1. Gasitul mancarii:

Scop:

- Antrenarea pestelui sa se deplaseze si sa colecteze mancare

Initiere:

- AI-ul primeste:
  - Pozitia sa
  - Pozitia mancarii
- Ai-ul primeste recompensa de -0.001 la fiecare frame pentru a descuraja statul pe loc
- La fiecare episod nou:
  - mancarea este mutata la o pozitie aleatoare din acvariu
  - Pestele isi pastreaza pozitia
- Recompense:
  - +1 pentru ajungerea la mancare,
  - -1 pentru atingerea peretilor

Rezultat:

- AI-ul a invatat initial cu succes sa gaseasca mancarea cea mai apropiata de el
- In lipsa mancarii, ii era trimisa o valoare aleatorie pentru inot, altfel, miscari haotice pe loc.

2. Balans inot si mancat in functie de foame:

Scop:

- Antrenarea pestelui sa manance doar cand ii e foame, altfel sa inoate

Modificarea datelor de intrare:

- AI-ul primeste aditional:
  - Pozitie de inot
  - O valoare bool daca exista mancare sau nu
  - Parametrii interni (foame, stress, etc) initiati cu 0 momentan (cu exceptia foamei)

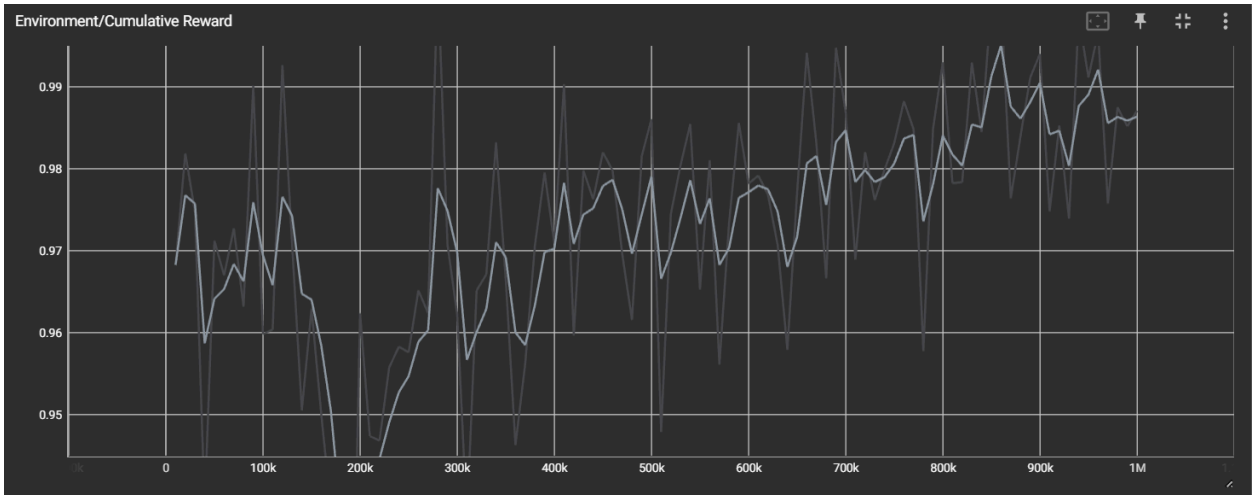
Initiere:

- La fiecare episod nou:
  - Mancarea este mutata aleator in acvariu
  - Se da o valoare random intre 0 si 1 parametrului foame
  - Booleanul de mancare este implicit true in cadrul antrenamentului
- Recompense:
  - 1 daca a atins mancarea cand foamea e peste jumate
  - 1 daca a inotat cand foamea e sub jumate

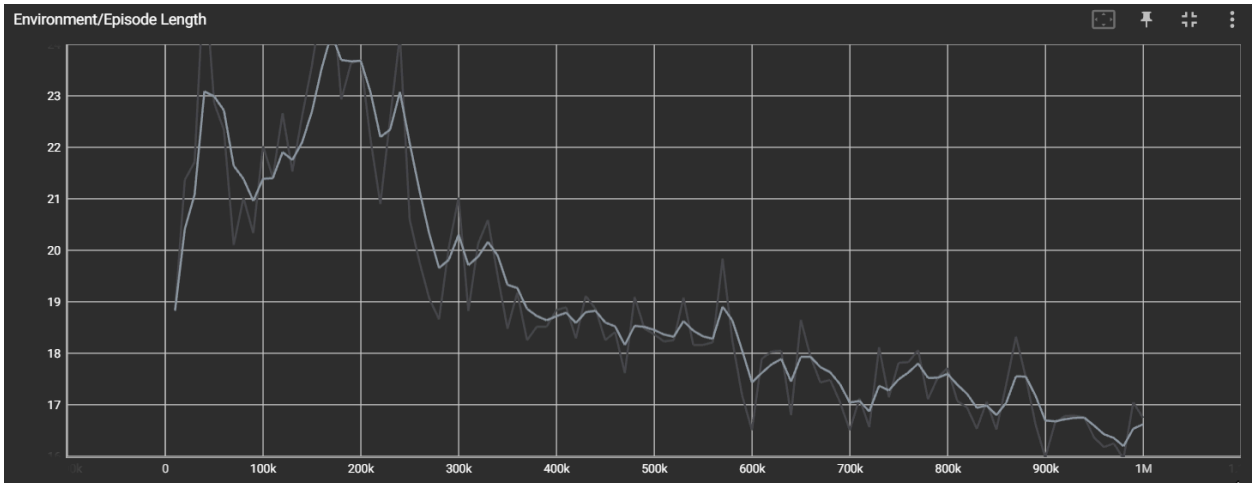
Rezultat:

- Ai-ul a invatat sa aleaga mancarea cand foamea este peste ~0.5 si inotul cand foamea este sub 0.5
- In lipsa unei mancari in acvariu, cazul in care booleanul de mancare este false, pestele se misca haotic, nu stie ce sa faca, indiferent de valoarea foamei
- In cazul in care ceilalti parametrii interni au valori diferite de valoarea 0, pestele se misca haotic, nu stie ce sa faca

- Grafic al recompensei dupa al celui de-al 4lea model antrenat (reusita de aproape 1):



- Grafic al vitezei de atingere a tintei:



3. Obisnuirea modelului cu variatie in parametrul de stress

Scop:

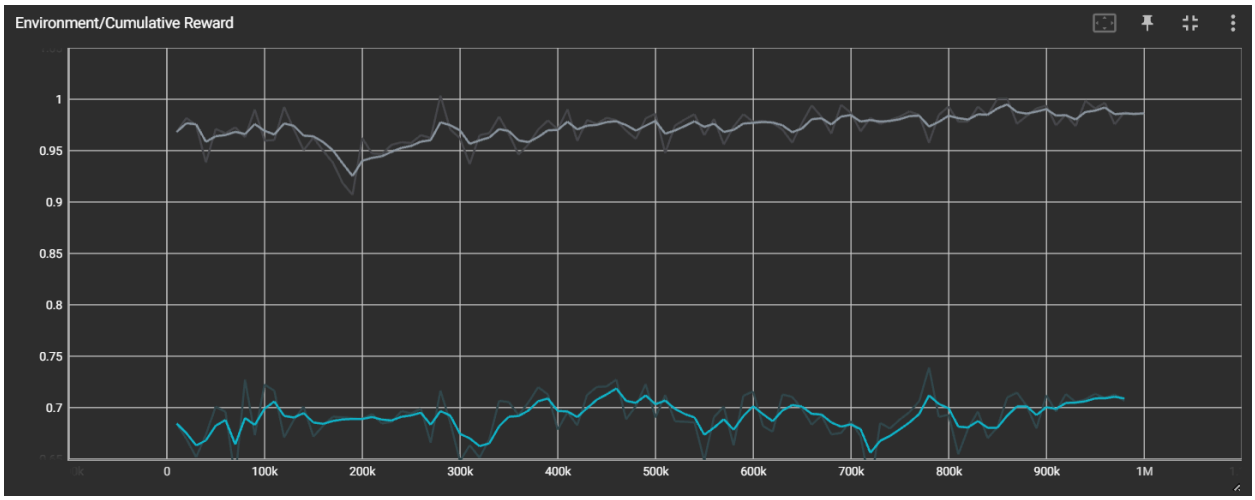
- Obisnuirea modelului cu valori diferite de stres pentru a facilita viitoare antrenari fara pierderea comportamentului deja invatat

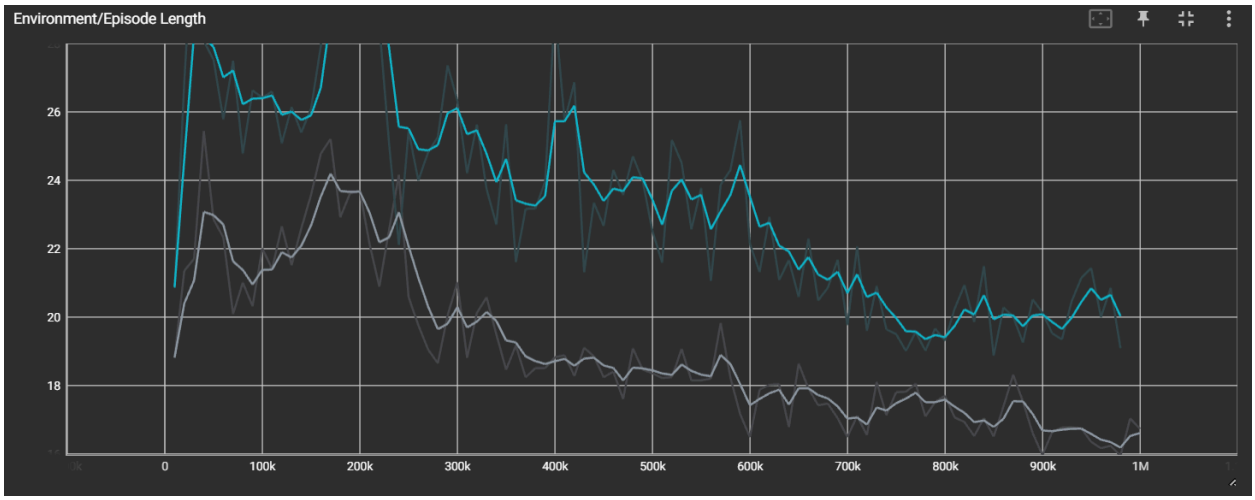
Initiere:

- La fiecare episod nou:
  - Mancarea este mutata aleator in acvariu
  - Se da o valoare random intre 0 si 1 parametrului foame
  - Booleanul de mancare este implicit true in cadrul antrenamentului
  - Se initiaza o valoare de stres aleatoare de la 0 la 1
- Recompense modificate:
  - Valoare foame pentru gasirea mancarii,
  - 1 - valoare foame pentru gasirea pozitiei de inot

Rezultat:

- Ai-ul a invatat sa se adapteze la noile valori de stres si sa actioneze similar cu modelul anterior, insa cu o performanta mai slaba
- Ar fi necesar mai mult antrenament de obisnuire insa nu voi insista, din moment ce stresul va avea un impact mai puternic in viitor
- Grafic vizual al rezultatului fata de varianta antrenata cu valoarea de stres 0, recompensa medie a scazut de la 0.95 la 0.7 (datorita schimbarii modului de calcul a recompensei)





(gri - 2.4, albastru - 3.3 antrenat pe baza 2.4)

4. Antrenarea pentru inot in cazul in care nu exista mancare

Scop:

- In cazul in care nu exista mancare in acvariu, indiferent de nivelul de foame, pestele ar trebui sa aleaga inotul

Incercari anterioare:

- Recompensa:
  - neschimbata pentru cazul anterior
  - 1 pentru atingerea locatiei de inot cand nu exista mancare
- Rezultat:

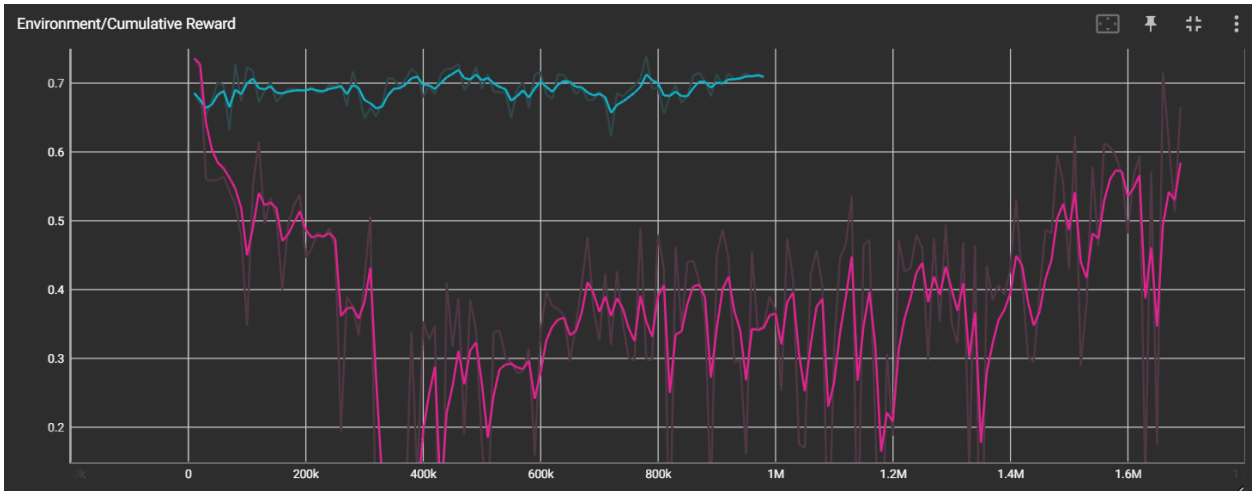
        - Pestele continua sa se miste haotic mai mult pe loc decat deplasandu-se.
        - Dupa ajungerea la modelul 3 antrenat, nici o imbunatatire nu a fost facuta
        - Pestele nu reusea sa descopere noua recompensa

Initiere:

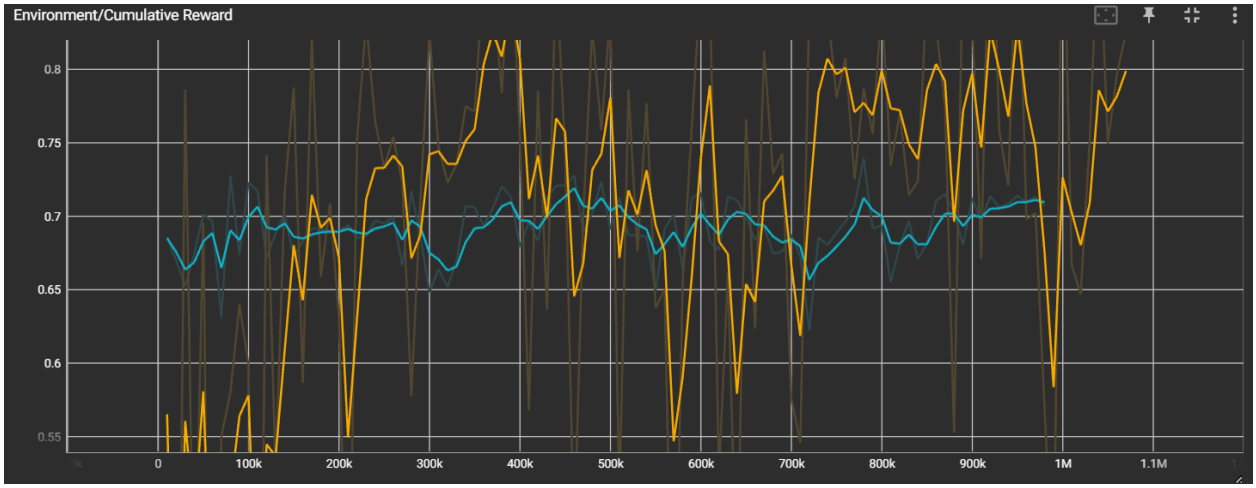
- Date de intrare neschimbate
- Modificarea regizorului de antrenament pentru a lasa mancare sau nu aleator, cu o proportie de 50% (ne dorim sa continue sa foloseasca ce a invatat anterior, sa nu uite, asa ca lasam posibilitatea de antrenare a acelei aptitudini)
- Recompense:
    - Neschimbate pentru cazul in care exista mancare
    - Pentru cazul in care nu exista mancare:
        - cu cat pestele se apropie mai mult de locatia de inot, cu atat primeste o recompensa aditionala mai mare
        - Recompensele sunt oferite o singura data la atingerea unor praguri de dinstanta fata de destinatie
        - La atingerea destinatiei se ofera o recompensa mult mai mare
        - Toate recompensele adunate se acumuleaza la valoarea totala de 1
        - Pentru cazurile in care pestele are de parcurs mai putina distanta totala, se calculeaza recompensa care ar fi fost obtinuta si se adauga la recompensa finala
        - Formula gasita: sum((1-goalProcent) * stepRewardMultiplier) + finalreward = 1

Rezultat:

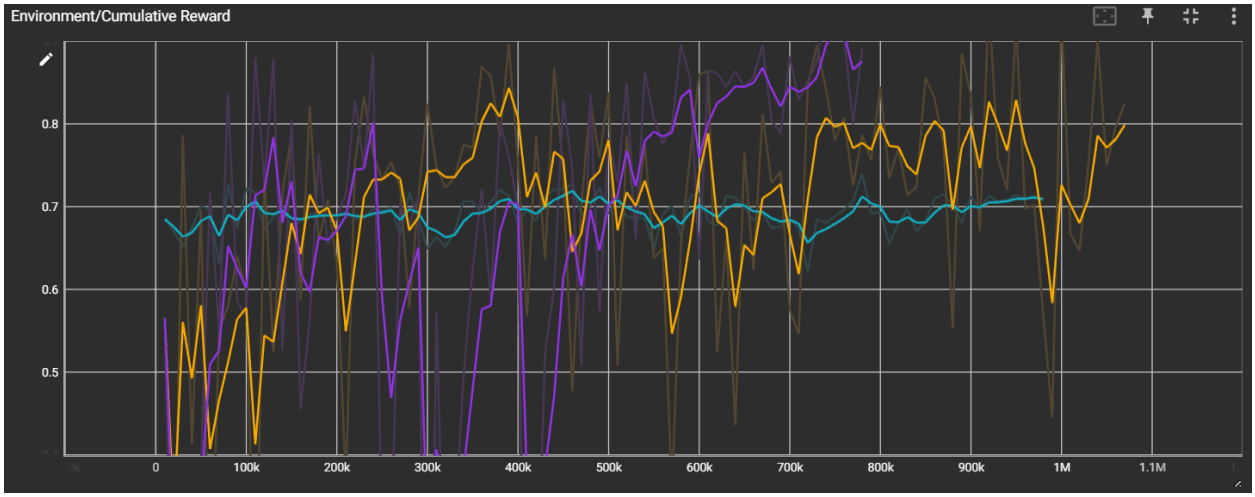
- Ai-ul are un progres aparent haotic, datorita schimbarii dintre cele 2 scenarii
- Este nevoie de mai mult antrenament pentru a ajunge la rezultate bune
- Grafic al recompensei relativ la albastru-3:



(roz-4.1 antrenat pe baza 3.3)

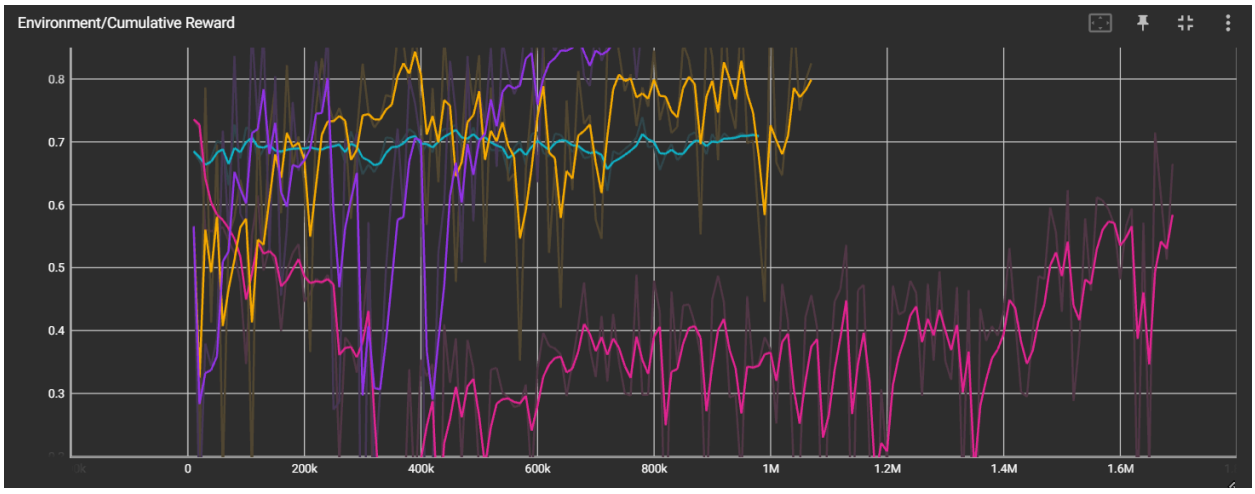


(galben-4.2, antrenat pe baza 4.1)



(mov-4.3, antrenat pe baza 4.2)

- Comparativ intre cele 3:



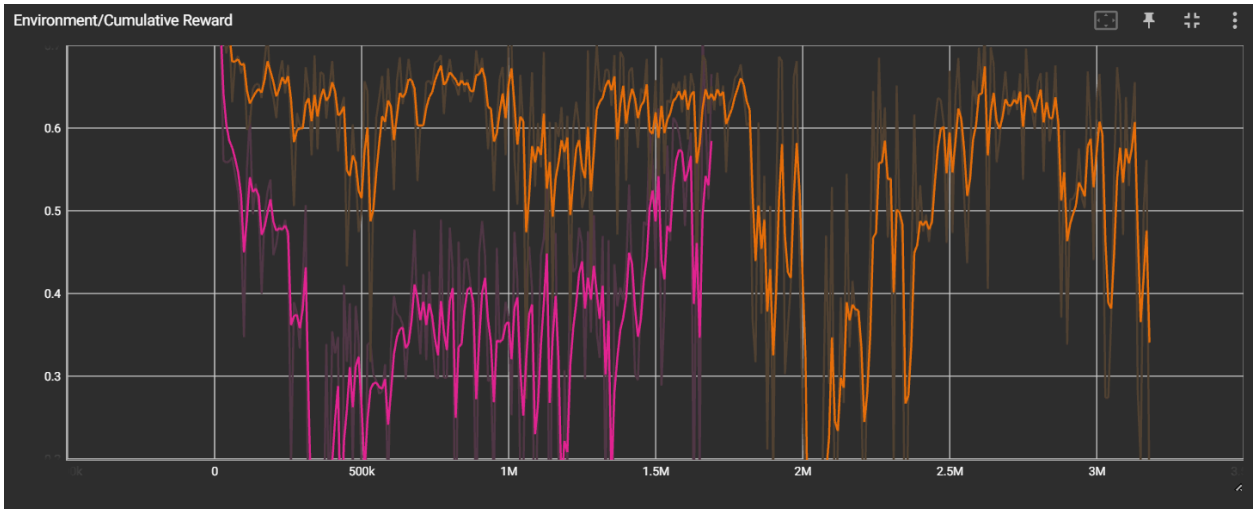
4r. Reglaj Antrenarea pentru inot in cazul in care nu exista mancare

Initiere:

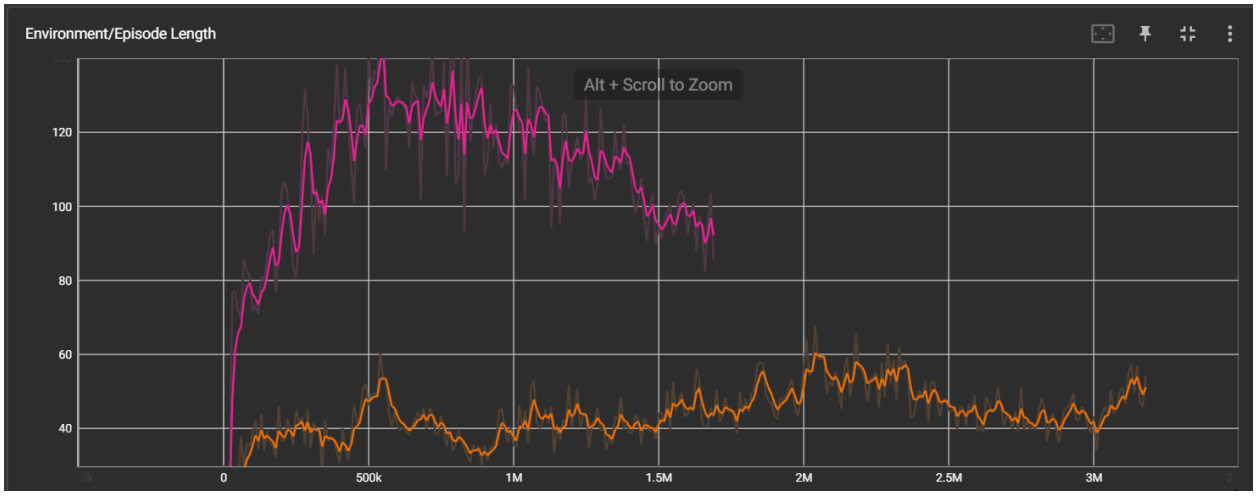
- Am modificat valorilor parametrilor din formula de recompensa pentru inot in cazul lipsei mancarii
- In urma modificarilor, recompensele sunt acordate si la praguri mai dese, dar sunt recompense mai mici decat cele precedente
- Suma totala este in continuare 1
- S-a reantrenat pe baza lui 3.3 pentru comparatie

Rezultat:

- Rezultatul initial pare sa fi dat rezultate mai bune decat rezultatul initial al valorilor anterioare:



(roz-4.1 antrenat pe baza 3.3 , portocaliu-4.1r antrenat pe baza 3.3)



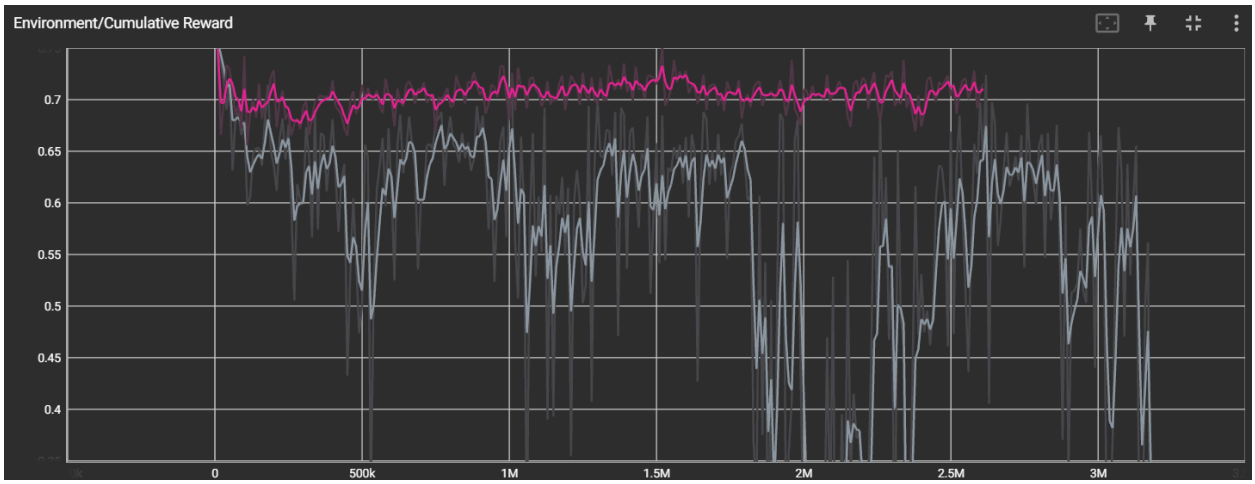
4r. Reglaj - Final Antrenarea pentru inot in cazul in care nu exista mancare

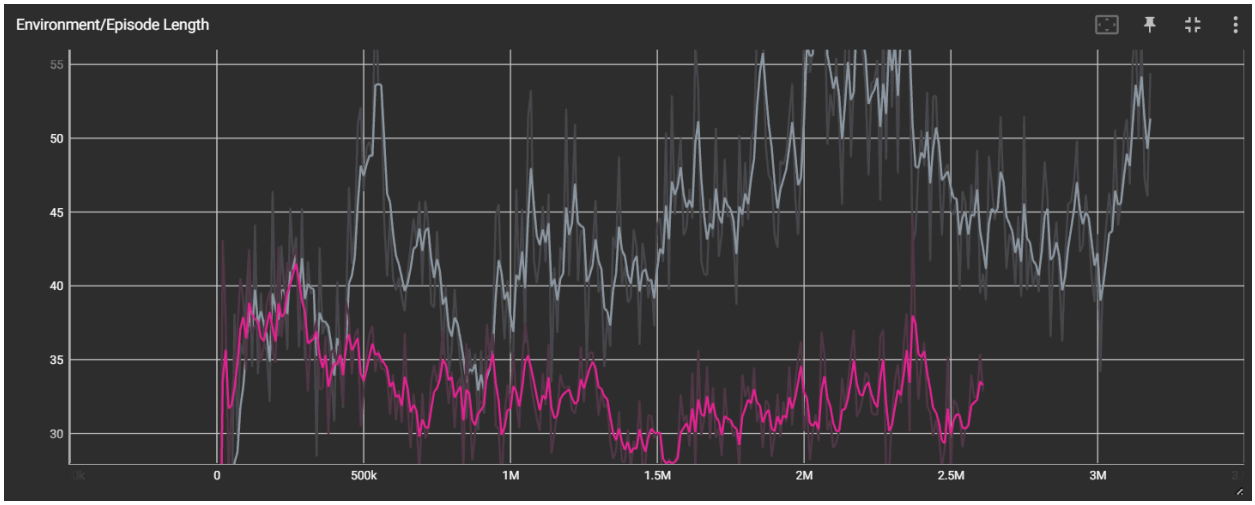
Initiere:

- Date de intrare neschimbate
- Modificarea regizorului de antrenament pentru a lasa mancare sau nu aleator, cu o proportie de 70%

Rezultat:

- Ai-ul are un randament majoritar pozitiv cu cateva situatii de confuzie





(roz-4.3r antrenat pe baza 4.2r, gri-4.1r antrenat pe baza 3.3)

5. Agresivitate in functie de stress

Scop:

- Ai-ul trebuie sa invete sa faca decizia de atac atunci cand stresul este mai mare, in functie de niste thresholduri (ce vor fi adaptate pentru antranarea de variatii de agresivitate)
- **Incercarea 1:**
    - **Initiere:**
    - Daca ai-ul face alegerea de attack, se transmite pozitia unui peste in locul pozitie de mancare
    - Recompense:
        - In cazul in care se face dicizia de attack si coliziune cu alt peste:

        - Stress < Tmin : reward = - (agresivityMinThreshold - stress) / agresivityMinThreshold
        - Tmin < Stress < Tmax: reward = (stress - agresivityMinThreshold) / (agresivityMaxThreshold - agresivityMinThreshold)
        - Tmax < Stress: reward = 1 - ((1 - stress) / (1 - agresivityMaxThreshold))
    - Cand stresul e mai mic de Tmin, rewardul va fi negativ
    - Pentru un stress mai mare de Tmin, rewardul variaza intre 0 si 1
- **Rezultat:**
- Pestele nu a invatat sa atace
- Trebuie invatata decisia de atac separat de coliziunea cu pestele

- **Incercarea 2:**
    - Modificarea recompensei oferite constant in cadrul episodului, imediat dupa decidere (ex: 0.005f * Time.deltaTime)
    - Recompensa oferita pozitiva sau negativa in functie de stress
    - **Rezultat**: pestele schimba in continuu decizia, recompensele se anuleaza constant, nu se face o diferenta semnificativa
- **Incercarea 3:**
    - Adaugarea unui multiplicator pentru a marii rewardul daca decizia luata este aceasi cu cea antetioara
    - Pentru decizia corecta, rewardul creste semnificativ, pentru decizia gresita, rewardul scade puternic, pe negativ
    - **Rezultat:** valuarea medie a recompensei a crescut enorm fata de celelalte incercari, iar ai-ul nu mai tine cont la fel de bine de recompensele de inot si mancare (care sunt normalizate la [0-1])
- **Incercarea 4:**
    - Micsoarea cresterii factorului de recompensa
    - Adaugarea unei limite de maxim 1p total pentru rewardul dat de decizie
    - **Rezultat**: AI-ul este confuzat din cauza suprapunerii de recompense si actiuni, inotul devine haotic

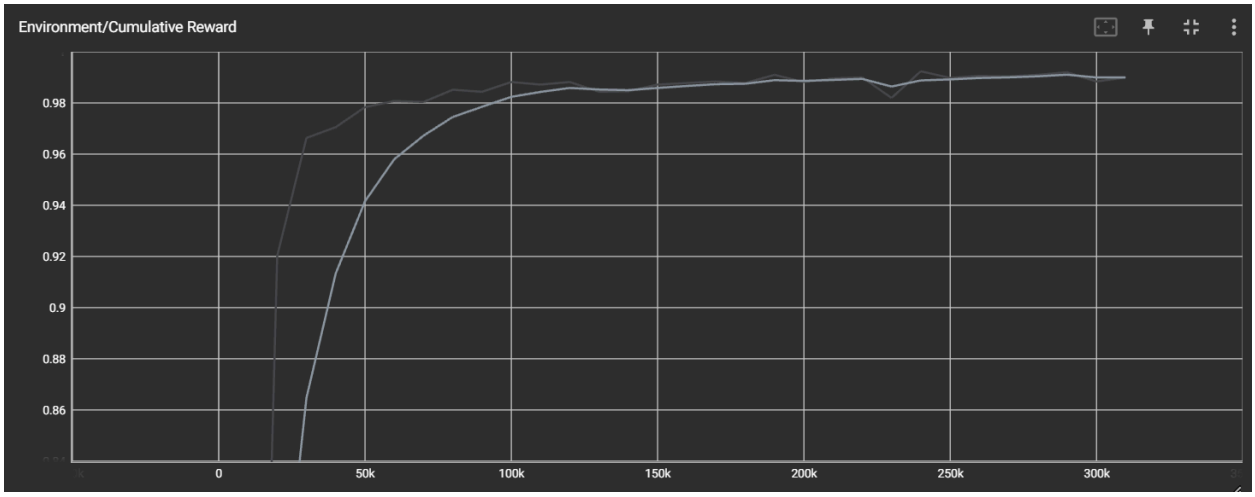
5.1. Agresivitate in functie de stress - Agent Separat

Initiere:

- Creerea unui nou agent ml care sa se concentreze doar pe decizia de atac
- Initierea unui nou enviorment de antrenament
- Date de intrare: stresul si un threshold de stress pentru care se va face decizia

Rezultat:

- Invatarea deciziei este rapida, si corecta
- Utilizarea de treshold variabil ofera posibilitarea de obtinere a nivele de agresivitate setabile
- Integrarea cu agentul pentru intot si hrana este mult mai usoara decat antrenarea ambelor actiuni intr-un singur model, si mai modulara

- 

6. Reantrenare Inot - Model separat

Scop:

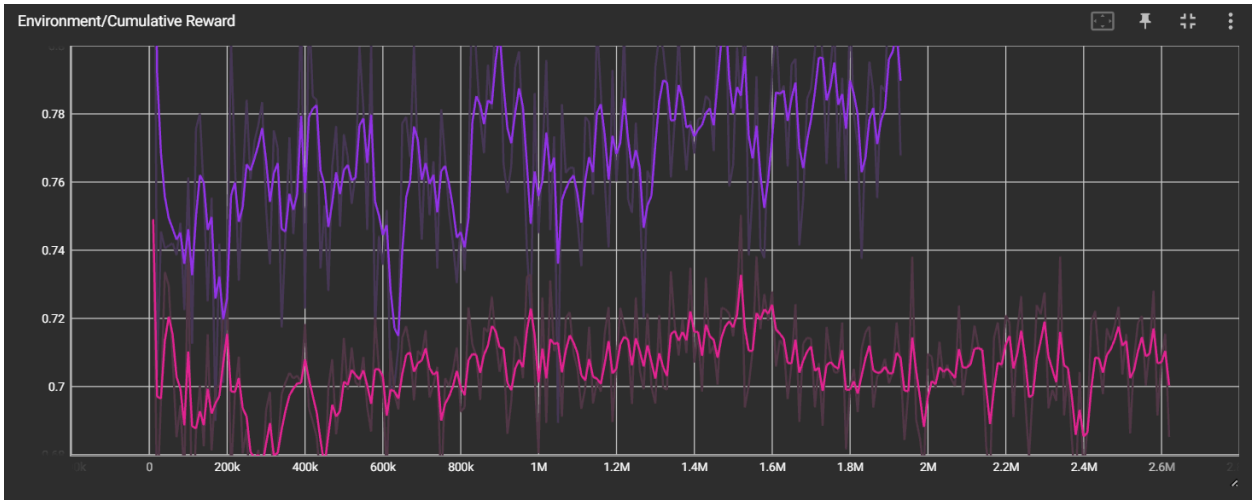
- Inotul pestelui nu este suficient de accurate, se chinuie mult la deplasare
- Pentru integrarea completa a atacului este necesar un AI de miscare mai bun
- Separarea in agent de miscare/hrana si agent de decizie de attack necesita modificarea parametrilor primiti de agentul de miscare si reantrenarea acestuia

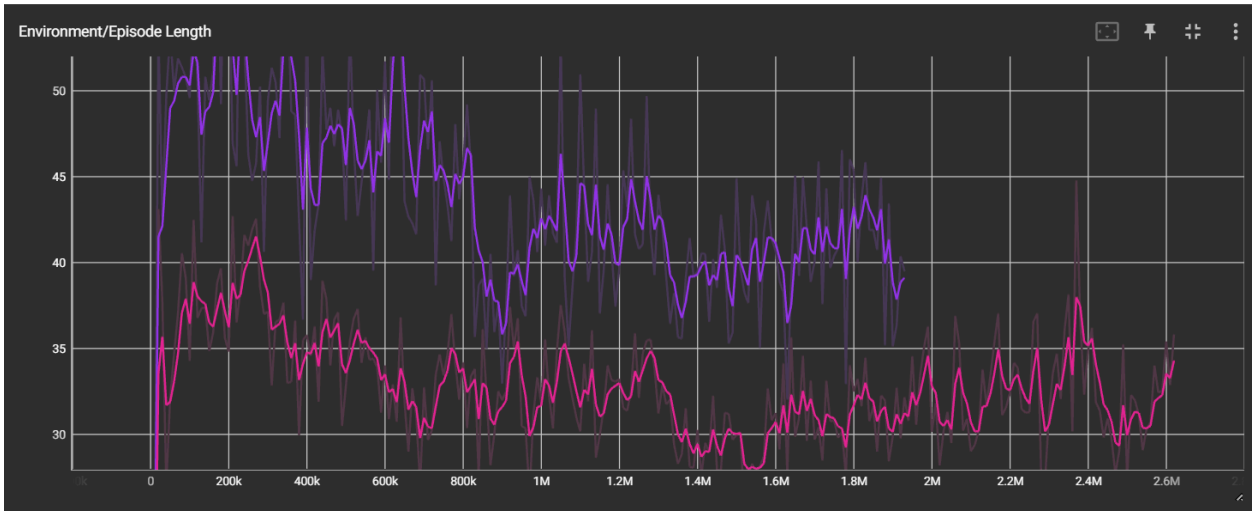
Initiere:

- Mici ajustari la recompense, dar in principiu neschimbat
- Accent pus pe reantrenare si perfectionare
- Observatii primite:
    - Pozitia pestelui
    - Pozitia mancarii
    - Pozitia de inot
    - Un boolean reprezentand daca exista mancare
    - Valoarea foamei
- Recompense: neschimbate / putin ajustate fata de modelul precedent

Rezultat:

- Modelul de inot are in continuare gafe, cu un scor mediu de 0.75
- Trebuie regandit in totalitate modelul
- Dupa mai multe generatii de antrenament, cel mai bun rezultat optinut este doar cu putin mai bun decat modelul precedent, dar mai lent





(mov-6.4r antrenat dupa 8 generatii si 2 restructurari de cod, roz-4.3r modelul precendent)

7. Restructurarea agentului de Inot

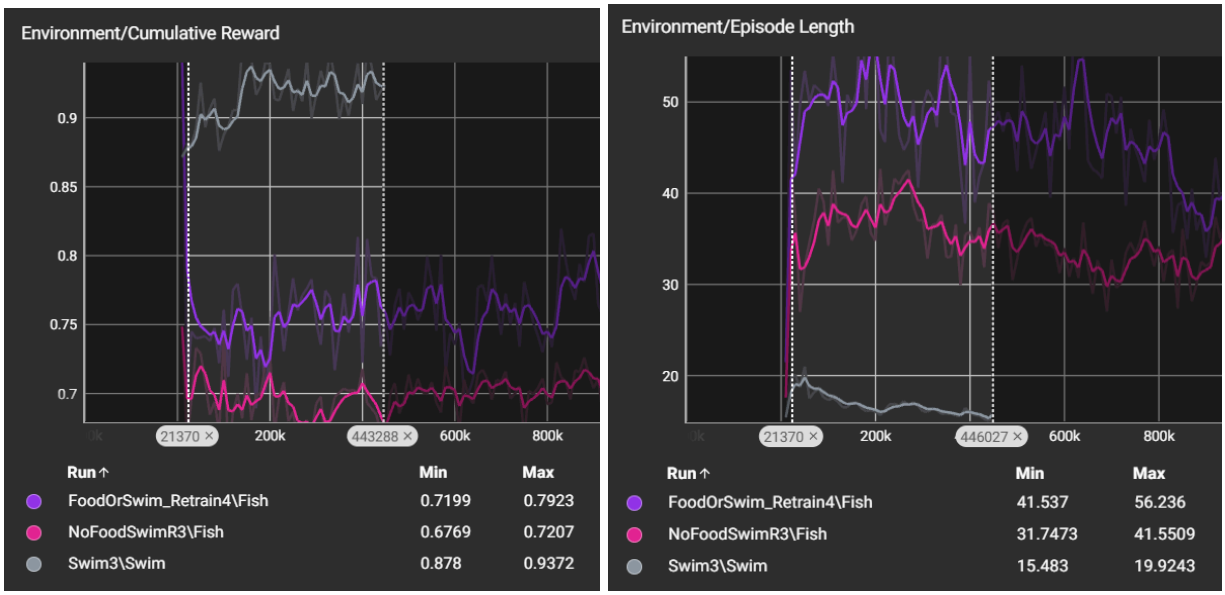
Initiere:

- Restructurarea completa a agentului de inot
- Oferirea de observatii aditionale
- Modelul primeste ca observatii:
    - Pozitia pestelui
    - Pozitia de inot
    - Pozitia mancarii
    - Un boolean reprezentand daca exista mancare
    - Foamea si limita de foame
    - Limitele acvariului
    - Velocitatea de deplasare a pestelui
    - Directia dintre pozitia pestelui si tinta
- Recompense:

- Atingerea mancarii: +1 daca foamea este peste treshold, 0 altfel
- Atingerea destinatiei de inot:
    - +1 daca nu exista hrana
    - +1 daca exista hrana dar foamea este sub treshold
    - -1 daca exista hrana dar foamea este peste treshold
- Atingerea limitelor acvariului: -1

Rezultat:

- Pestele a invatat mult mai bine sa inoate, si si-a marit si acuratetea, iar timpul de atingere a tintei s-a micsorat enorm
- Dupa un timp mult mai scurt decat celelalte modele, progresul este extrem de evident
- Limita de foame este acum setabila



(gri-7.3 antrenat dupa restructurarea completa, mov-6.4r, roz-4.3)