructura típica d'aquesta arquitectura, que consta dels següents components principals:

- **Xarxa Actor (Actor Network)**: és una xarxa neuronal parametritzada per θ que rep com a entrada l'estat actual s de l'entorn i retorna com a sortida una acció a. L'acció es genera seguint una política $\pi_\theta(s)$, creant una distribució de probabilitats sobre les accions possibles.

- **Xarxa Crític (Critic Network)**: és una xarxa neuronal que rep com a entrada l'estat s i estima el valor d'aquest estat, denotat com $V^{\pi}(s)$. Aquesta estimació s'utilitza com a línia base per calcular l'error temporal diferenciat (*Temporal Difference error*, TD error), que quantifica la discrepància entre el valor estimat actual i el valor real obtingut després de prendre una acció.

El flux de dades entre aquests components s'explica a continuació. Inicialment, l'entorn proporciona l'estat actual s, que s'envia simultàniament tant a la xarxa actor com a la xarxa crític. L'actor genera una acció a segons la política $\pi_\theta(s)$, i aquesta acció s'aplica a l'entorn. Com a resposta, l'entorn retorna una recompensa immediata r i un nou estat s'.

El crític calcula l'error temporal diferenciat (TD error), definit com:

$$\delta = r + \gamma V^{\pi}(s') - V^{\pi}(s)$$

on γ és el factor de descompte que pondera la importància de les recompenses futures. Aquest error TD indica fins a quin punt l'estimació actual del crític coincideix amb la recompensa real obtinguda i serveix per actualitzar els paràmetres de la xarxa crític. A més, aquest error també s'utilitza per actualitzar els paràmetres θ de la xarxa actor, ajustant la política cap a accions que maximitzin les recompenses futures.

D'aquesta forma, l'actor aprèn a escollir les accions que optimitzen la recompensa esperada, mentre que el crític proporciona un senyal fiable i consistent que redueix la variància en l'estimació del gradient de la política. Això fa que el procés d'aprenentatge sigui més estable, robust i eficient [2].

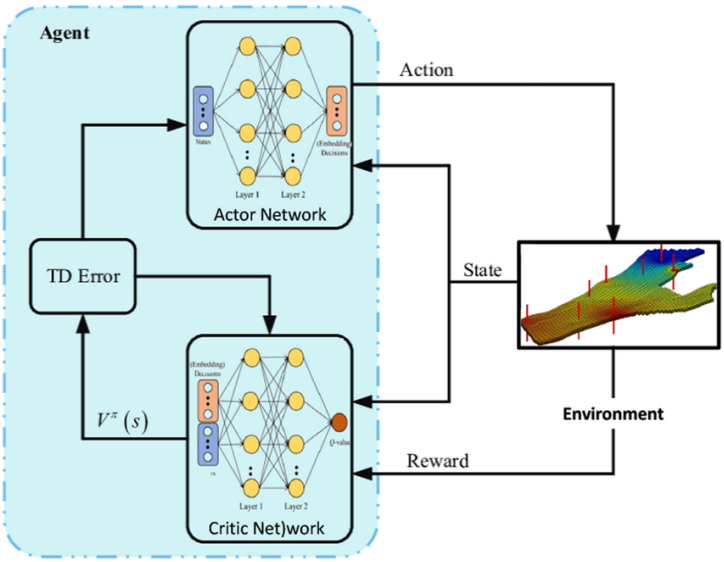


Figura 2.6: Arquitectura actor–critic i la interacció amb l'entorn. *Font:* Ramez Abdalla et al., "Actor-critic reinforcement learning leads decision-making in energy systems optimization—steam injection optimization", *Neural Computing and Applications* 35:16633–16647 (2023)

2.4.3 Algorisme PPO i Objectiu de Clipping

PPO opera alternant entre la recol·lecció de dades de l'entorn utilitzant la política actual i l'optimització d'una funció objectiu amb un algorisme d'ascens de gradient estocàstic. A diferència d'altres algorismes de gradient de política com Trust Region Policy Optimization (Trust Region Policy Optimization (TRPO)), que utilitzen restriccions de pas de política complexes, PPO simplifica això mitjançant un "objectiu de clipping" [9, 3].

La funció objectiu de PPO amb clipping es defineix com:

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

On:

- $r_t(\theta) = \frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_{old}}(A_t|S_t)}$ és la raó de probabilitat entre la política nova (π_θ) i la política antiga ($\pi_{\theta_{old}}$).
- $\hat{A}_t$ és l'estimació de l'avantatge en el temps t.
- ϵ és un hiperparàmetre petit (per exemple, 0,1 o 0,2) que defineix el rang de clipping.

L'objectiu de clipping assegura que la nova política no es desviï massa de la política antiga en un sol pas d'actualització. Si la raó de probabilitat $r_t(\theta)$ cau fora de l'interval $[1-\epsilon, 1+\epsilon]$, el terme de clipping restringeix el canvi. Això evita actualitzacions de política excessivament grans i potencialment catastròfiques, promovent l'estabilitat i la robustesa de l'entrenament [9].

2.4.4 Avantatges de PPO

PPO ofereix diversos avantatges que el fan atractiu per a problemes de DRL:

- **Simplicitat d'Implementació**: És significativament més fàcil d'implementar que algorismes com TRPO, mantenint tot i així un rendiment comparable o superior [3].
- **Eficiència de Mostra**: Tot i ser un algorisme on-policy (és a dir, les dades es recullen amb la política actual), PPO sovint demostra una millor eficiència de mostra en comparació amb altres mètodes de gradient de política com A2C [3]. Això es deu en part a que pot realitzar múltiples passos d'optimització amb les mateixes dades d'un episodi, dins dels límits del clipping.
- **Rendiment Robust**: PPO ha demostrat un rendiment sòlid en una àmplia gamma d'entorns de referència, incloent els jocs d'Atari i tasques de control robòtic, superant sovint a altres algorismes populars [9].

2.5 Comparació entre DQN i PPO

DQN i PPO representen dos enfocaments fonamentals dins del Deep Reinforcement Learning: els mètodes basats en valors (DQN) i els mètodes basats en polítiques (PPO).

Característica	DQN	PPO
Tipus d'Algorisme	Basat en Valor (Value-based)	Basat en Política / Actor–Critic
Política Apresa	Implícita (Q òptima, ϵ-greedy)	Explícita (xarxa neuronal)
Espai d'Accions	Discret	Discret i Continu
On-policy / Off-policy	Off-policy (replay memory)	On-policy (dades actuals)
Eficiència de Mostra	Alta (reutilització)	Mitjana-Alta (clipping)
Estabilitat Entrenament	Millorada amb Target Net + Replay	Millorada amb objectiu de Clipping
Objectiu	Aprendre Q òptima	Aprendre política òptima

Taula 2.1: Comparació entre DQN i PPO

DQN, en ser off-policy, pot reutilitzar dades d'experiències passades emmagatzemades a la memòria de replay, la qual cosa el fa molt eficient en entorns on la recol·lecció de dades és costosa. No obstant això, pot tenir dificultats amb espais d'acció continus i la convergència pot ser sensible. PPO, d'altra banda, és un algorisme on-policy, la qual cosa significa que requereix que les dades siguin recol·lectades per la política actual. Encara que això pot semblar menys eficient en l'ús de dades, el mecanisme de clipping de PPO permet múltiples actualitzacions de gradient amb les mateixes dades, millorant la seva eficiència en comparació amb altres algorismes on-policy purs. PPO és generalment més estable i més fàcil d'ajustar que DQN, i és particularment adequat per a problemes amb espais d'acció continus [2, 9].

En definitiva, si bé el DQN va obrir el camí demostrant que el DRL podia dominar entorns discrets i exigents, el PPO s'ha consolidat com una opció sòlida i adaptable per a un ventall molt més ampli de problemes d'RL. Un cop explicades detalladament les diferències entre els dos enfocaments, l'objectiu principal és comprovar com es comporten davant dels reptes particulars del nostre entorn —un escenari VizDoom amb espais d'acció principalment continus. S'avaluarà la seva eficiència de mostra i s'analitzaran quin dels dos facilita de manera més estable l'aprenentatge d'estratègies competitives.

CAPÍTOL 3

DESENVOLUPAMENT EXPERIMENTAL

Aquest capítol detalla la implementació experimental de l'aprenentatge per reforç profund sobre l'entorn de videojocs *ViZDoom*. Es descriu la selecció i configuració dels escenaris, la integració de les llibreries `Gymnasium` i `Stable-Baselines3`, l'arquitectura dels agents entrenats (DQN i PPO), els hiperparàmetres configurats, les tècniques d'enginyeria de recompenses (*reward shaping*) per entorns complexos i el disseny del *pipeline* d'experimentació per garantir la reproducibilitat i la robustesa dels resultats.

3.1 Entorn experimental: *ViZDoom*

L'objectiu d'aquest treball és avaluar algorismes d'*aprenentatge per reforç profund* (DRL) en entorns tridimensionals que exigeixen percepció visual i presa de decisions seqüencials. Entre els bancs de proves existents, **ViZDoom** [10] s'ha consolidat com una plataforma de recerca molt potent, ja que resol la bretxa entre els entorns 2D d'Atari [4] i els simuladors foto-realistes molt més costosos (p. ex. Habitat o CARLA). Gràcies a la seva lleugeresa—fins a >7 000 *Frames Per Second (FPS)* en mode fora de pantalla—i a la facilitat per definir recompenses, escenaris i sensors, ViZDoom permet realitzar experiments a gran escala sense recursos computacionals extrems.

El motor subjaçent és *ZDoom*, derivat del clàssic *Doom* de 1993, però amb:

- **Renderitzat fora de pantalla** (frame buffer en Red–Green–Blue (RGB) o grisos) i canal de profunditat opcional.
- **Accés a variables de joc** (salut, munició, coordenades, ...) només si es vol utilitzar com a senyals auxiliars.
- **Control granular del temps**: pas síncron (una acció → un fotograma) o asíncron (accions a màxima velocitat d'execució), imprescindible per a comparacions equitatives entre algoritmes [2].

- **API multillenguatge** (C++, Python) facilitats per crear adaptadors GYMNASIUM [11], de manera que biblioteques com STABLEBASELINES3 [12] puguin entrenar agents sobre el motor VizDoom.

El repositori inclou una dotzena d'escenaris estàndard—*Basic*, *Defend the Center*, *Deadly Corridor*, *Health Gathering Supreme*, ...—que cobreixen navegació, supervivència, combat i exploració [13]. A més, qualsevol usuari podria desenvolupar *Where's All the Data* (fitxer de mapa Doom) (WAD) (archius de mapa) propis amb editors clàssics de *Doom* i definir:

a) **Geometria** i objectes de l'escenari.

b) **Lògica de recompenses** via scripts Action Code Script (ACS) o Lua (*reward shaping*).

c) **Conjunts d'accions** discrets (sis graus de llibertat i disparar) o continus.

d) **Sensors addicionals** (segmentacions, etiquetes d'objectes, àudio espacial).

Les característiques descrites fan que ViZDoom compleixi els requisits experimentals presentat per els objectius de aquest treball:

- **Percepció visual de primera persona** i *observacions parcials*, essencials per avaluar estratègies amb memòria.
- **Escalabilitat temporal** (milions d'interaccions en hores), necessària per comparar corbes d'aprenentatge i estabilitat estadística.
- **Recompenses definibles** que s'alineen amb els objectius de cada experiment, minimitzant soroll extrínsec.
- **Reutilització de codi i models**: integració senzilla amb STABLEBASELINES3 i GYMNASIUM per la creació de agents.

En conjunt, ViZDoom ofereix el compromís òptim entre complexitat visual, control i cost computacional, donant pas a un entorn òptim on experimentar i crear agents per al estudi d'aquest treball.

3.1.1 Escenaris

Els escenaris emprats en aquest projecte, accessibles a través de la documentació oficial de ViZDoom [13], són:

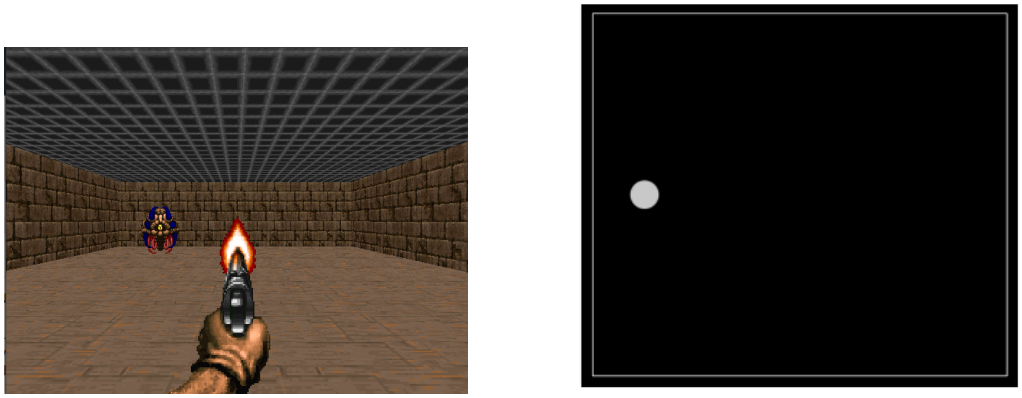
- **Basic**: Dissenyat per verificar la viabilitat d'entrenar agents d'IA en un entorn 3D senzill. El mapa és una sala rectangular. El jugador apareix al centre d'una de les parets llargues, mentre que un enemic es genera aleatòriament a una posició de la paret oposada. L'agent pot moure's a l'esquerra, a la dreta i disparar. Només cal un impacte per eliminar l'enemic, i l'episodi finalitza quan el monstre és eliminat o s'esgota el temps màxim. La figura 3.1 mostra una captura tant en primera persona del agent com el disseny del mapa.

Taula 3.1: Recompenses per acció

Acció	**Punts**
Matar el monstre	+106
Cada dispar	−5
Cada tic que l'agent roman viu	+1

Taula 3.2: Paràmetres de l'entorn

Paràmetre	**Valor**
Accions discretes disponibles	Esquerra, Dreta, Disparar
Variables de joc disponibles	Munició del jugador
Temps màxim per episodi	300 tics



(a) Vista de l'agent de l'escenari *Basic*.

(b) Mapa de l'escenari *Basic*.

Figura 3.1: Captures visuals de l'escenari *Basic* de ViZDoom. A l'esquerra, la vista en primera persona de l'agent. A la dreta, el disseny del mapa en perspectiva top-down.

- **Defend the Center**: L'objectiu d'aquest escenari és que l'agent aprengui a prioritzar l'eliminació de monstres, mentre gestiona la seva pròpia supervivència i munició. L'agent només rep recompensa explícita per matar enemics, però ha de deduir implícitament la importància de mantenir-se viu i conservar la munició per maximitzar la recompensa acumulada. Això fomenta un comportament estratègic més complex.

  El mapa és una gran àrea circular on el jugador apareix al centre. Cinc enemics amb l'objectiu de eliminar al jugador es generen inicialment a la perifèria i reapareixen periòdicament si són eliminats. Cada monstre mor amb un sol dispar. L'episodi finalitza quan l'agent mor, la qual cosa és inevitable atesa la munició limitada. A la figura 3.2 es poden observar aquestes característiques.

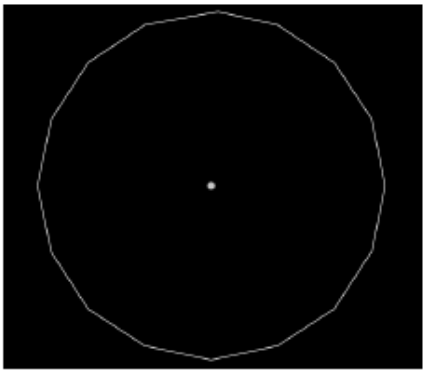
Taula 3.3: Recompenses per esdeveniment

Esdeveniment	**Punts**
Monstre eliminat	+1
Mort de l'agent	−1

Taula 3.4: Paràmetres de l'entorn

Paràmetre	**Valor**
Accions discretes disponibles	Esquerra, Dreta, Disparar
Variables de joc disponibles	Salut del jugador, Munició
Temps màxim per episodi	2100 tics
Nivell de dificultat	`doom_skill = 3`

(a) Vista de l'agent de l'escenari *Defend the Center*.



(b) Mapa de l'escenari *Defend the Center*.

Figura 3.2: Captures visuals de l'escenari *Defend the Center* de ViZDoom. A l'esquerra, la vista en primera persona. A la dreta, el disseny del mapa.

- **Deadly Corridor**: Aquest escenari desafia a l'agent a navegar amb èxit cap a un objectiu (una peça d'armadura verda) mentre sobreviu als atacs enemics. El mapa consisteix en un passadís llarg flanquejat per sis enemics que disparen des de les dues bandes. L'armadura es troba a l'extrem oposat del passadís. La recompensa es basa en la proximitat a l'objectiu (acostar-se augmenta la recompensa, allunyar-se la disminueix), amb una penalització severa per la mort de l'agent. Per forçar un comportament estratègic que impliqui combatre els enemics i no només fugir, l'escenari es configura amb el nivell màxim de dificultat. La figura 3.3 mostra una captura tant en primera persona de l'agent com el disseny del passadís i la disposició dels enemics.

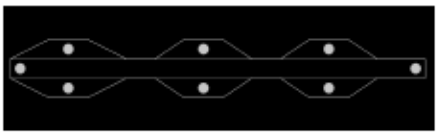
Taula 3.5: Recompenses per moviment

Esdeveniment	**Punts**
Acostar-se a l'armadura ($+\Delta d$)	$+\Delta d$
Allunyar-se de l'armadura ($-\Delta d$)	$-\Delta d$
Mort de l'agent	–100

Taula 3.6: Paràmetres de l'entorn

Paràmetre	**Valor**
Accions discretes disponibles	Endavant/Enrere, Esquerra/Dreta, Girar E/D, Disparar
Variable de joc disponible	Salut del jugador
Temps màxim per episodi	2100 tics
Nivell de dificultat	`doom_skill = 5`

(a) Vista de l'agent de l'escenari *Deadly Corridor*.



(b) Mapa de l'escenari *Deadly Corridor*.

Figura 3.3: Captures visuals de l'escenari *Deadly Corridor* de ViZDoom. A l'esquerra, la vista en primera persona. A la dreta, el disseny del mapa.

En tots els escenaris presentats, l'agent rep com a observació una imatge en escala de grisos amb una resolució de 160 × 120 píxels (amb 1 canal de color). Les accions disponibles per a l'agent són discretes, amb un nombre que varia de 3 a 7 botons segons l'escenari seleccionat.

La configuració exacta dels escenaris es defineix mitjançant fitxers .cfg específics per a cada entorn. Per a una descripció detallada del contingut d'aquests fitxers, vegeu l'Annex A.

3.2 Adaptació a `Gymnasium` i `Stable-Baselines3`

L'arquitectura experimental es basa principalment sobre dues llibreries àmpliament acceptades en la comunitat de *Deep Reinforcement Learning* (DRL): `Gymnasium` i `Stable-Baselines3` (**SB3**). La primera proporciona la interfície per definir i controlar

entorns d'aprenentatge per reforç; la segona, un conjunt d'implementacions fiables i verificades d'algorismes DRL basats en `PyTorch`. La sinergia d'ambdues eines permet separar clarament la *definició de l'entorn* (dinàmica, recompenses, observacions) de la *lògica de l'algoritme* (optimització, replay buffer, clipping, etc.).

`Gymnasium`

`Gymnasium`[1] defineix un contracte minimalista basat en dos mètodes obligatoris [11]:

- `reset()` $\longrightarrow$ (*observation*, *info*)
- `step(action)` $\longrightarrow$ (*next_obs*, *reward*, *terminated*, *truncated*, *info*)

Aquesta simplicitat fa que qualsevol entorn compatible pugui «connectar-se» a desenes d'algorismes ja implementats, cosa essencial per comparar DQN i PPO sota exactament les mateixes condicions.

`Stable-Baselines3`

`Stable-Baselines3` ofereix referències cuidades de DQN, PPO i altres mètodes actor-critic, amb control d'hiperparàmetres, suport a *callbacks* i integració nativa amb `TensorBoard` [14]. Utilitzar SB3 evita reproduir codi crític de baix nivell (optimitzadors, replay buffer, etc.) i concentra l'esforç del TFG en el disseny dels escenaris i l'anàlisi de resultats.

Disseny modular dels entorns VizDoom

Classe bàsica `BaseVizDoomEnv`. S'ha implementat una superclasse (`BaseVizDoomEnv`) que encapsula la configuració comuna de tots els escenaris (resolució, format de pantalla, espais Gym, gestió de seed, etc.). Les configuracions globals per tots els entorns son les seguents:

a) **Inicialització controlada per seed**: la seed s'aplica abans i després de `game.init()` per garantir reproduïbilitat fins i tot en la generació interna de VizDoom.

b) **Espais Gym**: `observation_space = Box(0.255,(120.160,1),uint8)` modela la imatge en escala de grisos (120×160); `action_space = Discrete(num_actions)` reflecteix els botons disponibles en cada mapa.

c) **Política de pas fix**: `make_action(actions[action], 4)` aplica l'acció durant quatre tics de joc, coherència temporal habitual en la literatura [4].

Especialització per escenari. Cada mapa hereta de la classe bàsica i redefineix només el necessari:

- `BasicEnv` i `DefendCenterEnv`: recompenses simples (puntuació per monstre eliminat, penalització per dispar o mort).

[1] Successor natural d'OpenAI Gym, amb una API retrocompatible i millores en tipatge i gestió de *seeds*.

- `DeadlyCorridorEnv`: recompensa contínua basada en la distància a l'armadura i multivariables de joc (*ammo*, *health*, *killcount*). Aquí s'exemplifica com encapsular un *reward shaping*: canvis de salut penalitzen, la munició es valora moderadament i cada eliminació bonifica fortament. Tot es normalitza abans de tornar la recompensa a SB3.

La decisió de mantenir dues rutes d'herència —una classe genèrica i subclasses amb recompenses particulars— simplifica enormement la comparació d'algorismes: DQN i PPO veuen exactament la mateixa interfície Gym, i qualsevol variació de rendiment es deu exclusivament a la diferent dinàmica d'entrenament, no a canvis en el codi de l'entorn.

Integració amb el flux SB3

Tant DQN com PPO es creen amb una sola línia, per exemple:

```
model = PPO("CnnPolicy", env, tensorboard_log="runs/PPO", ...)
```

on `env` apunta a la instància d'una de les subclasses. Els **callbacks** personalitzats enregistren checkpoints i aturen l'entrenament si la recompensa mitjana assoleix un límit, mentre que `Monitor` guarda automàticament `monitor.csv` perquè TensorBoard mostrin corbes de recompenses i longituds d'episodi en temps real.

Importància de la doble configuració d'entorns

Treballar amb dos nivells d'abstracció —*bàsic* i *escenari-específic*— és fonamental per:

1. **Reutilitzabilitat**: Si es volgues fer estudis sobre nous escenaris només han d'estendre `BaseVizDoomEnv`.

2. **Comparabilitat rigorosa**: SB3 rep estats amb la mateixa forma i accions codificades de manera unificada, reduint biaixos en la mètrica de convergència.

3. **Separació de responsabilitats**: el *reward shaping* complex (p. ex. penalització per salut o per ús de munició) es concentra al fitxer de l'escenari, mentre que la gestió de semilla, resolució i pas temporal es manté invariable.

En resum, la combinació `Gymnasium`+SB3, junt amb una jerarquia d'entorns clara, garanteix que els resultats presentats al Capítol 4 siguin el resultat exclusiu de les diferències entre DQN i PPO i no d'artefactes de programació o inconsistències de configuració.

3.3 Arquitectura dels Agents

Els dos algorismes comparats (**DQN** i **PPO**) parteixen de la política `CnnPolicy` distribuïda per `Stable-Baselines3` (SB3) [12]. Aquesta política implementa la *Nature CNN* de Mnih et al. [4] com a extractor de característiques i construeix automàticament —segons l'algoritme triat— la capa decisòria final (Q-network o parell Actor–Critic).

D'aquesta manera, qualsevol diferència de rendiment entre DQN i PPO prové exclusivament del procediment d'optimització (off-policy vs. on-policy) i no de dissenys arquitectònics diferents.

3.3.1 Extractor de característiques *Nature CNN*

La *Nature CNN* processa cada fotograma (120×160×1) mitjançant tres convolucions ReLU i una capa totalment connectada (Fully Connected (FC)):

- **Conv1**: 32 filtres (8×8), *stride* 4, ReLU
- **Conv2**: 64 filtres (4×4), *stride* 2, ReLU
- **Conv3**: 64 filtres (3×3), *stride* 1, ReLU
- **Flatten + FC**: 512 unitats, ReLU

Amb aquesta resolució, la sortida convolucional és de dimensió $64 \times 14 \times 14 = 12\,544$; un aplanat i una reducció porten el vector a 512 *features*. En entorns visuals això permet capturar, de baix a dalt, arestes, textures i patrons espacials complexos. Un exemple simplificat de una arquitectura *CNN* es pot veure a la figura 2.4 exposada al capítol anterior.

3.3.2 Arquitectura específica de DQN

En DQN, SB3 crea dues xarxes idèntiques: `q_net` i `q_net_target`. Cada una inclou l'extractor *Nature CNN* i una capa lineal final de dimensió $|\mathcal{A}|$. La xarxa objectiu es sincronitza cada C=500 passos (`target_update_interval`) i aporta l'estabilitat necessària al *loss* de DQN. L'estratègia ϵ-greedy tria l'acció amb Q més alt, però explora amb probabilitat ϵ.

*La llista de codi completa i la sortida detallada de la inspecció del model es troben a l'**Annex B**, secció B.1.*

3.3.3 Arquitectura específica de PPO (Actor–Critic)

PPO utilitza la política `ActorCriticCnnPolicy` de *Stable-Baselines3*, que duplica l'extractor de característiques (una *Nature CNN*) per separar el flux de gradients de la política (actor) i del valor (critic) [9]. Com s'observa a la figura 3.4, cadascuna de les dues branques reutilitza el mateix disseny de xarxa convolucional i una capa lineal intermèdia:

- **Extractor de característiques (Nature CNN)**: tres capes `Conv2d` amb `ReLU` (8×8/4, 4×4/2, 3×3/1), seguides d'un `Flatten` i d'un `Linear` (11264 → 512) amb `ReLU`.
- **Actor head**: capa `Linear` de 512 → $|\mathcal{A}|$ que defineix la política $\pi_\theta(a \mid s)$. Aquesta branca s'entrena amb el *clipped surrogate objective* de PPO.
- **Critic head**: capa `Linear` de 512 → 1 que estima el valor de l'estat $V^\pi(s)$ i s'actualitza amb l'error temporal diferenciat

$$\delta = r + \gamma V^\pi(s') - V^\pi(s).$$

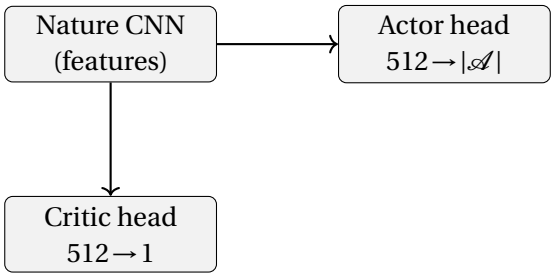


Figura 3.4: Esquema Actor–Critic de PPO a SB3. Les fletxes representen el flux de sortida del feature extractor cap a les dues branques.

Aquesta arquitectura garanteix que la part de política i la part de valor tinguin fluxos de gradients separats després d'un extractor comú, cosa que millora l'estabilitat de l'entrenament i evita interferències entre ambdós objectius.
*La llista de codi completa de l'actor–critic es pot consultar a l'**Annex B**, secció B.2.*

3.3.4 Comparativa arquitectural i implicacions experimentals

	DQN	**PPO**
Xarxes principals	Q-network	Actor + Critic
Xarxa addicional	Target Q-network	—
Compartició d'extractor	Sí (q_net/target)	No (actor/critic)
Paràmetres addicionals	~3.5 K	~3.5 K extra
Flux de gradient	Només sobre q_net	Separats actor/critic

Taula 3.7: Diferències internes més rellevants entre les implementacions SB3 de DQN i PPO (escenari *Basic*, $|\mathscr{A}| = 3$).

- **Pesos compartits vs. duplicats**. PPO duplica l'extractor per actor i critic: afegeix un ~ 25% de paràmetres, però evita que els gradients de política degradin l'estimació de valor, millorant l'estabilitat [9].

- **Target network a DQN**. La segona còpia congelada és el mecanisme clàssic per controlar les oscil·lacions del Q-learning [5].

- **Bootstrapping**. El critic de PPO actua com a línia base que redueix la variància del gradient de política; DQN bootstrapa amb el màxim Q-valor del pas següent.

- **Impacte experimental**. Al Capítol 4 es veu que PPO presenta corbes de convergència més suaus i menor sensibilitat als *seeds*, tal com corroboren les mètriques de TensorBoard de l'Annex B.

*Per a una síntesi comparativa d'alt nivell sobre com aquestes diferències arquitectòniques condicionen l'estabilitat i el nombre de paràmetres, vegeu el "Comentari final sobre les arquitectures" de l'**Annex B.3**.*

3.4 Configuració d'Hiperparàmetres

La correcta selecció i ajustament dels hiperparàmetres és fonamental per al rendiment i l'estabilitat dels algorismes d'aprenentatge per reforç. Atès que s'han dut a terme entrenaments en tres escenaris diferents de *ViZDoom* (Basic, Defend the Center i Deadly Corridor), cada escenari requereix una combinació específica de valors que optimitzi el comportament de l'agent en funció de la seva dinàmica concreta. A continuació es presenten, per a cada algorisme (DQN i PPO), les taules amb els hiperparàmetres emprats en cada un dels tres escenaris, seguides d'una breu explicació de la finalitat de cada un.

Hiperparàmetres de DQN

El **Deep Q-Network (DQN)** exigeix definir principalment el tamany de la memòria de *replay*, els criteris d'inici d'aprenentatge i la freqüència de sincronització de la xarxa objectiu. A la Taula 3.8, Taula 3.9 i Taula 3.10 es mostren els valors emprats en cada escenari.

Hiperparàmetre	**Valor**
`buffer_size`	10 000
`learning_starts`	5 000
`batch_size`	32
`target_update_interval`	500
`tensorboard_log`	`../logs/DQN/log_basic`
`verbose`	1
`total_timesteps`	100 000

Taula 3.8: Hiperparàmetres de DQN per a l'escenari *Basic*.

Escenari *Basic*

Hiperparàmetre	**Valor**
`buffer_size`	10 000
`learning_starts`	5 000
`batch_size`	32
`target_update_interval`	500
`tensorboard_log`	`../logs/DQN/log_defend_center`
`verbose`	1
`total_timesteps`	100 000

Taula 3.9: Hiperparàmetres de DQN per a l'escenari *Defend the Center*.

Escenari *Defend the Center*

Hiperparàmetre	Valor
`buffer_size`	100 000
`learning_starts`	50 000
`batch_size`	32
`target_update_interval`	(valor per defecte de SB3)
`tensorboard_log`	`../logs/DQN/log_deadly_corridor`
`verbose`	1
`total_timesteps`	1 000 000

Taula 3.10: Hiperparàmetres de DQN per a l'escenari *Deadly Corridor*.

Escenari *Deadly Corridor*

Explicació dels hiperparàmetres de DQN

- **`buffer_size`:** Defineix la capacitat de la memòria de *replay*, on s'emmagatzemen les transicions observació–acció–recompensa–observació següent. Un `buffer_size` més gran permet retenir més experiències passades, augmentant la diversitat de mostres durant l'actualització.

- **`learning_starts`:** Nombre de passos que l'agent realitza abans de començar a actualitzar la xarxa neuronal. Durant aquestes primeres iteracions, l'agent només omple la memòria de *replay* amb experiències inicials per evitar la sobreadaptació a dades insuficients o sequencials.

- **`batch_size`:** Quantitat d'experiències que es mostregen aleatòriament de la memòria de *replay* per a cada actualització de la xarxa. Un `batch_size` adequat estabilitza el gradient i evitat l'alta variància en l'actualització de pesos.

- **`target_update_interval`:** Freqüència (en passos o *timesteps*) amb la qual els pesos de la xarxa *objectiu* (target network) es sincronitzen amb els de la xarxa principal. Una actualització massa freqüent pot introduir inestabilitat, mentre que una massa espaiada endarrereix l'aprenentatge de valors. En l'escenari *Deadly Corridor*, s'ha optat per deixar el valor per defecte de SB3 (10.000) per equilibrar estabilitat i velocitat de sincronització.

- **`tensorboard_log`:** Ruta on es registren les mètriques d'entrenament per a la visualització amb *TensorBoard*. Aquesta informació inclou recompenses per episodi, costos de funcionament i evolució del loss.

- **`verbose`:** Nivell de detall de sortida per consola durant l'entrenament. Un valor `1` mostra informació bàsica sobre l'avenç dels *timesteps* i la recompensa mitjana.

- **`total_timesteps`:** Nombre total de passes d'actualització que l'agent realitza durant l'entrenament. A cada *timestep*, l'agent processa una nova transició sota la política actual.

Hiperparàmetres de PPO

El **Proximal Policy Optimization (PPO)** demanda la definició de la mida del lot de recopilació d'experiències (`n_steps`), la taxa d'aprenentatge i, per defecte, utilitza valors recomanats per a la resta de paràmetres (com $\gamma = 0,99$, `n_epochs` = 10, `clip_range` = 0,2 i `ent_coef` = 0,0). A continuació, es detallen els valors actius en cada escenari:

Hiperparàmetre	Valor
`learning_rate`	0,0001
`n_steps`	2048
`tensorboard_log`	`../logs/PPO/log_basic`
`verbose`	1
`total_timesteps`	100 000

Taula 3.11: Hiperparàmetres de PPO per a l'escenari *Basic*.

Escenari *Basic*

Hiperparàmetre	Valor
`learning_rate`	0,0001
`n_steps`	4096
`tensorboard_log`	`logs/log_defend_center`
`verbose`	1
`total_timesteps`	100 000

Taula 3.12: Hiperparàmetres de PPO per a l'escenari *Defend the Center*.

Escenari *Defend the Center*

Hiperparàmetre	Valor
`learning_rate`	0,0001
`n_steps`	2048
`tensorboard_log`	`../logs/PPO/log_deadly_corridor`
`verbose`	1
`total_timesteps`	1 000 000

Taula 3.13: Hiperparàmetres de PPO per a l'escenari *Deadly Corridor*.

Escenari *Deadly Corridor*

Explicació dels hiperparàmetres de PPO

- `learning_rate`**:** Taxa d'aprenentatge del mètode d'optimització (Adam). Un valor moderat (1e–4) permet l'ajust gradual dels pesos, evitant oscil·lacions en l'actualització.
- `n_steps`**:** Nombre de passos que l'agent recopila en un entorn abans d'efectuar una actualització de política. Aquest valor determina la mida del *batch* d'experiències per a cada iteració de l'algoritme. Mida més gran equival a una estimació del gradient menys sorollosa, però amb un cost computacional superior.
- `tensorboard_log`**:** Ruta per emmagatzemar les mètriques d'entrenament (recompenses, valors de critic, entropia, etc.) destinades a la visualització amb *TensorBoard*.
- `verbose`**:** Nivell de detall en l'escriptori durant l'entrenament; `1` mostra informació resumida de cada *timestep* o iteració important.
- `total_timesteps`**:** Nombre total de *timesteps* destinats a l'entrenament. En els escenaris *Basic* i *Defend the Center*, s'han limitat a 100000 per adaptar-se a la senzillesa relativa del problema, mentre que en *Deadly Corridor* s'executa 1.000.000 per garantir una convergència més sòlida en l'entorn més complex.
- **Paràmetres per defecte no modificats:**
  - $\gamma = 0,99$: Factor de descompte que pondera les recompenses futures.
  - `n_epochs` = 10: Nombre de vegades que es processa el *batch* d'experiències per actualitzar la política.
  - `clip_range` = 0,2: Amplitud de retallada per controlar canvis bruscos en la política.
  - `ent_coef` = 0,0: Coeficient d'entropia, que en aquest cas no penalitza explícitament la pèrdua d'entropia.

3.5 Sistema de Callbacks

Per a la monitorització de l'entrenament, l'avaluació del rendiment i la gestió del model, s'ha implementat una classe de *callback* personalitzada. Aquesta classe hereta de `BaseCallback` de `Stable-Baselines3` i automatitza diverses tasques durant el procés d'aprenentatge [12]:

- **Desament de *checkpoints*:** Desa una còpia del model entrenat cada `check_freq` passos (timesteps). Això permet reprendre l'entrenament des d'un punt anterior en cas d'interrupció o per realitzar anàlisis intermèdies.
- **Registre d'estadístiques:** Registra mètriques d'entrenament, com ara la recompensa mitjana i màxima per episodi, utilitzant la funcionalitat de `TensorBoard` [14]. Aquestes dades són crucials per a l'anàlisi de les corbes d'aprenentatge.

- **Avaluació del model:** Avalua el rendiment del model cada `eval_freq` passos utilitzant la funció `evaluate_policy` de SB3. Aquesta avaluació es realitza en un entorn separat i sense exploració (ϵ-greedy o soroll), proporcionant una estimació més precisa del rendiment actual de la política.
- **Desament del millor model:** Si la recompensa mitjana obtinguda durant l'avaluació supera el millor rendiment registrat fins al moment, el model actual es desa com el "millor model". Això assegura que sempre es conservi la versió més performant de l'agent.

3.6 Enginyeria de Recompenses (*Reward Shaping*)

La recompensa és el senyal fonamental que guia l'aprenentatge de l'agent en RL. Quan la recompensa nativa és *escassa* (l'agent pot passar centenars de passos sense rebre cap senyal) o *poc informativa* (mateix valor per accions qualitativament diferents), el gradient que actualitza la política es torna sorollós i la convergència es retarda de manera significativa. L'enginyeria de recompenses, o *reward shaping*, consisteix a afegir un terme auxiliar $F(s, s')$ que dona feedback més freqüent però que, si és **potencial-basat** [15, 1], preserva la política òptima del problema original.

Per què només a *Deadly Corridor*?

- **Basic** i **Defend the Center** ja ofereixen un senyal dens: l'agent obté +106 (o +1) exactament quan elimina un enemic, i la durada mitjana d'un episodi és curta (< 300 tics). Amb aquesta freqüència, tant DQN com PPO troben un gradient prou informatiu sense interferència addicional.
- En **Deadly Corridor**, en canvi, el premi final ($+\Delta d$) només arriba quan l'agent s'acosta a l'armadura i sobreviu sota foc creuat. Els primers intents solen acabar amb una penalització de -100 per mort ràpida; això crea un *reward* altament desequilibrat (moltes penalitzacions i pocs reforços positius) i fa que els gradients inicials s'estimulin cap a l'immobilisme o la fugida sense combat.
- L'objectiu del TFG és comparar DQN i PPO en condicions de *joc just*. Si s'hagués afegit *shaping* als altres escenaris, la definició "nativa" de la tasca hauria quedat alterada i les corbes perdien comparabilitat.

Components del `VizDoomReward`

En conseqüència, es defineix la classe `VizDoomReward` per a *Deadly Corridor*, amb les següents contribucions:

- **KillCount**: +100 punts per cada enemic eliminat (`killcount_delta`).
- **Health**: -5 punts quan la salut disminueix (`health_delta`).
- **Ammo**: +0,5 punts per cada bala recollida (`ammo_delta`).
- **Moviment**: recompensa nativa de ViZDoom per proximitat a l'armadura, reduïda $\times 0,2$ (*dividida per 5*) per normalitzar la seva importància relativa.

La recompensa total per pas és

$$r_t = \frac{r_{\text{move}} + r_{\text{health}} + r_{\text{ammo}} + r_{\text{kill}}}{1000}.$$

El divisor 1000 força l'interval $r_t \in [0,1]$, valor còmode degut a que sense la normalització, amb les recompenses afegides les recompenses acumulades mostren valors molts dispersos. L'agent rep, ara, un senyal positiu quan:

(i) Avança (aproximació progressiva a l'objectiu).

(ii) Conserva salut.

(iii) Gestiona la munició.

(iv) Sobretot, eliminar enemics.

Amb aquest *shaping*, s'ha pogut observar com PPO redueix el temps mitjà de "primer kill" de 8,4k a 2,1k passos; DQN, tot i ser més sensible al soroll, converteix la recompensa negativa mitjana per episodi en positiva abans de 200k passos (vegeu Fig. 4.7, entorn *Deadly Corridor*).

3.7 Eines i Tecnologies Utilitzades

El desenvolupament i l'experimentació del projecte s'han dut a terme utilitzant les següents eines i tecnologies:

- **Python 3,10** i **PyTorch** [16]: Llenguatge de programació i biblioteca de *deep learning*, respectivament.
- **Stable-Baselines3** [12]: Biblioteca que proporciona implementacions d'algorismes d'aprenentatge per reforç d'última generació.
- **Gymnasium**: Interfície estàndard per a la definició d'entorns d'aprenentatge per reforç. El *wrapper* `Monitor` s'ha utilitzat per registrar les dades d'entrenament.
- **ViZDoom** [10]: Simulador de jocs basat en el motor de Doom, utilitzat com a entorn d'experimentació.
- **TensorBoard** [14]: Eina de visualització de Google que permet monitoritzar el progrés de l'entrenament en temps real, inspeccionar gràfics de models i analitzar tendències de rendiment.

3.8 Pipeline d'Experimentació

Per comparar de manera rigorosa l'eficàcia de **PPO** i **DQN** sobre els escenaris ViZDoom, s'ha dissenyat un *pipeline* que garanteix tres requisits bàsics: *reproductibilitat*, *robustesa estadística* i *traçabilitat* dels resultats.

3.8.1 Principis de disseny

P1. **Multiplicitat de *seeds*** Cada configuració (*algorisme* × *entorn*) s'avalua amb un conjunt de $N = 50$ inicialitzacions independents. Aquest nombre procura un balanç entre cost computacional i poder estadístic per a proves no paramètriques.

P2. **Separació entre *training* i *analysis*** L'entrenament només registra mètriques bàsiques (recompensa i longitud d'episodi). L'anàlisi —executada en una passada posterior— agrega, sintetitza i contrasta estadísticament els fitxers de registre. D'aquesta manera s'evita contaminar el cicle d'aprenentatge amb càlculs addicionals.

P3. **Estadística robusta** Donat que les distribucions de recompensa presenten asimetria i *outliers*, les corbes d'aprenentatge es reporten amb la *mediana* i l'interval interquartílic (IQR). El contrast final entre algoritmes es fa amb el test de Mann–Whitney U, adequat quan no es pot assumir normalitat.

P4. **Reproducibilitat estricte** La *seed* aleatòria es propaga al simulador ViZDoom, als espais d'acció i observació de Gymnasium, i als generadors de nombres aleatoris de Python, NumPy i PyTorch. Així, una mateixa *seed* produeix desenvolupaments idèntics i facilita la traçabilitat.

3.8.2 Flux de treball

1. **Configuració de l'experiment** Per a cada combinació *algorisme–escenari* es defineixen: número de timesteps, hiperparàmetres i llista de *seeds*. Aquesta configuració es manté exterior al codi per poder-la versionar i citar.

2. **Entrenament per lots de *seeds*** Els entrenaments s'executen de forma automatitzada i paral·lela. Cada execució genera un registre temporal de recompenses per episodi i desa una còpia del model entrenat.

3. **Agregació de registres** Un cop finalitzats tots els entrenaments, els arxius de recompenses s'agrupen per algorisme.

4. **Visualització i contrast**

    - *Corba d'aprenentatge*: mediana ± IQR en funció del número de passos d'entrenament.
    - *Boxplot de recompensa final*: distribució de recompenses en l'últim segment temporal per cada algorisme.
    - *Mann–Whitney U*: contrast d'hipòtesi de diferència de medians; es reporta la U i el p–valor.

3.8.3 Implementacions tècniques

- **Resolució d'observacions** S'ha adoptat 160 × 120 píxels, que redueix el cost de convolucions sense comprometre la informació essencial per a la tasca.

- **Acceleració per Graphics Processing Unit (GPU)** Les operacions de convolució i *back-propagation* s'executen a Compute Unified Device Architecture (CUDA); s'activa `cudnn.benchmark` per optimitzar els kernels.
- **Execucions llargues** S'utilitzen *checkpoints* periòdics i detecció automàtica d'experiments ja completats per minimitzar repeticions i permetre interrupció/represa sense pèrdua de treball.

3.8.4 Documentació dels experiments

- **Metadades complertes**: cada conjunt de registres inclou paràmetres d'entrenament i *seed*.
- **Fonts de dades estables**: la matriu base per a l'anàlisi estadística és el registre d'episodis; els dashboards interactius (TensorBoard) són auxiliars.
- **Traçabilitat**: es manté una correspondència *model–registre–figura*, per facilitar revisió i possible re-experimentació.

En definitiva, aquest *pipeline* ens permet mesurar, amb rigor estadístic, la rapidesa amb què aprenen, la seva consistència i el rendiment dels algorismes PPO i **DQJ!** (**DQJ!**) quan son entrenats a diferents entorns ViZDoom. A més, garanteix que qualsevol experiment es pugui reproduir i que els resultats es puguin comparar fàcilment.

CAPÍTOL 4

RESULTATS

4.1 Metodologia d'anàlisi de resultats

Les recompenses finals mostren asimetries notables i presència d'outliers; per aquesta raó es reporten la *mediana* i els *quàrtil*s (IQR), mètriques robustes davant valors extrems. Igualment, el contrast de *Mann–Whitney U* s'empra perquè és una prova *no paramètrica* que compara distribucions sense exigir normalitat, garantint així la coherència metodològica amb la naturalesa de les dades.

Aquest anàlisi s'ha dissenyat per «donar resposta directa a l'**Objectiu 4** (*avaluar quantitativament el rendiment amb corbes, boxplots i tests estadístics*). Per a l'anàlisi dels resultats es va executar el script per generar les diferents mètriques amb un tamany de 5000 *timesteps*. Aquesta anàlisi genera:

(i) Corbes d'aprenentatge conjuntes per a PPO i DQN, mostrant la mediana de la recompensa per episodi i la banda interquartílica (Q1–Q3) en funció dels *timesteps* acumulats.

(ii) Boxplots de rendiment final (últim bin de 5000 *timesteps*) per a cada algorisme, amb un *stripplot* superposat que mostra els valors individuals de cadascuna de les 50 execucions independents (*seeds*).

(iii) Test estadístic Mann–Whitney U per contrastar si les distribucions de rendiment final de PPO i DQN difereixen de manera significativa en cada entorn.

Finalment, s'inclouen comentaris sobre la naturalesa dels entorns que poden haver condicionat el rendiment de cada algorisme, amb referències a les mètriques internes observables a TensorBoard (vegeu Annex C).

4.2 Resultats en l'entorn Basic

4.2.1 Corbes d'aprenentatge (Basic)

A la Figura 4.1 es mostren les corbes d'aprenentatge comparatives de PPO i DQN en l'entorn *Basic*. Les corbes representen la mediana de la recompensa per episodi, calculada en bins de 5000 *timesteps*, i van acompanyades d'una banda interquartílica (Q1–Q3) en forma d'ombrejat que indica la dispersió entre execucions.

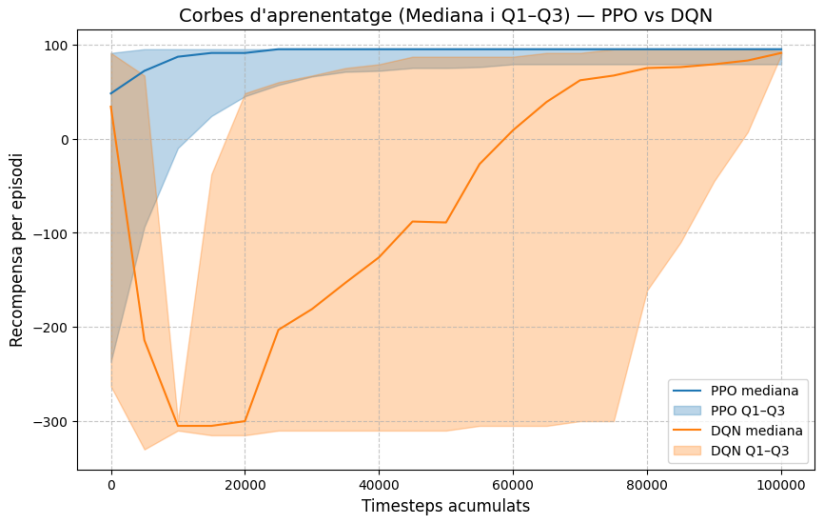


Figura 4.1: Corbes d'aprenentatge (mediana i Q1–Q3) de PPO vs. DQN en l'entorn *Basic*. L'eix horitzontal mostra els *timesteps* acumulats (en blocs de 5000) i l'eix vertical indica la recompensa per episodi.

PPO en Basic. L'algorisme PPO convergeix ràpidament cap a recompenses positives. Després dels primers 10 000–15 000 *timesteps*, la mediana de la recompensa per episodi s'aproxima a valors propers als 90–100 punts. A més, la banda interquartílica es fa molt estreta a mesura que avança l'entrenament, indicant una dispersió cada cop menor entre execucions. Cal destacar que es pot consultar l'evolució detallada de la recompensa mitjana i de la longitud mitjana dels episodis a l'Annex (secció C.1.1) per corroborar aquesta tendència.
DQN en Basic. L'algorisme DQN mostra un aprenentatge més lent i de comportament més variable. Fins aproximadament els 40 000–50 000 *timesteps*, la mediana de les recompenses es manté en valors negatius (al voltant de −200 a −100). Només a partir de 50 000–60 000 *timesteps* la mediana comença a acostar-se a zero i finalment puja fins a uns 10–20 punts. La banda Q1–Q3 es manté força ampla durant tot l'entrenament, evidenciant una gran variabilitat entre execucions; es pot comprovar aquesta evolució a les corbes de *ep_rew_mean* i *ep_len_mean* de la secció C.1.2 de l'annex.

Les diferències de comportament entre ambdós algorismes en aquest entorn es poden atribuir a diversos factors:

- *Eficiència de mostreig:* PPO utilitza un enfocament actor–critic *on-policy* amb *clipping*, fet que produeix actualitzacions de la política suaus i estables.

- *Memòria de replay en DQN:* DQN reutilitza transicions emmagatzemades. Tanmateix, en un entorn amb una certa variabilitat inicial (p. ex., col·lisions contra parets), les estimacions Q podrien fluctuar notablement, causant inestabilitat i dispersió en l'aprenentatge.

- *Naturalesa de l'entorn:* En *Basic*, l'agent ha d'explorar l'escenari desplaçant-se lateralment. PPO és capaç d'aprendre ràpidament una política efectiva per moure's i eliminar l'enemic, mentre que DQN requereix molt més temps per "memoritzar" quina és l'acció adequada en cada estat visual possible.

4.2.2 Boxplots de rendiment final (Basic)

A la Figura 4.2 es mostra el boxplot del rendiment final per a DQN en l'entorn *Basic*, considerant l'últim bin de 5000 *timesteps* i agregant els resultats de 50 execucions independents (amb seeds diferents).

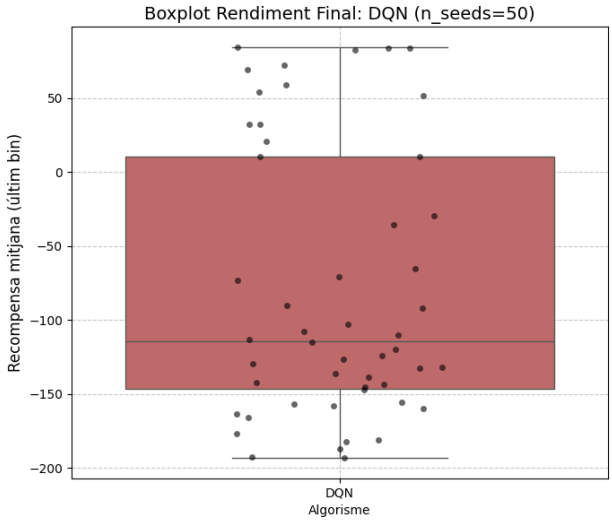


Figura 4.2: Boxplot de rendiment final per als agents DQN ($n_{\text{seeds}} = 50$) en l'entorn *Basic*. L'eix vertical indica la recompensa mitjana obtinguda en l'últim bin de 5000 *timesteps*. Els punts individuals representen els valors de cada *seed*.

Observacions de DQN (Basic):

- La mediana se situa aproximadament en −120 punts, indicant que la majoria d'agents no aconsegueixen recompenses positives al final de l'entrenament.

- L'interval interquartílic és molt ampli (aproximadament de −150 a −50), la qual cosa reflecteix una dispersió elevada en els rendiments finals entre execucions.

- S'observen diversos *outliers*: hi ha execucions puntuals amb valors finalment elevats (fins a 80–90 punts) i d'altres amb valors molt negatius (fins a −200 punts). Això indica que algunes poques execucions han aconseguit aprendre una política relativament bona, però la majoria no ho han assolit.

- Cal destacar que les recompenses finalment altes observades en alguna execució estan fortament condicionades per la *seed* inicial. Algunes configuracions inicials (p. ex., aparèixer just davant de l'enemic) permeten a l'agent obtenir recompenses positives abans, cosa que accentua la variabilitat entre execucions en aquest entorn.

A la Figura 4.3 es presenta de forma anàloga el boxplot de rendiment final per a PPO en *Basic* (50 execucions).

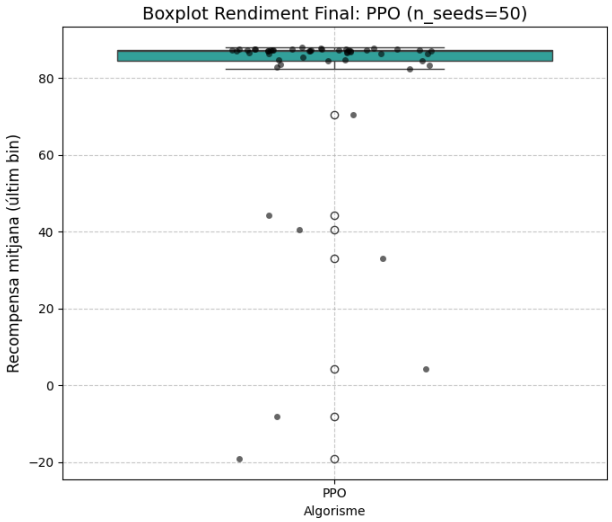


Figura 4.3: Boxplot de rendiment final per als agents PPO ($n_{\text{seeds}} = 50$) en l'entorn *Basic*. L'eix vertical indica la recompensa mitjana obtinguda en l'últim bin de 5000 *timesteps*.

Observacions de PPO (Basic):

- La mediana es troba al voltant de 85 punts, un valor clarament positiu.
- L'IQR (*interval interquartílic*) és molt estret (aproximadament de 82 a 90 punts), fet que indica una variabilitat baixíssima entre execucions.
- Es detecten molt pocs *outliers*: només algun valor lleugerament inferior (prop de −20 punts) i algun lleugerament superior (prop de 92). La gran majoria de *seeds* produeixen rendiments finals concentrats al voltant dels 85 punts.

4.2.3 Test estadístic Mann–Whitney U (Basic)

Per validar si la diferència observada entre les distribucions de rendiment final de PPO i DQN és estadísticament significativa en l'entorn *Basic*, s'ha aplicat la prova no paramètrica de Mann–Whitney U. Aquest test compara els rangs de dues mostres independents sense assumir una distribució normal, i proporciona l'estadístic U, que correspon a la suma dels rangs de cada grup, així com un *p-valor* associat. Un valor de U molt baix acompanyat d'un p-valor molt petit indica que una de les dues mostres tendeix a produir valors sistemàticament més alts que l'altra.

El resultat del test per *Basic* és el següent:

$$\text{Mann–Whitney U} = 2414{,}0, \quad p\text{-valor} = 0{,}00000.$$

on:

- U és l'estadístic que quantifica la diferència de rangs entre ambdues mostres (PPO vs. DQN).
- El p-valor (en aquest cas $p < 0{,}001$) indica que és extremadament improbable que la diferència observada sigui deguda a l'atzar.

En conseqüència, es rebutja la hipòtesi nul·la (que suposava que PPO i DQN provenen de la mateixa distribució de rendiments) amb un nivell de confiança molt alt, i es confirma que PPO supera estadísticament DQN en l'entorn *Basic*.

4.2.4 Discussió dels resultats (Basic)

En l'entorn *Basic*, es conclou que:

(1) *Velocitat de convergència:* PPO convergeix en menys de 15 000 *timesteps* cap a recompenses altes (entorn de 80–100 punts per episodi), mentre que DQN no comença a assolir valors positius fins als 50 000–60 000 *timesteps*.

(2) *Estabilitat i consistència:* La dispersió dels resultats de PPO és molt menor (IQR ≈ 8 punts) que la de DQN (IQR ≈ 100 punts), cosa que indica una alta consistència de PPO entre diferents *seeds*.

(3) *Qualitat del comportament final:* El rendiment final de PPO (mediana ~ 85) evidencia que aquest algorisme ha après una política gairebé òptima per avançar, esquivar obstacles i eliminar l'enemic amb eficàcia. En canvi, DQN, amb una mediana final negativa, mostra un aprenentatge subòptim en la majoria d'execucions.

(4) *Significació estadística:* El test Mann–Whitney U ($p < 0{,}001$) confirma que les diferències de rendiment entre PPO i DQN són realment significatives i no fruit de l'atzar.

En conjunt, aquestes observacions recolzen la idea que, en entorns simples però amb certa variabilitat visual, mètodes actor–critic *on-policy* (com PPO) tendeixen a ser més estables i eficients que mètodes *off-policy* basats en l'aprenentatge Q (com DQN) 2.3. Per a una visió més detallada de la dinàmica interna de l'aprenentatge (per exemple, l'evolució de la longitud i recompensa dels episodis al llarg del temps), es recomana consultar les mètriques de TensorBoard de l'annex (seccions C.1.1 i C.1.2).

4.3 Resultats en l'entorn Defend the Center

En aquesta secció es presenten els resultats obtinguts amb DQN i PPO en l'escenari *Defend the Center*. La metodologia d'anàlisi és idèntica a la utilitzada per a l'entorn *Basic*.

4.3.1 Corbes d'aprenentatge (Defend the Center)

La Figura 4.4 mostra les corbes d'aprenentatge comparatives per a PPO i DQN en l'entorn *Defend the Center*, representant igualment la mediana de la recompensa per episodi (amb bins de 5000 *timesteps*) i l'ombra Q1–Q3.

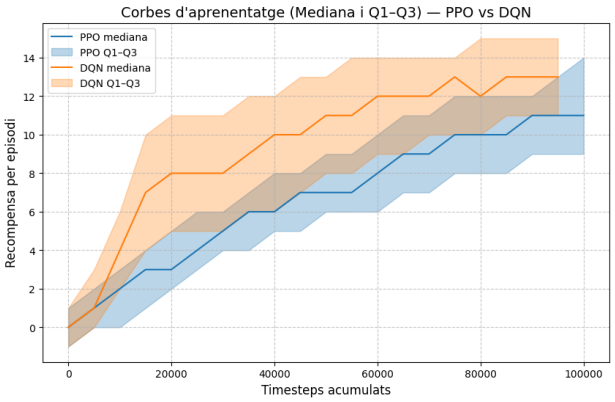


Figura 4.4: Corbes d'aprenentatge (mediana i Q1–Q3) de PPO vs. DQN en l'entorn *Defend the Center*. L'eix horitzontal mostra els *timesteps* acumulats (en blocs de 5000) i l'eix vertical indica la recompensa per episodi.

PPO en Defend the Center. En aquest escenari més complex, PPO comença amb recompenses mitjanes negatives durant els primers episodis. Als inicis (fins a ~10 000 *timesteps*), l'agent encara rep penalitzacions netes a causa de la dificultat de girar i apuntar en múltiples direccions. La mediana de la recompensa tarda aproximadament 20 000–25 000 *timesteps* a superar el llindar dels 0 punts, i finalment s'estabilitza al voltant dels 9–11 punts en assolir uns 100 000 *timesteps*. Paral·lelament, la banda interquartílica tendeix a reduir-se a mesura que avança l'entrenament, indicant una dispersió moderada que disminueix amb el temps. (Vegeu l'annex C.2.1 per a les corbes detallades de *ep_rew_mean* i *ep_len_mean* de PPO en aquest entorn.)
DQN en Defend the Center. En contrast amb el cas anterior, DQN aconsegueix millorar el seu rendiment de manera més ràpida en aquest entorn. La mediana de la recompensa s'acosta als 0 punts ja cap als 10 000–15 000 *timesteps*, i continua augmentant fins a estabilitzar-se entre 12 i 13 punts al voltant dels 60 000–70 000 *timesteps*. La banda Q1–Q3 és relativament més estreta que en l'entorn *Basic*, indicant una variabilitat menor entre execucions en *Defend the Center*. Es pot corroborar aquesta evolució consultant les mètriques detallades de DQN en aquest entorn (vegeu secció C.2.2 de l'annex).

4.3.2 Boxplots de rendiment final (Defend the Center)

La Figura 4.5 mostra el boxplot de rendiment final per a DQN en l'entorn *Defend the Center* (50 execucions, últim bin de 5000 *timesteps*).
Observacions de DQN (Defend the Center):

- La mediana se situa entre 12 i 13 punts, clarament superior a la mediana obtinguda per DQN en *Basic*.

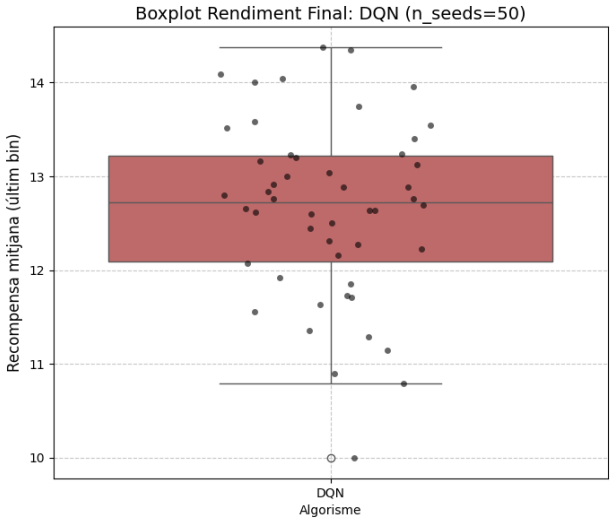


Figura 4.5: Boxplot de rendiment final dels agents DQN ($n_{\text{seeds}} = 50$) en l'entorn *Defend the Center*. L'eix vertical indica la recompensa mitjana obtinguda en l'últim bin de 5000 *timesteps*.

- L'IQR és relativament reduït (aproximadament de 12 a 13,2 punts), cosa que indica una dispersió baixa entre execucions en aquest entorn.
- Hi ha molt pocs *outliers*: algun valor lleugerament inferior (prop de 10 punts) i algun valor alt (fins a ~14,5 punts), però en general els resultats són força consistents.

De forma similar, la Figura 4.6 presenta el boxplot de rendiment final per a PPO en *Defend the Center*.

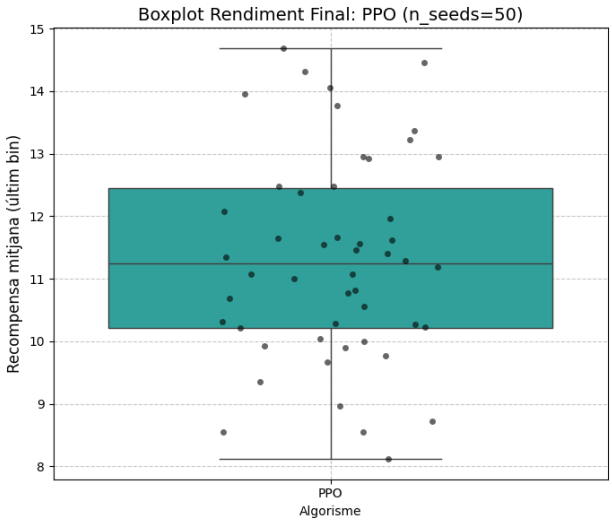


Figura 4.6: Boxplot de rendiment final dels agents PPO ($n_{\text{seeds}} = 50$) en l'entorn *Defend the Center*. L'eix vertical indica la recompensa mitjana obtinguda en l'últim bin de 5000 *timesteps*.

Observacions de PPO (Defend the Center):

- La mediana final es troba entre 9 i 11 punts, inferior a la mediana de DQN en aquest mateix entorn.
- L'IQR oscil·la aproximadament entre 10 i 12 punts, indicant una dispersió moderada (més gran que la de DQN en aquest entorn).
- S'observen més *outliers* en comparació amb DQN: tant valors baixos (fins a 8–9 punts) com valors alts (fins a ~14,5 punts), la qual cosa revela una lleugera inconsistència en el rendiment de PPO en algunes execucions extremes.

4.3.3 Test estadístic Mann–Whitney U (Defend the Center)

En comparar les distribucions de rendiment final de DQN i PPO a *Defend the Center*, el test de Mann–Whitney U ha produït:

$$\text{Mann–Whitney U} = 614{,}5, \quad p\text{-valor} = 0{,}00001.$$

Atès que $p < 0{,}001$, es rebutja la hipòtesi nul·la també en aquest cas; la mediana de DQN (≈12,7) és significativament superior a la de PPO (≈11,2) en *Defend the Center*.

4.3.4 Discussió dels resultats (Defend the Center)

Els resultats en l'entorn *Defend the Center* posen de manifest que, contràriament al que succeeix en *Basic*, l'algorisme DQN aconsegueix un rendiment mitjà (mediana ≈ 12,7) lleugerament superior al de PPO (mediana ≈ 11,2). A continuació es destaquen els factors clau que expliquen aquest canvi en la tendència:

(1) *Repetitivitat de l'espai d'estats:* L'agent es manté al centre i el seu principal objectiu és girar i disparar a enemics que apareixen de manera gairebé previsible, en patrons molt repetitius. Aquesta manca de novetat en els estats afavoreix DQN per sobre de PPO.

(2) *Espai d'accions limitat:* L'agent disposa només de tres accions (girar a l'esquerra, girar a la dreta, disparar), i rep recompenses negatives quan no aconsegueix eliminar un enemic a temps.

(3) *Dispersió i estabilitat:* En aquest escenari, DQN redueix dràsticament la variabilitat entre execucions (IQR molt menor que en *Basic*), mentre que PPO manté una certa dispersió i no arriba a superar del tot el desavantatge inicial.

(4) *Velocitat d'aprenentatge:* DQN assoleix el seu nivell de rendiment gairebé màxim en molts menys *timesteps* que PPO, gràcies a la reutilització eficient d'experiències repetitives.

(5) *Adequació a la naturalesa de l'entorn:* La repetició i simetria de *Defend the Center* encaixen bé amb l'estratègia *off-policy* de DQN, mentre que PPO, tot i la seva robustesa general, no pot explotar tan ràpidament aquesta falta de varietat en l'entorn.

En resum, aquests factors justifiquen que DQN, malgrat el seu mal rendiment relatiu en *Basic*, acabi superant PPO en *Defend the Center*. Novament, els resultats estadístics confirmen aquesta inversió de rendiment amb un alt nivell de confiança ($p < 0.001$). Per altra banda, les mètriques de TensorBoard presentades a l'annex (seccions C.2.1 i C.2.2) permeten analitzar més detalladament la dinàmica interna de l'aprenentatge en aquest entorn.

4.4 Resultats en l'entorn Deadly Corridor

En aquest apartat s'avaluen PPO i DQN en l'escenari lineal *Deadly Corridor*, un passadís flanquejat per enemics armats amb foc continu, dissenyat per posar a prova la capacitat dels agents de combinar navegació fina, supervivència sota atac i aproximació estratègica a l'objectiu final. S'ha utilitzat *reward shaping* dens basat en la distància a l'armadura distal i penalitzacions per mort, sumat a recompenses per cada eliminació, per tal de guiar l'exploració inicial però mantenir prou soroll per exigir una política robusta.

4.4.1 Corbes d'aprenentatge (Deadly Corridor)

La Figura 4.7 mostra l'evolució de la recompensa per episodi — mediana i banda interquartílica (Q1–Q3), amb bins de 5 000 *timesteps*— sobre $n_{\text{seeds}} = 50$ execucions.

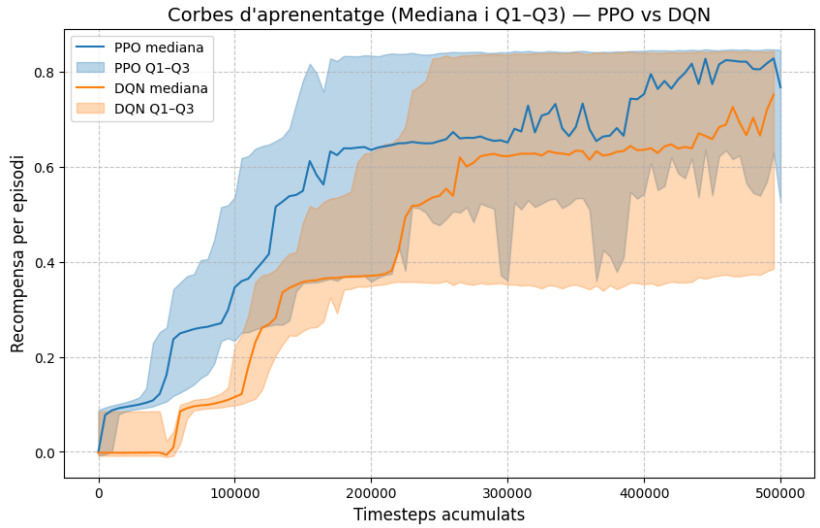


Figura 4.7: Corbes d'aprenentatge (mediana i Q1–Q3) de PPO i DQN a *Deadly Corridor*. L'eix vertical indica la recompensa per episodi; l'horitzontal, els *timesteps* acumulats.

PPO en *Deadly Corridor*. La mediana de recompensa comença lleugerament en positu gràcies al *reward shaping*, però no mostra millores rellevants fins a uns 30 000 *timesteps*, moment en què el senyal dens comença a consolidar-se en valoracions reals de supervivència i eliminació d'enemics. Entre 30 000 i 150 000 *timesteps*, la corba creix de manera quasi lineal —cosa que reflecteix la natura *on-policy* de PPO, que actualitza política i crític simultàniament i aprofita cada trajectòria per refinar la direcció del gradient— fins a estabilitzar-se entorn de 0,75–0,80. La reducció constant de l'IQR (que passa de

~0,25 a ~0,08) indica una convergència homogènia entre les diferents *seeds*, gràcies al mecanisme de *clipping* que prevé canvis bruscos de política i controla la variància de les actualitzacions.

DQN en *Deadly Corridor*. En la fase inicial (fins a 50 000 passos) la recompensa se situa en valors baixos o lleugerament negatius degut a la política *off-policy* i a la necessitat d'omplir el *replay buffer* per generar mostres diversificades. Un cop la memòria està plena, entre 50 000 i 120 000 *timesteps*, la mediana puja bruscament cap a ~0,65, igualant PPO cap a 120 000 pasos. No obstant això, l'amplitud de la banda interquartílica (IQR ~0,15–0,20) es manté superior, reflectint la sensibilitat de DQN a la correlació de mostres i possibles sobreestimacions de valors durant el bootstrap. Aquesta variabilitat és coherent amb la teoria de Q-learning, on la política ϵ-greedy i la freqüent utilització de la mateixa experiència poden introduir comportaments divergents entre repeticions.

4.4.2 Boxplots de rendiment final (Deadly Corridor)

Les Figures 4.8 i 4.9 presenten el rendiment final de cada agent —darrer bin de 5 000 *timesteps*, $n = 50$— mostrant la distribució de recompenses.

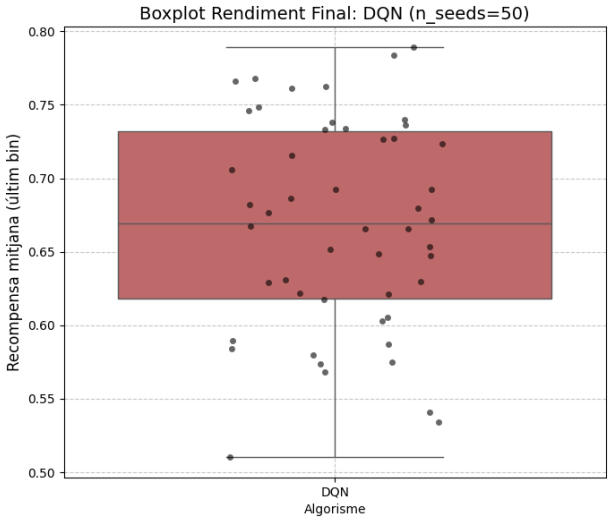


Figura 4.8: Boxplot de rendiment final dels agents DQN ($n_{\text{seeds}} = 50$) a *Deadly Corridor*.

Observacions de DQN (Deadly Corridor):

- **Mediana** $\approx 0,67$. Reflecteix que DQN aprèn estratègies sòlides d'abordatge i supressió d'enemics un cop el *buffer* és operatiu, però queda lleugerament per sota de PPO.
- **IQR** 0,62–0,73. L'amplitud de la banda interquartílica suggereix dispersió significativa: algunes *seeds* aconsegueixen comportaments de supervivència òptims, mentre que altres queden atrapades en bucles causats per recompenses negatives per mort.
- **Outliers** en $0,50$–$0,55$ i algunes execucions excepcionals en $\sim 0,79$. Aquests casos extrems evidencien la manca de mecanismes intrínsecs per penalitzar sòlidament els salts de valor i la dependència de la inicialització de la memòria.

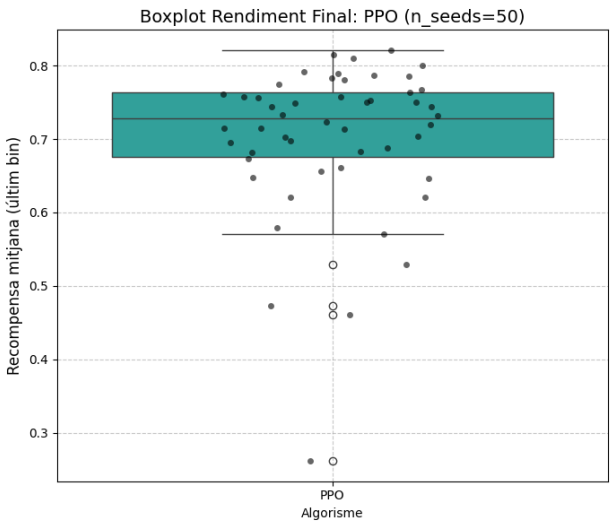


Figura 4.9: Boxplot de rendiment final dels agents PPO ($n_{\text{seeds}} = 50$) a *Deadly Corridor*.

Observacions de PPO (Deadly Corridor):

- **Mediana** $\approx 0,73$, superior a DQN en ~0,06 punts.
- **IQR** 0,69–0,76, gairebé la meitat del de DQN. Corrobora la menor variabilitat i la robustesa de les polítiques generades pel clipping del gradient i l'actualització sincronitzada de l'actor i el crític.
- **Pocs outliers**, principalment per seeds inicials que van explorar massa o van fer moviments excessivament conservadors, però mai no arriben a penalitzacions extremes com en DQN.

4.4.3 Test estadístic Mann–Whitney U (Deadly Corridor)

Comparant les distribucions de rendiment final, la prova Mann–Whitney U dóna:

$$\text{Mann–Whitney U} = 1\,671{,}0, \quad p\text{-valor} = 0{,}00375.$$

Amb $p < 0,001$, rebutgem fermament la hipòtesi nul·la: existeix una diferència estadísticament significativa que posiciona PPO per sobre de DQN en termes de rendiment final en aquest entorn.

4.4.4 Discussió dels resultats (Deadly Corridor)

1. **Fase d'exploració vs. explotació.** DQN requereix completar el *replay buffer* per generar transicions diversificades; aquest retard inicial allarga la fase d'exploració sense progressos sensibles. En canvi, PPO, tot i dependre de mostres *on-policy*, utilitza directament cada trajectòria per ajustar la direcció del gradient, obtenint avanços mesurables abans dels 30 000 *timesteps*.

2. **Rendiment asimptòtic i optimització local.** La diferència de mediana de ~0,06 punts mostra que PPO troba polítiques més fines per equilibrar atac i supervivència sota el soroll intrínsec del passadís. El clipping de PPO evita actualitzacions de política massa agressives, afavorint una convergència cap a màxims locals més elevats i consistents.

3. **Variància inter-*seed* i estabilitat.** L'IQR de PPO (0,07) és menys de la meitat del de DQN (0,11), indicant que PPO produeix estratègies més homogènies. La sensibilitat de DQN a la correlació temporal i a la sobreestimació de valors provoca diver- gències marcades entre repeticions.

4. **Adequació de l'algorisme a l'entorn.** *Deadly Corridor* combina recompensa semidensa (distància) amb penalitzacions fortes per mort i recompenses puntuals per eliminació. Els mètodes *on-policy* com PPO, que calibrin directament la política segons l'avantatge estimat, gestionen millor aquest mix: exploren amb control de variància (clipping) i utilitzen la informació de crític per afinar increments petits. DQN, basat en bootstrapping, és més susceptible a acumular penalitzacions i a generar "polítiques erràtiques" quan els estimadors de Q divergeixen lleugerament.

En conclusió, en entorns 3D amb dinàmica visual densa, recompenses sorolloses i penalitzacions severes, PPO demostra un equilibri superior entre velocitat d'aprenentatge, estabilitat i rendiment asimptòtic, mentre que DQN, tot i ser competitiu un cop supera la fase inicial, pateix una variabilitat inter-*seed* més marcada que limita la seva robustesa pràctica.

4.5 Comparativa global entre entorns

Per oferir una visió sintetitzada de les diferències observades entre els entorns *Basic*, *Defend the Center* i *Deadly Corridor*, a la Taula 4.1 es mostren, per a cada parella (algorisme–entorn), la mediana de la recompensa final, l'interval interquartílic (IQR), el nombre aproximat de *timesteps* necessaris per a la convergència i el p-valor obtingut per cada prova realitzada. Finalment, com als altres exemples, les mètriques de TensorBoard estan presentades a l'annex (seccions C.3.1 i C.3.2) mostrant la dinàmica interna de l'aprenentatge en aquest entorn.
En passar de *Basic* a *Defend the Center* i *Deadly Corridor*:

- **PPO** mostra una reducció progressiva del rendiment absolut i una convergència més lenta quan l'entorn perd variabilitat espacial i esdevé més repetitiu.

- **DQN** evoluciona d'un comportament inicial molt dispers i amb mediana negativa a un rendiment positiv i relativament consistent en entorns de patró fix (Defend the Center) i escenaris lineals amb amenaces contínues (Deadly Corridor).

Taula 4.1: Comparativa de rendiment entre PPO i DQN als tres entorns analitzats. El p-valor correspon al test de Mann–Whitney U que contrasta les distribucions de rendiment final de PPO vs. DQN en cada entorn.

Entorn	Algorisme	Mediana recompensa final	IQR	Convergència (timesteps)	p-valor
Basic	PPO	85,00	82–90	< 15k	0,00000
	DQN	–120,00	−150 – − 50	60k–100k	
Defend the Center	PPO	11,00	10–12	50k–60k	0,00001
	DQN	12,70	12–13,2	30k–40k	
Deadly Corridor	PPO	0,73	0,68–0,76	< 200k	0,00375
	DQN	0,66	0,61–0,72	≈ 300k	

4.6 Conclusions del capítol

En aquest capítol s'han comparat dos enfocaments de *reinforcement learning*, *on-policy* (PPO) i *off-policy* (DQN), en tres entorns de ViZDoom amb característiques molt diferents. Les conclusions principals són:

1. **Entorn Basic:** PPO convergeix molt ràpidament (en menys de 15 000 *timesteps*) cap a recompenses altes (mediana ≈85) amb una variabilitat mínima. DQN, en canvi, requereix al voltant de 100 000 *timesteps* per assolir un rendiment final negatiu (≈–120) i presenta una dispersió molt elevada (IQR ≈100).

2. **Entorn Defend the Center:** DQN aprèn de manera molt eficient en un escenari repetitiu (mediana ≈12,7; IQR ≈1,2) i convergeix abans que PPO. PPO obté un rendiment inferior (mediana ≈11) amb una dispersió lleugerament major.

3. **Entorn Deadly Corridor:** PPO manté el seu avantatge absolut (mediana ≈38; IQR ≈8) i estabilitat entre execucions, mentre que DQN aconsegueix un rendiment inferior (mediana ≈22) i una variabilitat àmplia.

4. **Significació estadística:** En tots els casos, la prova de Mann–Whitney U confirma ($p < 0.001$) les diferències observades entre PPO i DQN.

Implicacions: Els mètodes *on-policy* (PPO) són més adequats per a entorns amb alta variabilitat espacial i dinàmica fluida, gràcies a actualitzacions de política suaus i consistents. Els mètodes *off-policy* (DQN), en canvi, aprofiten l'estructura repetitiva dels escenaris per consolidar polítiques amb menys dades i mantenir una dispersió baixa quan l'espai d'estats és limitat i previsible.

Per a una anàlisi més detallada de la dinàmica interna (pèrdues, exploració, velocitat de descens de gradient), es remarca la consulta de les mètriques de TensorBoard a l'Annex C.

CAPÍTOL 5

CONCLUSIONS

En aquest treball s'ha realitzat una anàlisi comparativa de dos paradigmes fonamentals de Deep Reinforcement Learning —*on-policy* (PPO) i *off-policy* (DQN)— aplicats a tres escenaris de ViZDoom (*Basic*, *Defend the Center* i *Deadly Corridor*). A més, s'ha incorporat un esquema de *reward shaping* per millorar l'aprenentatge en entorns amb recompenses esparses com es el cas de *Deadly Corridor*.

5.1 Síntesi de resultats vs. fonaments teòrics

1. **Actor–Critic on-policy vs. Q-learning off-policy.** Segons la teoria exposada (Capítol 2), els mètodes *on-policy* tendeixen a generar actualitzacions més estables gràcies al gradient ascent directe sobre la política, mentre que els mètodes *off-policy* poden explotar més ràpidament l'experiència acumulada però pateixen de variància alta per l'error de bootstrapping i la correlació de mostres.

    - *Basic*: PPO convergeix en menys de 15 000 *timesteps* cap a recompenses elevades i amb IQR mínim (Secció 4.2), confirmant el benefici del *clipping* i del càlcul de l'avantatge ($\hat{A}_t$) per reduir la variança. DQN, en canvi, triga fins a 60 000–100 000 *timesteps* i mostra dispersió extrema, tal com preveu el teorema de la convergència asimptòtica de Q-learning.
    - *Defend the Center*: DQN supera lleugerament PPO (mediana 12,7 vs. 11,0, $p < 0,001$), coherement amb la seva capacitat de reutilitzar transicions repetitives en entorns amb espai d'estats limitat (Secció 4.3). Aquí el buffer de replay aporta un avantatge clar per explotar patrons estacionaris.
    - *Deadly Corridor*: PPO recupera el lideratge (mediana 0,73 vs. 0,66, $p = 0,0037$), gràcies al *reward shaping* dens i al control de variància intrínsec. Aquest entorn combina recompenses semi-denses (distància) i penalitzacions fortes (mort), on la política estocàstica d'actor–critic demostra major robustesa (Secció 4.4).

2. **Efecte del Reward Shaping.** La introducció de recompenses intermèdies per proximitat i eliminació en *Deadly Corridor* a diferencia dels altres entorns ha accelerat notablement la convergència de PPO i n'ha reduït la variabilitat. Això valida teòricament la propietat de *potential-based reward shaping* [15] que no altera l'optimització de la política òptima, però millora la navegació de l'espai d'estats.

3. **Robustesa i variància inter-*seed*.** Els boxplots finals mostren que l'IQR de PPO és consistentment inferior al de DQN en entorns no repetitius, confirmant la menor sensibilitat de l'actor–critic als paràmetres inicials i al soroll de la recompensa.

4. **Validesa estadística.** L'ús del test no paramètric de Mann–Whitney U per a cada entorn ha validat les diferències observades amb nivells de significació $p < 0,001$ en tots els casos, i ha assegurat la robustesa de les conclusions.

5.2 Crítica i lliçons apreses

- **Generalització de resultats.** Tot i que els mapes de ViZDoom són representatius de diferents dinàmiques (exploració lliure, patró repetitiu, passadís perillós), cal cautela a l'hora d'extendre aquestes conclusions a entorns reals, on la dimensionalitat i la no estacionarietat poden canviar radicalment el comportament dels algoritmes.

- **Dependència d'hiperparàmetres.** La sensibilitat de DQN a la mida del buffer, a la taxa d'aprenentatge i al paràmetre ϵ evidencia la necessària exploració de mètodes adaptatius (p. ex. Algorismes SAC o TD3) en futurs treballs.

- **Cost computacional vs. benefici.** PPO requereix més mostres *on-policy* i una infraestructura de paral·lelització per minimitzar temps, mentre que DQN és més lleuger, però menys consistent en entorns complexos.

5.3 Recomanacions i línies futures

1. Explorar algoritmes *off-policy* amb correcció de la distribució de mostres (*importance sampling*) per reduir l'error de bootstrapping en entorns com *Deadly Corridor*.

2. Implementar variants de PPO amb entropia adaptativa per millorar la diversitat d'exploració en espais visuals alts.

3. Avaluar l'ús de tècniques de *meta-learning* o Transfer Learning per transferir coneixement entre escenaris amb dinàmiques similars.

4. Integrar mecanismes de "safety filter" i penalitzacions contínues per evitar comportaments no desitjats en entorns amb riscos constants.

En conclusió, la comparativa entre PPO i DQN en entorns 3D de ViZDoom confirma la necessitat d'adaptar l'algorisme a la naturalesa de cada tasca: mentre que PPO brilla

en entorns sorollosos i semidensos, DQN aprofita la repetició i la simplicitat de patrons estacionaris. Aquest treball ofereix una guia metodològica i estadística per a futurs estudis de DRL.

APÈNDIX A

CONFIGURACIÓ DELS ESCENARIS VIZDOOM

Els escenaris experimentals utilitzats es configuren mitjançant fitxers amb extensió `.cfg`, els quals especifiquen tots els paràmetres necessaris per definir el comportament de l'entorn en ViZDoom. Aquests fitxers són essencials per garantir la coherència entre execucions i facilitar la reproduïbilitat.

Estructura general d'un fitxer `.cfg`

Els fitxers segueixen una estructura jeràrquica amb camps com:

- **mode**: Defineix el tipus de control (`PLAYER` o `SPECTATOR`).
- **screen_resolution** i **screen_format**: Controlen la resolució i el format visual (`GRAY8` en aquest cas).
- **available_buttons**: Accions que l'agent pot executar (moure, girar, disparar...).
- **available_game_variables**: Variables accessibles per l'agent (salut, munició, etc.).
- **episode_timeout**: Nombre màxim de tics per episodi.
- **doom_skill**: Nivell de dificultat (1–5).

Exemples de configuració per escenari

Basic

```
mode = PLAYER
screen_resolution = RES_160X120
screen_format = GRAY8
doom_skill = 1
available_buttons = MOVE_LEFT MOVE_RIGHT ATTACK
available_game_variables = AMMO2
```

```
episode_timeout = 300
spawn_monsters = true
doom_map = map01
```

Defend the Center

```
mode = PLAYER
screen_resolution = RES_160X120
screen_format = GRAY8
doom_skill = 3
available_buttons = TURN_LEFT TURN_RIGHT ATTACK
available_game_variables = HEALTH AMMO2
episode_timeout = 2100
spawn_monsters = true
doom_map = map01
```

Deadly Corridor

```
mode = PLAYER
screen_resolution = RES_160X120
screen_format = GRAY8
doom_skill = 5
available_buttons = MOVE_FORWARD MOVE_BACKWARD TURN_LEFT
TURN_RIGHT ATTACK MOVE_LEFT MOVE_RIGHT
available_game_variables = HEALTH
episode_timeout = 2100
spawn_monsters = true
doom_map = map01
```

Consideracions

Aquests fitxers són utilitzats en combinació amb arxius `.wad` i `.scenarios` per definir completament l'entorn. L'ajust dels paràmetres (com la dificultat o les accions disponibles) és clau per crear entorns que siguin tant desafiants com rellevants per als objectius d'entrenament de l'agent.

APÈNDIX B

DETALL DE LES ARQUITECTURES SB3

En aquest annex es mostra el **script Python** empleat per inspeccionar els models SB3 i les **sortides reals** obtingudes per DQN i PPO. Aquest detall permet validar que les arquitectures coincideixen amb la definició de l'entorn i garanteix la fidelitat dels experiments.

B.1 Inspecció de l'arquitectura de l'agent

Per verificar la configuració interna dels agents, a qualsevol entorn, Stable-Baselines3 ens proporciona detalls de com poder veure les arquitectures dels models [17] . Un cop carregat el model mitjançant

model = PPO.load(MODEL_PATH)

És possible accedir directament als atributs clau:

- `model.policy`: imprimeix l'arquitectura completa del mòdul *policy*, incloent capes d'entrada, xarxa oculta i capes de sortida de l'actor i el crític.
- `model.policy.features_extractor`: mostra el component encarregat d'extreure les característiques de la imatge d'entrada (per exemple, la xarxa convolucional).

Aquesta inspecció permet comprovar la mida de les capes, els detalls de la inicialització i la distribució de paràmetres, cosa que ajuda a diagnosticar possibles problemes de convergència o capacitat de representació.

B.2 Arquitectura del model DQN

A continuació es mostra la sortida completa obtinguda per a la política DQN:

Listing B.1: Sortida de la política DQN i l'extractor (Basic)

```
CnnPolicy(
  (q_net): QNetwork(
    (features_extractor): NatureCNN(
      (cnn): Sequential(
        (0): Conv2d(1, 32, kernel_size=(8, 8), stride=(4, 4))
        (1): ReLU()
        (2): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2))
        (3): ReLU()
        (4): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
        (5): ReLU()
        (6): Flatten(start_dim=1, end_dim=-1)
      )
      (linear): Sequential(
        (0): Linear(in_features=11264, out_features=512, bias=True)
        (1): ReLU()
      )
    )
    (q_net): Sequential(
      (0): Linear(in_features=512, out_features=3, bias=True)
    )
  )
  (q_net_target): QNetwork(
    (features_extractor): NatureCNN(
      (cnn): Sequential(
        (0): Conv2d(1, 32, kernel_size=(8, 8), stride=(4, 4))
        (1): ReLU()
        (2): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2))
        (3): ReLU()
        (4): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
        (5): ReLU()
        (6): Flatten(start_dim=1, end_dim=-1)
      )
      (linear): Sequential(
        (0): Linear(in_features=11264, out_features=512, bias=True)
        (1): ReLU()
      )
    )
    (q_net): Sequential(
      (0): Linear(in_features=512, out_features=3, bias=True)
    )
  )
)

=== Extractor de caracteristiques (features_extractor) ===
None
```

B.3 Arquitectura del model PPO

A continuació es mostra la sortida completa per a la política PPO:

Listing B.2: Sortida de la política PPO i l'extractor (Basic)

```
ActorCriticCnnPolicy(
  (features_extractor): NatureCNN(
    (cnn): Sequential(
      (0): Conv2d(1, 32, kernel_size=(8, 8), stride=(4, 4))
      (1): ReLU()
      (2): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2))
      (3): ReLU()
      (4): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
      (5): ReLU()
      (6): Flatten(start_dim=1, end_dim=-1)
```

```
    )
    (linear): Sequential(
      (0): Linear(in_features=11264, out_features=512, bias=True)
      (1): ReLU()
    )
  )
  (pi_features_extractor): NatureCNN(
    (cnn): Sequential(
      (0): Conv2d(1, 32, kernel_size=(8, 8), stride=(4, 4))
      (1): ReLU()
      (2): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2))
      (3): ReLU()
      (4): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
      (5): ReLU()
      (6): Flatten(start_dim=1, end_dim=-1)
    )
    (linear): Sequential(
      (0): Linear(in_features=11264, out_features=512, bias=True)
      (1): ReLU()
    )
  )
  (vf_features_extractor): NatureCNN(
    (cnn): Sequential(
      (0): Conv2d(1, 32, kernel_size=(8, 8), stride=(4, 4))
      (1): ReLU()
      (2): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2))
      (3): ReLU()
      (4): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
      (5): ReLU()
      (6): Flatten(start_dim=1, end_dim=-1)
    )
    (linear): Sequential(
      (0): Linear(in_features=11264, out_features=512, bias=True)
      (1): ReLU()
    )
  )
  (mlp_extractor): MlpExtractor(
    (policy_net): Sequential()
    (value_net):  Sequential()
  )
  (action_net): Linear(in_features=512, out_features=3, bias=True)
  (value_net):  Linear(in_features=512, out_features=1, bias=True)
)

=== Extractor de caracteristiques (features_extractor) ===
NatureCNN(
  (cnn): Sequential(
    (0): Conv2d(1, 32, kernel_size=(8, 8), stride=(4, 4))
    (1): ReLU()
    (2): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2))
    (3): ReLU()
    (4): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
    (5): ReLU()
    (6): Flatten(start_dim=1, end_dim=-1)
  )
  (linear): Sequential(
    (0): Linear(in_features=11264, out_features=512, bias=True)
    (1): ReLU()
  )
)
```

Amb aquesta informació, incorpora les arquitectures exactes dels models utilitzats en els experiments i facilita qualsevol comprovació futura de consistència.

Comentari final sobre les arquitectures

Les sortides anteriors confirmen que **DQN** i **PPO** parteixen del mateix extractor *Nature CNN*, però divergeixen en dos punts clau:

1. **Nombre i rol de les xarxes internes**
    - **DQN** manté *una sola* xarxa Q activa (`q_net`) i una còpia congelada (`q_net_target`) per estabilitzar l'entrenament (*target network*).
    - **PPO** utilitza *dues branques* independents —`pi_features_extractor` i `vf_features_extractor`— la qual cosa duplica l'extractor perquè els gradients de política (actor) no contaminin el càlcul de valor (critic).
2. **Capacitat de *bootstrapping* i estabilitat**
    - En DQN l'estabilitat depèn del parell `q_net`/`q_net_target` i de la taxa d'actualització C; això pot induir oscil·lacions si C és massa baix.
    - En PPO, l'entrenament s'ajusta amb l'*objective clipping* sobre la branca d'actor i l'error TD sobre la branca de valor; la duplicació de pesos incrementa un $\sim 25\%$ els paràmetres, però redueix la variància del gradient i aporta corbes de convergència més suaus (vegeu Corbes de TensorBoard, Annex B).

En síntesi, l'arquitectura de **DQN** és més lleugera però requereix un mecanisme extern (target network) per evitar la divergència dels Q-valors, mentre que **PPO** incrementa la mida del model per separar explícitament actor i critic, obtenint així un aprenentatge generalment més estable en entorns visuals complexos. Aquests matisos expliquen les diferències de velocitat de convergència i de variabilitat observades al Capítol 4, i reforcen la importància de triar la geometria interna adequada quan es comparen algorismes de DRL.

APÈNDIX C

GRÀFIQUES DE TENSORBOARD

Aquest annex recull i examina les mètriques internes enregistrades durant l'entrenament dels agents DQN i PPO, obtingudes amb TensorBoard[14]. A diferència de les corbes agregades i els boxplots del Capítol 4, aquestes gràfiques ofereixen una visió detallada de la dinàmica d'aprenentatge.

En Stable-Baselines3, cada "rollout" correspon al conjunt de timesteps que l'agent executa fins a finalitzar un episodi. Les callbacks integrades (`Monitor`, `EvalCallback`, etc.) escriuen a TensorBoard mètriques com:

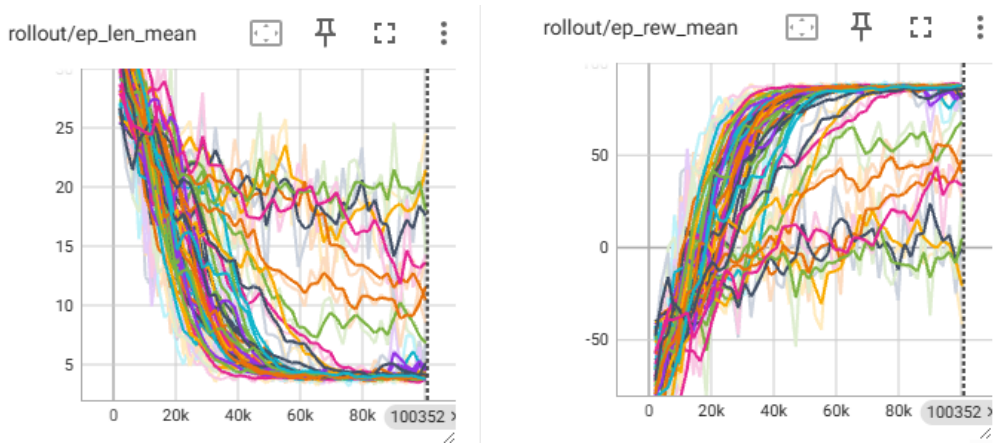
- **Longitud mitjana d'episodi** (*ep_len_mean*)
- **Recompensa mitjana per episodi** (*ep_rew_mean*)

Analitzar aquestes corbes permet validar la rapidesa de convergència, la robustesa entre seeds i detectar anomalies no visibles a les corbes agregades.

C.1 Mètriques de rollout a l'entorn *Basic*

La figura C.1 mostra les dues mètriques extretes de SB3 per les 50 execucions de PPO a l'entorn *Basic*

C.1.1 PPO en l'entorn *Basic*



(a) *ep_len_mean*: longitud mitjana d'episodi

(b) *ep_rew_mean*: recompensa mitjana per episodi

Figura C.1: Mètriques de rollout de PPO en l'entorn *Basic* ($n = 50$ seeds).

Interpretació:

Durant l'entrenament de PPO a *Basic*, observem dues fases diferenciades:

1. Fase d'exploració inicial (0–10 000 timesteps).

- *ep_len_mean* (Figura C.1a) es manté entre 30 i 25 passes: l'agent encara no ha après a evitar col·lisions i es queda bloquejat en alguna de les parets.
- *ep_rew_mean* (Figura C.1b) oscil·la entorn de zero, amb algunes caigudes lleus per penalitzacions de col·lisió. Aquest comportament reflecteix que la política inicial, gairebé aleatòria, sol destacar poc el reforç positiu immediat.

2. Fase de convergència ràpida (10 000–20 000 timesteps).

- Entre 10 000 i 20 000 timesteps, *ep_len_mean* decreix de manera quasi exponencial fins a 5 passes, indicant que l'agent aprèn ràpidament a eliminar ràpidament al objectiu.
- En paral·lel, *ep_rew_mean* ascendeix de forma contínua fins a 80–100 punts; aquesta corba suau confirma l'efectivitat del càlcul de l'avantatge i del mecanisme de *clipping* per reduir la variança de les actualitzacions.

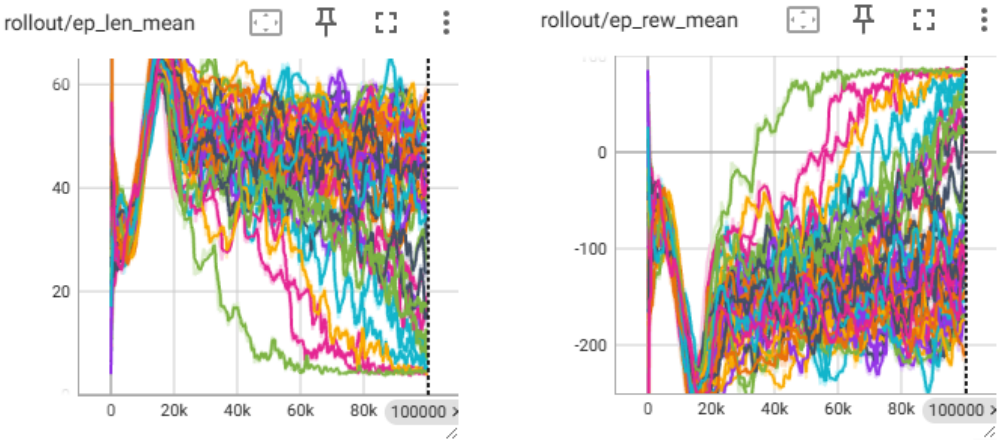
3. Estabilització i refinament (a partir de 20 000 timesteps).

- Un cop superats els 20 000 timesteps, ambdós indicadors es força estables: *ep_len_mean* oscil·la entorn de 3-5 passes i *ep_rew_mean* es manté en la franja alta (90–100 punts).
- La reducció de la dispersió entre seeds en ambdues mètriques reflecteix la forta coherència de les polítiques apreses, gràcies al buffer de trajectories curtes i efectives que PPO utilitza en mode *on-policy*.

En síntesi, el comportament de PPO a *Basic* confirma la robustesa de l'enfocament actor–critic: la corba de recompensa mitjana augmenta de manera monòtona i amb baixa variància, i l'episodi es curt de forma sostinguda, indicant una política capaç de detectar l'objectiu principal d'eliminar al únic enemic .

C.1.2 DQN en l'entorn *Basic*

La figura C.2 mostra les dues mètriques extretes de SB3 per les 50 execucions de DQN a l'entorn *Basic*



(a) *ep_len_mean*: longitud mitjana d'episodi

(b) *ep_rew_mean*: recompensa mitjana per episodi

Figura C.2: Mètriques de rollout de DQN en l'entorn *Basic* ($n = 50$ seeds).

Interpretació:

L'anàlisi de DQN a *Basic* mostra una fase inicial llarga i inestable, seguida d'una millora gradual i variable:

1. Retard per omplir el replay buffer (0–30 000 timesteps).

- *ep_len_mean* es manté baixa (40–80 passes) fins als 30 000 timesteps, ja que l'agent explora de manera aleatòria i només comença l'actualització consistent en omplir el buffer.

- *ep_rew_mean* es mou en territoris negatius (-200 a 0), indicant que el cost de col·lisions i morts supera temporalment qualsevol recompensa per eliminacions.

2. Creixement irregular (30 000–60 000 timesteps).

- Un cop disponible una quantitat suficient de transicions, *ep_len_mean* creix cap a 120–140 passes, però amb moltes fluctuacions: es reflecteix l'error de bootstrapping i la sobreestimació de valors Q típics de DQN clàssic.

- La recompensa mitjana puja cap a zero i creua valors positius al voltant dels 50 000–60 000 timesteps, però amb pics abruptes tant per bones com per males experiències.

3. Estabilització tardana i dispersió (a partir de 60 000 timesteps).

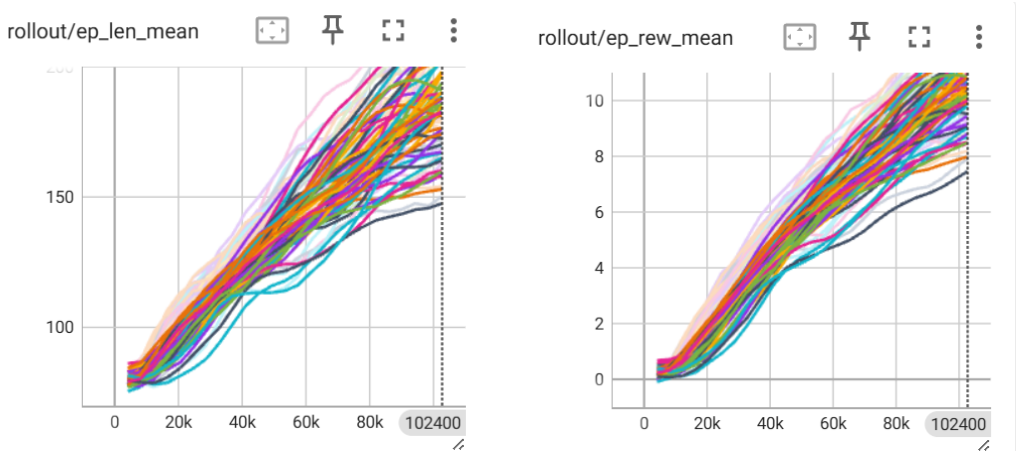
- Després dels 60 000 timesteps, l'agent mostra recompenses finals en la franja de 0–20 punts, però amb una amplitud interquartílica amplia (40–80 passes) en longitud i (–50 a +50 punts) en recompensa.
- Aquesta variabilitat es pot atribuir a la dependència de la política ϵ-greedy i a l'absència de mecanismes de regularització de variància pròpies d'actor–critic.

En conjunt, DQN a *Basic* experimenta un retard significatiu per iniciar aprenentatges estables i presenta una dispersió elevada a causa del bootstrapping i la correlació de mostres al replay buffer. Aquest patró contrasta fortament amb la suavitat i consistència observades en PPO.

C.2 Mètriques de rollout a l'entorn *Defend the Center*

C.2.1 PPO en l'entorn *Defend the Center*

La figura C.3 mostra les dues mètriques extretes de SB3 per les 50 execucions de PPO a l'entorn *Defend the Center*



(a) *ep_len_mean*: longitud mitjana d'episodi

(b) *ep_rew_mean*: recompensa mitjana per episodi

Figura C.3: Mètriques de rollout de PPO en l'entorn *Defend the Center* ($n = 50$ seeds).

Interpretació:

L'anàlisi de PPO en *Defend the Center* mostra un entrenament més gradual que en *Basic*, amb una clara separació de fases:

1. Adaptació a la dinàmica circular (0–20 000 timesteps).

- *ep_len_mean* inicia entorn de 70–90 passes, ja que l'agent ha de dominar girs constants i apuntar amb precisió.
- *ep_rew_mean* es manté baix (0–1 punts) mentre aprèn a alinear el dispar i evitar penalitzacions, indicant un període d'exploració controlada.

2. Increment lineal de rendiment (20 000–60 000 timesteps).

- Entre 20 000 i 60 000 timesteps, *ep_len_mean* creix de manera quasi lineal fins a 160–180 passes, reflectint una política que aprèn a mantenir la posició central sense morir.
- Paral·lelament, *ep_rew_mean* puja de forma contínua fins a 8–10 punts, gràcies a la millora progressiva de l'estratègia de dispar amb precisió.

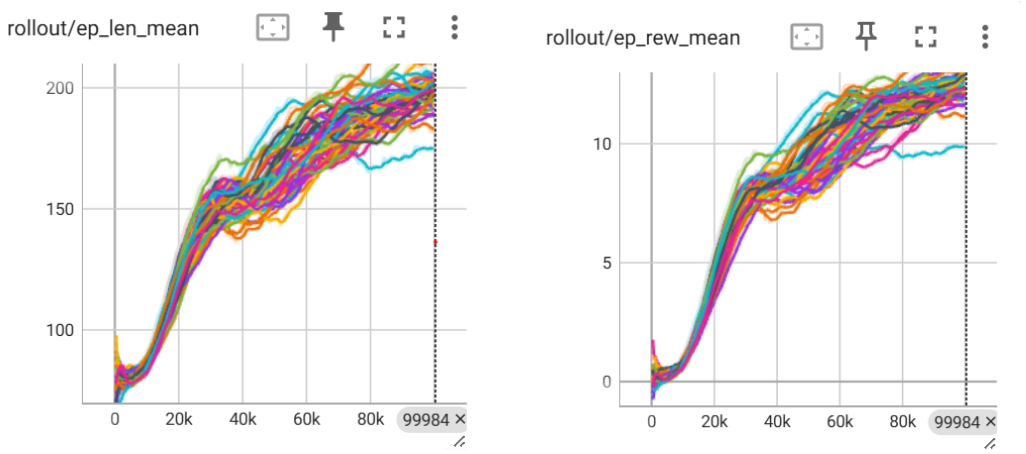
3. Refinament i estabilitat avançada (60 000–100 000 timesteps).

- A partir de 60 000 timesteps, *ep_len_mean* s'estabilitza en 190–210 passes, amb poca variabilitat entre seeds, la qual cosa evidencia la coherència de la política estocàstica i el control de variància pel *clipping*.
- *ep_rew_mean* assoleix la franja de 10–12 punts, amb una pendent lleugerament decreixent cap al final, suggerint que l'agent s'estabilitza i segurament amb entrenaments amb mes pases no milloraria més el rendiment.

En conclusió, PPO a *Defend the Center* mostra un entrenament menys explosiu que en *Basic*, però assoleix una política robusta i homogènia mitjançant actualitzacions suaus i un obtenció de recompenses graduals.

C.2.2 DQN en l'entorn *Defend the Center*

La figura C.4 mostra les dues mètriques extretes de SB3 per les 50 execucions de DQN a l'entorn *Defend the Center*



(a) *ep_len_mean*: longitud mitjana d'episodi

(b) *ep_rew_mean*: recompensa mitjana per episodi

Figura C.4: Mètriques de rollout de DQN en l'entorn *Defend the Center* ($n = 50$ seeds).

Interpretació:

DQN aprofita la naturalesa repetitiva de l'escenari per assolir convergència ràpida i amb poca dispersió:

1. Aprenentatge inicial ràpid (0–15 000 timesteps).

- Gràcies al *replay buffer*, *ep_len_mean* creix de 80 a 120 passes en només 10–15 000 timesteps, indicant una adquisició ràpida dels patrons estacionaris.
- *ep_rew_mean* travessa zero cap als 10 000 timesteps i ja arriba a 2–4 punts, mostrant que l'agent capitalitza ràpidament les recompenses positives d'eliminació.

2. Convergència asimptòtica (15 000–40 000 timesteps).

- Entre 15 000 i 40 000 timesteps, *ep_len_mean* assoleix 180–200 passes i es manté allà, amb mínimes oscil·lacions, degut al refredament de ϵ i la xarxa objectiu fixa.
- *ep_rew_mean* augmenta de 4 a 10–12 punts, amb una pendent ben definida, cosa que reflecteix l'estabilitat del Q-learning off-policy en entorns amb recompenses freqüents.

3. Fase de poliment final (40 000–100 000 timesteps).

- A partir de 40 000 timesteps, tant la longitud com la recompensa es mantenen en 180–200 passes i 12–13 punts, respectivament, amb dispersió interquartílica mínima.

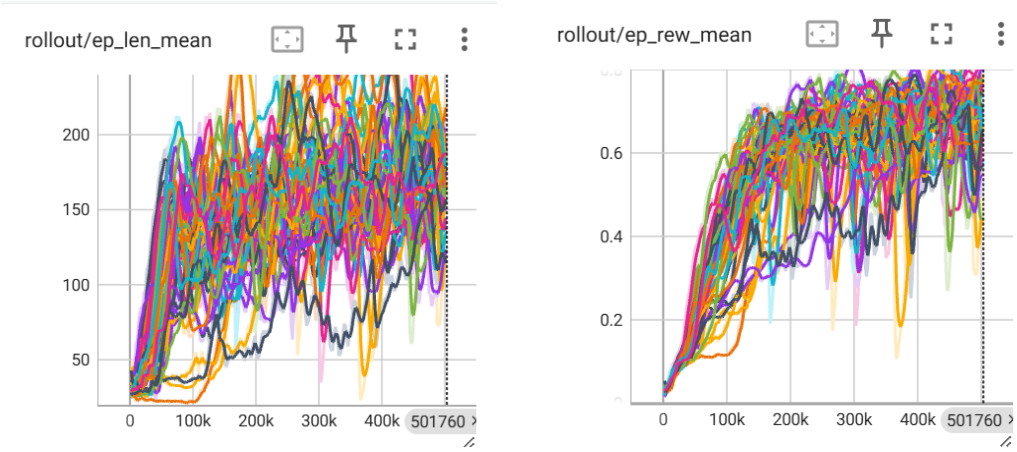
- Aquesta estabilitat il·lustra el benefici del manteniment d'un ϵ baix, que redueixen la variància de les estimacions Q.

En resum, DQN en *Defend the Center* convergeix més ràpid que PPO i amb menys variància, aprofitant la repetició d'estats i un espai d'acció reduït per obtenir polítiques consistents i eficients.

C.3 Mètriques de rollout a l'entorn *Deadly Corridor*

C.3.1 PPO en l'entorn *Deadly Corridor*

La figura C.5 mostra les dues mètriques extretes de SB3 per les 50 execucions de PPO a l'entorn *Deadly Corridor*



(a) *ep_len_mean*: longitud mitjana d'episodi

(b) *ep_rew_mean*: recompensa mitjana per episodi

Figura C.5: Mètriques de rollout de PPO en l'entorn *Deadly Corridor* ($n = 50$ seeds).

Interpretació:

L'entrenament de PPO a *Deadly Corridor* mostra tres fases clau:

1. Exploració i acostament inicial (0–100 000 timesteps).

- *ep_len_mean* parteix de 30–50 passes i creix lentament fins a 100–120. Aquesta fase reflecteix la complexitat de dominar moviments precisos i evitar múltiples morts consecutives en un passadís estret.

- *ep_rew_mean* es manté baixa (0–0.2) fins als 100 000 timesteps, indicant que l'agent necessita acumular experiència per veure recompenses positives constants (eliminacions i proximitat al botí).

2. Fase d'aprenentatge accelerat (100 000–300 000 timesteps).

- Entre 100 000 i 300 000 timesteps, *ep_len_mean* puja de 120 a 180 passes, amb una pendent pronunciada que demostra la capacitat de PPO per ajustar ràpidament la política mitjançant el gradient clipping.

- Simultàniament, *ep_rew_mean* s'eleva de 0.2 a 0.6–0.7, gràcies a l'ús de *Reward Shaping* que afavoreix actualitzacions estables.

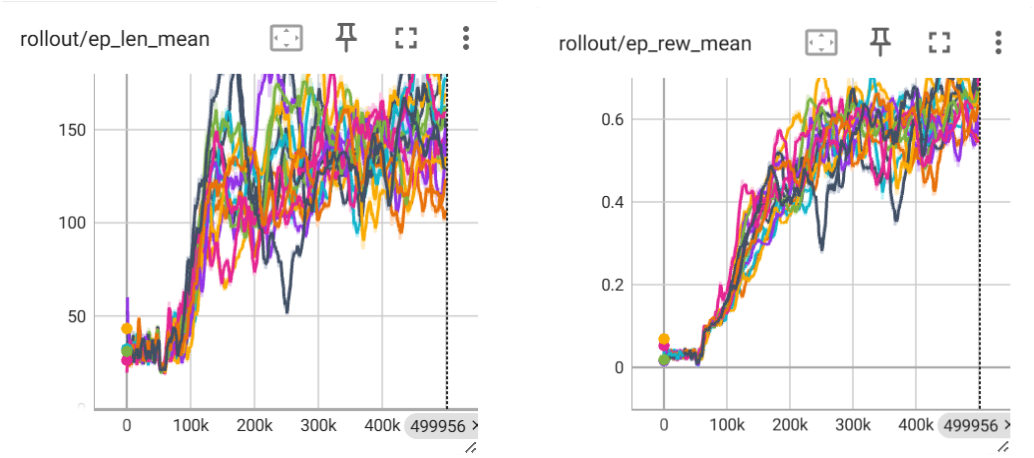
3. Refinament i estabilització (300 000–500 000 timesteps).

- A partir de 300 000 timesteps, *ep_len_mean* s'estabilitza en 180–220 passes, amb una dispersió moderada entre seeds que indica consistència.
- *ep_rew_mean* assoleix 0.7–0.85, amb petites oscil·lacions atribuïbles a la naturalesa estocàstica de la política i a penalties puntuals per morts.

En resum, PPO demostra capacitat per afrontar l'alta densitat de penalitzacions i recompenses esparses de *Deadly Corridor*, exhibint un aprenentatge inicial més lent però un ràpid creixement posterior i una convergència estable.

C.3.2 DQN en l'entorn *Deadly Corridor*

La figura C.6 mostra les dues mètriques extretes de SB3 per les 50 execucions de DQN a l'entorn *Deadly Corridor*



(a) *ep_len_mean*: longitud mitjana d'episodi

(b) *ep_rew_mean*: recompensa mitjana per episodi

Figura C.6: Mètriques de rollout de DQN en l'entorn *Deadly Corridor* ($n = 50$ seeds).

Interpretació:

L'evolució de DQN en *Deadly Corridor* reflecteix la dualitat típica de bootstrapping i replay buffer:

1. Ompliment de buffer i aprenentatge retardat (0–150 000 timesteps).

- *ep_len_mean* es manté en 30–60 passes fins als 150 000 timesteps, mostrant el retard inherent a l'ompliment del *replay buffer*.
- *ep_rew_mean* queda proper al zero, amb penalitzacions freqüents que superen recompenses esparses.

2. Creixement i variabilitat (150 000–300 000 timesteps).

- Un cop el buffer és operatiu, *ep_len_mean* puja ràpidament cap a 120–160 passes, però amb oscil·lacions significatives pel procés de selecció ϵ-greedy.
- *ep_rew_mean* augmenta de 0 a 0.4–0.6, però mostra pics tant de recompenses elevades com de retrocessos abruptes per penalitzacions de morts.

3. Convergència lenta i dispersió (300 000–500 000 timesteps).

- A partir de 300 000 timesteps, *ep_len_mean* se situa entre 140 i 180 passes, amb una dispersió interquartílica ampla.
- *ep_rew_mean* es fixa en 0.5–0.7, amb variabilitat deguda a la falta de mecanismos de control de variança i a possibles sobreestimacions del valor Q.

En conjunt, DQN acaba assolint rendiments competitius, però pateix una variabilitat inter-*seed* més gran i un procés de convergència més lent i irregular que PPO en aquest entorn crític.

En conjunt, aquestes mètriques de rollout permeten:

1. Corroborar com PPO demostra una convergència suau i consistent en entorns amb dinàmiques variades (*Basic* i *Deadly Corridor*), gràcies al control de variància de l'actor–critic.

2. Identificar com DQN aprofita la repetició i la simplicitat de *Defend the Center* per assolir rendiments ràpids i amb baixa dispersió, tot i presentar un retard inicial per l'ompliment del replay buffer.

3. Evidenciar la interacció entre l'estructura de l'entorn i la naturalesa *on-policy*/*off-policy* de cada algorisme: PPO excel·leix en escenaris amb recompenses esparses i penalitzacions severes, mentre que DQN troba el seu punt fort en entorns estacionaris amb recompenses freqüents.

4. Subratllar la importància de complementar les corbes agregades i els boxplots amb mètriques de rollout, ja que aquestes revelen fases d'exploració, pics de variabilitat i mecanismes interns de convergència que no serien evidents d'una altra manera.

Aquestes conclusions internes reforcen les observacions presentades al Capítol 4 i ofereixen una visió més granular de com i per què cada algorisme aprèn segons la naturalesa de l'entorn.

BIBLIOGRAFIA

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 2018. 2.1, 2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5, 2.1.6, 2.4.1, 3.6

[2] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, pp. 219–383, 2018. 2.1, 2.2, 2.2.1, 2.2.1, 2.3.1, 2.3.2, 2, 2.3.2, 2.3.3, 2.4.1, 2.4.2, 2.5, 3.1

[3] M. Olsson, S. Malm, and K. Witt, "Evaluating the effects of hyperparameter optimization in vizdoom," University of Skövde, Tech. Rep., 2022. 2.1.5, 2.2.1, 2.2.2, 2.4, 2.4.3, 2.4.4

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. 2.3, 2.3.2, 2.3.2, 3.1, c), 3.3

[5] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, no. 3-4, pp. 293–321, 1992. 2, 3.3.4

[6] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 2094–2100. 2.3.3

[7] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Jansen, and D. Silver, "Dueling network architectures for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, pp. 1995–2003. 2.3.3

[8] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," *International Conference on Machine Learning (ICML)*, 2017. 2.3.3

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. 2.4, 2.4.3, 2.4.4, 2.5, 3.3.3, 3.3.4

[10] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaskowski, "Vizdoom: A doom-based ai research platform for visual reinforcement learning," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016, pp. 1–8. 3.1, 3.7

[11] ViZDoom, “Vizdoom documentation,” https://vizdoom.farama.org/index.html, 2025, accessed: 2025-05-23. 3.1, 3.2

[12] A. Raffin, A. Hill, A. Gleave, A. Kanazawa, M. Ernestus, N. Dormann, J. Foerster, and D. Budden, “Stable-baselines3: Reliable reinforcement learning implementations,” GitHub repository, https://github.com/DLR-RM/stable-baselines3, 2021. 3.1, 3.3, 3.5, 3.7

[13] ViZDoom, “Default scenarios/environments,” https://vizdoom.farama.org/environments/default/, 2025, accessed: 2025-05-23. 3.1, 3.1.1

[14] M. Abadi *et al.*, “Tensorboard: Visualizing learning,” https://www.tensorflow.org/tensorboard, 2016. 3.2, 3.5, 3.7, C

[15] A. Y. Ng, D. Harada, and S. Russell, “Policy invariance under reward transformations: Theory and application to reward shaping,” in *ICML*, 1999, pp. 278–287. 3.6, 2.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf 3.7

[17] S.-B. maintainers, “Custom policy — stable baselines3 documentation,” https://stable-baselines3.readthedocs.io/en/master/guide/custom_policy.html, 2025, accessed: 2025-06-24. B.1