Universitatea POLITEHNICA din București
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
060042 București, Splaiul Independenței, nr. 313, sector 6

LUCRARE DE DISERTAȚIE

RAPORT DE CERCETARE – ANUL II SEMESTRUL I

LOAD BALANCING NEURAL NETWORK: O ABORDARE BAZATĂ PE DEEP REINFORCEMENT LEARNING PENTRU OPTIMIZAREA APLICAȚIILOR WEB

Autor: Ștefan PATRICHE

Conducători științifici:

Profesor Dr. Ing. Mihai DASCĂLU

Conf. Dr. Ing. Ștefan RUȘETI

București

Februarie 2026

Cuprins

Lista Tabelelor

Lista Figurilor

Lista Abrevierilor

Abreviere	Denumire	Traducere
LBNN	Load Balancing Neural Network	N/A
DRL	Deep Reinforcement Learning	Învățare Profundă prin Întărire
UtilStd	Utilization Standard Deviation	Abaterea Standard a Utilizării
DQN	Deep Q-Network	N/A
ReLU	Rectified Linear Unit	Unitate Liniară Rectificată
SGD	Stochastic Gradient Descent	N/A
CPU	Central Processing Unit	Unitate Centrală de Procesare
LSTM	Long Short-Term Memory	Memorie pe Termen Lung și Scurt
PPO	Proximal Policy Optimization	N/A

SAC	Soft Actor-Critic	N/A
A3C	Asynchronous Advantage Actor-Critic	N/A

1 INTRODUCERE ȘI CONTEXT

Acest raport prezintă progresul realizat în implementarea și antrenarea unui agent bazat pe Deep Reinforcement Learning pentru Load Balancing Neural Network (LBNN). Având infrastructura validată în semestrul anterior, am trecut la faza de implementare a inteligenței artificiale capabile să optimizeze distribuirea sarcinilor în aplicații web.

1.1 Recapitulare: Fundamentul din Anul I Semestrul II

În semestrul al doilea am construit fundamentul tehnic complet:

- **Infrastructură distribuită:** sistem containerizat Docker cu agent, 3 servere eterogene, generator de trafic, trainer și monitorizare.
- **Sistem de tick-uri:** simulare deterministă fără dependențe hardware.
- **Recompense cu gradient:** contribuție majoră - funcție invariantă la plafonul maxim de încărcare, dar care redă posibilitatea de a avea un gradient de valori în funcție de cât de optimă a fost decizia.
- **Flux de date automat:** coordonare episoade și colectare date.
- **Referință de bază – Round Robin:** ~40% decizii optime.

1.2 Obiectivele Anului II semestrul I

Cu infrastructura validată și funcțională, această etapă s-a concentrat pe trei obiective majore care transformă sistemul de la o platformă de simulare pasivă la un sistem inteligent capabil de optimizare autonomă:

1. **Primul obiectiv** a constat în implementarea interfeței între mediul de simulare și modelul de învățare. Deși infrastructura dezvoltată în semestrul anterior oferă toate componentele necesare (agenți, servere, generare trafic, sisteme de monitorizare), aceasta nu era pregătită pentru interacțiunea directă cu algoritmi de Deep Reinforcement Learning. A fost necesară crearea unui Wrapper Gymnasium care să traducă stările sistemului distribuit în reprezentări vectoriale normalizate, să transforme acțiunile modelului în comenzi HTTP concrete și să sincronizeze perfect ciclul de antrenare cu sistemul bazat pe tick-uri.
2. **Al doilea obiectiv** a vizat implementarea și antrenarea efectivă a agentului DRL. Aceasta a inclus: proiectarea arhitecturii rețelei neuronale (dimensiuni intrări/ieșiri, numărul și mărimea straturilor ascunse), implementarea algoritmului

DQN cu toate mecanismele sale de stabilizare (Experience Replay, Target Network, Gradient Clipping), configurarea strategiei de explorare epsilon-greedy, și definirea curriculum-ului de antrenare cu creștere progresivă a complexității. Antrenarea efectivă presupune rularea a minimum 1000 de episoade pentru a permite convergența modelului.

3. **Al treilea obiectiv**, cel mai important din perspectiva validării cercetării, a fost obținerea de rezultate care să demonstreze superioritatea clară a abordării DRL față de algoritmii tradiționali simpli de load balancing. Ținta concretă stabilită: depășirea pragului de 80% decizii optime, comparativ cu baseline-ul Round Robin de 40%. Această îmbunătățire de minimum 100% ar valida atât designul sistemului de recompense cu gradient, cât și viabilitatea abordării agent ca load balancer direct.

Împreună, aceste trei obiective marchează tranziția fundamentală de la infrastructură la inteligență, de la simulare la optimizare autonomă, de la concept teoretic la validare experimentală.

2 IMPLEMENTAREA AGENTULUI DRL

2.1 Arhitectura Rețelei Neuronale

Componentă	Specificație
Input Layer	12 neuroni (stare normalizată sistem)
Hidden Layer 1	64 neuroni + ReLU
Hidden Layer 2	64 neuroni + ReLU
Hidden Layer 3	32 neuroni + ReLU
Output Layer	3 neuroni (Q-values pentru fiecare server)
Tabel 1 - Specificații Arhitectură Neuronală	

2.1.1 Spațiul de stări (12 dimensiuni)

Starea sistemului este reprezentată printr-un vector de 12 valori normalizate în intervalul [0, 1]:

- Starea serverelor (9 dimensiuni): Pentru fiecare dintre cele 3 servere - utilizare CPU, utilizare memorie, și număr de conexiuni active.
- Caracteristicile cererii curente (3 dimensiuni): Cost CPU estimat, cost memorie estimat, și durata preconizată de procesare.

2.1.2 Spațiul de acțiuni (3 dimensiuni)

Agentul alege dintr-un set de acțiuni discrete, fiecare corespunzând rutării cererii către unul dintre cele trei servere disponibile.

Acțiunea 0 reprezintă Server-1, acțiunea 1 reprezintă Server-2, iar acțiunea 2 reprezintă Server-3.

2.2 Algoritmul DQN și Mecanisme de Stabilizare

Am implementat următoarele mecanisme pentru stabilitatea antrenării:

- **Experience Replay Buffer:** Stochează ultimele 10,000 tranziții (stare, acțiune, recompensă, stare următoare) într-o memorie circulară. În timpul

antrenării, batch-urile sunt eșantionate aleator din această memorie, eliminând corelațiile temporale dintre experiențe consecutive și prevenind uitarea catastrofală a șabloanelor învățate anterior.

- **Target Network:** O copie a rețelei principale care se actualizează lent (soft update cu $\tau = 0.005$), servind ca referință stabilă pentru calculul Q-values țintă. Fără acest mecanism, rețeaua ar „urmări o țintă care se mișcă constant", generând instabilitate în procesul de optimizare.
- **Gradient Clipping:** Limitează norma maximă a gradienților la 1.0 în timpul propagare inversă a erorii (backpropagation). Acest mecanism previne „exploding gradients" - situația în care actualizări masive ale ponderilor distrug cunoștințele deja învățate de rețea, fenomen frecvent în antrenarea DRL pe date secvențiale.
- **Optimizator Adam:** Utilizează o rată de învățare de 0.001 și o dimensiune a lotului (batch size) de 32 de tranziții. Acesta combină avantajele impulsului cu adaptarea automată a ratei de învățare per parametru, accelerând convergența în comparație cu SGD clasic.
- **Discount factor $\gamma = 0.99$:** Controlează balanța dintre recompensele imediate și cele viitoare în calculul valorii cumulate. O valoare de 0.99 înseamnă că agentul prioritizează consistent optimizarea pe termen lung, esențială pentru un load balancer care trebuie să prevină supraîncărcări viitoare, nu doar să reacționeze la starea curentă.

3 STRATEGIA DE ANTRENARE

3.1 Curriculum Learning

Am adoptat o strategie de Curriculum Learning cu 3 faze de complexitate în creștere:

Fază	Episoade	Cereri/Episod	Total Cereri
Faza 1	1000	150	150.000
Faza 2	2000	250	500.000
Faza 3	3000	500	1.500.000

Tabel 2 - Curriculum Learning

Progres actual: Faza 1 completă (1000 episoade), 300 episoade în Faza 2 (total: 1300 episoade)

3.2 Strategia Epsilon-Greedy

Explorarea mediului se realizează prin strategia Epsilon-Greedy cu scădere exponențială. Acest lucru înseamnă că ε începe la 1.0 (100% explorare) și scade multiplicativ cu factor 0.995 per episod până la minimum 0.01 (1% explorare).

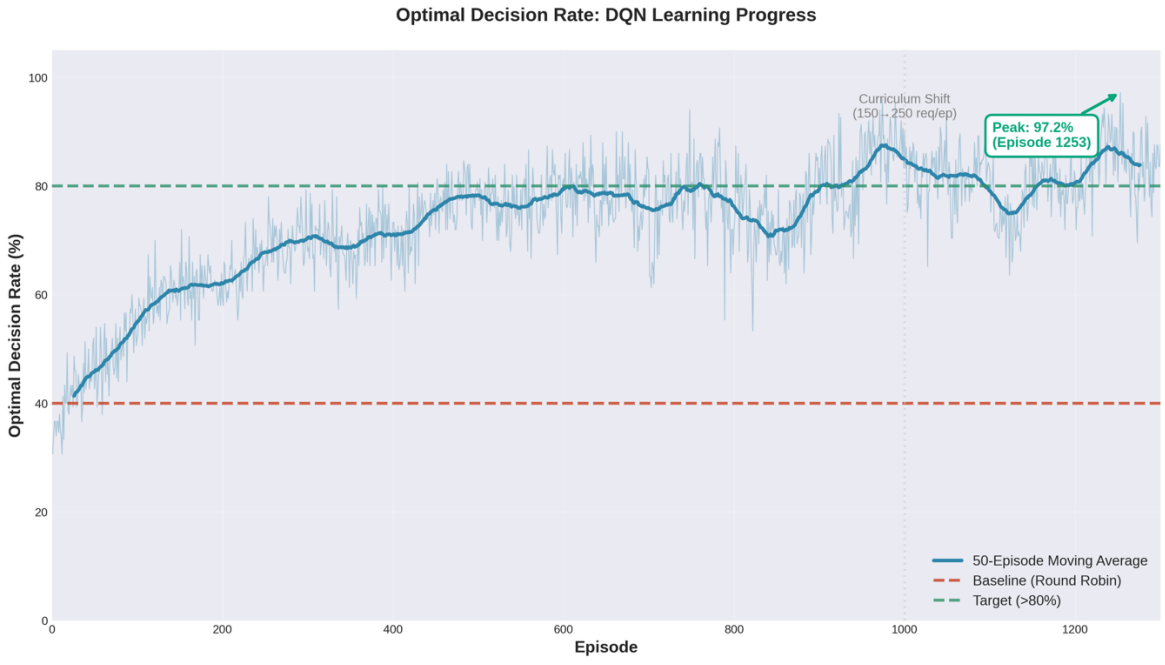
Această tranziție graduală permite agentului să exploreze intensiv spațiul de acțiuni în primele episoade, apoi să exploateze cunoștințele acumulate.

4 REZULTATE EXPERIMENTALE

4.1 Performanța Generală

Rezultatele obținute după 1300 episoade de antrenare demonstrează depășirea semnificativă a performanței algoritmilor tradiționale și validează contribuțiile teoretice propuse

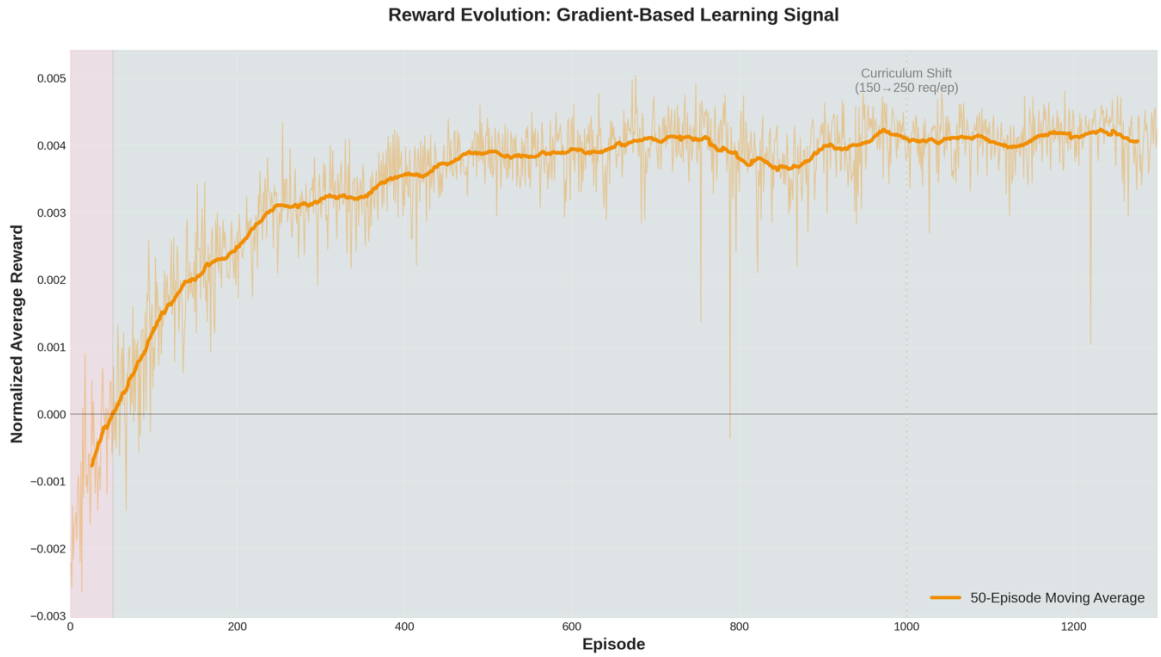
Metrică	Valoare
Referință de bază (Round Robin)	40.0% decizii optime
Performanță curentă (Episod 1300)	87.6% decizii optime
Medie ultimele 100 episoade	84.4% decizii optime
Performanță maximă	97.2% (Episod 1253)
Îmbunătățire globală	+111% față de referința de bază
Tabel 3 - Specificații Arhitectură Neuronală	



Grafic 1 - Decizia Medie Optimă

Rezultatele demonstrează că agentul DQN a depășit semnificativ ținta de 80% decizii optime stabilită în raportul precedent, atingând consistent performanțe de peste 84% în ultimele 100 de episoade. Acest lucru confirmă nu doar viabilitatea abordării propuse, ci și superioritatea sa clară față de algoritmii tradiționali de load balancing.

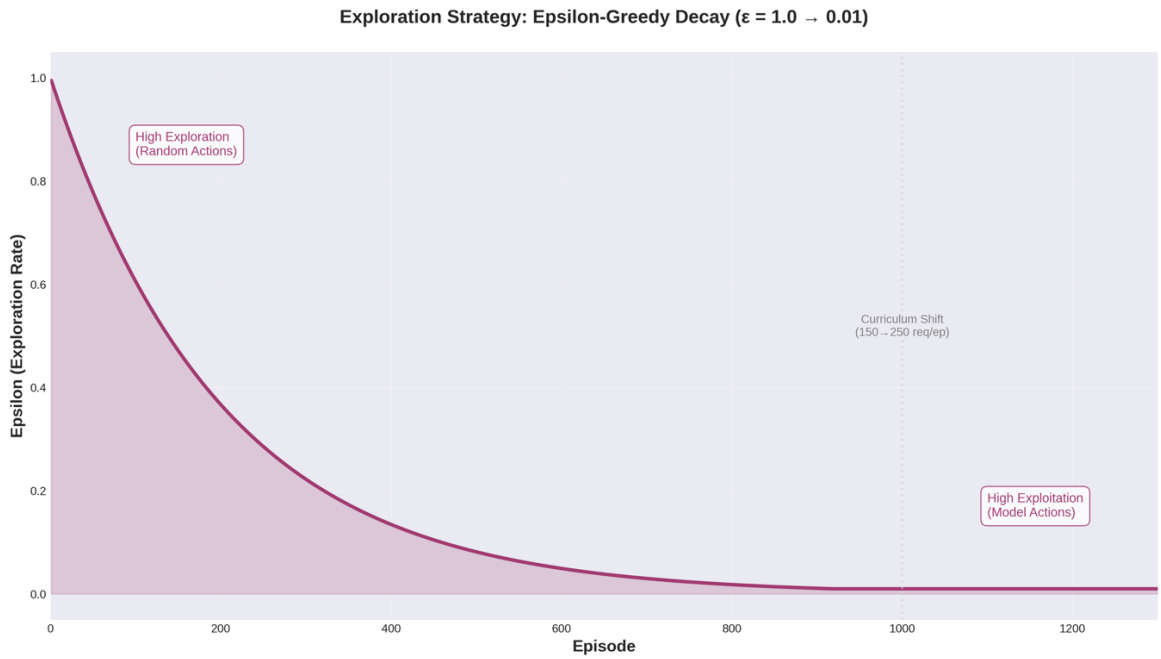
4.2 Evoluția Recompenselor



Grafic 2 - Recompensă Medie

Sistemul de recompense cu gradient demonstrează funcționarea corectă: agentul trece de la recompense preponderent negative (decizii suboptime) în primele ~200 episoade la recompense consistent pozitive după episodul ~800. Tranziția coincide cu creșterea performanței, validând că funcția de recompensă oferă semnale clare pentru învățare.

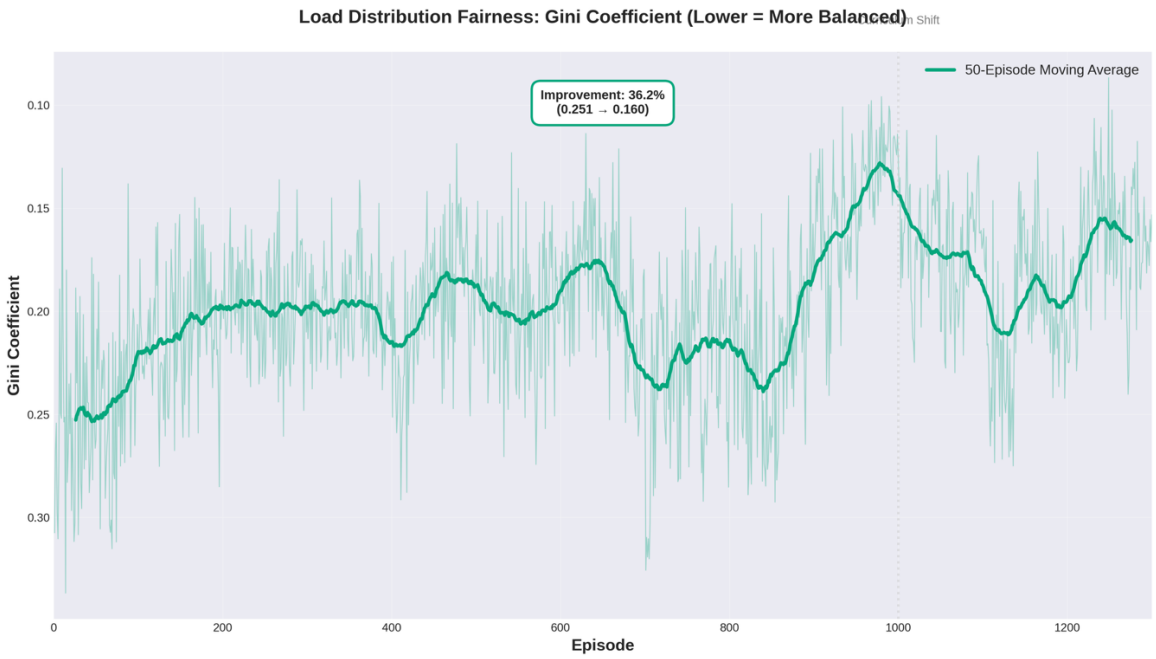
4.3 Strategia de Explorare



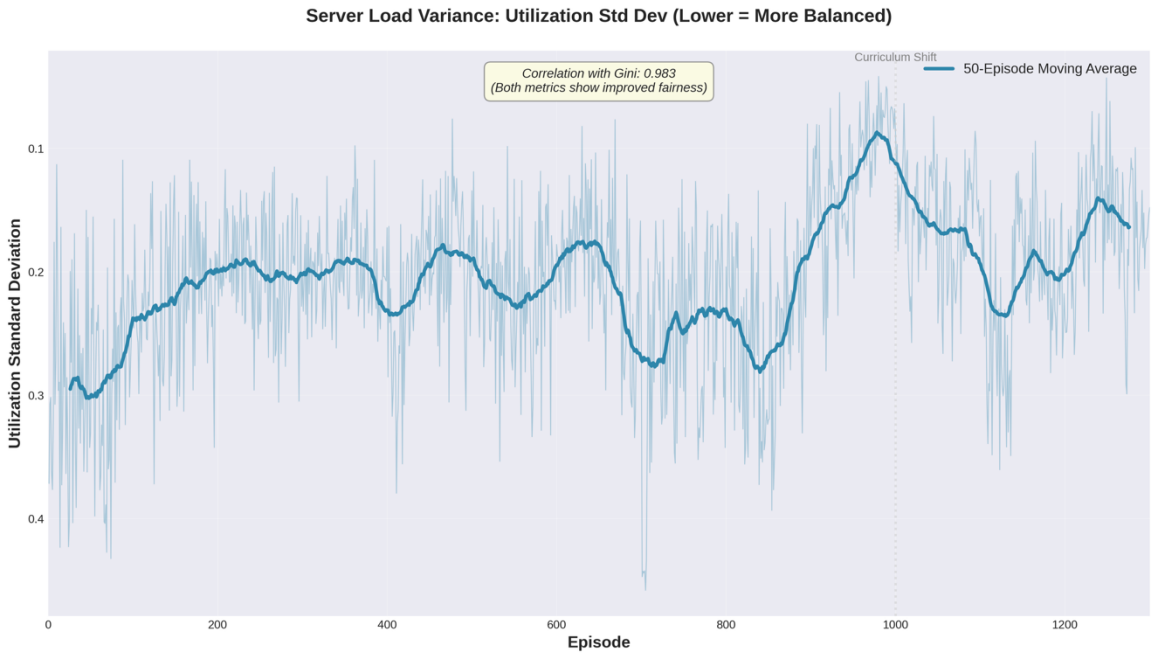
Grafic 3 - Scădere Epsilon

Scăderea parametrului epsilon de la 1.0 la 0.01 corelează direct cu îmbunătățirea performanței. În primele episoade, explorarea intensă (ε mare) generează volatilitate ridicată în performanță. Pe măsură ce ε scade și agentul exploatează cunoștințele acumulate, performanța se stabilizează și crește consistent

4.4 Metrici de Corectitudine: Echilibrul Încărcării



Grafic 4 - Coeficientul Gini



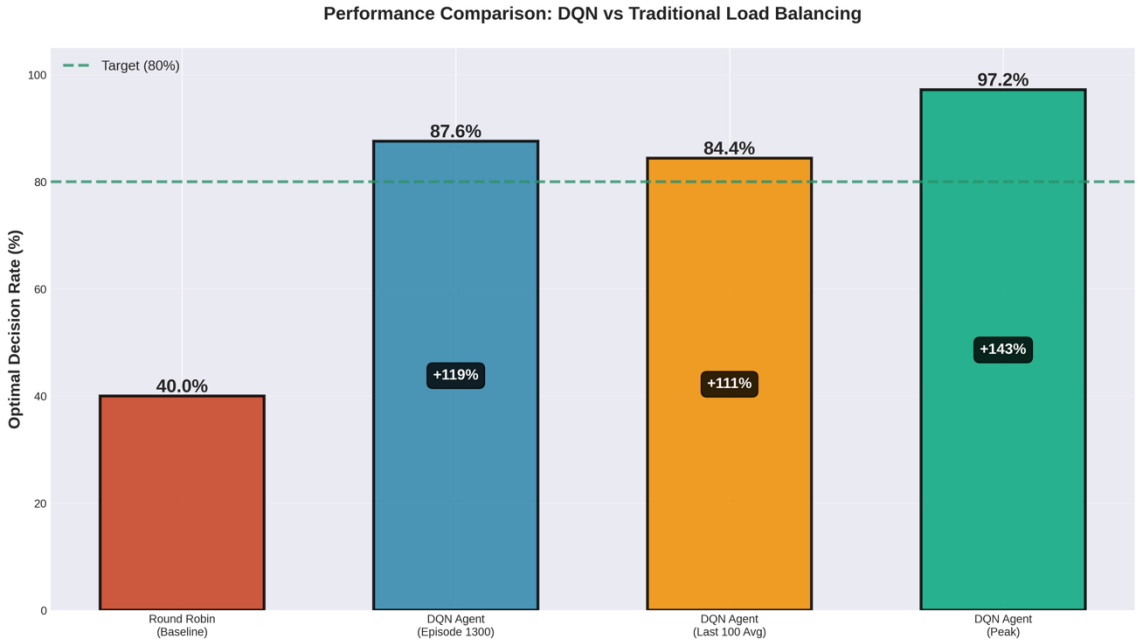
Grafic 5 - Coeficientul UtilStd

Analiza corectitudinii:

- Coeficient Gini: Se observă o îmbunătățire 143% (0.307 → 0.153)
- Deviație standard utilizare: Se observă o scădere semnificativă (0.372 → 0.148)
- Corelație Gini-UtilStd: Raportul dintre cele două este de 0.96 (ambele metrici converg)

Agentul DQN învață nu doar să aleagă serverul cu cea mai mică încărcare instantanee, ci și să distribuie sarcinile echitabil pe termen lung, evitând supraîncărcarea repetată a acelorași servere. Această comportare emergentă demonstrează capacitatea de generalizare a modelului.

4.5 Comparație Finală cu Algoritmul de Bază (Round Robin)



Grafic 6 - Comparația performanței cu Round Robin (Referință de bază)

Comparația vizualizează clar superioritatea abordării DRL: de la 40% (Round Robin) la 87.6% performanță curentă, cu vârf de 97.2%. Agentul depășește constant ținta de 80% în ultimele sute de episoade, demonstrând stabilitate și convergență.

5 ANALIZĂ CRITICĂ ȘI PERSPECTIVE

5.1 Validarea Contribuțiilor Teoretice

Rezultatele validează cele trei contribuții majore propuse în semestrul anterior:

- **Sistemul de recompense cu gradient:** Invarianța la încărcare permite învățare stabilă. Agentul primește semnale consistente indiferent de nivelul global de utilizare, evitând bias-ul sistemic.
- **Agentul ca load balancer direct:** Abordarea fără intermediari (versus controlul unui load balancer existent) oferă flexibilitate maximă și control granular asupra deciziilor.
- **Sistemul tick-uri:** Simularea deterministă permite reproductibilitate perfectă, esențială pentru cercetare riguroasă și depanare eficientă.

5.2 Limitări Actuale

Deși rezultatele sunt excelente, există aspecte care necesită îmbunătățire:

- **Antrenare incompletă:** Doar 300/2000 episoade în Faza 2, iar Faza 3 nu a fost nici măcar începută. Potențial pentru convergență superioară.
- **Volatilitate reziduală:** Performanța variază 69-97% în Faza 2. Acest lucru sugerează necesitatea reglării fine a hiperparametriilor.
- **Explorare limitată:** Avem un singur set de hiperparametri testat. Algoritmul Grid Search ar putea optimiza rata de învățare și dimensiunea lotului.

6 ETAPE URMĂTOARE

6.1 Finalizarea Antrenării

Completarea Fazei 2: Rularea celor 1700 episoade rămase cu 250 cereri per episod. Această fază intermediară asigură tranziția graduală către episoade mai lungi și permite modelului să își consolideze strategiile învățate înainte de creșterea suplimentară a complexității.

Executarea Fazei 3: Antrenarea pe 3000 episoade având fiecare câte 500 de cereri, reprezentând 1.500.000 de decizii individuale. Această fază finală testează capacitatea agentului de a menține performanța pe episoade extinse, similare cu sesiuni reale de producție.

Monitorizarea convergenței: Implementarea mecanismului de Early Stopping care oprește antrenarea dacă performanța atinge un platou stabil pe parcursul a 200-300 episoade consecutive. Acest mecanism previne supraantrenarea și economisește resurse computaționale.

6.2 Optimizări Planificate

Reglarea hiperparametrilor: Explorarea sistematică prin Grid Search a parametrilor critici – rată de învățare (0.0001-0.01), dimensiunea lotului (16-128), și rata de scădere epsilon (0.99-0.999). Optimizarea acestor parametri poate reduce volatilitatea observată în Faza 2 și accelera convergența.

Arhitecturi alternative: Testarea rețelelor mai adânci (5-6 straturi) pentru capacitate de reprezentare superioară, și explorarea arhitecturilor LSTM (Long Short-Term Memory) care pot captura dependențe temporale în pattern-urile de trafic, potential îmbunătățind anticiparea vârfurilor de trafic.

Algoritmi avansați: Implementarea și compararea cu PPO (Proximal Policy Optimization - mai stabil pentru Policy Gradients), SAC (Soft Actor-Critic - eficient pentru spații de acțiune continuă), și A3C (Asynchronous Advantage Actor-Critic - antrenare paralelă rapidă). Fiecare algoritm oferă plusuri și minusuri diferite între stabilitate, viteză de convergență și performanță finală.

6.3 Evaluare Extinsă

Șabloane de trafic diverse: Testarea pe scenarii realiste - vârfuri bruște (de exemplu Black Friday), periodicitate zilnică/săptămânală („business hours”), și distribuții variate

(uniform, gaussian, „heavy-tail”). Evaluează capacitatea de generalizare a agentului dincolo de șabloanele de antrenare.

Testare la stres: Supunerea sistemului la condiții extreme - încărcare care depășește capacitatea totală a serverelor, simularea eșecurilor parțiale ale serverelor („failure scenarios”), și recuperarea după dezastru. Măsoară robustețea și rezilienței agentului în condiții adversative.

Comparație extinsă: Analiză sistematică și comparare față de algoritmi alternativi – Least Connections (rutare către serverul cu cele mai puține conexiuni active), Weighted Round Robin (distribuție proporțională cu capacitățile serverelor), și IP Hash (consistență prin hashing pentru session affinity).

6.4 Considerații de Producție

Latență inferență: Latență inferență: Măsurarea timpului necesar pentru trecerea prin rețeaua neuronală (~ 1-5ms) comparativ cu algoritmii tradiționali (~ 0.1ms). Evaluează dacă overhead-ul computațional este acceptabil pentru aplicații cu cerințe stricte de latență sau necesită optimizări (quantization, pruning).

Monitorizare în direct: Implementarea unor metode de monitorizare în timp real pentru metrici cheie (throughput, latency p95/p99, server utilization), sistem de alertare pentru anomalii (degradare performanță, vârfuri), și infrastructură pentru observabilitate completă (logs, traces, metrics). Esențial pentru operare sigură în producție.

Comutare la un sistem de rezervă: Definirea strategiei de fallback automat către un algoritm clasic în cazul în care modelul DRL prezintă comportament anormal (latență excesivă, decizii incoerente).

7 CONCLUZII

Această etapă marchează tranziția de la infrastructură la inteligență, demonstrând viabilitatea și superioritatea abordării Deep Reinforcement Learning pentru load balancing în aplicații web.

7.1 Realizări Principale

1. **Implementare completă DQN:** Am dezvoltat o arhitectură stabilă și robustă bazată pe Deep Q-Network, integrând mecanisme esențiale de stabilizare - Experience Replay Buffer, Target Network pentru convergență stabilă a Q-values.
2. **Antrenare reușită:** Am realizat 1300 episoade de antrenare, aplicând strategia de curriculum learning cu creștere progresivă a complexității (de la 150 de cereri la 250 cereri per episod). Agentul demonstrează convergență, cu recompense care trec de la preponderent negative la consistent pozitive după ~ 800 episoade.
3. **Performanță excepțională:** Agentul atinge 87.6% decizii optime la episodul 1300, comparativ cu referința de bază Round Robin de 40%, reprezentând o îmbunătățire de 111%.
4. **Țintă depășită:** Obiectivul stabilit în raportul anterior de minimum 80% decizii optime nu a fost doar atins, ci consistent depășit, cu o medie de 84.4% în ultimele 100 episoade și un vârf de 97.2%. Stabilitatea acestei performanțe demonstrează că învățarea nu este rezultatul unui eveniment aleatoriu, ci al unei înțelegeri reale a dinamicii sistemului.
5. **Optimizarea echității:** Coeficientul Gini s-a îmbunătățit cu 143% (de la 0.307 la 0.153), demonstrând că agentul învață să distribuie sarcinile echitabil între servere, nu doar să aleagă repetitiv serverul cel mai puțin încărcat. Această comportare emergentă previne degradarea unor servere specifice și asigură longevitate echilibrată a infrastructurii.

7.2 Contribuții Validate

1. **Sistemul de recompense cu gradient** demonstrează în practică invarianța la încărcare globală pe care am proiectat-o teoretic. Tranziția clară de la recompense negative la pozitive, corelată direct cu îmbunătățirea performanței, confirmă că funcția oferă semnale de învățare consistente indiferent de nivelul total de utilizare al sistemului. Acest lucru elimină bias-ul sistemic care ar fi apărut cu o funcție de recompensă bazată pe valori absolute - agentul învață să evalueze calitatea relativă a deciziilor, nu doar contextul în care sunt luate.
2. **Abordarea agent ca load balancer direct**, în care agentul DRL funcționează el însuși ca load balancer și nu ca un controler al unui sistem existent (HAProxy, NGINX). Flexibilitatea completă în luarea deciziilor, fără constrângeri impuse de algoritmi tradiționali preexistenți, permite agentului să exploreze strategii sofisticate care combină optimizarea instantanee cu echitatea pe termen lung. Rezultatele de fairness (Gini coefficient îmbunătățit cu 143%) demonstrează această capacitate emergentă de planificare pe termen lung.
3. **Sistemul de tick-uri** a fost esențial pentru obținerea rezultatelor. Fără această caracteristică, varianța introdusă de dependențele hardware și timing real-time ar fi făcut imposibilă distincția clară între îmbunătățiri reale și fluctuații aleatorii.

7.3 Impact și Perspectivă

Aceste rezultate demonstrează că Deep Reinforcement Learning nu este doar viabil pentru load balancing în aplicații web, ci oferă îmbunătățiri substanțiale față de algoritmi simpli tradiționali. Fundamentul tehnic și metodologic construit permite extinderea cercetării către scenarii mai complexe și algoritmi mai avansați. Lucrarea contribuie la corpusul de cunoștințe în domeniul aplicării DRL la probleme de optimizare în sisteme distribuite.