Creació del Joc Snake i Implementació d'una IA Bàsica.

Víctor Valero Carrasco

Resum—Aquest projecte presenta la implementació del joc Snake utilitzant Pygame i la integració d'una IA bàsica basada en Reinforcement Learning (RL) mitjançant Pytorch. Primerament, es va crear el joc Snake per a que el jugui una persona humana, amb el seu corresponent testing que va aconseguir un 99% de coverage. Posteriorment, es va desenvolupar un agent que aprèn a jugar al joc Snake utilitzant l'algorisme de Deep Q-Learning, el qual aprofita les xarxes neuronals profundes per avauar les accions en diferents estats del joc. Durant l'entrenament, l'agent utilitza una política ε-greedy per exploar i explotar estratègies, millorant progressivament el seu rendiment. Els resultats mostren una millora en les puntuacions mitjanes i màximes al llarg de 10.000 episodis d'entrenament. Tot i això, l'agent té molt marge de millora i no està completament desenvolupat, fent que l'agent no reconegui bé les accions o faci el mateix moviment durant l'avaluació. Aquestes limitacions indiquen la necessitat d'ajustar els paràmetres d'entrenament i de continuar desenvolupant l'avaluador per garantir una millor adaptació i rendiment de l'agent.

Paraules clau—Python, Aprenentatge per Reforçament, Pygame, Pytorch ,Deep Q-Learning, Intel·ligència Artificial, Xarxes Neuronals, Snake, ε-greedy, Testing de Software.

Abstract— This project presents the implementation of the Snake game using Pygame and the integration of a basic AI based on Reinforcement Learning (RL) using PyTorch. Firstly, the Snake game was created for a human player, with its corresponding testing achieving 99% coverage. Subsequently, an agent was developed to learn to play the Snake game using the Deep Q-Learning algorithm, which leverages deep neural networks to evaluate actions in different game states. During training, the agent uses an ε-greedy policy to explore and exploit strategies, progressively improving its performance. The results show an improvement in average and maximum scores over 10,000 training episodes. However, the agent has a lot of room for improvement and is not fully developed, as it struggles to recognize actions correctly or performs the same movement during evaluation. These limitations indicate the need to adjust training parameters and continue developing the evaluator to ensure better adaptation and performance of the agent.

Index Terms— Python, Reinforcement Learning, Pygame, PyTorch, Deep Q-Learning, Artificial Intelligence, Neural Networks, Snake, ε-greedy, Software Testing.

1 Introducció - Context del treball

AQUEST projecte sorgeix de la meva motivació per aprendre a implementar un videojoc des de zero i adquirir coneixements per a l'execució eficient d'algoritmes d'intel·ligència artificial. La selecció del joc Snake com a base per al desenvolupament d'aquest projecte no ha estat casual. Aquesta elecció es fonamenta en diversos factors que van influir en la presa de decisió.

En primer lloc, la popularitat del joc Snake el converteix en una elecció idònia, ja que existeix una àmplia documentació i recursos disponibles per a la seva implementació. Aquesta abundància d'informació facilita l'aprenentatge i la resolució de problemes que puguin sorgir durant el desenvolupament..
D'altra banda, la simplicitat en la implementació del joc Snake no ha de ser confosa amb una manca de complexitat. Tot i la seva estructura aparentment senzilla, aquest joc ofereix elements distintius que proporcionen una base sòlida per adquirir les nocions bàsiques del desenvolupament de videojocs. Aquesta combinació de simplicitat i complexitat permetrà explorar i comprendre els fonaments essencials de la creació de jocs.

A més dels objectius centrats en la implementació del joc, aquest projecte també es planteja explorar altres conceptes rellevants en el món dels videojocs i la tecnologia. Amb l'auge dels "Serious Games" i la "Gamification", aquest projecte busca entendre com els videojocs poden transcendir en l'àmbit de l'entreteniment i ser utilitzats amb finalitats educatives, de formació o resolució de problemes tant autònoms com en grups de treball. Aquesta exploració pot proporcionar una perspectiva més àmplia sobre el paper dels videojocs en la societat actual i les seves aplicacions diverses més enllà de la recreació.
Finalment, l'aplicació de tècniques de "Reinforcement Learning" en el desenvolupament d'una Intel·ligència Artificial per al joc busca portar el projecte a un nivell superior. Això implica la creació d'una entitat virtual capaç

- *E-mail de contacte: victorvalerocarrasco@gmail.com*
- *Menció realitzada: Enginyeria del Software*
- *Treball tutoritzat per: Coen Antens (Àrea de Ciències de la Computació i Intel·ligència Artificial)*
- *Curs 2023-24.*

d'entendre i millorar les seves accions mitjançant la interacció amb l'entorn del joc. Aquesta dimensió d'intel·ligència artificial proporcionarà una experiència més dinàmica i desafiadora i, per a això, em serviré de la llibreria PyTorch per implementar algorismes d'IA i Reinforcement Learning.

2 Objectius

Amb l'objectiu de concloure de manera satisfactòria el projecte, m'he proposat assolir els següents objectius específics:

- **Objectiu 1:** Formació en les llibreries Pygame i Pytorch
- **Objectiu 2:** Implementació del joc Snake i jugabilitat humana.
- **Objectiu 3:** Realització de tests unitaris per al joc base.
- **Objectiu 4:** Distribució eficient de classes per a la implementació de la IA.
- **Objectiu 5:** Implementació de la IA de manera acurada.
- **Objectiu 6:** Estudi i implementació de Reinforcement Learning.

3 Estat de l' art

En aquest apartat el focus estarà en fer una petita introducció a la història del Reinforcement Learning (RL), introduir els principals mètodes d'aquest i, finalment, parlar sobre projectes coneguts que fan servir Reinforcement Learning. Abans d'entrar a explicar això, faré una petita explicació sobre què és el Reinforcement Learning.

El Reinforcement Learning o RL és una àrea de l'aprenentatge automàtic on un agent aprèn a prendre decisions mitjançant la interacció amb un entorn. L'objectiu de l'agent és maximitzar una recompensa acumulada al llarg del temps, ajustant les seves accions basant-se en les conseqüències observades.

3.1 Història del Reinforcement Learning.

El Reinforcement Learning té les seves arrels en la teoria del condicionament operant de la psicologia, introduïda per B.F. Skinner a mitjans del segle XX. El camp es va desenvolupar significativament amb l'aportació de Richard Sutton i Andrew Barto, que van publicar el llibre "Reinforcement Learning: An Introduction" l'any 1998. En aquest llibre es van establir les bases teòriques i pràctiques del camp, incloent els algorismes clàssics com el Q-Learning i les tècniques de control de Monte Carlo.

Amb l'avenç de la capacitat computacional i les xarxes neuronals profundes, el camp del Reinforcement Learning ha experimentat un renaixement en les últimes dècades. Algorismes com DQN, Actor-Critic i Proximal Policy Optimization han estat desenvolupats per abordar problemes més complexos.

3.2 Mètodes de Reinforcement Learning.

3.2.1 Q-Learning

El Q-Learning és un algorisme senzill, com s'ha comentat abans era dels primers, però potent en l'àmbit del Reinforcement Learning. Aquest algorisme es basa en actualitzar els valors de Q, que representen la qualitat d'una acció en un estat particular, per tal de trobar la millor política d'acció. La seva formulació es defineix per l'equació de Bellman, i les actualitzacions es realitzen iterativament a mesura que l'agent interactua amb l'entorn.

3.2.2 Deep Q-Networks (DQN)

Deep Q-Networks (DQN) és una extensió del Q-Learning que utilitza xarxes neuronals profundes per aproximar els valors Q. Va ser introduït per Google DeepMind, i va ser un avenç significatiu perquè permet que els agents aprenguin polítiques d'acció a partir d'observacions d'alta dimensionalitat, com ara píxels d'imatges en jocs. Aquesta tècnica és molt utilitzada en reptes de RL en el camp dels videojocs.

3.2.3 Actor-Crític.

Actor-Crític és una família d'algorismes que combina aspectes com les estratègies basades en valors, com Q-Learning, i les basades en polítiques. En aquesta metodologia, l'actor aprèn la política (quina acció prendre en cada estat), mentre que el crític aprèn el valor (quant de bona és una acció en un estat). Aquesta separació permet una major flexibilitat i estabilitat durant l'entrenament, ja que cada component es pot optimitzar de manera més especialitzada.

3.3 Aplicacions actuals conegudes

Actualment hi ha moltes aplicacions del Reinforcement Learning en el camp dels videojocs, però una que destaca molt entre les altres són els bots que trobem a les pàgines d'escacs. Un dels exemples més famosos és AlphaZero, desenvolupat per Google DeepMind. AlphaZero va utilitzar una variant del RL anomenada Monte Carlo Tree Search (MCTS) combinada amb xarxes neuronals profundes per aprendre a jugar a escacs (entre altres jocs) des de zero, simplement jugant contra si mateix i millorant a cada partida.

Aquest enfocament va permetre a AlphaZero superar programes d'escacs tradicionals com Stockfish, que es basen principalment en força bruta i l'exploració exhaustiva de possibles moviments. L'èxit d'AlphaZero, a part de demostrar la potència del Reinforcement Learning en combinació amb l'aprenentatge profund, va obrir nous horitzons a més aplicacions de RL en videojocs i fins i tot més enllà d'aquests.

4 Joc Snake

En aquesta secció explicaré com funciona el joc Snake, les regles del joc i com apliquem l'algorisme de Reinforcement Learning a aquest.

4.1 Funcionament del joc Snake

En el joc Snake, el jugador (o agent en el cas de Reinforcement Learning) controla una serp que es desplaça a una velocitat constant dins d'un pla delimitat, recollint aliments i evitant col·lisionar amb les parets que envolten l'escenari de joc o amb la seva pròpia cua.

4.2 Regles del Joc

Les principals regles són les següents:

- **Inici del Joc:** El joc comença amb la serp del tamany de 4 pixels en una posició prefixada. A més, es genera la primera poma de forma aleatòria.
- **Moviment:** La serp només es pot moure en 4 direccions: amunt, abaix, dreta i esquerra. Aquest moviment sol estar associat a les fletxes del teclat o, per altra banda, al moviment conegut com a WASD, que és amb les tecles: W, A, S i D.
- **Puntuació:** Cada vegada que el cap de la serp col·lisiona amb una poma, sumarà un punt i s'afegirà un píxel a la cua.
- **Penalització:** Si el cap de la serp col·lisiona amb els extrems del escenari o amb la seva pròpia cua perdrà i haurà de començar de nou.
- **Objectiu:** El objectiu final és menjar totes les pomes possibles sense perdre.

4.3 Reinforcement Learning aplicat a Snake

En el desenvolupament d'aquest projecte, s'ha investigat l'aplicació de tècniques de reforç per entrenar l'agent de la serp. L'aprenentatge per reforç és una branca de l'aprenentatge automàtic que es centra en com un agent pot aprendre decisions òptimes mitjançant la interacció amb un entorn. Es basa principalment en el concepte de recompenses i penalitzacions per guiar el comportament cap a l'objectiu desitjat.

Si entrem més en detall en el Reinforcement Learning, es basa en un procés de presa de decisions on l'agent, quan interactua amb l'entorn, es defineixen els següents elements::

- Un conjunt d'estats d'entorn S;
- Un conjunt d'accions A;
- Regles de la transició entre els estats;
- Regles que determinen la recompensa immediata
- Regles que descriuen el que observa l'agent.

Les regles de transició entre estats venen donades per una fórmula anomenada probabilitat de transició que és la següent:

$$Pa(s, s') = Pr(st + 1 = s' \mid st = s, at = a)$$

Aquesta fórmula indica la probabilitat de passar de l'estat s a l'estat s' en prendre l'acció a en el temps t.

També tenim el cas de les regles de recompensa immediata que venen definides per la següent expressió:

$$Ra(s, s')$$

Aquesta recompensa, s'obté després de la transició de l'estat s a l'estat s' en prendre l'acció a. En el cas de la meva aplicació al joc, la recompensa immediata serà de 10 o -10, depenent de si ha menjat una poma o si ha mort.

En cada pas discret de temps t, l'agent interactua amb el seu entorn. Rep l'estat actual s_t i la recompensa r_t. A partir d'això, selecciona una acció a_t d'un conjunt d'accions disponibles i l'envia a l'entorn. Com a resultat, l'entorn passa a un nou estat s_{t+1} i proporciona una recompensa r_{t+1} associada a la transició (s_t, a_t, s_{t+1}).

L'objectiu de l'agent per aprenentatge per reforç és aprendre una política $\pi: A \times S \rightarrow [0,1]$ on $\pi(a, s) = \Pr(a_t = a \mid s_t = s)$, que maximitzi la recompensa acumulativa esperada. És a dir, l'agent vol trobar una estratègia òptima per a seleccionar accions en funció dels estats actuals, maximitzant la suma de recompenses en el llarg termini.

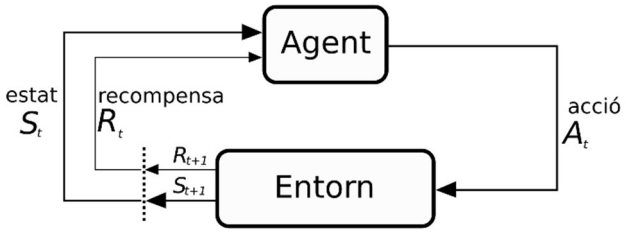

Fig 1: Funcionament del Reinforcement Learning.

Per la part del desenvolupament amb Reinforcement Learning, vaig prendre la decisió de fer servir PyTorch ja que és una llibreria pròpia de Python i directament suportada per la comunitat de recerca en l'aprenentatge automàtic. PyTorch ofereix una gran flexibilitat i eficiència per implementar i entrenar models de xarxes neuronals, cosa que el fa especialment adequat per al desenvolupament del model de Deep Q-Learning utilitzat en aquest projecte.

A continuació, es va definir l'arquitectura del model neuronal. Aquest model consta de diverses capes lineals intercalades amb funcions d'activació ReLU per introduir no linealitat en les prediccions. La xarxa neuronal rep com a entrada un vector d'estat que descriu l'entorn actual del joc, incloent la posició de la serp, la direcció actual, la ubicació de la poma i dels extrems, els perills al voltant, entre altres factors.

Durant l'entrenament, l'agent utilitza una política ϵ-greedy per seleccionar accions. Inicialment, l'agent pren accions de manera majoritàriament aleatòria per explorar l'entorn i, a mesura que l'entrenament avança, el valor de ϵ es redueix gradualment per afavorir l'explotació de les accions que han demostrat ser més rendibles segons les experiències acumulades. Aquesta estratègia permet equilibrar l'exploració de noves accions amb l'explotació de les accions conegudes que maximitzen la recompensa.

5 METODOLOGIA

Pel que fa a la metodologia, reconec que la selecció d'una metodologia adequada per a un projecte individual pot presentar certa complexitat. Malgrat això, he optat per implementar SCRUM com a metodologia preferent, ja que ofereix la possibilitat de realitzar un bon seguiment de les tasques. Per dur a terme SCRUM, faré servir l'eina "Jira", la qual em permetrà fer un seguiment detallat de les tasques. A més, tinc a la meva disposició un taulell KANBAN. Així, podré utilitzar simultàniament les metodologies SCRUM i KANBAN per a una planificació efectiva.

Amb l'objectiu de posar en pràctica la metodologia SCRUM, planejo realitzar Sprints de dues setmanes, començant el 29 de febrer. Quan arribi al quart Sprint, aquests s'extendran a 3 setmanes. Aquest canvi de dues a tres setmanes es deu al fet que, a mesura que avanço en etapes més avançades del desenvolupament, experimentaré dificultats per complir amb els terminis, tenint en compte la documentació, implementació, etc. Per tant, per ajustar-se a les entregues d'informes, els Sprints passaran de dues a tres setmanes. Aquesta elecció també pretén una sincronització amb les reunions de tutoratge, utilitzant l'últim dia de cada Sprint per a aquestes sessions de feedback.

Aquesta estructura em permetrà aprofitar al màxim les reunions amb el tutor i disposar d'una setmana intermèdia per a la discussió dels progressos o possibles retards experimentats en el desenvolupament del projecte.

6 PLANIFICACIÓ

Per poder portar a terme el projecte, també faré servir l'eina "Jira", així mantindré centralitzada una eina on podré controlar tot el que està relacionat amb el meu projecte.

Per establir una planificació precisa del projecte, he introduït conceptes coneguts com a "Epic", que representen les tasques de major envergadura. Cada Epic es descompon en subtasques més específiques, permetent-nos un seguiment detallat i establir un cronograma que faciliti l'estimació del temps necessari per a la seva realització..

Les tasques Epic a realitzar són les següents:

1. Entrega informe previ
2. Crear joc amb jugabilitat humana
3. Testing del joc amb jugabilitat humana.
4. Estudi d'algorismes d'IA.
5. Implementació d'algorismes d'IA.
6. Testing d'algorismes d'IA.
7. Entrega informe I
8. Estudi d'algorismes RL (Reinforcement Learning).
9. Implementació d'algorimes RL.
10. Testing d'algorismes RL.
11. Entrega informe II.
12. Conclusions i tancament del projecte.
13. Presentacions finals.

El detall del cronograma associat a aquestes tasques es troba disponible a la secció d'annexos d'aquest informe.

7 REQUERIMENTS

Per a la realització del projecte, he fet una recaptació de requeriments fent servir la metodologia MOSCOW per prioritzar-los. Aquesta metodologia divideix els requeriments en Must (Imprescindibles), Should (Importants) i Could (Desitjables). A continuació, veurem les llistes amb els requisits, separats en funcionals i no funcionals, i els seus respectius codis on entenem RF com a Requisit Funcional i RNF com a Requisit No Funcional, seguit d'un número que farà de codi identificador. Per tant, el conjunt seria RF_00 o RNF_00 per identificar els diferents Requisits.

7.1 Requeriments funcionals (RF)

Must (Imprescindibles):

- **RF_01:** El joc ha de tenir una interfície que permeti als jugadors interactuar amb la serp fent servir el teclat.
- **RF_02:** La serp ha de poder moure's en les quatre direccions principals: amunt, avall, esquerra i dreta.
- **RF_03:** S'han de generar aliments aleatoris dins del límit del joc.
- **RF_04:** El joc ha de mostrar la puntuació del jugador en pantalla, actualitzant-se cada vegada que la serp s'alimenti.
- **RF_05:** Quan la serp col·lisioni amb un aliment, ha de créixer i augmentar la puntuació del jugador.
- **RF_06**: Quan la serp col·lisioni amb el final de la pantalla, o amb el seu propi cos, morirà i el jugador perdrà la partida.

Should (Importants):

- **RF_07:** El joc ha de permetre reiniciar la partida quan la serp col·lisioni amb els límits del joc o amb el seu propi cos.
- **RF_08:** La IA quan jugui ha d'aconseguir una puntuació mínima de 15 punts.

Could (Desitjables):

- **RF_09**: Incorporar efectes de so com a reacció a les accions del jugador, com ara la recollida d'aliments o col·lisió de la serp.
- **RF_10:** Incorporar un apartat visual atractiu al jugador.

7.2 Requeriments no funcionals

Must (Imprescindibles):

- **RNF_01:** El joc ha de ser lleuger i mantenir un bon rendiment, fins i tot en equips amb recursos limitats.
- **RNF_02:** El joc ha de ser compatible amb la majoria d'equips disponibles a l'actualitat.
- **RNF_03**: El joc ha de ser clar i fàcil d'entendre per a totes les persones.

Should (Importants):

- **RNF_04:** El codi ha de ser testejat, almenys cada línia de codi.

Could (Desitjables):

- **RNF_05:** Incorporar un disseny visual atractiu i

coherent per millorar l'experiència de l'usuari i augmentar la immersió en el joc.

8 DESENVOLUPAMENT

A continuació, s'explica la implementació del joc Snake, la implementació basada en el jugador humà, la implementació basada en Reinforcement Learning i la realització de testing que s'ha dut a terme per poder assegurar un bon funcionament del projecte.

8.1 Joc Snake amb interacció humana

En aquesta secció, descriuré tot el que s'ha desenvolupat de la part de creació del joc Snake, entenent així les classes creades.

- **Classe Direction:** És una classe que fa el mapeig a través d'un enum, on assignem la direcció UP al valor 1, la direcció DOWN al valor 2, la direcció LEFT al valor 3 i la direcció RIGHT al valor 4.
- **Classe Snake:** És la classe responsable de gestionar el comportament relacionat amb la serp al joc. Guarda informació sobre la longitud, direcció, cos de la serp, mida del bloc, color, límit, puntuació i temps d'inici.
- **Classe Food**: És responsable de gestionar la generació i el comportament dels aliments al joc. Guarda informació sobre la mida, color, coordenades i límit del joc.
- **Classe MainHuman:** Aquesta classe gestiona el flux principal del joc per a la interacció humana. Ens serveix per tenir un punt d'entrada centralitzat a aquesta funció del joc, separant-lo del punt d'interacció amb la IA.

Fig X: Joc Snake amb acció humana

8.2 Testing del joc Snake amb interacció humana.

Una vegada ja hem creat el joc bàsic, s'ha de començar a fer proves sobre aquesta part. Per tant, el que hem fet és crear diferents classes que ens ajudaran a cobrir tot el codi..

- Classe TestSnakeFood: Classe que farà test a totes les funcions que hi ha a la classe del joc Food.
- Classe TestSnakeGame: Classe que farà test a totes les funcions que hi ha a la classe del joc Snake..
- Classe MainTest: És responsable de l'execució de proves unitàries per a les classes TestSnakeFood i TestSnakeGame. És l'entorn centralitzat de proves que ens servirà per crear un arxiu de cobertura.

La classe MainHuman no cal ser testejada ja que és només el control de flux del joc. Per tant, un correcte funcionament de les classes per separat ens garanteix un correcte funcionament de MainHuman.

Sobre l'arxiu de cobertura, gràcies a una llibreria anomenada Coverage.py, hem pogut crear un arxiu HTML que ens proporcionarà una estimació de la cobertura que passen els nostres tests.

Module	statements	missing	excluded	coverage
V:\TFG_Snake_Victor\food.py	19	0	0	100%
V:\TFG_Snake_Victor\game.py	74	1	0	99%
maintest.py	17	1	0	94%
testfood.py	33	0	0	100%
testgame.py	96	0	0	100%
Total	239	2	0	99%

Fig 2: Cobertura del joc amb agent humà

Com podeu observar, aquesta cobertura indica que falten 2 línies per provar. Fent una revisió dels tests podem observar que sí que s'executen aquelles línies, per tant podem veure que hi ha un petit marge d'error.

6.3 Implementació del Reinforcement Learning

Prèviament ja he comentat una mica com he implementat el Reinforcement Learning i el per què de la decisió de ferho en PyTorch i fent servir un model de Deep Q-Learning amb xarxes neuronals. Per tant, en aquesta subsecció parlaré sobre els detalls del desenvolupament, el funcionament de les classes i sobretot del bucle d'entrenament, que és la part central d'aquest projecte. Començaré per les classes noves que he implementat:

- **Classe SnakeGameAI**: Gestiona la lògica del joc per a la IA, incloent la inicialització del joc, el reinici, els passos del joc, la interfície d'usuari i la detecció de col·lisions.
- **Classe Linear_QNet:** Defineix la xarxa neuronal utilitzada per al model de Q-Learning profund. Inclou la definició de les capes de la xarxa i les funcions per guardar i carregar el model.
- **Classe QTrainer**: Gestiona l'entrenament del model, incloent l'optimització i la funció de pèrdua.
- **Classe Agent**: Gestiona l'aprenentatge de l'agent, incloent la memòria, la selecció d'accions i l'entrenament a curt i llarg termini.
- **Classe Evaluator:** Realitza l'avaluació del model entrenat jugant diverses partides i calculant les estadístiques com la puntuació mitjana i màxima.

Aquesta classe d'avaluació, veurem més endavant per què no funciona correctament.perquè no funciona correctament.

Seguint amb el pas a pas de l'entrenament, comentaré les passes principals que realitza el model durant les iteracions d'entrenament.

Comencem amb la part d'inicialització, on primer de tot, just al moment de començar l'entrenament, es crea un agent amb una xarxa neuronal que, inicialment, prendrà accions de manera totalment aleatòria (fase d'exploració). Seguidament, s'inicialitzaran tots els paràmetres d'entrenament com la memòria, la taxa d'aprenentatge, el descompte gamma i el decaïment d'epsilon.

Paràmetres d'entrenament:

- **Memòria**: La memòria es refereix a l'emmagatzematge de les experiències de l'agent, que inclouen transicions (estat, acció, recompensa, estat següent). Aquestes experiències es guarden en una cua de doble finalitat (deque) amb una mida màxima predeterminada. La memòria permet a l'agent entrenar-se amb mostres aleatòries d'experiències passades (replay buffer) per trencar la correlació temporal entre les transicions i estabilitzar l'aprenentatge. Li donem un valor de 100.000 transicions.
- **Taxa d'aprenentatge (α):** Hiperparàmetre que determina la velocitat amb la qual el model ajusta els seus pesos en resposta a l'error entre les prediccions i els valors objectius durant l'entrenament. En aquest projecte li donem un valor de 0,001.
- **Descompte Gamma (γ):** El factor de descompte determina la importància de les recompenses futures en comparació amb les recompenses immediates. Un valor més proper a 1 farà que es valorin més les recompenses a llarg termini, mentre que un valor més baix farà que es centri en les immediates. Li donem un valor de 0,9.
- **Decaïment d'Epsilon:** Paràmetre que controla el balanç entre exploració (provar noves accions) i explotació (seleccionar les millors accions conegudes). Inicialment, epsilon és alt per afavorir l'exploració, però es redueix gradualment per afavorir l'explotació a mesura que avança l'entrenament. Li donem un valor de 0,995 amb un mínim valor d'epsilon de 0,01.

Una vegada explicats els paràmetres i la part d'inicialització, comencem amb el bucle principal d'entrenament.

1. Inicialització: Es crea l'agent i la xarxa neuronal, i s'inicialitzen els paràmetres d'entrenament. Aquesta part ja l'he explicada prèviament.
2. Episodi de joc: Per a cada episodi, es reinicia l'entorn del joc i l'agent comença en un estat inicial.
3. Selecció d'accions: L'aget selecciona accions utilitzant una política ϵ-greedy, inicialment amb molta exploració i després amb més explotació a mesura que épsilon disminueix.
4. Interacció amb l'entorn: L'agent executa l'acció seleccionada, observa la recompensa obtinguda i el nou estat del joc.
5. Emmagatzematge de la transició: La transició (estat, acció, recompensa, estat següent) s'emmagatzema a la memòria de l'agent.
6. Entrenament a curt termini: L'agent entrena el model amb la transició més recent per ajustar els pesos de la xarxa neuronal.
7. Entrenament a llarg termini: Quan el joc finalitza, l'agent realitza un entrenament a llarg termini amb un lot de transicions seleccionades aleatòriament de la memòria.
8. Actualització de paràmetres: Es redueix el valor de epsilon per afavorir l'explotació de les accions conegudes a mesura que avança l'entrenament.
9. Replicació i guardat de model: Durant l'entrenament, es guarda periòdicament el model per evitar pèrdues de dades i permetre la continuïtat de l'entrenament.

Finalment, en aquest apartat de desenvolupament, acabaré comentant per què la classe d'avaluació no funciona. Durant el desenvolupament del projecte, vaig tenir problemes a l'hora de realitzar correctament la xarxa neuronal i que l'agent entrenés de forma eficient, és a dir, que l'agent aprengués dels seus entrenaments passats. Molta part del temps dedicat a la implementació del Reinforcement Learning va estar a la part d'ajustar correctament el model i l'agent. La part de l'avaluador, tot i veure que l'agent aprenia (com veurem als resultats), no funcionava correctament ja que semblava que no carregués bé el model o no reconegués bé les accions. Durant les 100 iteracions d'avaluació, sempre feia el mateix moviment. Per tant, es va decidir en conjunt amb el tutor avaluar el model de manera diferent, com veurem al següent apartat..

9 RESULTATS

Com he comentat prèviament, els resultats els avaluarem d'una forma diferent de l'habitual. El que faré serà executar el bucle d'entrenament 10.000 vegades i veure com evoluciona l'agent i si millora amb el temps. Així també podré veure tendències durant l'entrenament i concretar si està entrenant correctament o si, per altra banda, tot és de forma aleatòria i separada.

Per fer aquest procés, he dividit els entrenaments en execucions de 200 episodis. Aquest nombre és degut al fet que així puc fer un seguiment constant de les execucions i assegurar-me que l'agent realment està en un bon camí. Fer més episodis per execució suposaria més temps per cada una, a més d'analitzar més dades de cop. Seguidament, he guardat cada puntuació que obtingui a cada episodi (1 poma és igual a 1 punt) en un bloc de notes. Del bloc de notes anomenat result.txt, passaré les dades d'aquells 200 episodis a un document Excel on calcularé la suma de les puntuacions en aquells 200 episodis, a més d'altres

mètriques que em seran útils per fer l'anàlisi posterior.

Les mètriques que he fet servir per aquest anàlisi són:

- **Puntuació total:** Suma de puntuacions dels 200 episodis.
- **Desviació estàndard:** Mesura la variabilitat del conjunt de dades.
- **Puntuació mitjana:** Idea general del rendiment de l'agent durant els 200 episodis.
- **Mediana de puntuació:** Comprendre el valor central de les puntuacions.
- **Coeficient de variació:** Mesura la variabilitat relativa de l'agent. Un valor més baix indica que les puntuacions són més consistents.
- **Puntuació màxima:** Veure quin valor màxim ha assolit durant la fase d'explotació.

però s'ha de tenir en compte el següent: aquestes mètriques no són un absolut per avaluar el rendiment, ja que, com he comentat prèviament, l'agent passa per unes fases d'exploració i explotació. Per tant, durant les fases d'exploració la puntuació serà més baixa o directament zero, mentre que durant les fases d'explotació la puntuació serà més elevada.

Per tant, és important interpretar aquestes mètriques dins del context del procés d'entrenament. A mesura que l'agent avança en l'entrenament, hauríem de veure una tendència de millora en puntuacions mitjanes i màximes, així com una reducció de la variabilitat de les puntuacions, indicada per la desviació estàndard i el coeficient de variació. Aquestes tendències podrien indicar que l'agent està aprenent de manera consistent i efectiva.

9.1 Anàlisi i interpretació de resultats

A partir d'aquest punt, referenciaré els resultats que estan a l'arxiu "Resultats.xlsx" que hi ha al GitHub[15], ja que estan totes les mètriques analitzades. Per tant, per a més informació consultar aquesta font. A més, a l'apartat d'annexos trobareu els gràfics realitzats per a cada mètrica.

Per fer aquest anàlisi, revisarem cada mètrica per separat i els resultats que ens dona per als blocs de 200 episodis.

Puntuació total:

- Tendència: En general, es pot observar una tendència ascendent en la puntuació total a mesura que avancen els blocs d'entrenament. Poc a poc, però la tendència és creixent.
- Excepcions: Hi ha algunes excepcions on hi ha caigudes en la puntuació total. Això podria ser degut a fluctuacions en el rendiment de l'agent durant aquestes fases.

Desviació estàndard:

- Tendència: La desviació estàndard varia molt al llarg dels blocs, però hi ha una tendència a augmentar en alguns casos. Una desviació estàndard alta indica una alta variabilitat en les puntuacions, suggerint que l'agent encara està explorant i ajustant el seu comportament.
- Estabilització: En altres blocs més darrers, com del 8600 al 8800, la desviació estàndard és més baixa, indicant que l'agent està aconseguint una major consistència.

Mitjana:

- Tendència: La mitjana de puntuació generalment augmenta amb el temps, mostrant com l'agent està aprenent i millorant el seu rendiment.
- Anàlisi: Els blocs amb mitjanes més altes suggereixen que l'agent està explotant millor les seves estratègies per obtenir punts.

Puntuació Màxima

- Tendència: La màxima puntuació és de 40 punts al bloc 9400-9600 i la segona és al bloc 4600-4800 amb una puntuació de 37.
- Anàlisi: Cal comentar que la puntuació màxima ve donada per diversos factors, com la etapa d'explotació i el fet de la col·locació de les pomes. Per tant, és un valor que ens pot ajudar però no només s'han de mirar els pics, sinó la consistència d'aquests.
- Consistència: La presència constant de puntuacions altes (25-40) a la majoria de blocs ens indica que l'agent té potencial d'obtenir puntuacions elevades i no quedar-se en valors baixos.

Mediana

- Tendència: La mediana de puntuació es manté relativament estable al voltant de valors baixos inicialment, entre 1 i 2. Però tendeix a augmentar en execucions finals i es manté constant en una mediana de 2 i 3 i a vegades 4.
- Anàlisi: Una mediana més alta ens indicarà que hi ha més consistència de valors elevats, per tant el model estarà aprenent millor.

Coeficient de variació:

- Tendència: El coeficient de variació disminueix, sobretot baixant de 1 en els blocs finals.
- Anàlisi: Per tant, això és un clar indicador que l'agent està adquirint un comportament més previsible i consistent, millorant considerablement.

9.2 Observacions dels resultats obtinguts

Primer de tot, podem veure com els resultats estan molt lligats a les fases d'exploració i explotació del bucle d'entrenament. Durant la fase d'exploració, com que l'agent està provant accions aleatòries, tenim una major variabilitat i mètriques com la desviació estàndard i el coeficient de variació es veuen molt afectades. A més, en la fase d'exploració, al donar valors tan baixos i normalment zero, la mitjana es veu molt afectada. Per altra banda, en fases

d'explotació on l'agent està actuant gairebé amb tot el que ha après, veiem que les puntuacions són més constants i més altes. Per tant, en conclusió, a aquestes mètriques hem d'anar amb compte ja que al ser dues fases molt diferents, les mètriques es veuen molt afectades.

Tret d'això, veiem una millora global i una tendència positiva en les puntuacions mitjanes i màximes, el que indica que l'agent efectivament està aprenent de les seves experiències i millorant el seu rendiment a mesura que avancen les execucions d'entrenament.

Tot i així, hem d'anar amb precaució dels punts de caiguda que encara són persistents. Hi ha blocs on la puntuació total és molt baixa i la desviació estàndard molt alta, el que indica que l'agent s'ha d'ajustar encara més.

9.3 Conclusions dels resultats

Finalment, com a conclusions dels resultats, podem veure que aquests mostren una evolució positiva en el rendiment de l'agent, amb una millora general en les puntuacions i una major consistència a mesura que avança l'entrenament. Tot i això, la presència de grans fluctuacions en alguns punts indica que hi ha espai per millorar l'estratègia d'entrenament per aconseguir una estabilitat encara més gran.

Per tant, gràcies a aquest anàlisi de les mètriques, hem pogut confirmar que l'agent està sent entrenat de forma positiva i que, si es continués, s'aconseguirien millors resultats. Tot i així, s'haurien de revisar els paràmetres d'entrenament per poder veure si l'agent necessita variar-los per aprendre més.

10 Conclusió

Gràcies a aquest treball he pogut aprendre diferents tecnologies que no hauria après d'una altra forma. Primer de tot, el desenvolupament fent servir la llibreria Pygame ha estat una experiència totalment nova i he hagut d'aprendre nous conceptes de Python des de zero.

A més, m'ha servit de base per poder començar a desenvolupar jocs senzills. A part de Pygame, he après PyTorch, que sí que ha estat un veritable repte, ja que no només he hagut d'aprendre una llibreria nova, sinó també tot el que comporta aplicar-la al Reinforcement Learning i, en el meu cas, al Deep Q-Learning.

Cal destacar també una part molt important: aprendre a estructurar de manera correcta el codi per facilitar la utilitat de l'agent i el model de Reinforcement Learning. Es pot veure a versions prèvies del codi que estava tot molt diferent estructurat respecte a la fase final, i ha estat un treball d'organització i disseny de software per aconseguir que el meu algorisme funcionés de manera correcta.

Aquest projecte no només m'ha permès adquirir coneixements tècnics en Pygame i PyTorch, sinó que també m'ha proporcionat habilitats valuoses en projectes de programació, la resolució de problemes complexos i l'aprenentatge autònom, ja que al final el projecte l'he desenvolupat de manera individual i el meu únic suport per arreglar problemes era la documentació de Pygame i PyTorch i exemples que he trobat a la xarxa.

Per millorar aquest projecte, s'hauria de crear un avaluador que funcionés correctament i pogués valorar de manera real el punt on està el model entrenat. Una vegada es vegi el punt real on està, s'hauria de decidir si cal canviar aspectes de l'entrenament o crear un model nou amb altres paràmetres directament. Per tant, és una bona feina a futur i segurament un repte que em proposaré de manera personal: acabar aquesta part per poder seguir augmentant els meus coneixements en aquest àmbit. Tot i així, estic molt content de com ha anat evolucionant el projecte i de com he gestionat els temps, ja que he anat gairebé sempre en línia amb el que vaig marcar a la planificació a l'inici del projecte i he aprofitat les setmanes restants per realitzar l'entrenament de manera correcta.

Finalment, gràcies a aquest treball de final de grau he aconseguit una bona base en tecnologies clau per al desenvolupament en Python i Pygame i he iniciat aquest camí en l'aprenentatge sobre el Reinforcement Learning.

Agraïments

Principalment, m'agradaria agrair a en Coen, amb qui gairebé cada setmana ens hem reunit i hem parlat sobre el TFG, així com sobre molts altres aspectes personals, cosa que m'ha fet sentir molt còmode. A més, m'agradaria agrair a la meva família, que m'ha donat un gran suport durant tota la carrera i que ha ajudat a que no la deixés fa ara mateix 3 anys.

Bibliografia

[1] Loeber, P. Snake AI Pytorch. Recuperat 27 de febrer de 2024, GitHub. https://github.com/patrickloeber/snake-ai-pytorch

[2] amnindersingh1414. Snake Game in Python – Using Pygame module. Recuperat 27 de febrer de 2024, Geeks for Geeks. https://www.geeksforgeeks.org/snake-game-in-python-using-pygame-module/

[3] Wikipedia contributors. Snake (vídeo game). Recuperat 28 de febrer de 2024, Wikipedia. https://en.wikipedia.org/wiki/Snake_(video_game)

[4] Kim, Y. Pygame v2.1.4 documentation. Recuperat 28 de febrer de 2024, Pygame. https://www.pygame.org/docs/

[5] David D. sNNake. Recuperat 24 d'abril de 2024, https://davidd-55.github.io/sNNake/

[6] Chang, A. Mastering the game of Snake with AI and Reinforcement Learning. Recuperat 28 d'abril de 2024, Medium. https://medium.com/@aidenchang/mastering-the-game-of-snake-with-ai-and-reinforcement-learning-17f03397b060

[7] Reddit contributors. AI beats Snake game with Deep Q-learning. Recuperat 28 d'abril de 2024, Reddit. https://www.reddit.com/r/reinforcementlearning/comments/zfvyq1/ai_beats_snake_game_with_deep_qlearning/

[8] CEAS Journal. AI beats Snake game with Deep Q-learning. Recuperat 29 d'abril de 2024, CEAS Journal. https://ceasjournal.com/index.php/CEAS/article/download/13/10

[9] Geeks for Geeks contributors. What is Reinforcement Learning? Recuperat 5 de maig de 2024, Geeks for Geeks. https://www.geeksforgeeks.org/what-is-reinforcement-learning/

[10] PyTorch contributors. PyTorch tutorials. Recuperat 2 de maig de 2024, PyTorch. https://pytorch.org/tutorials/

[11] PyTorch contributors. Introduction to PyTorch. Recuperat 3 de maig de 2024, PyTorch. https://pytorch.org/tutorials/beginner/introyt.html

[12] PyTorch contributors. PyTorch documentation. Recuperat 15 de maig de 2024, PyTorch. https://pytorch.org/docs/stable/index.html

[13] Wikipedia contributors. Aprenentatge per reforç. Recuperat 27 de maig de 2024, Wikipedia. https://es.wikipedia.org/wiki/Aprendizaje_por_refuerzo

[14] S.G. Excel Formulas Cheat Sheet. Recuperat 12 de juny de 2024, Cheatography. https://cheatography.com/s-g/cheat-sheets/excel-formulas/

[15] Valeris9 (Victor Valero Carrasco). TFG Snake Victor. GitHub. https://github.com/Valeris9/TFG_Snake_Victor

ANNEXOS

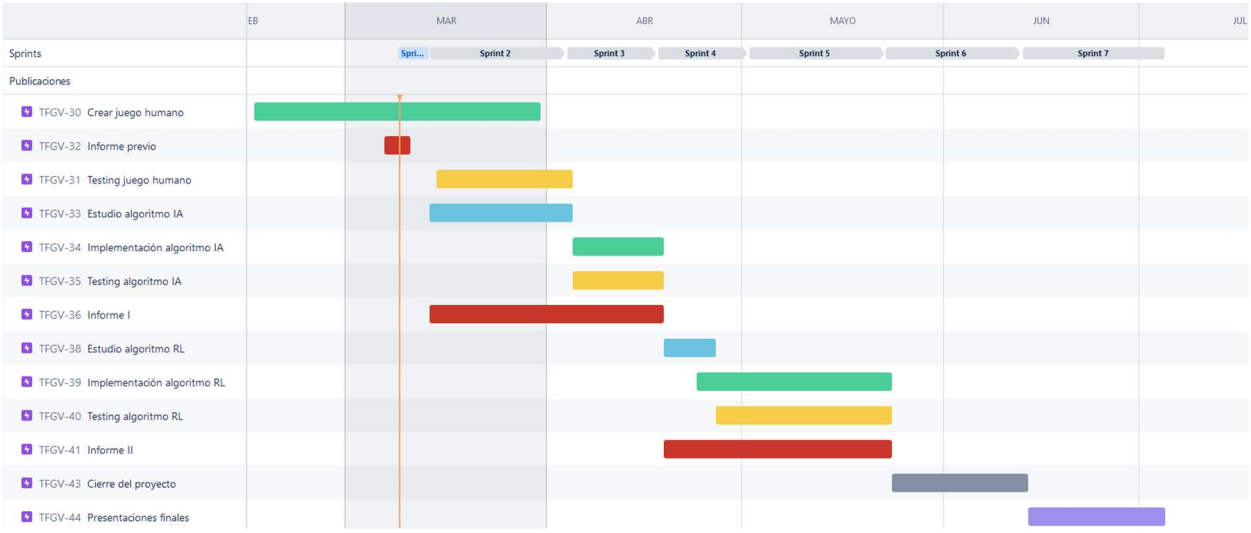

Fig. 4: Diagrama de la planificació

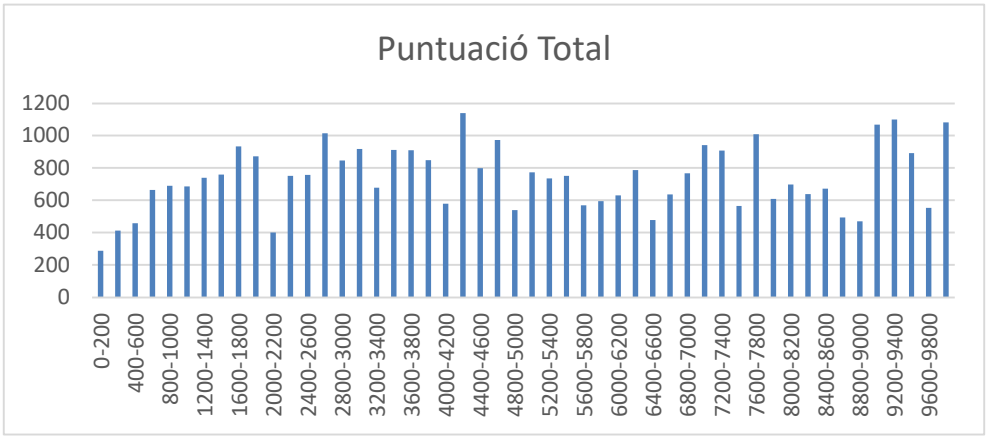

Fig. 5: Gràfic de barres de la puntuació total.

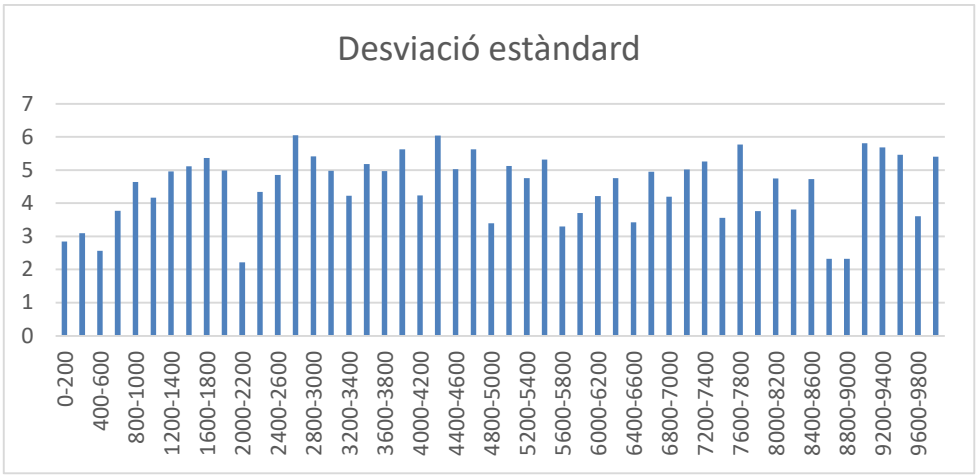

Fig. 6: Gràfic de barres de la desviació estàndard

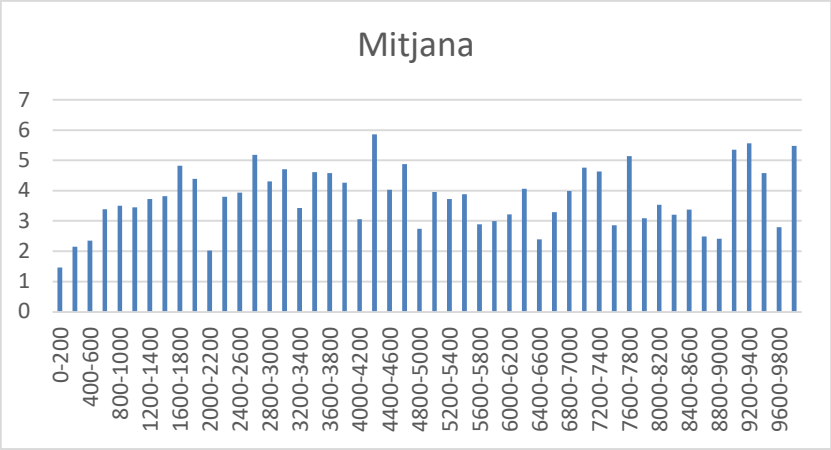

Fig 7: Gràfic de barres de la mitjana.

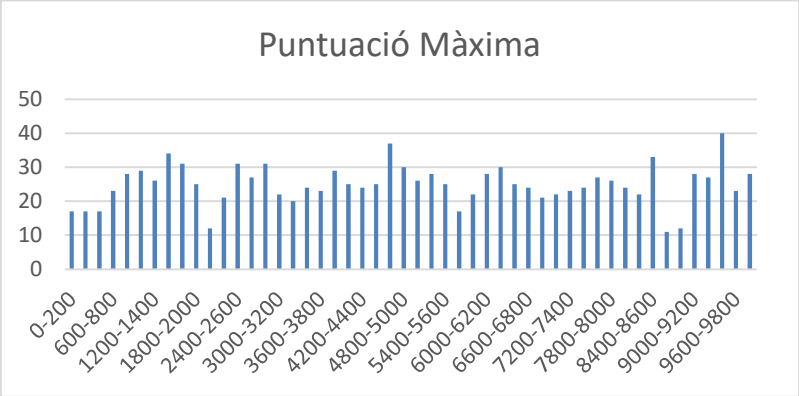

Fig 8: Gràfic de barres de la puntuació màxima.

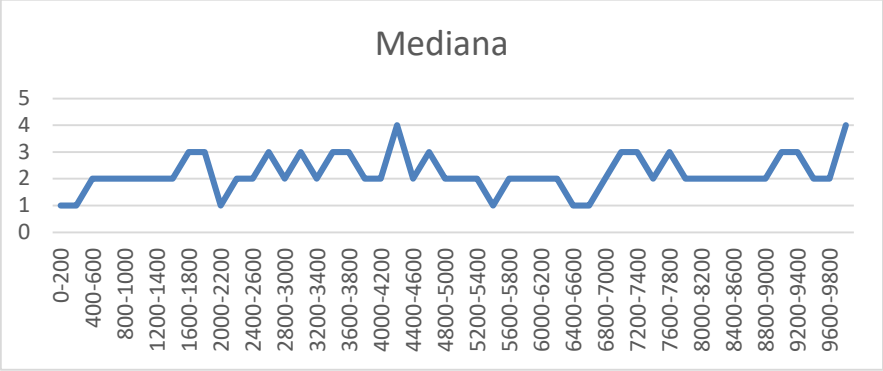

Fig 9: Gràfic de línies de la mediana.

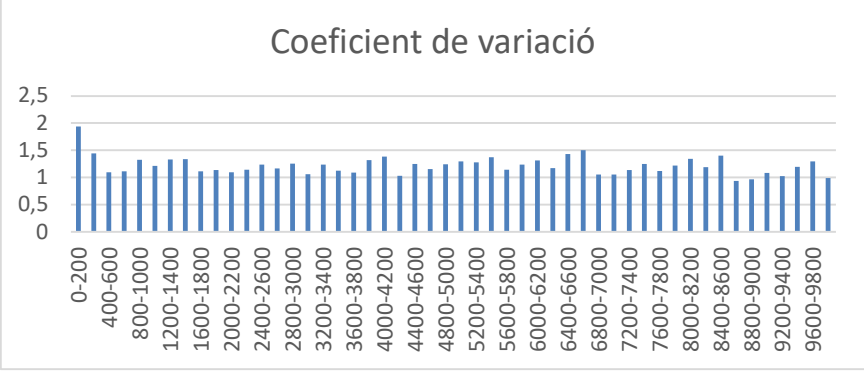

Fig 10: Gràfic de barres del coeficient de variació