Optymalizacja Agentów RL w Środowiskach Gier

Teoria i Praktyka: DQN, A2C, PPO

Jan Zakroczymski, Jan Zadrąg, Sławek Brzózka

January 17, 2026

Plan Prezentacji

1. Definicja Problemu i Cel

Cel Projektu:
Stworzenie i optymalizacja agentów uczenia ze wzmocnieniem (ang. RL) dla środowisk Gymnasium (`LunarLander`, `CarRacing`, `CartPole`).

Wyzwanie:
Znalezienie balansu między stabilnością a szybkością uczenia poprzez:

- Dobór odpowiedniego algorytmu (Model Selection).
- Automatyczną optymalizację hiperparametrów (Optuna).

2. Dane w RL i Źródła

- **Źródło danych:** interakcja agenta ze środowiskiem Gymnasium (symulator).
- **Dane są generowane online:** kolejne stany powstają w czasie treningu.
- **Różne przestrzenie stanów:** wektor stanu (CartPole, LunarLander) vs obraz RGB 96x96 (CarRacing).
- **Powtarzalność:** ustalony seed dla treningu, ewaluacji oraz optymalizacji hiperparametrów.

3. Przygotowanie Danych

- **Preprocessing CarRacing:** grayscale + resize do 84x84 + frame stacking (4 klatki).
- **Normalizacja obrazu:** model normalizuje piksele na wejsciu.
- **Dostosowanie akcji:** dyskretyzacja sterowania tylko dla DQN w CarRacing.
- **Wektoryzacja środowisk:** DQN(16 env), A2C (16 env), PPO (16 env).

4. EDA w RL

W RL nie ma klas, a dane powstaja w trakcie interakcji. EDA oznacza analize:

- **Trajektorii:** sekwencji stanów.
- **Rozkładu nagród w czasie (episodic return):** wykresach sumy nagród względem kroków symulacji.
- **Stabilność względem długosc epizodów:** wykrywanie oscylacji nagród.
- **Efektywność względem czasu treningu.**
- **Epsilon względem czasu (w przypadku DQN).**

Uzasadnienie Wyboru Modelu

- **DQN:** najlepszy dla dyskretnych akcji i prostszych przestrzeni stanów.
- **A2C:** prosty algorytm on-policy obsługujący akcje ciągłe.
- **PPO:** zawaansowana wersja A2C poprawiająca stabilność algorytmu.

RL opiera się na założeniu że rozpatrywany proces jest procesem markova. Przyjęty model środowiska: Agent w stanie s_t podejmuje akcję a_t, otrzymuje nagrodę r_t i przechodzi do stanu s_{t+1}.

Cel Agenta: Maksymalizacja sumy zdyskontowanych nagród:

$$G_t = \sum_{k=1}^{\infty} \gamma^k r_{t+k}$$

gdzie $\gamma \in [0, 1]$ to współczynnik dyskontujący (discount factor).

Podstawy Teoretyczne RL - Pojęcia

- **Polityka $\pi(s)$ lub $\pi(a|s)$:** wybrana akcja lub rozkład akcji w danym stanie.
- **Funkcja wartości $V_\pi(s)$:** oczekiwany zwrot z danego stanu przy polityce π.
- **Funkcja jakości $Q_\pi(s, a)$:** oczekiwany zwrot po wykonaniu akcji a w stanie s przy polityce π.
- **Eksploracja vs eksploatacja:** wybór między akcjami o najwyższej ocenie a nowymi.
- **On-policy vs off-policy:** czy dane do uczenia pochodzą z tylko aktualnej polityki.
- **Doświadczenie:** czwórka (s, a, r, s_{next}) gdzie s - stan obecny, a - akcja, r - nagroda ze środowiska, s_{next} - następny stan.

Deep Q-Network (DQN)

- Deep Q-Network (DQN), nazywany też Deep Q-Learning, to ulepszona wersja Q-Learning.
- Zamiast tabeli wartości używa sieci neuronowej, dzięki czemu działa w dużych przestrzeniach stanów.
- Nie rozwiązuje problemu ciągłych przestrzeni akcji, ponieważ sieć neuronowa ma skończoną liczbę wyjść.
- Mimo ograniczeń jest skuteczny i pozwala grać nawet w skomplikowane gry 3D poprzez dyskretyzacje akcji.

DQN - Funkcja straty

Funkcja straty (loss):

$$loss = (Q_{target} - Q_{current})$$

gdzie:

- $Q_{current}$ to wartość Q zwracana przez model dla stanu s.
- $Q_{target} = nagroda + \gamma \max_a Q(s_{next}, a)$.
- nagroda - zwrócona nagroda przez środowisko.

DQN - Kluczowe mechanizmy

- **Target Network:** osobna sieć do obliczania Q_{target} co stabilizuje uczenie.
- **Replay Buffer:** przechowuje doświadczenia i umożliwia losowe próbkowanie, co zmniejsza korelacje akcji.
- **Eksploracja ϵ-greedy:** ϵ zmniejszający się w czasie współczynnik akcji losowych - kompromis pomiędzy eksploracją a eksploatacją.

Advantage Actor-Critic (A2C)

- Składa się z dwóch sieci: **Aktor** steruje zachowaniem, **Krytyk** ocenia akcję.
- Podejście hybrydowe: łączy zalety on-policy (actor) i off-policy (critic).
- Rozwija algorytm strategii gradientowej.

A2C - Aktualizacja krytyka

Krytyk uczy się funkcji wartości $V(s)$ w której wagi są aktualizacjone za pomoca wzoru:

$$\Delta w = \beta(r + \gamma V(s_{next}) - V(s))\nabla_w V_w(s)$$

A2C - Aktualizacja aktora

Funkcja przewagi:

$$A(s, s_{next}) = r + \gamma V(s_{next}) - V(s)$$

Aktualizacja aktora:

$$\Delta\theta = \alpha[\nabla(\log \pi(s, a, \theta)) \cdot A(s, s_{next}) + c\nabla S_{\pi}(s)]$$

Jest to forma analogiczna do strategii gradientowej.

Proximal Policy Optimization (PPO)

- PPO to state of the art w RL, użyty m.in. do przełomu w Dota 2 (2018).
- Architektura podobna do A2C: aktor i krytyk.
- Główna idea: unikanie zbyt dużych aktualizacji aktora.

PPO - Funkcja zmiany

Stosunek prawdopodobieństw:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

Cel: utrzymać $r_t(\theta)$ w przedziale $[1-\epsilon, 1+\epsilon]$.

PPO - Funkcja clip

$$clip(r_t(\theta), \epsilon) = \begin{cases} 1-\epsilon & \text{gdy } r_t(\theta) < 1-\epsilon \\ r_t(\theta) & \text{gdy } r_t(\theta) \in (1-\epsilon, 1+\epsilon) \\ 1+\epsilon & \text{gdy } r_t(\theta) > 1+\epsilon \end{cases}$$

PPO - Funkcje loss

Loss polityki:

$$L_{clip} = \mathbb{E}_t[\min(r_t(\theta) \cdot A_t, clip(r_t(\theta), \epsilon) \cdot A_t)]$$

Wartość powyższej funkcji jest bardzo mocno zależna od $A_t(s_t, s_{t+1})$.

Loss krytyka:

$$L_V = r_t + \gamma V(s_{t+1}) - V(s_t)$$

Loss całkowity:

$$Loss = \mathbb{E}_t[L_V - c_1 L_{clip} + c_2 S_\pi(s_t)]$$

Kluczowe Hiperparametry - Wspólne

Wspólne parametry optymalizowane dla wszystkich modeli (Optuna):

- **Learning Rate (LR):** Szybkość uczenia (10^{-5} do 10^{-2}). Zbyt duży destabilizuje, zbyt mały spowalnia.
- **Gamma (γ):** Czynnik dyskontujący (0.9 do 0.9999). Określa, jak ważne są przyszłe nagrody (krótkowzroczność vs dalekowzroczność).
- **Batch Size:** Ilość próbek w jednej aktualizacji gradientu (16 do 512).
- **Architektura Sieci:** Rozmiar warstw ukrytych (Tiny, Small, Medium).

Kluczowe Hiperparametry - Specyficzne

DQN (Deep Q-Network):

- **Target Update Interval:** Częstotliwość aktualizacji sieci docelowej (1000-20000 kroków). Kluczowe dla stabilności.
- **Exploration Fraction:** Jak długo zmniejszać epsilon (eksplorację).

PPO / A2C (Actor-Critic):

- **Entropy Coefficient:** Premia za entropię. Zapobiega przedwczesnej zbieżności do suboptymalnej polityki.
- **GAE Lambda (λ):** Balans między bias a wariancją w estymacji przewagi.
- **N Steps:** Długość trajektorii zbieranej przed aktualizacją.

Optymalizacja w Praktyce - LunarLander (Case Study)

Obecnie Optuna optymalizuje na środowisku test (niezależne seedy), co redukuje przeuczenie do jednej ewaluacji.

- **Cel optymalizacji:** test/mean_reward.
- **Budżet:** domyślnie 60 prób, 300k kroków na próbę.
- **Różne algorytmy:** osobne przestrzenie hiperparametrów dla każdego algorytmu.

Trening Modeli

- **Budżet treningu:** konfigurowalny w `config.json` (aktualnie 1 000 000 kroków).
- **Podział środowisk:** trening seed=0, ewaluacja seed=42, test seed=1000.
- **Optymalizacja:** Optuna (domyślnie 40 prób, 150k kroków na próbę, inne dla każdej gry).

Ewaluacja Modeli

- **Mean Reward:** główna metryka skuteczności (eval/mean_reward, test/mean_reward).
- **Episodic Length:** dodatkowa kontrola stabilności.
- **Ewaluacja deterministyczna:** stałe warunki porównań, 10 epizodów na ewaluację.

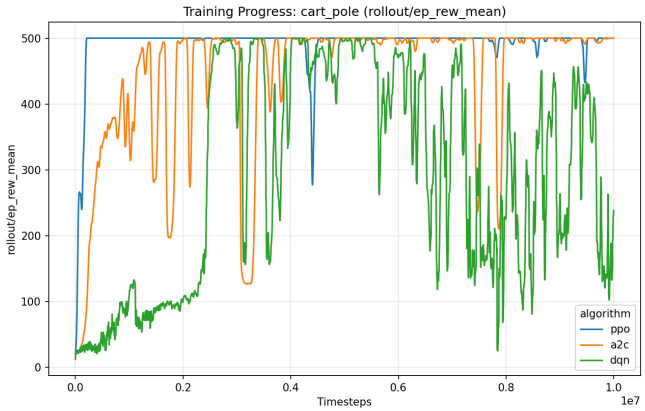
Analiza Błędów i Ograniczeń

- **Plateau nagrody:** widoczne wypłaszczenia na wykresach reward.
- **Katastrofalne zapominanie:** spadki po długich treningach.
- **CarRacing + DQN:** utrata płynności sterowania przy dyskretnych akcjach.
- **Diagnostyka:** porównanie reward/loss/epsilon między algorytmami.

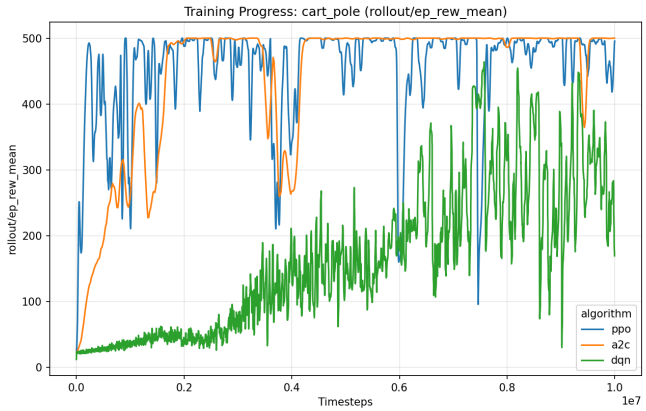
Optymalizacja i Ulepszenia

- **Preprocessing CarRacing:** grayscale + resize + frame stacking (4).
- **Dyskretyzacja akcji:** tylko dla DQN w CarRacing.
- **DQN buffer size:** redukcja do 50k dla danych obrazowych.
- **Tuning Optuna:** LR 10^{-5}–10^{-2}, γ 0.9–0.9999, batch 16–512, itd.

CartPole



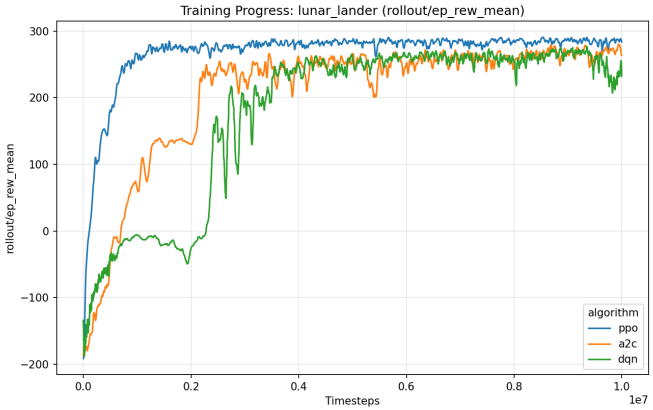
(a) Bez optymalizacji



(b) Po optymalizacji

Figure: CartPole: porownanie sredniej nagrody w epizodzie.

LunarLander



(a) Bez optymalizacji

(b) Po optymalizacji

Figure: LunarLander: porownanie sredniej nagrody w epizodzie.

CarRacing

(a) Bez optymalizacji

(b) Po optymalizacji

Figure: CarRacing: porownanie sredniej nagrody w epizodzie.

Porównanie Algorytmów w Projekcie

Cecha	**DQN**	**A2C**	**PPO**
Typ	Off-policy	On-policy	On-policy
Akcje	Dyskretne	Dyskretne/Ciągłe	Dyskretne/Ciągłe
Stabilność	Średnia	Średnia	**Wysoka**
Sample Eff.	**Wysoka** (Replay)	Średnia	Niska

Wnioski Końcowe

- **Stabilność:** PPO jest zwykle stabilniejsze niż A2C, ale wymaga sensownego budżetu kroków.
- **Obrazy:** DQN w środowiskach obrazowych wymaga preprocessing (grayscale, resize, frame stack).
- **Automatyzacja:** Optuna przyspiesza tuning, ale kosztuje czas i wymaga kontroli hold-out.