Treball Final de Màster

Màster en Enginyeria Informàtica

Àrea: Intel·ligència Artificial

Aprenentatge profund per reforç aplicat al control automàtic de la locomoció de robots bípedes simplificats en entorns simulats

Professor responsable de l'assignatura:

Carles Ventura Royo

Estudiant:

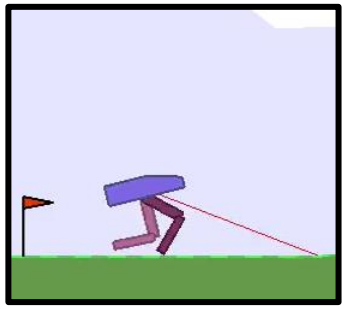
Rafael Jesús Castaño Ribes

Consultor:

Samir Kanaan Izquierdo

Gener de 2021

Definició del problema



- **Problema:**

  Control autònom de la locomoció bípeda en robots hominoides

- **Objectiu d'aquest TFM:**

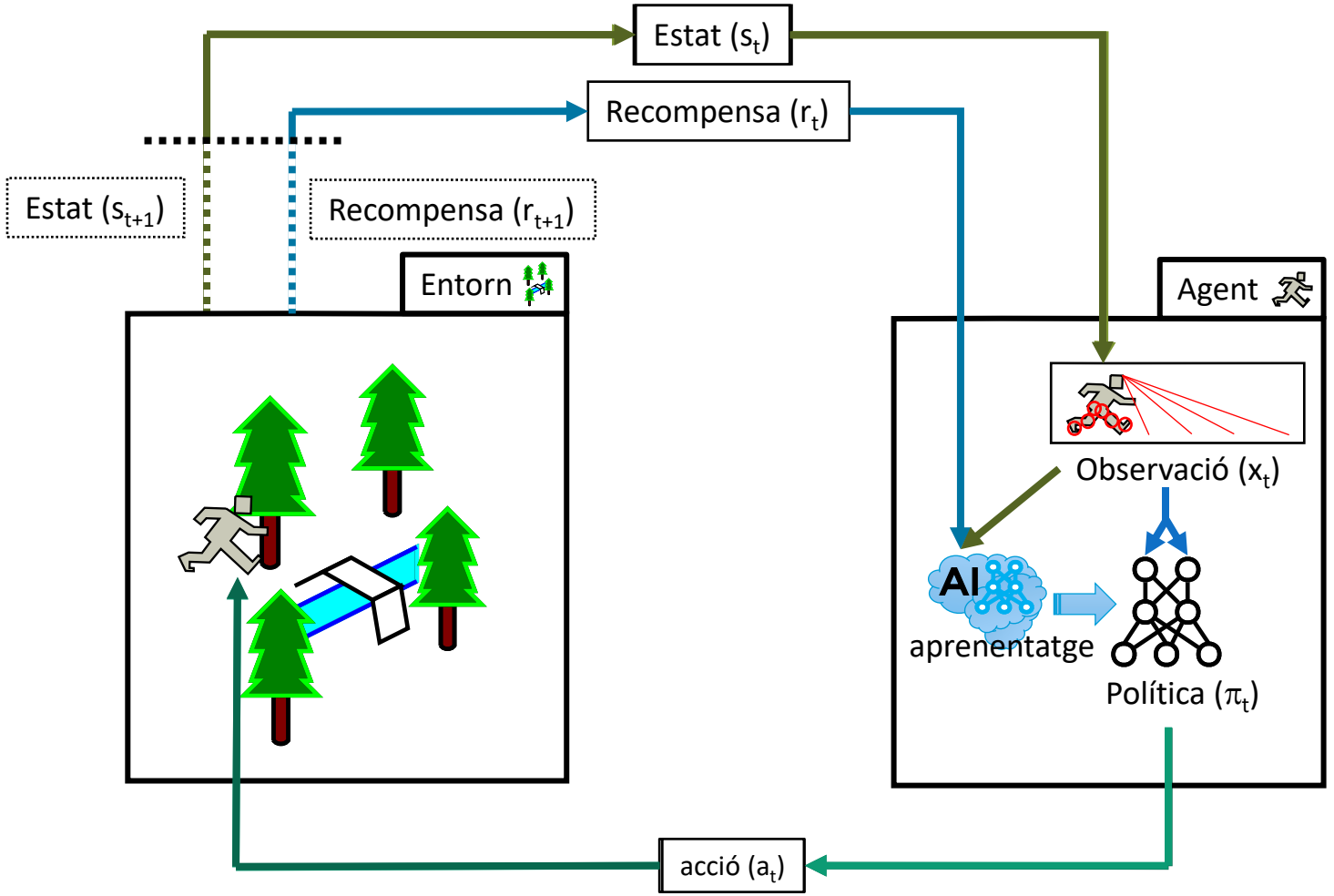
  Obtenir un agent intel·ligent per resoldre aquest problema en un entorn simplificat i simulat per programari, tot integrant diferents enfocaments actuals.

- **Disciplina en la qual s'encabeix:**

  Aprenentatge per Reforç (Profund) – *(Deep) Reinforcement Learning – (Deep) RL*

- **Rellevància:**
  - Hiperautomatització com a tendència tecnològica estratègica (Gartner 2020)
  - Integració de diferents metodologies
  - Apropar la disciplina al camp de la *citizen science*

Marc conceptual de l'Aprenentatge per Reforç



Objectiu del l'Aprenentatge per Reforç

- **Valor de retorn** descomptat acumulat a llarg termini ($\boldsymbol{R_t}$)

$$\boldsymbol{R_t} = r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k}$$

on $\boldsymbol{\gamma}$ rep el nom de factor de descompte i es defineix com:

$$\{ \gamma \in \mathbb{R} \mid 0 \leq \gamma \leq 1 \}$$

- **Política òptima π^***

$$\pi^* = \arg\max_{\pi} \mathbb{E}[R_t|\pi]$$

- L'objectiu de l'RL és **trobar (o aproximar-se a)** $\boldsymbol{\pi^*}$

Reformulacions de l'objectiu de l'RL

Reformulació de l'R_t	Reformulació de l'objectiu de l'RL
• Funció **valor-estat (*V*)**: $v_\pi(s) = \mathbb{E}_\pi[R_t \mid s_t = s]$ $V_\pi(s) \xrightarrow{aproxima} v_\pi(s)$	• En funció del **valor-estat (*V-Learning*)**: $\pi^* = \arg\max_\pi v_\pi(s)$
• Funció **valor-acció (*Q*)**: $q_\pi(s,a) = \mathbb{E}_\pi[R_t \mid s_t = s, a_t = a]$ $Q_\pi(s) \xrightarrow{aproxima} q_\pi(s,a)$	• En funció del **valor-acció (*Q-Learning*)**: $a^*(s) = \arg\max_a q_*(s,a)$
• Funció **avantatge (*A*)**: $A_\pi(s_t,a) = Q_\pi(s_t,a) - V_\pi(s_t)$	• En funció de l'**avantatge**: $A_\pi(s,a) = r_{t+1,a} + \gamma V_\pi(s_{t+1}) - V_\pi(s_t)$ *Reformulació en base a la recompensa i al valor-estat (V)*

Simplificacions del problema de l'RL

- **Discretització del temps**
  - Cada instant de temps rep el nom de ***time step***.
  - Si la durada és limitada, una seqüència completa de *time steps* rep el nom d'**episodi**.

- **Propietat de Markov**
  - Es compleix quan qualsevol estat futur (s_{t+1}) depèn **únicament** del seu estat immediatament anterior (s_t) i de l'acció (a_t) que s'hi hagi executat.
  - El problema es pot formalitzar com un ***Markov Decision Process (MDP)*** *i permet aplicar-hi els mètodes de la* ***Programació Dinàmica (DP)*** *i la* ***Diferència Temporal (TD)*** *per intentar resoldre'l.*

Principals enfocaments en l'RL

- Programació Dinàmica (PD)
- Monte Carlo (MC)
- Diferència Temporal (TD)
- Optimització directa de la política
- Actor-Crític (AC)
- Mètodes basats en models

En aquest TFM s'ha:

Implementat un algorisme TD (*Continuous Q-Learning with NAF*)

Provat un algorisme AC (*Soft Actor-Critic*)

TD-Learning

- S'utilitza en els problemes de tipus MDP
- El mètode es basa en què, idealment:

$$\mathbb{E}_\pi[r + \gamma v_\pi(s')] - v_\pi(s) = 0$$

- Aleshores, durant k iteracions (*bootstrapping*):

$$V_{\pi,k}(s) \leftarrow V_{\pi,k-1}(s) + \alpha \underbrace{\left[\left[r_k + \gamma V_{\pi,k-1}(s')\right] - V_{\pi,k-1}(s)\right]}_{TD}$$

- Existeixen múltiples variants, segons la funció valor utilitzin, com ara *SARSA* i *Q-Learning*.

Deep Reinforcement Learning

- Utilitza mètodes de ***Deep Learning***, com ara **xarxes neuronals**, per aproximar alguna/es de les funcions necessàries per a l'RL, com són:
  - Les **funcions valor** (V, Q, A)
  - Les **polítiques** (ja siguin deterministes o estocàstiques)
  - Les funcions de transició i/o de recompensa (per **modelar l'entorn**)

- Pren popularitat i rellevància a partir del disseny i resultats de l'algorisme ***Deep Q-Learning (DQN)*** (Mnih, et al., 2013 i 2015)

Deep Q-Learning

- ***Integració del Deep Learning en l'RL:***
  - **Q** és una **xarxa neuronal**, amb un output per al valor Q de cada acció possible
  - Una segona xarxa (***xarxa target* o Q⁻**), actualitzada a partir de Q, calcula els *targets*
  - L'objectiu d'aprenentatge per a la xarxa neuronal Q és **minimitzar el *Loss (L)***:

$$L = \left(\underbrace{\left(r'_j + \gamma \max_{a'} Q_*^-(s'_j, a'_j; \theta^-) \right)}_{target} - Q_*(s_j, a_j; \theta) \right)^2$$

equivalent al TD

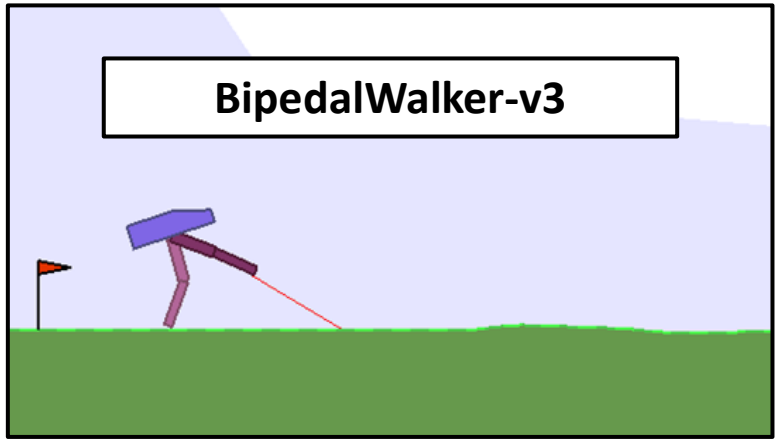
  - Cada cert nombre d'iteracions, Q^- s'actualitza mitjançant: $\theta^- = \tau \cdot \theta + (1 - \tau) \cdot \theta^-$

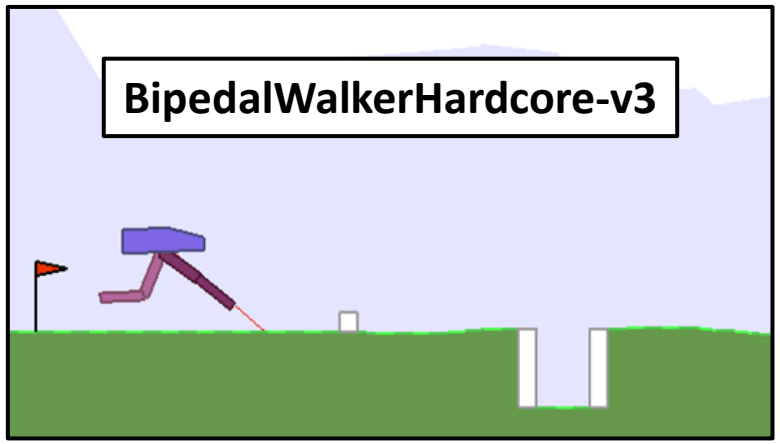
- ***Buffer replay:***
  - Per al l'aprenentatge s'agafa un conjunt (*batch*) de *n* experiències prèvies a l'atzar per disminuir el biaix de les correlacions temporals
- **Limitació:** només és vàlid en espais d'accions discrets i finits

La plataforma OpenAI Gym

https://gym.openai.com/

Entorns d'interès

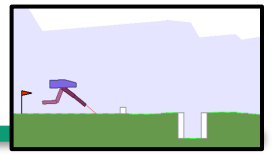




API

```
import gym

N_EPISODES = 10

policy = MyPolicy() #User defined

env = gym.make('BipedalWalker-v3')

for i_episode in range(N_EPISODES):

  observation = env.reset()
  done = False
  env.render()

  while not done:
    observation, reward, done, info =
      env.step(policy.getAction(observation))

    env.render()

env.close()
```

Propietats dels entorns *BipedalWalker*



Espai d'observacions (24 mesures):

- Angle d'inclinació i velocitats angular del cos
- Velocitats horitzontal i vertical del cos
- Angle i velocitat angular de malucs i genolls
- Contacte de cada peu amb el terra
- 10 mesures LIDAR[1] (sobre l'entorn)

Espai d'accions (4 valors):

- Força angular a aplicar a cada maluc i genoll

Tots dos espais són continus

Recompensa (1 valor):

- Valor **proporcional al desplaçament**
- Penalitzat per:
  - La torsió efectuada en els efectors
  - No mantenir el cap recte
- **Si** el robot **cau a terra** el valor és **-100**

Durada màxima dels episodis:

- L'episodi finalitza si:
  - S'aconsegueixen **300 punts** en:
    - 1600 *time steps* per a BipedalWalker-v3
    - 2000 *time steps* per a BipedalWalkerHardcore-v3
  - El robot **cau a terra**

[1] LIDAR = Sensor de tipus *Light Detection and Ranging*

Solució proposada

Plantejament algorísmic

Algorisme base:

Continuous Q-Learning with NAF (Gu *et al.*, 2016)

- Accepta espais continus d'accions
- Encara no implementat a TF-Agents
- Senzill d'implementar i modificar
- Eficient (entrena política i funcions valor a la vegada)

Possible integració amb altres paradigmes:

- *Safe RL (García & Shafie, 2020)*, com a política d'exploració basada en la predicció del risc de caiguda.
- Acceleració de l'aprenentatge mitjançant models, seguint *Dina-Q (Sutton, 1990).*

Plantejament tecnològic

- Biblioteca per a RL:

  TF-Agents

- Entorn per a la simulació

  OpenAI Gym

  (integrat en TF-Agents)

- Llenguatge:

  Python

- Monitoratge dels resultats:

  TensorBoard

Continuous Q-Learning with NAF

Algorithm 1 Continuous Q-Learning with NAF

Randomly initialize normalized Q network $Q(\boldsymbol{x}, \boldsymbol{u}|\theta^Q)$.
Initialize target network Q' with weight $\theta^{Q'} \leftarrow \theta^Q$.
Initialize replay buffer $R \leftarrow \emptyset$.
for episode=1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $\boldsymbol{x}_1 \sim p(\boldsymbol{x}_1)$
  for t=1, T **do**
    Select action $\boldsymbol{u}_t = \mu(\boldsymbol{x}_t|\theta^\mu) + \mathcal{N}_t$
    Execute $\boldsymbol{u}_t$ and observe r_t and $\boldsymbol{x}_{t+1}$
    Store transition $(\boldsymbol{x}_t, \boldsymbol{u}_t, r_t, \boldsymbol{x}_{t+1})$ in R
    for iteration=1, I **do**
      Sample a random minibatch of m transitions from R
      Set $y_i = r_i + \gamma V'(\boldsymbol{x}_{i+1}|\theta^{Q'})$
      Update θ^Q by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(\boldsymbol{x}_i, \boldsymbol{u}_i|\theta^Q))^2$
      Update the target network: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
    end for
  end for
end for

Extret de Gu *et al.* 2016

- En aquest algorisme:

$$Q(x, u|\theta^Q) = A(x, u|\theta^A) + V(x|\theta^V)$$

- L'avantatge (A) es defineix com:

$$A(x, u|\theta^A) = -\frac{1}{2}\left(u - \mu(x|\theta^\mu)\right)^T P(x|\theta^P)\left(u - \mu(x|\theta^\mu)\right)$$

- On P és definida positiva[1] i s'obté per:

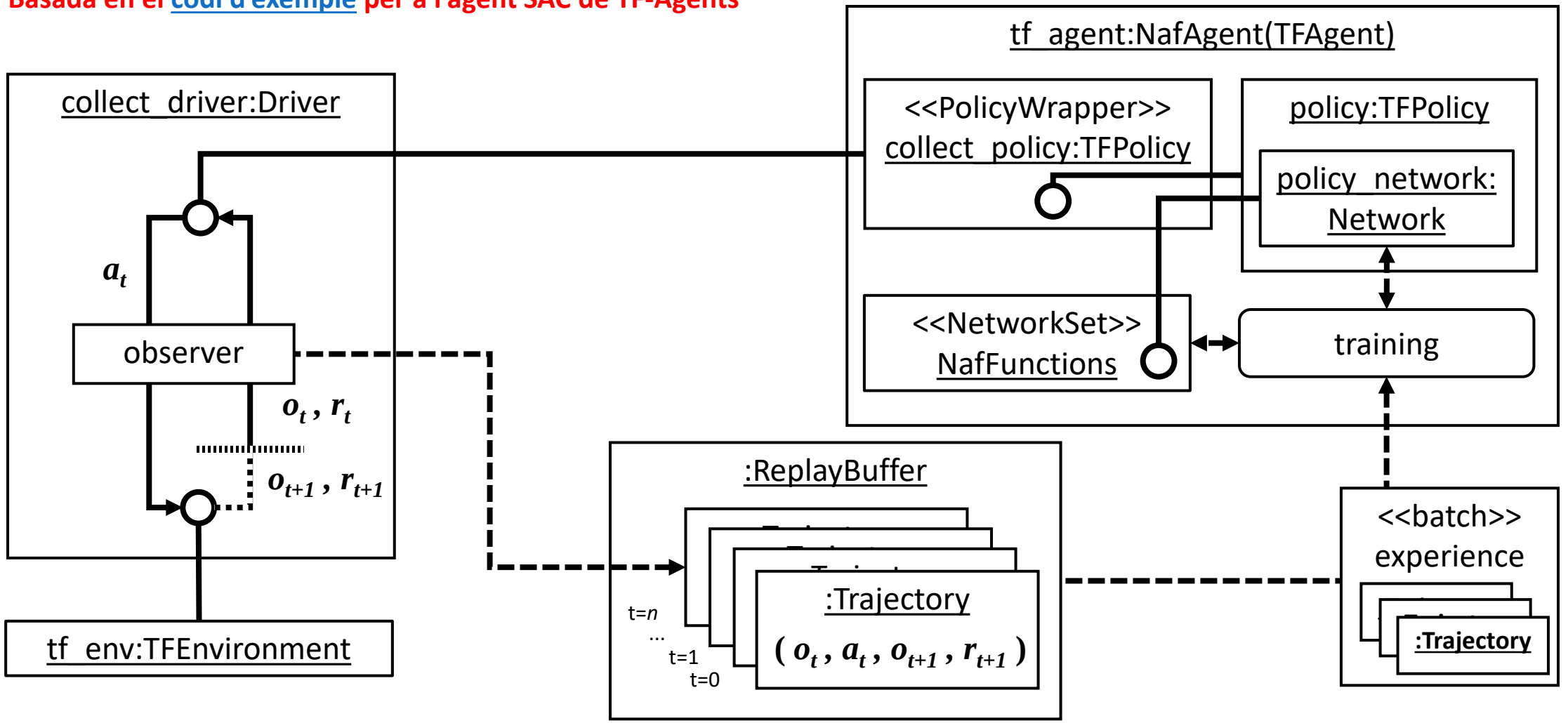
$$P(x|\theta^P) = L(x|\theta^P)L(x|\theta^P)^T$$

- On *L*:
  - És una matriu triangular inferior
  - Els seus elements són proporcionats per una xarxa neuronal amb paràmetres θ^P
  - Els elements de la diagonal fan d'exponent del nombre *e* (forçosament $\mathbb{R}^+$).

[1] P *és definida positiva* $\Leftrightarrow \boldsymbol{w}^T P \boldsymbol{w} \in \mathbb{R}^+, \forall \boldsymbol{w} \in (\mathbb{R}^{\boldsymbol{n}} - \{\boldsymbol{0}\})$

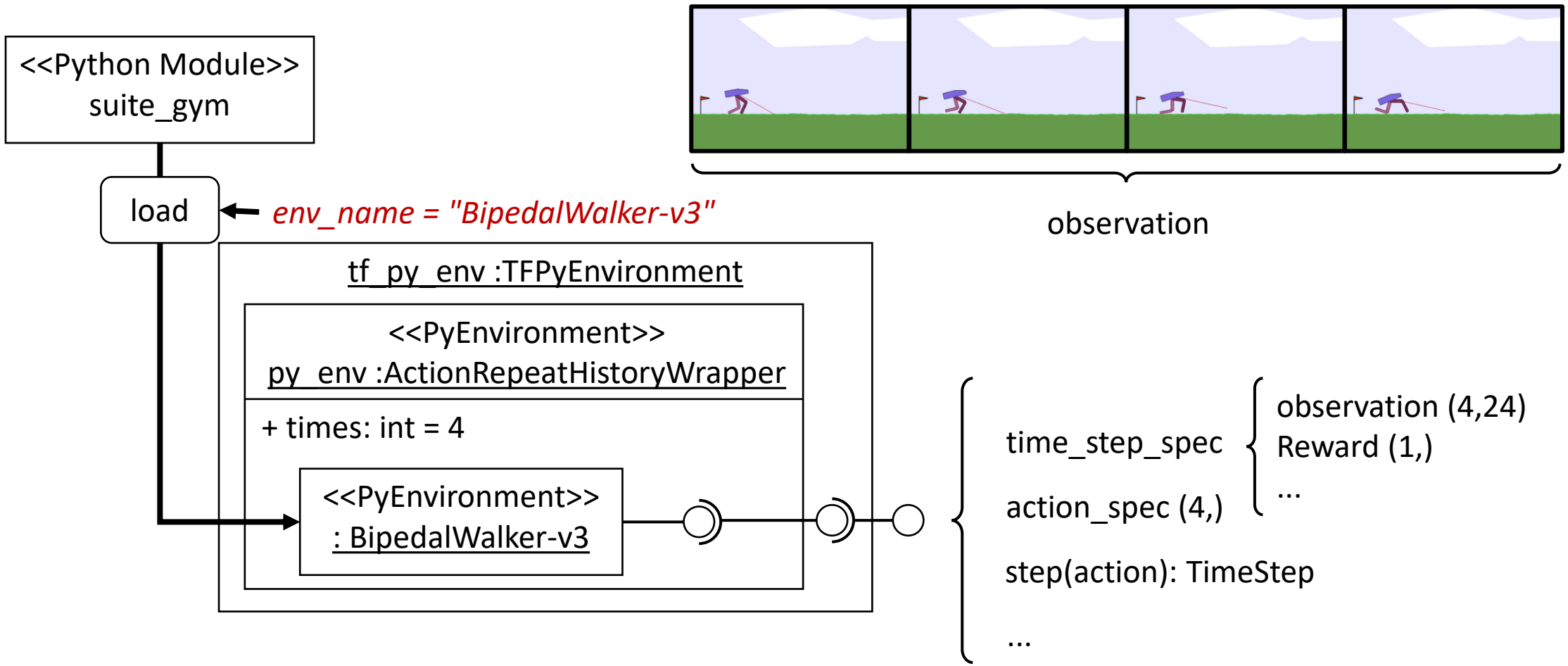
Arquitectura de la solució

Basada en el codi d'exemple per a l'agent SAC de TF-Agents

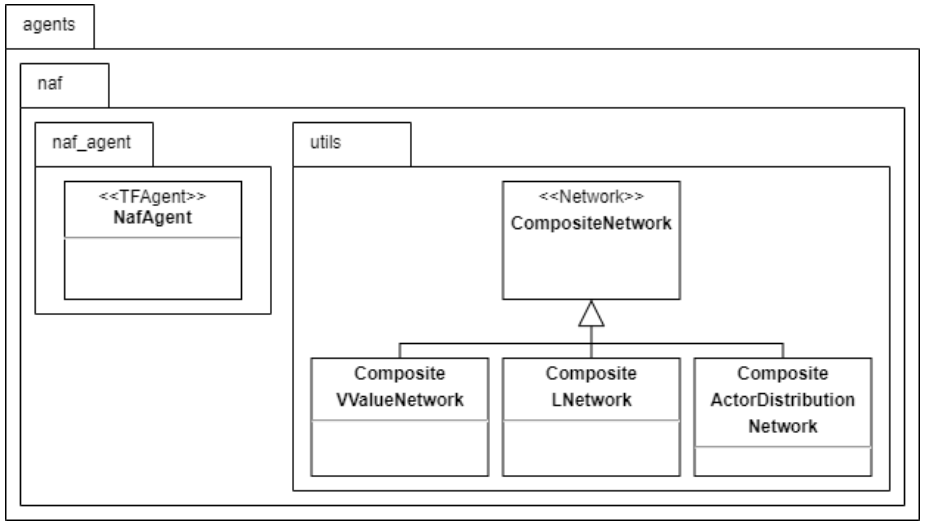


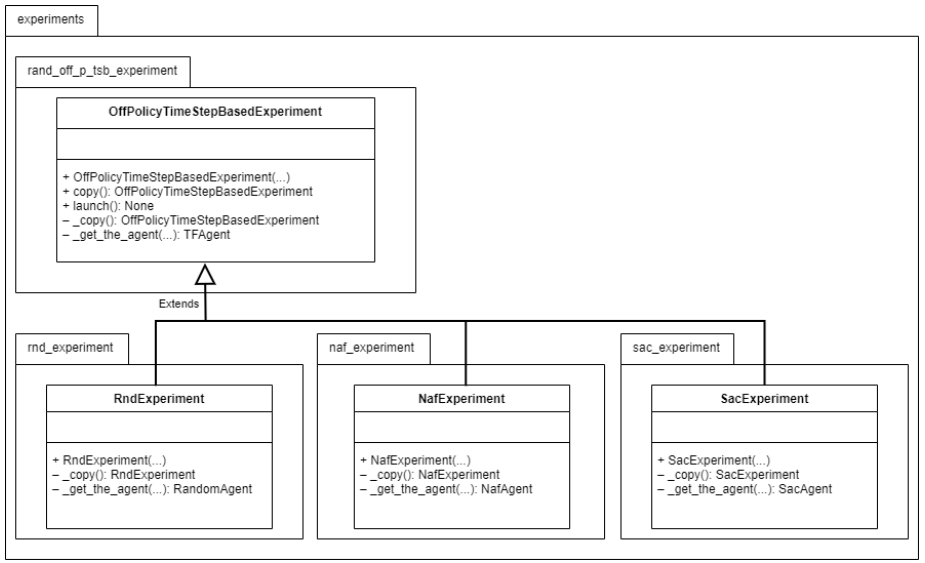
Component *ActionRepeatHistoryWrapper*

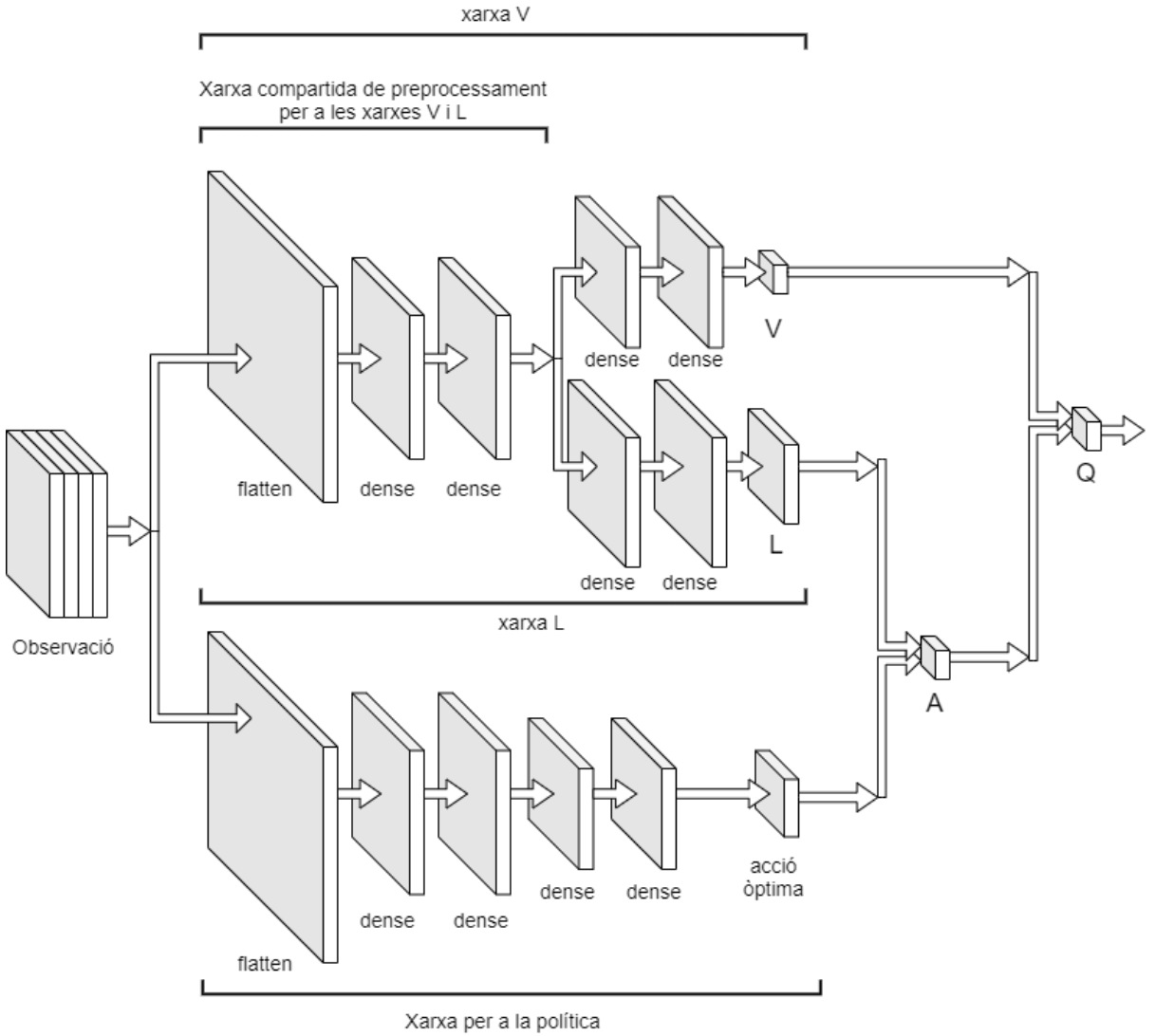
Basat en Mnih *et al.* 2015



Producte obtingut







Resultats

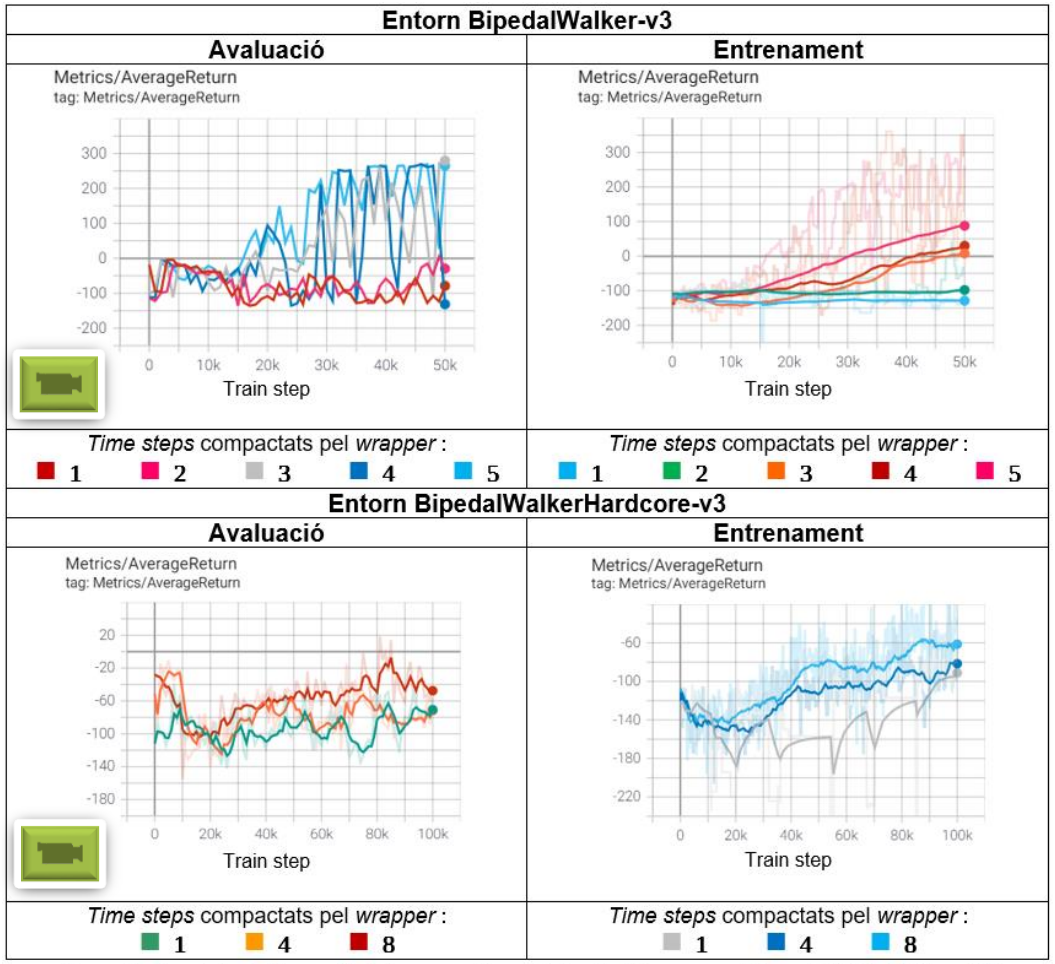


Figura 26. Resultats de l'execució del wrapper RepeatActionEnvWrapper amb l'agent SAC de TF-Agents[27]

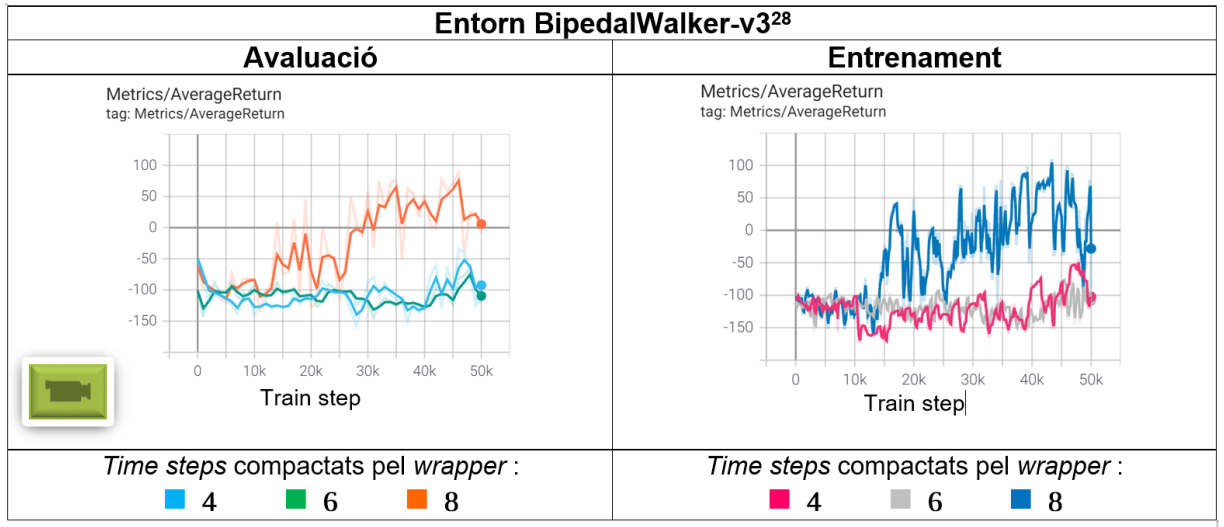


Figura 27. Resultats de l'execució del wrapper RepeatActionEnvWrapper amb l'agent NAF

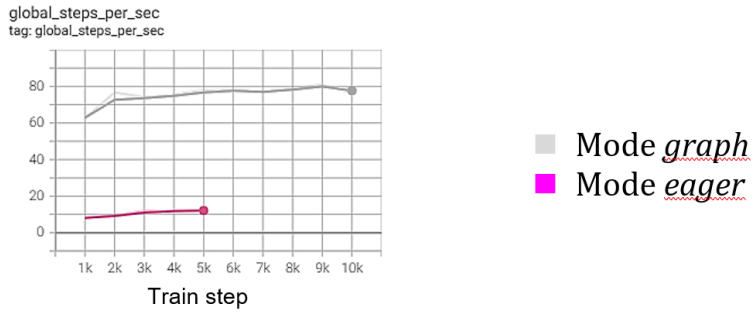


Figura 28. Velocitats d'execució de l'agent NAF en mode graph i mode eager per al mateix problema i configuració

Conclusions

- El *Deep RL* és una disciplina actual i complexa, amb una gran diversitat d'algorismes.
- Els espais d'accions continus incrementen la complexitat de la solució de *Deep RL*.
- TF-Agents és molt potent, però manca encara d'elements (compartició de capes entre xarxes).
- El *wrapper* per compactar diferents *time steps* com un de sol ha resultat una bona estratègia.
- El *Deep RL* té un elevat nombre de paràmetres i els que funcionen en un tipus d'agent poden no ser adequats en altres.
- El mode de grafs de la biblioteca TensorFlow és computacionalment molt eficient.
- NAF i SAC són molt eficients, amb resultats en pocs minuts (en general).
- OpenAI Gym ha resultat una plataforma adient com a base per al desenvolupament del TFM.
- L'objectiu general del projecte s'ha pogut assolir en la seva majoria, malgrat no ha estat possible, per haver esgotat el temps, integrar diferents enfocaments ni escalar la solució a un entorn 3D.

Línies de treball futur

- Acceleració de l'aprenentatge mitjançant models apresos a partir de l'experiència (Gu *et al.* 2016)
- Aproximació i ús d'una funció de predicció del risc de caiguda en la política d'exploració (com a wrapper) (García & Shafie, 2020).
- Adequar les classes CompositeNeuronalNetwork i derivades a la filosofia de la biblioteca TF-Agents.
- Posar el codi desenvolupat a disposició de l'equip de TF-Agents