Control prin învăţare

Master ICAF, An 1 Sem 2

Lucian Buşoniu, Ştefan Pîrje

Partea VIII

Deep Reinforcement Learning

Partea VI în plan

- Problema de învăţare prin recompensă
- Soluţia optimală
- Programarea dinamică exactă
- Învăţarea prin recompensă exactă
- Tehnici de aproximare
- Programarea dinamică cu aproximare (var. continue)
- Învăţarea prin recompensă cu aproximare (var. continue)
- Reţele neurale profunde
- **Învăţarea prin recompensă cu reţele neurale profunde**

Partea VI în plan: Categorii de algoritmi

După utilizarea unui model:

- Bazat pe model: f, ρ cunoscute
- Fără model: doar date (învăţarea prin recompensă)

După nivelul de interacţiune:

- Offline: algoritmul rulează în avans
- Online: algoritmul controlează direct sistemul

Exact vs. cu aproximare:

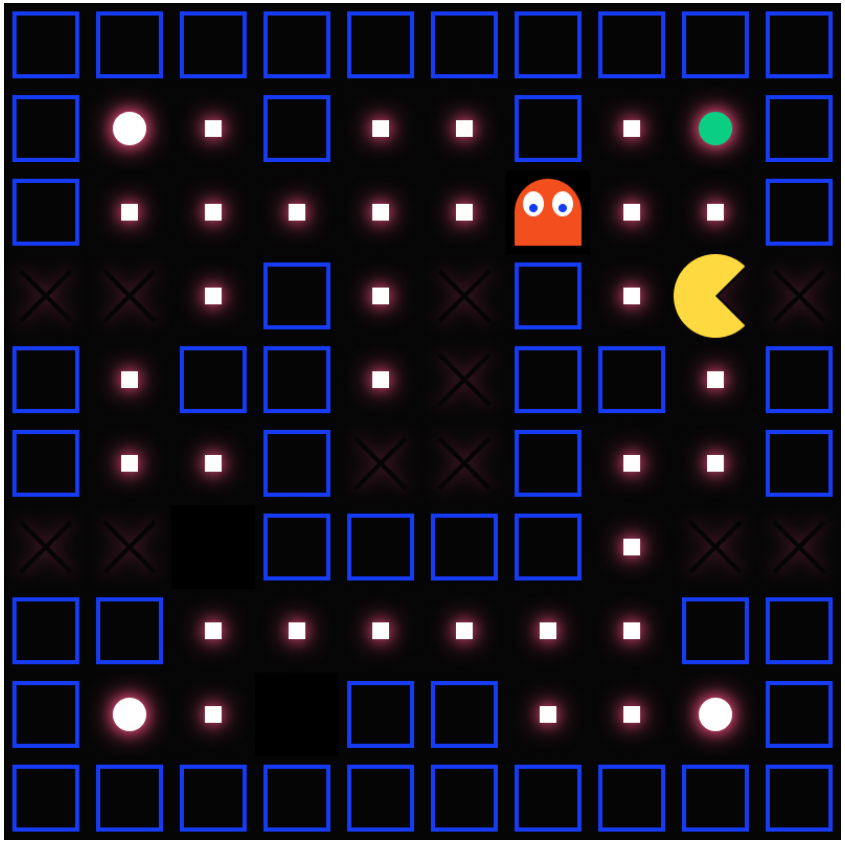
- Exact: x, u număr mic de valori discrete
- Cu aproximare: x, u continue (sau multe valori discrete)

De ce reţele neurale în învăţarea prin recompensă?

Motivaţie

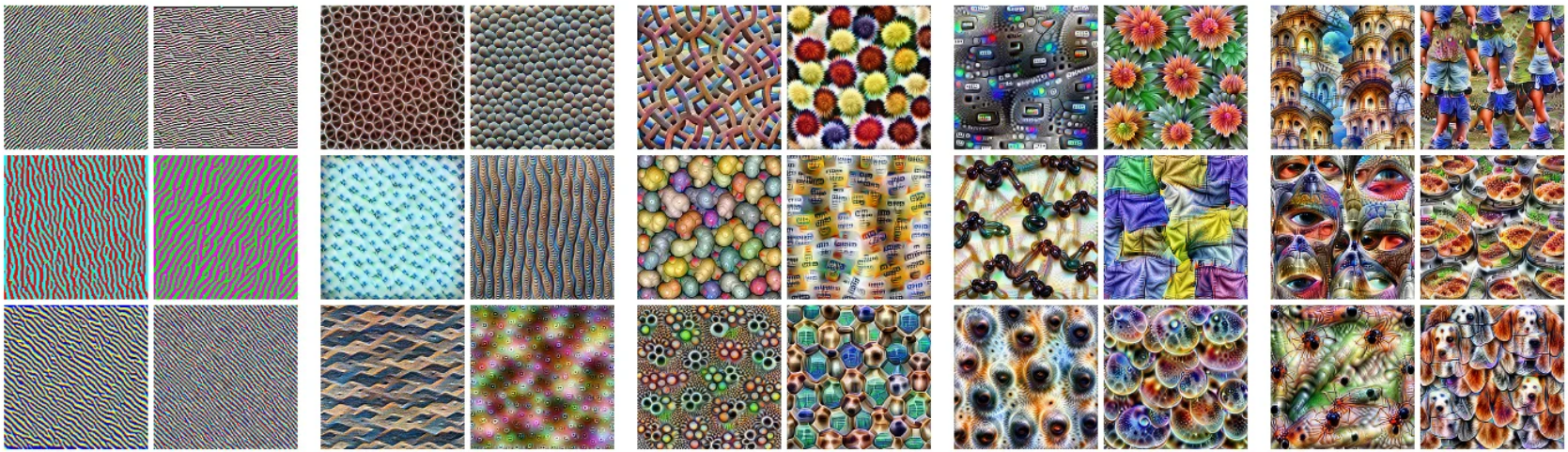
Chiar având acces la starea internă a jocului, ar trebui să extragem diverse atribute:

- distanţele până la fantome
- distanţa până la cel mai apropiat power-up
- distanţa până la pereţi
- dacă Pac-Man este blocat
- distanţa până la cele mai apropiate puncte
- puncte deja consumate



Aceste atribute sunt relevante doar pentru MDP-ul curent!

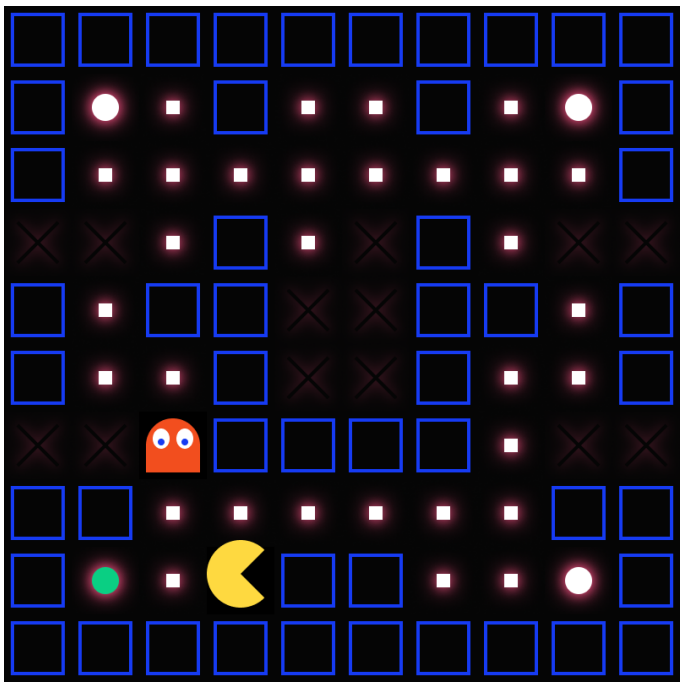
Reţele convoluţionale: atribute automate!



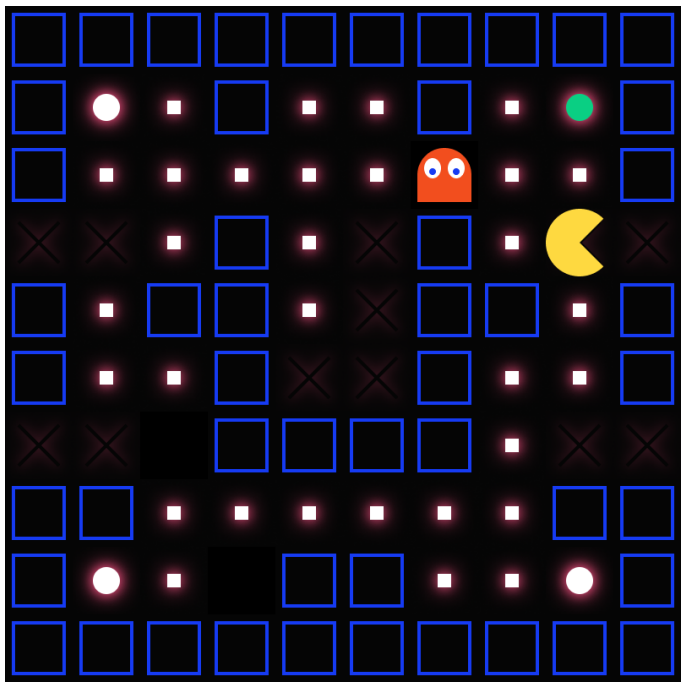
Reprezentări ierarhice învăţate de o reţea convoluţională dintr-un set de date cu imagini naturale

Generalizare

Dorim Q antrenată pe x ...



... să fie similară pentru alt x



Reamintim: Iteraţia Q bazată pe date

Iteraţia Q bazată pe date

obţine sau colectează setul de date
$\mathcal{D} = \{(x_s, u_s, r_s, x'_s), s = 1, \ldots, n_s\}$
repeat la fiecare iteraţie ℓ
  for $s = 1, \ldots, n_s$ **do**
    calculează ţinta cu bootstrap pentru $\widehat{Q}(x_s, u_s; \theta)$:
    $\hat{R}_s \leftarrow r_s + \gamma \max_{u'} \widehat{Q}(x'_s, u'; \theta_\ell)$
  end for
  $\theta_{\ell+1} \leftarrow \arg\min_\theta \sum_{s=1}^{n_s} \left[\hat{R}_s - \widehat{Q}(x_s, u_s; \theta)\right]^2 =: \mathcal{L}(\theta)$
until terminare

$\mathcal{L}$ **funcţie de pierdere**

Iteraţia Q neurală bazată pe date

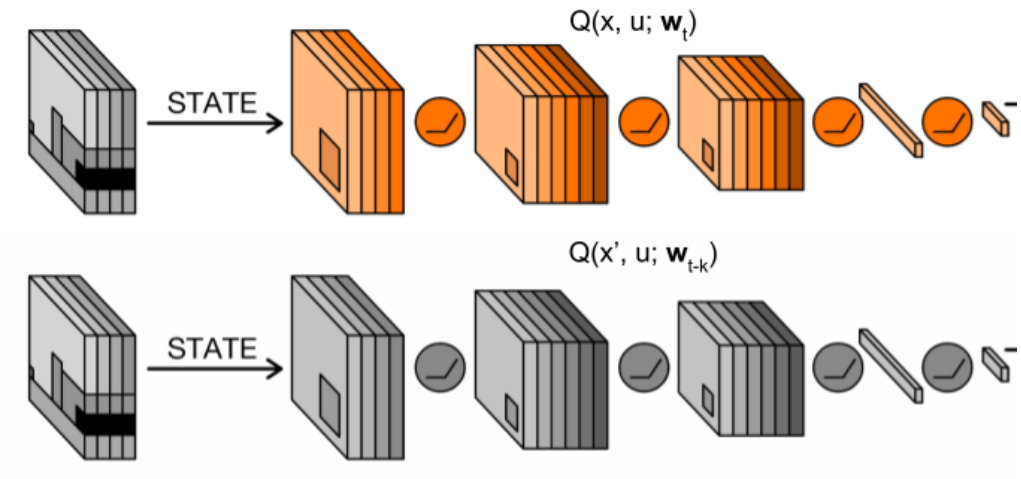
Utilizează NN pentru $\widehat{Q}(x, u; \theta)$ şi SGD pentru optimizare

Iteraţia Q neurală bazată pe date

iniţializează $\widehat{Q}(x, u; \theta)$ a.î. $Q(x, u; \theta) \approx 0, \forall x, u$
obţine sau colectează setul de date
$\mathcal{D} = \{(x_s, u_s, r_s, x'_s), s = 1, \ldots, n_s\}$
repeat la fiecare iteraţie ℓ
  for $s = 1, \ldots, n_s$ **do**
    calculează ţinta cu bootstrap pentru $\widehat{Q}(x_s, u_s; \theta)$:
    $\hat{R}_s \leftarrow r_s + \gamma \max_{u'} \widehat{Q}(x'_s, u'; \theta_\ell)$
  end for
  rezolvă $\theta_{\ell+1} = \arg\min_\theta \mathcal{L}(\theta)$ cu SGD pe batch-uri de b tranziţii din $\mathcal{D}$
until terminare

Problemă: actualizarea lui θ modifică ţintele într-un mod corelat, ducând la acumularea erorilor.

Deep Q-Networks. Fix 1: Reţea ţintă



DQN introduce o reţea suplimentară pentru calcularea ţintei de regresie, care “urmăreşte” reţeaua online

Această **reţea ţintă** nu este antrenată separat – parametrii săi sunt copiaţi din reţeaua online la fiecare C paşi

Deep Q-Networks. Fix 2: Reluarea experienţei

DQN utilizează un buffer ciclic de reluare a experienţei, care:

- permite antrenarea aproximatoarelor cu gradient stohastic descendent pe mini-batch-uri (ca în NFQI)
- permite îmbunătăţiri rapide, optimiste ale legii de control fără a aştepta convergenţa NN

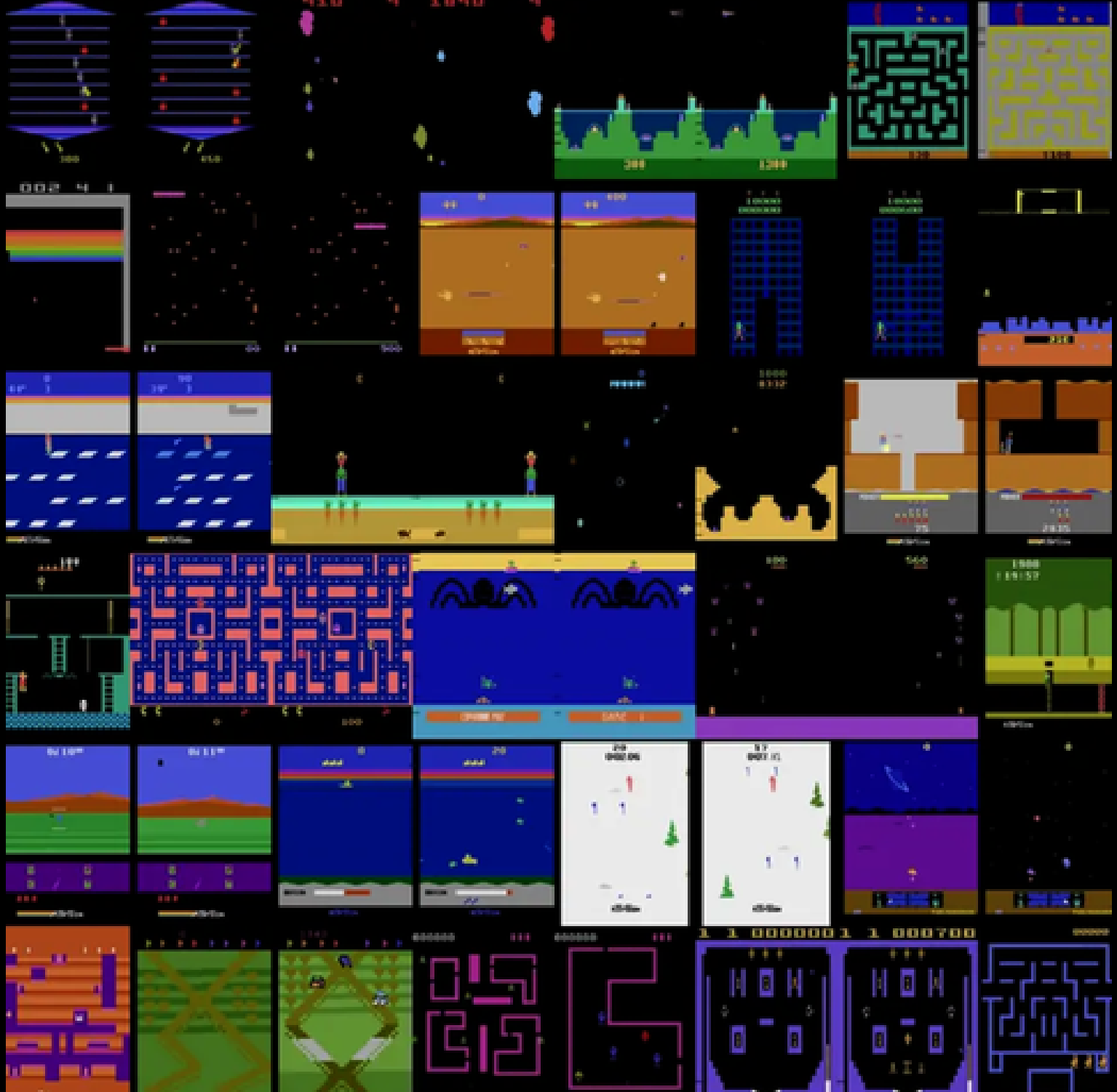
Alte ajustări importante

- recompensele sunt limitate la $[-1, 1]$
- un optimizator bun, precum Adam
- funcţia de pierdere este Huber/Smooth L1 (L2 pătratic aproape de 0, L1 liniar pentru valori mari)
- în unele implementări se limitează şi norma gradientului

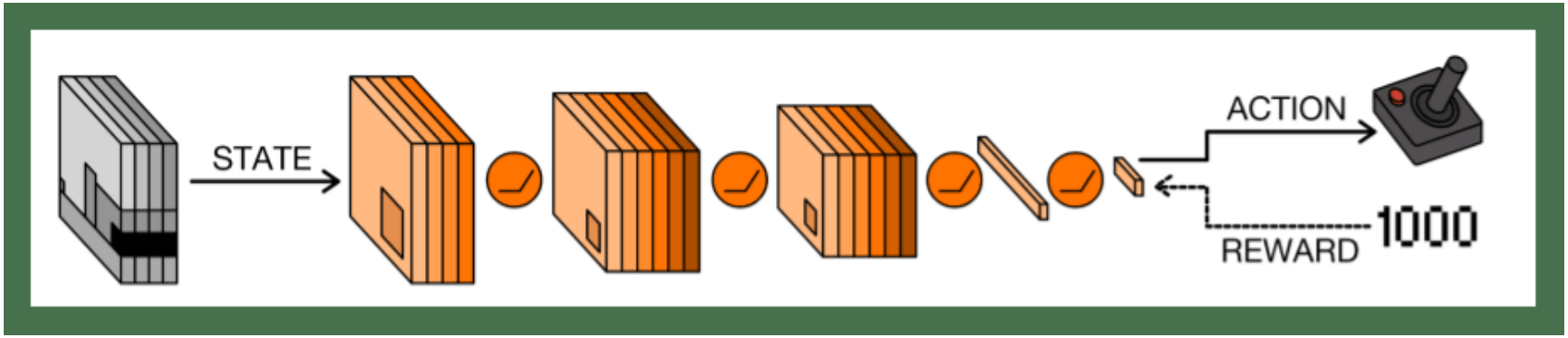
Algoritm

Deep Q-Networks

init reţeaua Q $\widehat{Q}(x, u; \theta)$ cu parametri aleatori θ
init reţeaua Q ţintă $\bar{Q}(x, u; \bar{\theta})$ cu aceiaşi parametri $\bar{\theta} = \theta$
init buffer-ul $\mathcal{D}$ cu capacitatea n_s
for fiecare traiectorie **do**
    init x_0
    repeat la fiecare pas k
        alege u_k, de ex., ε-greedy din $Q(x_k, \cdot; \theta)$
        aplică u_k, măsoară x_{k+1}, primeşte r_{k+1}
        adaugă (x_k, u_k, r_k, x_{k+1}) în $\mathcal{D}$; posibil eliminând eş. vechi
        eşantionează un batch de b tranziţii (x_s, u_s, r_s, x'_s) din $\mathcal{D}$
        calculează ţintele $\hat{R}_s = r_s + \gamma \max_{u'} \bar{Q}(x'_s, u'; \bar{\theta})$
        optimizează $\sum_{s=1}^{b} \left[\hat{R}_s - \widehat{Q}(x_s, u_s; \theta)\right]^2$ faţă de θ
        la fiecare C paşi: $\bar{\theta} = \theta$
    until traiectoria terminată
end for

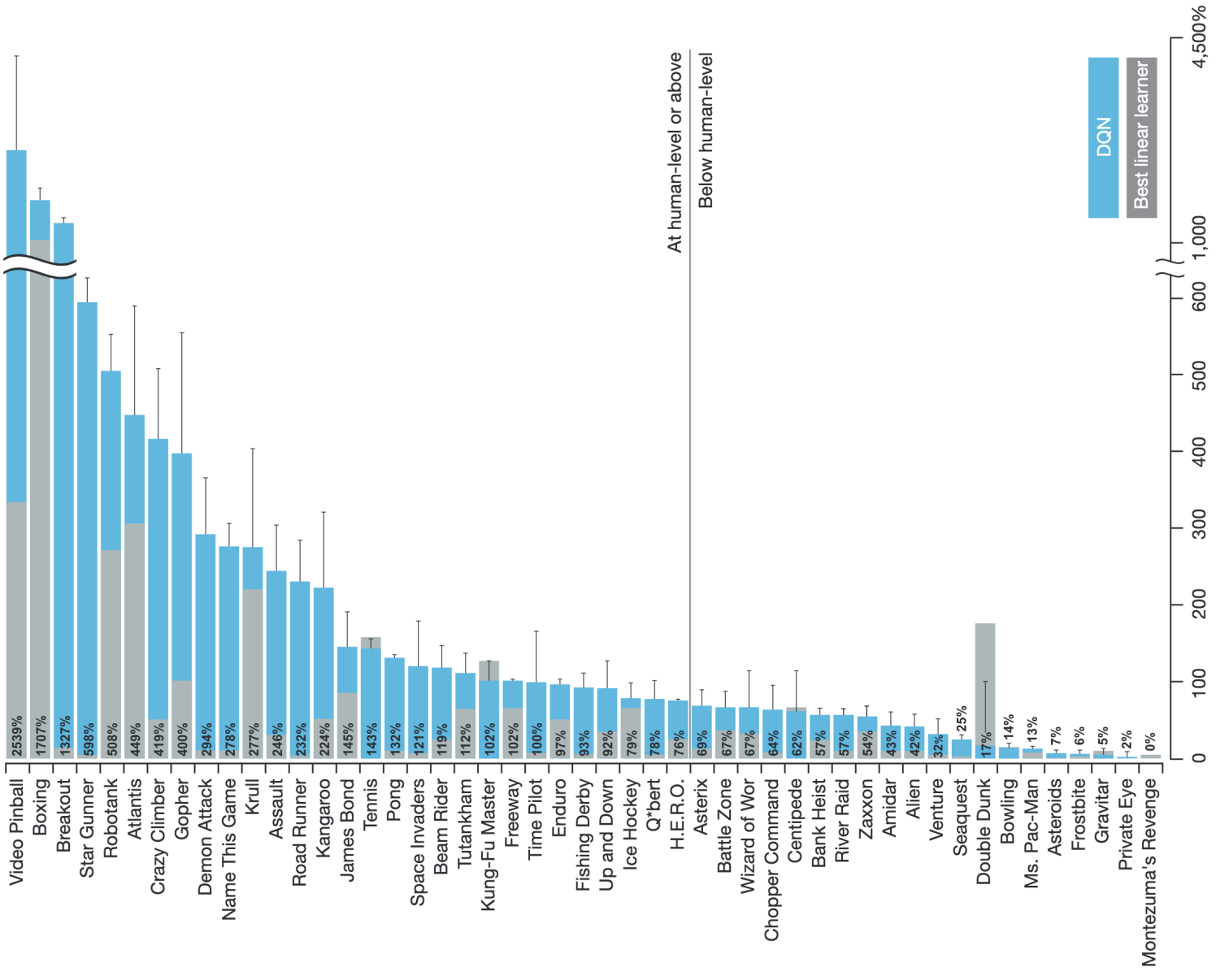


Arhitectura DQN pentru Atari



- primeşte ultimele 4 ecrane de joc în tonuri de gri
- trei straturi convoluţionale ascunse
- un strat liniar ascuns
- un strat liniar de ieşire (uneori cu un bias comun pentru toate acţiunile)
- câte o ieşire pentru valoarea Q a fiecărei acţiune
- activări RELU

Performanţa DQN relativ la un jucător uman



Dincolo de Deep Q-Networks

Multe direcţii pentru îmbunătăţirea DQN:

- obiective: Double-DQN, Dueling DQN, Munchausen-DQN, TD cu return pe n paşi
- distribuţii în loc de estimări punctuale: Categorical DQN, IQM
- eşantionare: Prioritized Experience Replay
- explorare: Random Network Distillation, Go-Explore, Bootstrapped DQN

Conexiuni: Control optimal

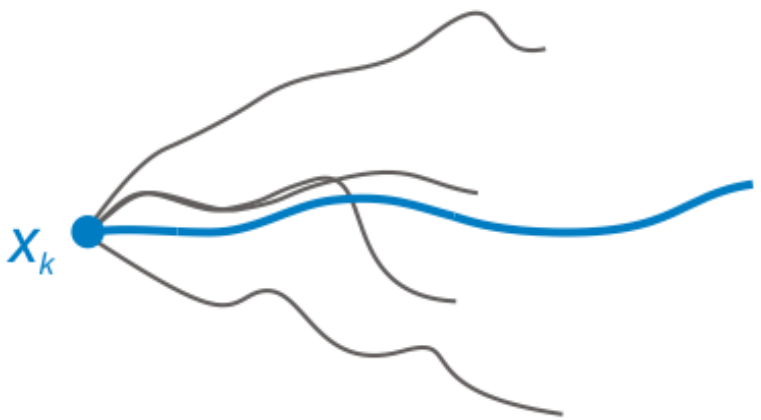
- Controlează un sistem minimizând costul J
- Bazat pe model
- Posibil în timp continuu, orizont finit

RL, DP **sunt control optimal**!

- Controlează un sistem maximizând returnul $V^h(x)$
- În timp discret, orizont infinit
- RL **fără model**, bazat pe date/interacţiune

Conexiuni: Control predictiv

- MPC: model-predictive control
- Variantă de control optimal, bazată pe model
- Principiu de bază: **receding horizon** (orizont alunecător)



Probleme deschise

RL & DP **în curs de dezvoltare**

Probleme deschise:

- Strategii de explorare
- Garanţii de siguranţă şi stabilitate
- Stări care nu pot fi măsurate
- Sisteme multiagent
- Transfer, învăţarea multi-task, învăţarea continuă