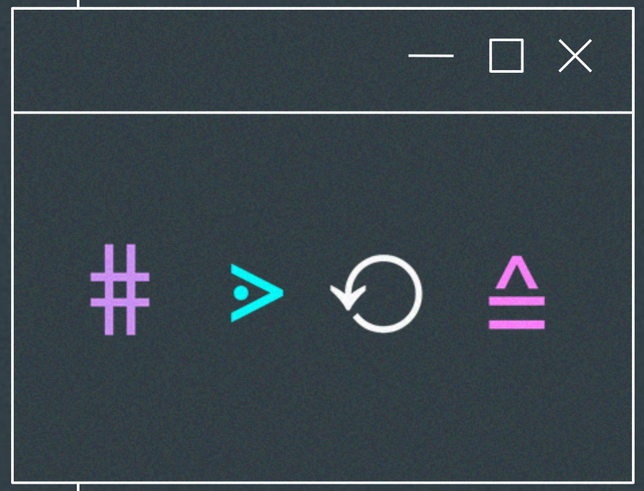
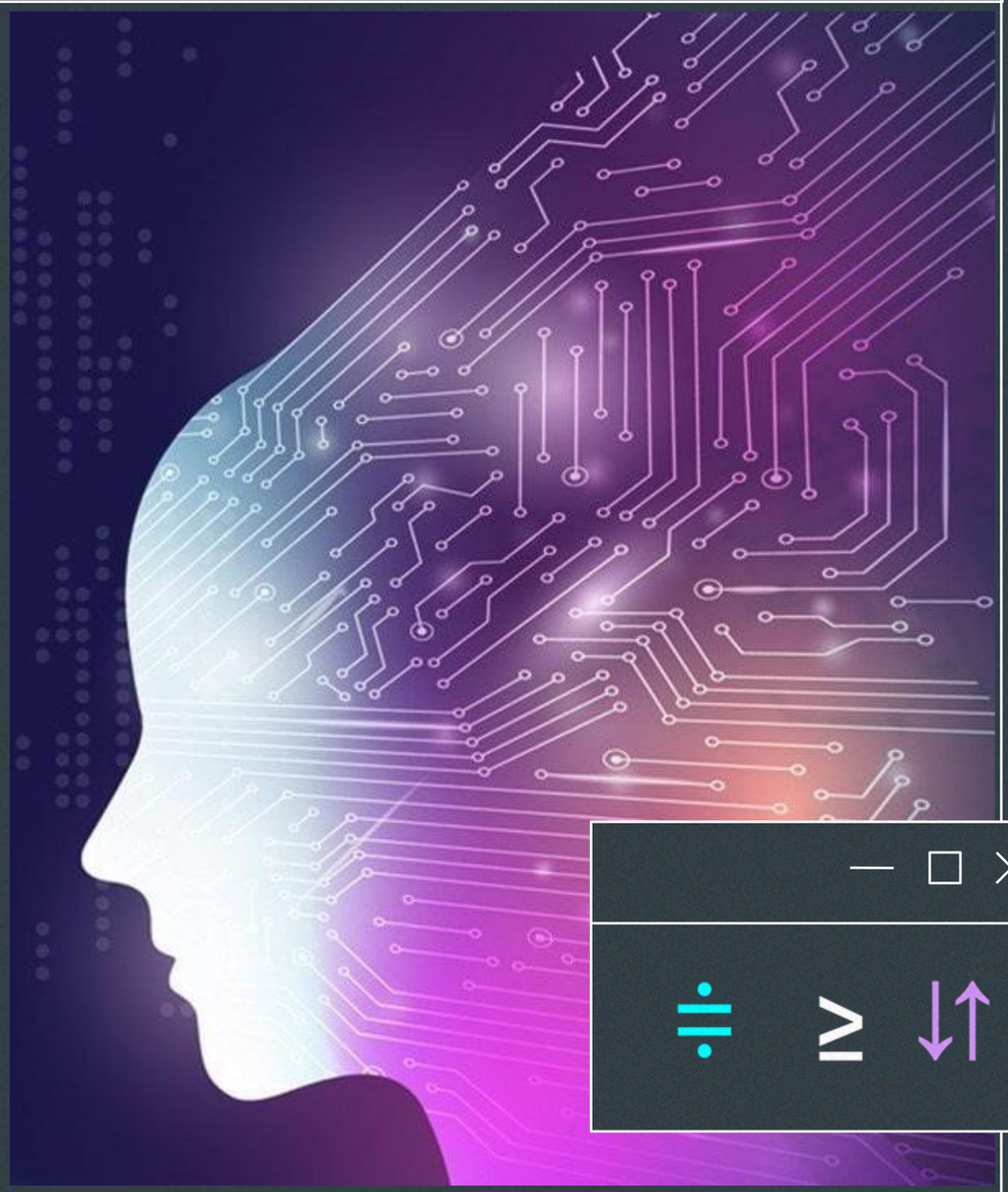
RL pentru algoritmi de publicitate online



Bejenaru Andrada (344)
Dobromirescu Mihaela (342)
Pincu Iulia (342)
Solomon Dragos (334)
Zamfirescu Alexandra (341)

Contextul și importanța utilizării RL în publicitatea online

Reinforcement Learning (RL) a devenit o tehnologie revoluționară în publicitatea online datorită capacității sale de a învăța și adapta în timp real în medii dinamice și complexe. Spre deosebire de metodele tradiționale, RL poate modela și învăța strategii de afișare a reclamelor care se bazează pe comportamentul individual al utilizatorilor și feedback-ul acestora.



Evoluția publicității digitale

Conform unui raport publicat de The Business Research Company, piața globală a publicității digitale a crescut rapid în ultimii ani, de la 734,24 miliarde USD în 2024 la 843,48 miliarde USD în 2025, cu o rată anuală de creștere (CAGR) de 14,9%.

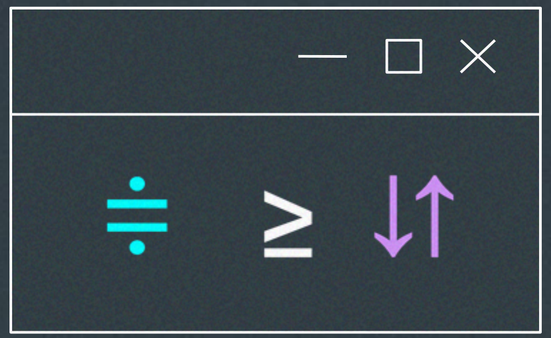
The Business Research Company

PROVOCĂRI

1. **Diversitatea utilizatorilor**
   Publicitatea online trebuie să țină cont de factori precum demografia, interesele și istoricul de navigare al utilizatorilor. RL poate personaliza experiența utilizatorilor prin afișarea reclamelor relevante.

2. **Optimizarea costurilor**
   Bugetele publicitare sunt limitate, iar RL ajută la maximizarea ROI prin afișarea reclamelor cu impact mai mare.

3. **Adaptarea în timp real**
   Comportamentul utilizatorilor poate varia în funcție de momentul zilei, context social sau economic. RL este ideal pentru medii dinamice datorită capacității de a se adapta rapid prin feedback continuu.



De ce RL este ideal pentru publicitate?

1. Exploration vs. Exploitation

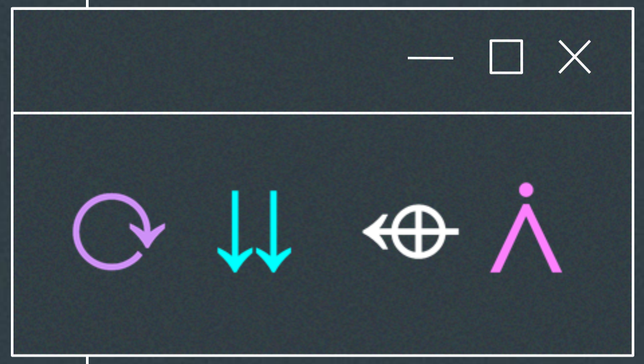
RL echilibrează explorarea noilor strategii publicitare și exploatarea celor deja cunoscute.

Exemplu: Dacă o reclamă nouă funcționează bine pentru un anumit segment, RL poate învăța să o afișeze mai des acestui public.

De ce RL este ideal pentru publicitate?

2. Învățare continuă

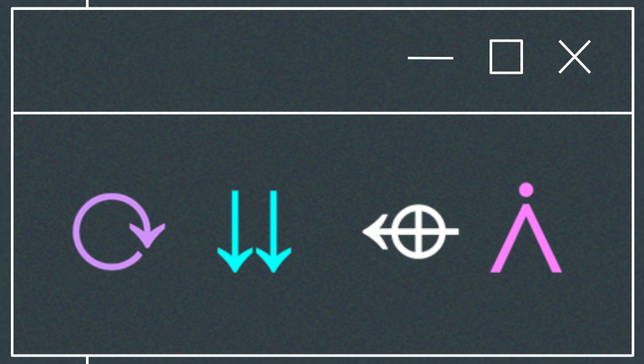
Învățare continuă Feedback-ul utilizatorilor (ex: clicuri, conversii) este folosit pentru a ajusta strategiile de afișare.

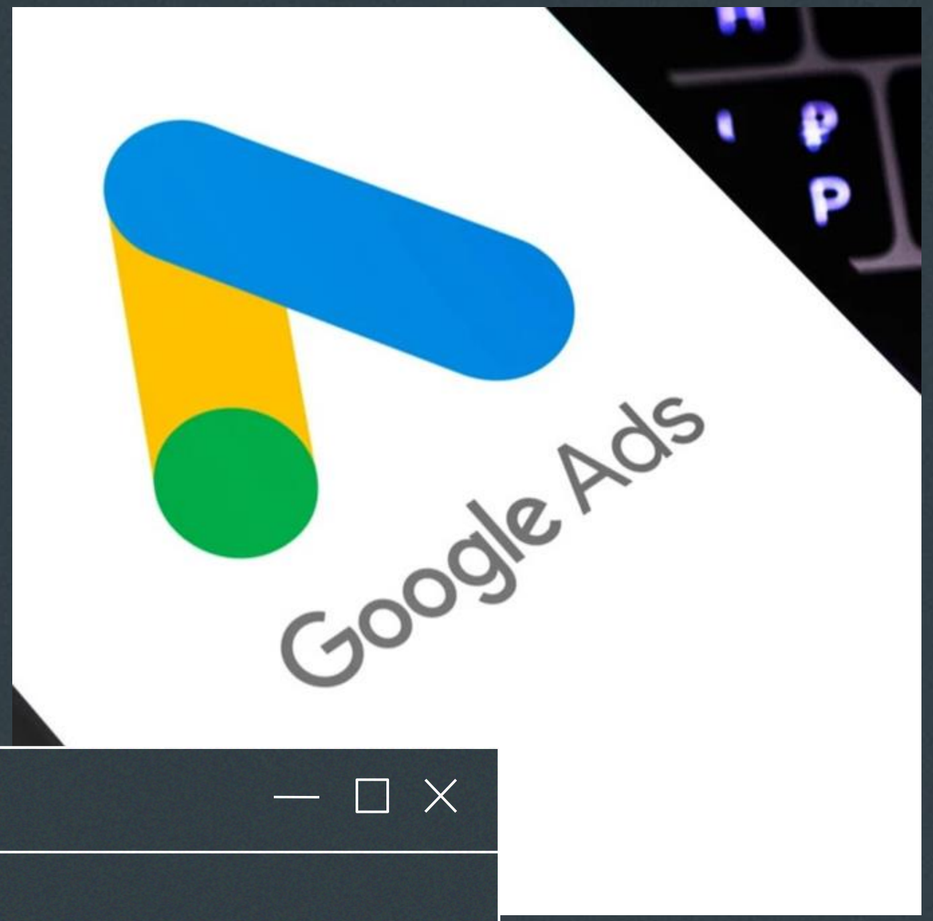


De ce RL este ideal pentru publicitate?

3. Scalabilitate

RL poate gestiona simultan milioane de utilizatori și combinații de reclame.





Impactul RL în industria publicității

1.Google Ads

Folosește RL pentru licitarea automată a reclamelor șiajustarea bugetelor.

Rezultate raportate: Creșterea CTR (Click-Through Rate) cu până la 20%.
De exemplu, un articol publicat pe platforma ClickGiant menționează că Google Ads folosește algoritmi de învățare automată pentru a analiza datele utilizatorilor și a ajusta licitațiile, ceea ce duce la creșterea ratelor de clic și, implicit, a conversiilor și a rentabilității investiției (ROI).

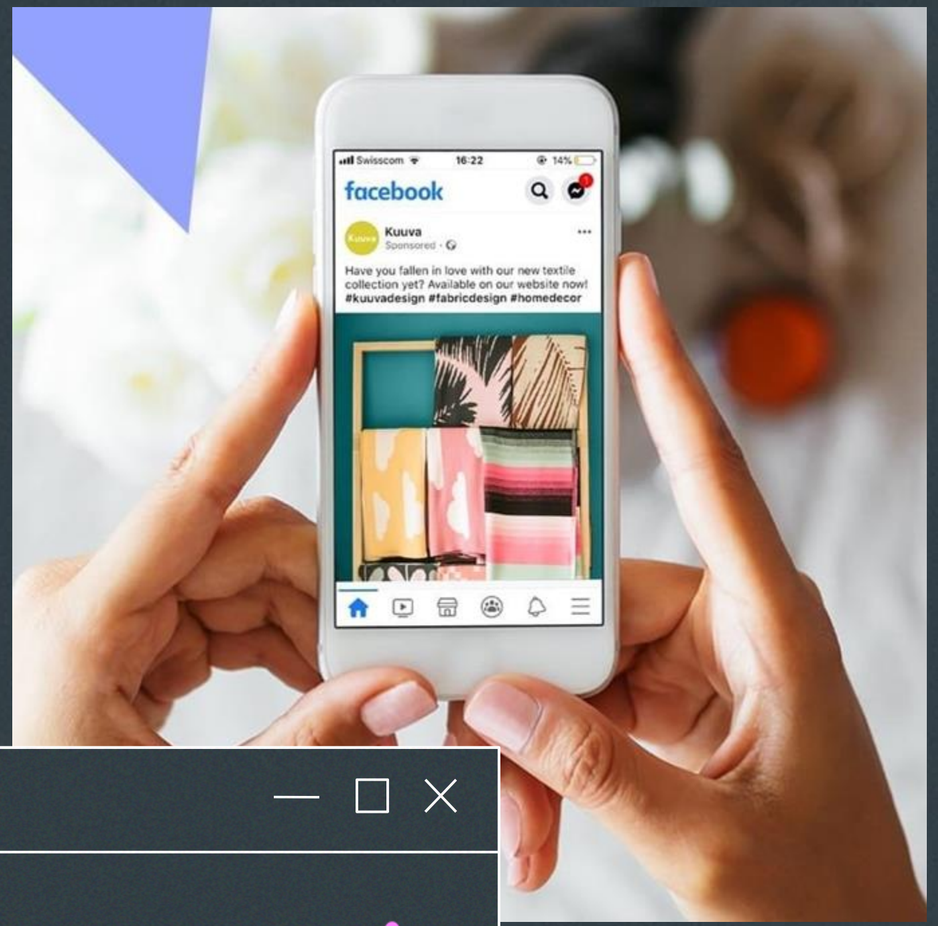
ClickGiant

Impactul RL în industria publicității

2. **Alibaba**

Integrarea RL în platformele lor a dus la o creștere semnificativă a veniturilor și a retenției utilizatorilor. Conform unui studiu, implementarea acestei tehnologii a dus la o creștere de 44,7% a performanței clicurilor în comparație cu metodele tradiționale.

Best Practice AI



Impactul RL în industria publicității

3. Facebook Ads

Utilizează RL pentru optimizarea afișării reclamelor în funcție de probabilitatea conversiei. Un aspect esențial al acestei optimizări este faza de învățare a campaniei, în care algoritmul încearcă să înțeleagă cel mai eficient mod de a livra reclamele. Pentru a ieși din această fază și a optimiza eficient livrarea, se recomandă atingerea a 50 de evenimente de conversie într-o perioadă de șapte zile.

TechBullion

Avantaje practice ale RL în publicitate

1. Creșterea CTR și ROI

Algoritmii RL pot ajusta afișarea reclamelor pentru a atrage atenția userilor care sunt mai predispuși să interacționeze. Un exemplu notabil este implementarea de către Alibaba a RL pentru optimizarea licitațiilor în timp real (RTB), în cadrul platformei lor de publicitate. Conform unui studiu, această tehnologie a dus la o creștere de 44.7 % a performanței clicurilor comparativ cu metodele tradiționale.

2. Personalizare avansată

Spre deosebire de modelele tradiționale, RL personalizează afișările la nivel individual.

3. Automatizare completă

Integrarea RL reduce nevoia intervenției manual în gestionarea campaniilor publicitare.

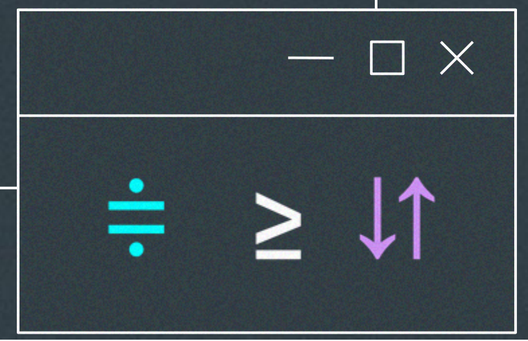
Algoritmi și tehnici relevante pentru RL în publicitatea online

Pentru optimizarea reclamelor online, Reinforcement Learning oferă o varietate de algoritmi care pot învăța strategii complexe de afișare. În acest context, vom detalia câțiva algoritmi esențiali, cu focus pe Deep Q-Network (DQN) și variantele acestuia, precum și alte tehnici relevante.

Deep Q-Network (DQN)

Definiție

DQN este o extensie a algoritmului clasic Q-learning, care folosește rețele neuronale profunde pentru a aproxima funcția Q, utilizată pentru a evalua valoarea unei acțiuni într-o stare specifică.

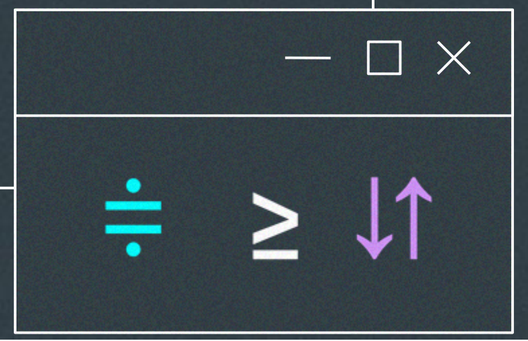


Deep Q-Network (DQN)

Aplicații în publicitatea online

- Alegerea reclamei optime pentru fiecare utilizator.
- Optimizarea ordinii reclamelor pentru a maximiza CTR (Click-Through Rate) și conversiile.

Un studiu care analizează provocările și soluțiile în implementarea algoritmului DQN: arXiv



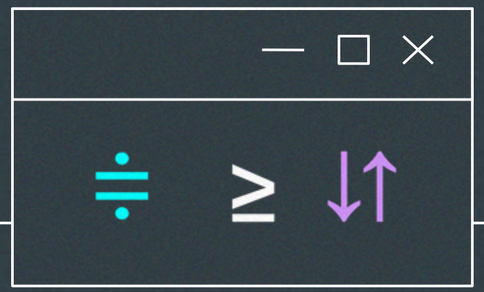
Deep Q-Network (DQN)

AVANTAJE

Scalabilitate în spații de stare mari. Capacitatea de a învăța relații complexe între stările utilizatorilor și acțiuni.

LIMITĂRI

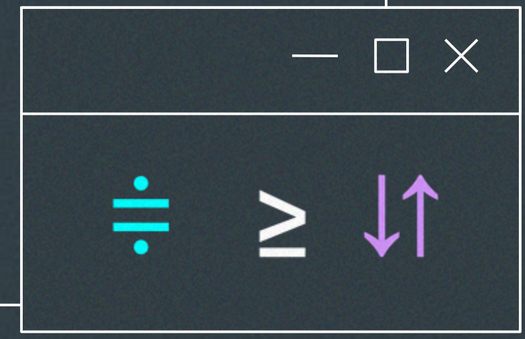
Problema supraestimării valorilor Q (abordată de Double DQN). Instabilitatea în medii foarte dinamice.



Double Deep Q-Network (Double DQN)

Definiție

Double DQN elimină tendința algoritmului DQN de a supraestima valorile Q, ceea ce duce la o alegere mai precisă a acțiunilor.



Double Deep Q-Network
(Double DQN)

Aplicații

Este ideal pentru medii unde există incertitudini mari sau variabilitate în feedback-ul utilizatorilor

Dueling Deep Q-Network (Dueling DQN)

Definiție

Dueling DQN îmbunătățește performanța algoritmului DQN prin separarea funcției Q în două componente:

- Valoarea stării **(V(s)V(s))**: cât de bună este o stare independent de acțiune.
- Avantajul acțiunii **(A(s,a)A(s,a))**: cât de bună este o acțiune relativ la alte acțiuni.

Dueling Deep Q-Network (Dueling DQN)

Aplicații

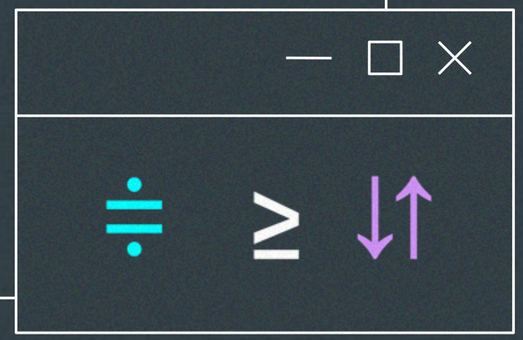
Optimizarea afișării reclamelor pentru campanii complexe.
Dueling DQN este utilizat în optimizarea afișării reclamelor pentru campanii complexe, unde este esențial să se evalueze atât valoarea stării utilizatorului, cât și avantajul diferitelor acțiuni de afișare a reclamelor.

Proceedings of Machine Learning Research

Multi-Armed Bandit Problems (MAB)

Definiție

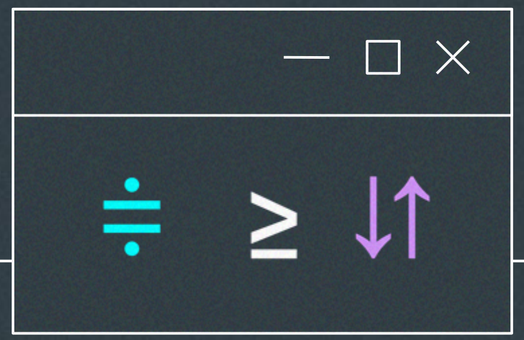
Un model simplu și eficient pentru situații în care trebuie să se aleagă între mai multe opțiuni (reclame), cu scopul de a maximiza recompensa.



Multi-Armed Bandit Problems (MAB)

Aplicații

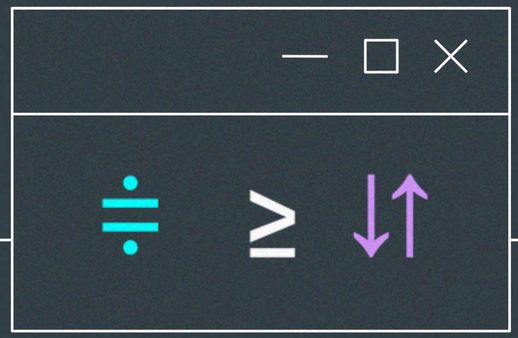
Alegerea reclamei cu cele mai mari șanse de clic bazat pe istoricul utilizatorilor.



Cross-Domain Reinforcement Learning

Definiție

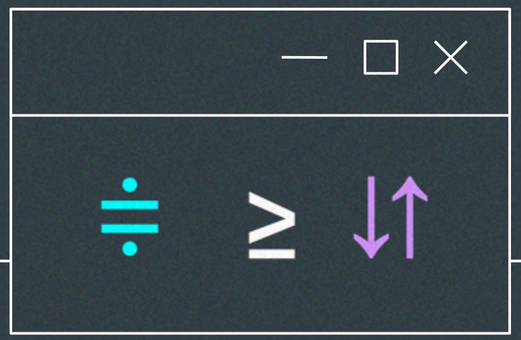
Abordare avansată care implică învățarea strategiilor RL pentru medii care împărtășesc caracteristici similare (ex: afișarea reclamelor pe diferite platforme, cum ar fi YouTube și Google Search).



Cross-Domain Reinforcement Learning

Aplicații

Campanii publicitare care implică multiple canale media.



Compararea algoritmilor

Algoritm	Avantaje	Limitări
DQN	Scalabilitate, învățare stabilă	Supraestimarea valorilor Q
Double DQN	Reducerea supraestimării	Necesită mai multă putere computațională
Dueling DQN	Separare clară între stare și acțiune	Configurare mai complexă
MAB	Simplitate, rapiditate	Limitat la probleme fără dependențe de stare
Cross-Domain RL	Învățare mai rapidă în medii noi	Necesită similaritate între medii

Studii recente

DEAR: Deep Reinforcement Learning for Online Advertising Impression in Recommender Systems

Obiectiv: Echilibrarea veniturilor din reclame cu experiența utilizatorilor pe platformele de recomandare.

Contribuție: Un model RL bazat pe DQN care decide:
- Dacă să insereze o reclamă;
- Care reclamă să fie inserată. În ce poziție să fie afișată.
- A crescut CTR cu peste 15% pe un set de date real.

Sursă: ArXiv

Studii recente

Optimizing AD Pruning of Sponsored Search with Reinforcement Learning

Obiectiv: Reducerea numărului de reclame afișate în motoarele de căutare sponsorizate pentru a îmbunătăți relevanța.

Contribuție:

- RL este utilizat pentru a selecta reclamele cele mai relevante dintr-un set mare.
- Modelul a redus costurile publicitare cu 12% și a crescut satisfacția utilizatorilor.

Sursă: ArXiv

Studii recente

Multi-Agent Reinforcement Learning for Advertising

Obiectiv: Integrarea RL într-un cadru multi-agent pentru a optimiza afișarea reclamelor pe mai multe platforme simultan.

Contribuție: Agenții RL cooperează pentru a împărți bugetul publicitar între diverse canale, maximizând veniturile totale.

Rezultate:

- Creșteri ale veniturilor publicitare cu până la 18%.
- Modelul a redus costurile publicitare cu 12% și a crescut satisfacția utilizatorilor.

Sursă: SpringerLink

IMPLEMENTARE

Pentru a demonstra aplicarea practică a algoritmilor, am dezvoltat și testat un model într-un mediu personalizat.

- Definirea mediului RL
- Alegerea algoritmilor
- Antrenare și testare
- Interpretarea rezultatelor

Definirea mediului RL

Descriere generală:

- **Tip de mediu:** Personalizat, bazat pe interacțiunea user-reclamă.

Structura mediului: (exemplu)

- **Număr de reclame:** 5
- **Număr de utilizatori:** 10
- **Caracteristici:** Fiecare utilizator și reclamă este definit printr-un vector cu 3 caracteristici.

Funcționalități principale:

- **Reset:**
  - Selectează un utilizator aleator și returnează caracteristicile acestuia ca observație inițială.
- **Step:**
  - Agentul alege o reclamă (acțiune).
  - Calculul recompensei: produsul scalar între vectorul utilizatorului și cel al reclamei.
  - Episoade de un singur pas (done=True).

Scop:

Simularea eficientă a interacțiunilor utilizator-reclamă pentru optimizarea strategiilor de afișare.

Testarea mediului RL

```
Observation: [0.3345578  0.82249363 0.18043462]
Action: 2
Reward: 0.897
Done: True
```

OUTPUT

Observație:

Utilizatorul este reprezentat prin acel array, care descrie caracteristicile acestuia (de exemplu, interese, preferințe).

Acțiune:

Agentul a ales reclama cu indexul 2 din setul disponibil. Alegerea acestei acțiuni sugerează că modelul fie a explorat, fie a exploatat politica învățată pentru a lua această decizie.

Recompensă:

Recompensa obținută (0.897) este relativ mare, ceea ce indică o potrivire bună între vectorul utilizatorului și caracteristicile reclamei selectate, sugerând o performanță pozitivă a agentului în contextul acestui utilizator specific.

Alegerea Algoritmilor

1. DQN (Deep-Q-Network)

antrenarea modelului

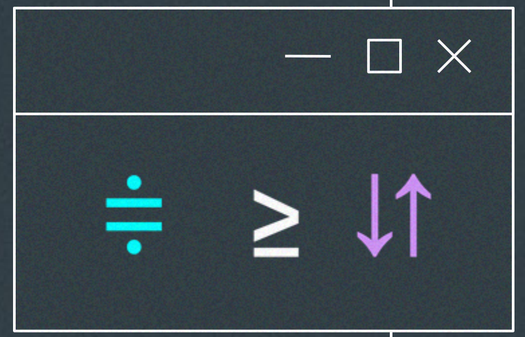
```
# vectorizarea mediului
env = make_vec_env(lambda: AdEnvironment(), n_envs=1)

# configurarea modelului DQN
model = DQN("MlpPolicy", env, verbose=1)

# antrenarea modelului pentru 10000 de pasi
model.learn(total_timesteps=10000)

# il salvam
model.save("dqn_ad_model")
print("Your model has been trained and saved successfully")
```

Alegerea Algoritmilor



DE CE AM ALES DQN?

- problema este de tip **discret** => numărul de reclame este finit, iar DQN este ideal pentru astfel de spații discrete de acțiune

- este **model-free** => DQN învață direct din interacțiunile cu mediul, fără a avea nevoie de un model al acestuia. Funcționează bine în medii simple

Interpretarea outputului în urma antrenării modelului DQN

- Modelul DQN a fost antrenat cu succes și a început să îmbunătățească performanța, reflectată în creșterea recompensei medii.
- Rata de explorare scade treptat, permițând agentului să exploateze mai eficient politica învățată.
- Loss-ul scăzut și numărul mare de actualizări arată că funcția Q este bine ajustată pentru acest mediu.
- Totuși, lungimea medie a episoadelor (**ep_len_mean = 1**) ar putea indica faptul că mediul este prea simplu sau politica învățată nu maximizează lungimea episoadelor. Ar fi utilă o analiză a naturii mediului și a obiectivelor de antrenament.

```
Using cpu device
----------------------------------
| rollout/              |          |
|    ep_len_mean        | 1        |
|    ep_rew_mean        | 0.353    |
|    exploration_rate   | 0.996    |
| time/                 |          |
|    episodes           | 4        |
|    fps                | 3816     |
|    time_elapsed       | 0        |
|    total_timesteps    | 4        |
----------------------------------
----------------------------------
| rollout/              |          |
|    ep_len_mean        | 1        |
|    ep_rew_mean        | 0.603    |
|    exploration_rate   | 0.992    |
| time/                 |          |
|    episodes           | 8        |
|    fps                | 4324     |
|    time_elapsed       | 0        |
|    total_timesteps    | 8        |
----------------------------------
----------------------------------
| rollout/              |          |
...
|    loss               | 5.31e-06 |
|    n_updates          | 2474     |
----------------------------------
Your model has been trained and saved successfully
```

Output is truncated.

testare

```python
model = DQN.load("dqn_ad_model")
env = AdEnvironment()

# testare
total_rewards = []
for _ in range(100):                                    # 100 episoade de test
    obs, info = env.reset()
    done = False
    while not done:
        action, _ = model.predict(obs, deterministic=True)
        obs, reward, done, truncated, info = env.step(action)
        total_rewards.append(reward)

# recompensa medie
avg_reward = sum(total_rewards) / len(total_rewards)
print(f"Average reward per episode using DQN: {round(avg_reward, 3)}")
```

Testare DQN

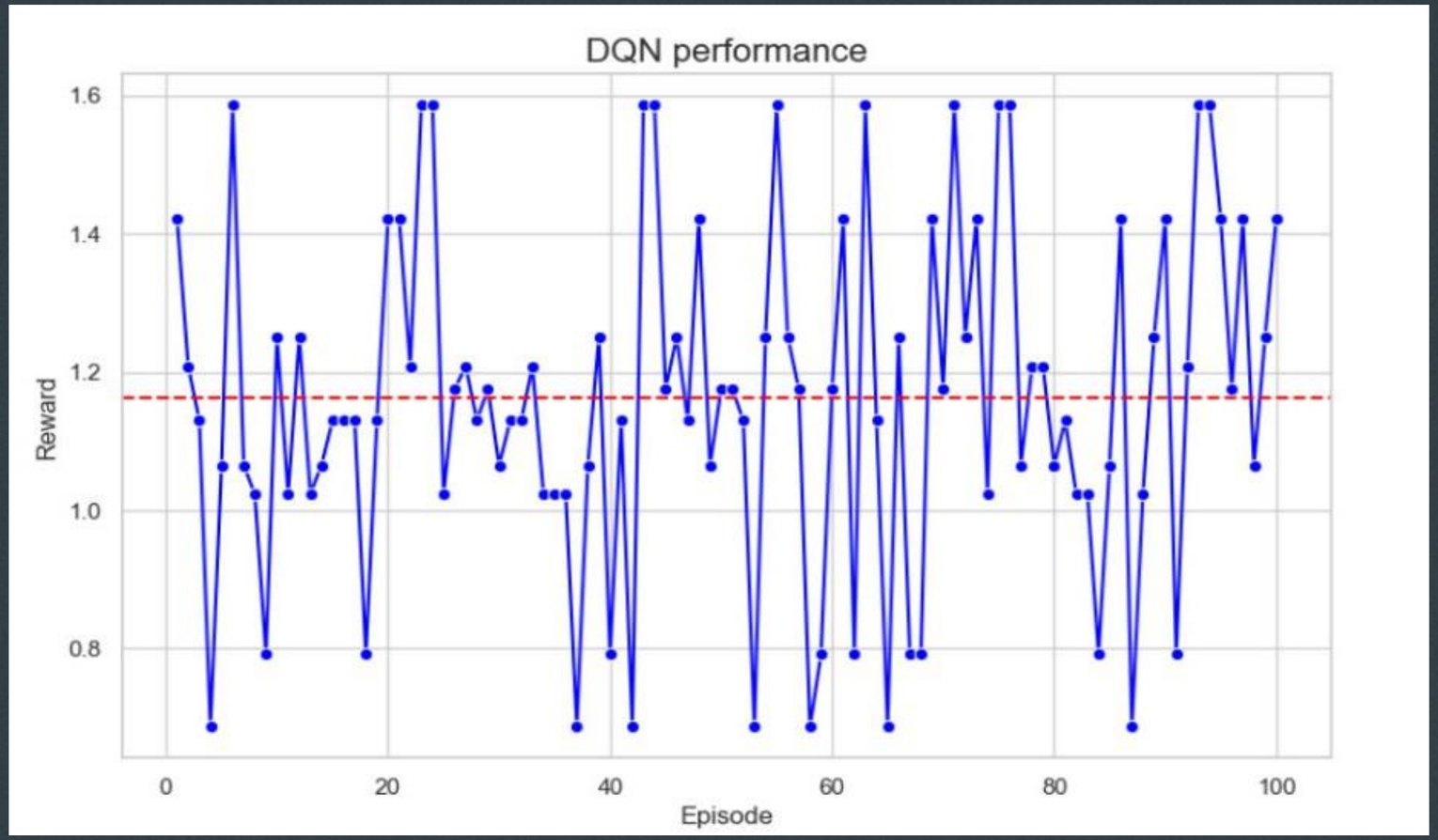
Average reward per episode using DQN: 1.162

Ce putem spune despre acest output?

- Pentru un mediu relativ simplu (cum este cel testat aici, cu doar 10 utilizatori și 5 reclame), această recompensă poate fi considerată un indicator pozitiv.
- O valoare mai mare a recompensei medii ar indica o corelare mai bună între caracteristicile utilizatorului și reclamele selectate.

- Recompensa medie de **1.162** arată că DQN funcționează bine în acest mediu, dar există potențial pentru îmbunătățirea performanței, mai ales dacă mediul devine mai complex sau este comparat cu alte metode mai avansate.

Plotare DQN



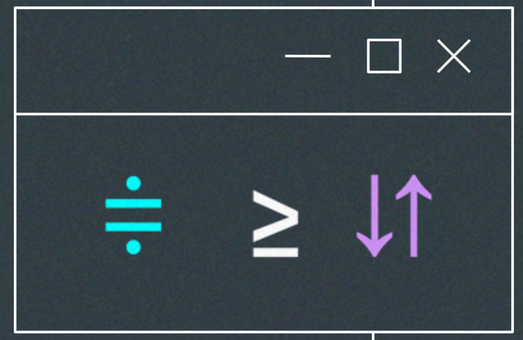
Interpretare Rezultate

Modelul DQN a fost testat pentru 100 de episoade.

Oscilațiile reward-urilor arată că agentul ia decizii bune în majoritatea cazurilor.

Agentul DQN a învățat o politică funcțională cu o recompensă medie decentă. Totuși, variabilitatea recompenselor indică faptul că performanța ar putea fi îmbunătățită prin:

- Ajustarea hiperparametrilor (de ex., rata de explorare ε).
- Utilizarea unui mecanism precum **Double DQN** sau **Dueling DQN** pentru reducerea supraestimărilor.
- Mai multe episoade de antrenament pentru o convergență mai stabilă.



Alegerea Algoritmilor

2. PPO (Proximal Policy Optimization)

antrenarea modelului

```
# vectorizarea mediului
env = make_vec_env(lambda: AdEnvironment(), n_envs=1)

# configurarea modelului PPO
ppo_model = PPO("MlpPolicy", env, verbose=1)

# antrenarea modelului
ppo_model.learn(total_timesteps=10000)

# save
ppo_model.save("ppo_ad_model")
print("Your model has been trained and saved successfully")
```

Alegerea Algoritmilor

DE CE AM ALES PPO?

- este un algoritm robust și eficient pentru problemele de RL
- este de tip policy-based, ceea ce înseamnă că încearcă să învețe direct o politică optimă, spre deosebire de DQN care învață o funcție de valoare
- stabilește limite clare pentru actualizări ale politicii, prevenind schimbările mari care pot destabiliza învățatrea
- este mai potrivit pentru medii **complexe** și **continuu-dinamice**

Interpretarea outputului în urma antrenării modelului PPO

- Modelul PPO a fost antrenat cu succes, iar performanța sa s-a îmbunătățit, reflectată în creșterea recompensei medii pe episod.
- Explorarea rămâne activă datorită entropiei ridicate, permițând agentului să învețe o politică mai robustă.
- Pierderile reduse și respectarea constrângerilor de proximitate (**approx_kl** scăzut) indică o politică stabilă și ajustări controlate.
- Totuși, lungimea medie a episoadelor (**ep_len_mean = 1**) sugerează că mediul este simplu sau că interacțiunile ar putea fi extinse pentru o complexitate mai mare. O analiză suplimentară ar putea evidenția noi direcții de optimizare.

```
Using cpu device
----------------------------------
| rollout/            |          |
|    ep_len_mean      | 1        |
|    ep_rew_mean      | 0.662    |
| time/               |          |
|    fps              | 6541     |
|    iterations       | 1        |
|    time_elapsed     | 0        |
|    total_timesteps  | 2048     |
----------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 1           |
|    ep_rew_mean          | 0.754       |
| time/                   |             |
|    fps                  | 4443        |
|    iterations           | 2           |
|    time_elapsed         | 0           |
|    total_timesteps      | 4096        |
| train/                  |             |
|    approx_kl            | 0.050017573 |
|    clip_fraction        | 0.625       |
|    clip_range           | 0.2         |
|    entropy_loss         | -1.57       |
...
|    policy_gradient_loss | -0.15       |
|    value_loss           | 0.0661      |
-----------------------------------------
Your model has been trained and saved successfully
```

testare

```python
# incarcarea modelului PPO
ppo_model = PPO.load("ppo_ad_model")
env = AdEnvironment()

# testare
ppo_rewards = []
for _ in range(100):
    obs, info = env.reset()
    done = False
    while not done:
        action, _ = ppo_model.predict(obs, deterministic=True)
        obs, reward, done, truncated, info = env.step(action)
        ppo_rewards.append(reward)

# average reward
ppo_avg_reward = sum(ppo_rewards) / len(ppo_rewards)
print(f"Average reward per episode PPO: {round(ppo_avg_reward, 3)}")
```

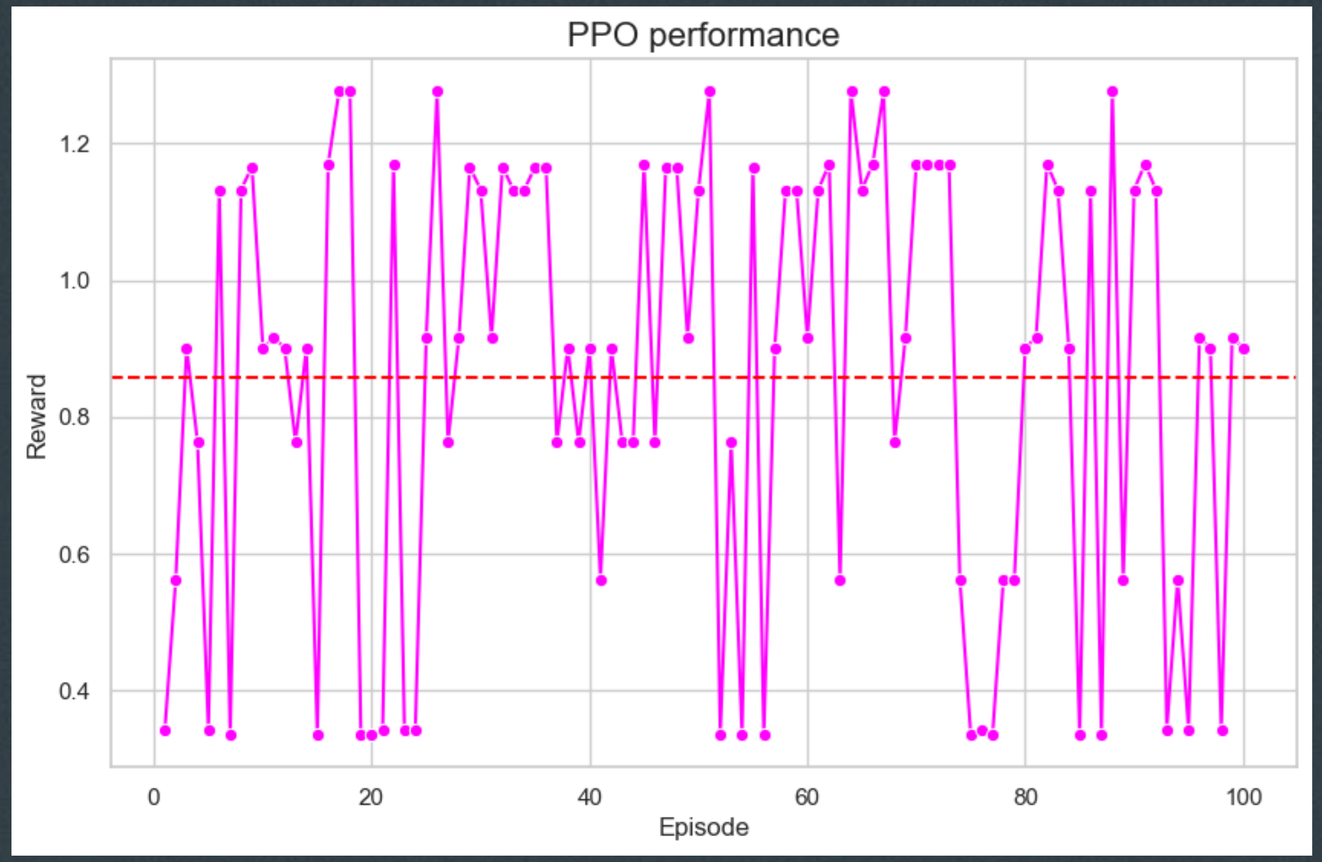
Testare PPO

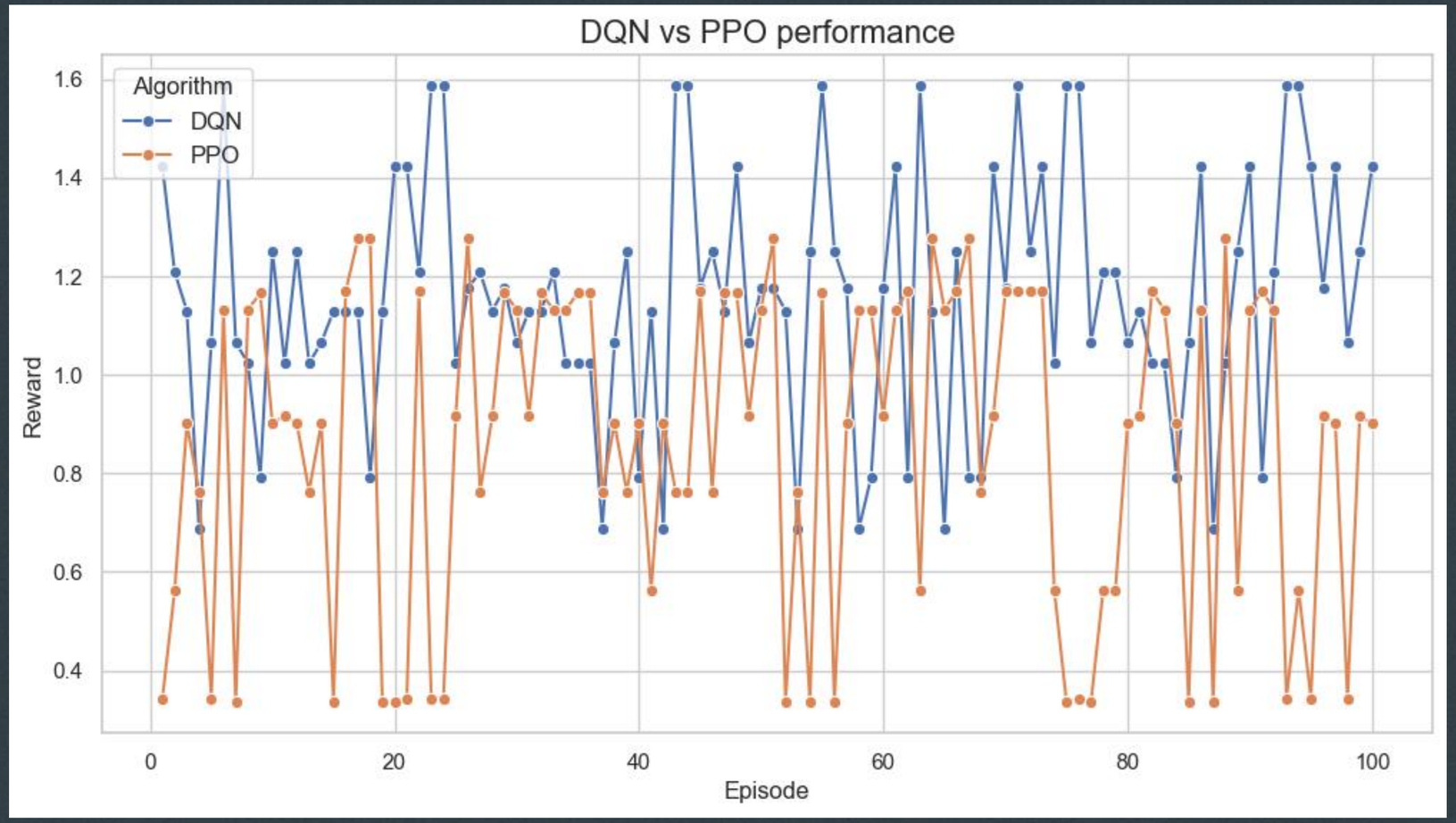
Average reward per episode PPO: 0.859

Ce putem spune despre acest output?

- Este o valoare mai mică decât recompensa obținută cu DQN (**1.162**), ceea ce sugerează că PPO poate fi mai puțin performant în acest mediu simplu sau că necesită ajustări suplimentare.
- O recompensă medie de **0.859** indică faptul că agentul PPO a învățat o politică rezonabilă, dar nu a reușit să maximizeze complet potențialul recompenselor în acest mediu.
- Această diferență ar putea fi explicată prin caracteristicile mediului sau prin necesitatea unor ajustări suplimentare pentru PPO. Analiza detaliată a rezultatelor și ajustarea hiperparametrilor sunt recomandate pentru a maximiza performanța acestui algoritm.

Plotare PPO




DQN vs PPO performance
Algorithm
DQN
PPO
Reward
Episode
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
20
40
60
80
100

DQN vs PPQ

DQN vs PPO

Avantajele PPO în medii dinamice:

- Robusteză superioară față de DQN:
  - Adaptare mai bună la medii dinamice datorită actualizării eficiente a politicii.
  - Mai puțin predispus la supraestimarea recompenselor.

Rezultate comparative între PPO și DQN:

- Dacă recompensa medie **PPO > DQN**:
  → PPO demonstrează o politică mai eficientă de afișare a reclamelor.
- Dacă recompensa medie **PPO ≈ DQN** sau **PPO < DQN**:
  → PPO nu aduce beneficii semnificative în acest mediu.
  → **DQN**, fiind mai simplu, este suficient pentru învățarea unei politici optime.