

Optymalizacja Agentów RL w Środowiskach Gier

Teoria i Praktyka: DQN, A2C, PPO

Jan Zakroczymski, Jan Zadrąg, Sławek Brzózka

January 14, 2026

Plan Prezentacji

- 1 Definicja Problemu
- 2 Dane i Przygotowanie
- 3 Analiza Danych (EDA)
- 4 Wybór Modelu
- 5 Podstawy Teoretyczne
- 6 Opisy Rozważanych Modeli
 - DQN (Deep Q-Network)
 - A2C (Advantage Actor-Critic)
 - PPO (Proximal Policy Optimization)
- 7 Kluczowe Hiperparametry
- 8 Trening i Ewaluacja
- 9 Analiza Błędów
- 10 Udoskonalenia
- 11 Porównanie i Wyniki
 - Optymalizacja vs bez
- 12 Podsumowanie

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 wejściu.
- **Dostosowanie akcji:** dyskretyzacja sterowania tylko dla DQN w CarRacing.
- **Wektoryzacja środowisk:** DQN(1 env), A2C (16 env), PPO (8 env).

4. EDA w RL

W RL nie ma klas, a dane powstają w trakcie interakcji. EDA oznacza analizę:

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

- **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 markowa. 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).

- **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 dyskretyzację akcji.

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.

- **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.

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

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

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.

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 $[1 - \epsilon, 1 + \epsilon]$.

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

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:

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

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):** Czynn timer 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).

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.

Obecnie Optuna optymalizuje z metryką hold-out (niezależne seedy), co redukuje przeuczenie do jednej ewaluacji.

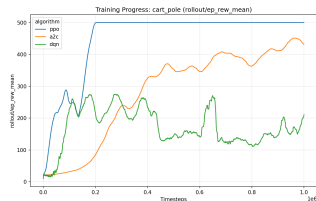
- **Cel optymalizacji:** holdout/mean_reward.
- **Budżet:** domyślnie 10 prób, 100k kroków na próbę (CLI); skrypt batch używa wartości per środowisko.
- **Różne algorytmy:** osobne przestrzenie hiperparametrów dla DQN oraz PPO/A2C.

- **Budżet treningu:** konfigurowalny w `config.json` (aktualnie 5 000 000 kroków).
- **Podział środowisk:** trening `seed=0`, ewaluacja `seed=42`, hold-out `seed=1000`.
- **Optymalizacja:** Optuna (domyślnie 10 prób, 100k kroków na próbę w CLI; batch ma inne wartości).

- **Eval vs Hold-out:** eval służy do wyboru modelu, hold-out do kontroli generalizacji.
- **Mean Reward:** główna metryka skuteczności (eval/mean_reward, holdout/mean_reward).
- **Episodic Length:** dodatkowa kontrola stabilności.
- **Ewaluacja deterministyczna:** stałe warunki porównań, 10 epizodów na ewaluację.

- **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.

- **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.



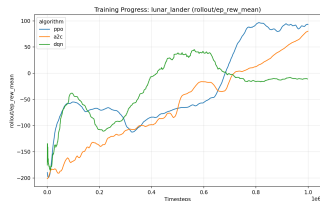
(a) Bez optymalizacji



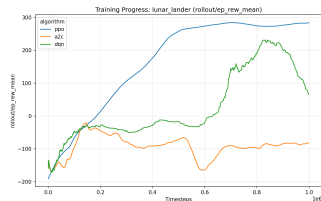
(b) Po optymalizacji

Figure: CartPole: porównanie sredniej nagrody w epizodzie.

LunarLander

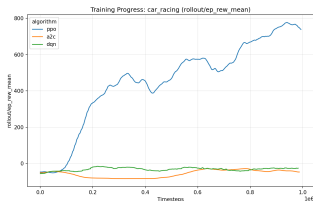


(a) Bez optymalizacji

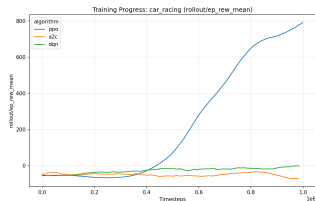


(b) Po optymalizacji

Figure: LunarLander: porównanie sredniej nagrody w epizodzie.



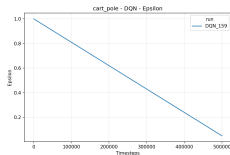
(a) Bez optymalizacji



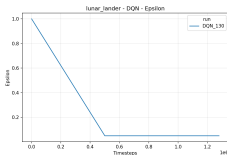
(b) Po optymalizacji

Figure: CarRacing: porównanie średniej nagrody w epizdzie.

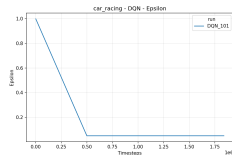
DQN - Epsilon Decay (Porównanie)



(a) CartPole



(b) LunarLander



(c) CarRacing

Optuna - DQN: Najlepsze parametry

Środowisko	Parametry
CartPole	learning_rate=0.0001793261, gamma=0.9534737990, net_arch=medium, batch_size=16, optimizer_class=RMSprop, activation_fn=Tanh, weight_decay=0.0000216215, target_update_interval=1000, train_freq=16, gradient_steps=2, exploration_fraction=0.4929325775, exploration_final_eps=0.0154726366
LunarLander	learning_rate=0.0082072604, gamma=0.9897746172, net_arch=tiny, batch_size=32, optimizer_class=Adam, activation_fn=ReLU, weight_decay=0.0000032910, target_update_interval=1000, train_freq=16, gradient_steps=4, exploration_fraction=0.3144025881, exploration_final_eps=0.0910880037
CarRacing	learning_rate=0.0000119164, gamma=0.9907858308, net_arch=medium, batch_size=512, optimizer_class=AdamW, activation_fn=ELU, weight_decay=0.0002437466, target_update_interval=10000, train_freq=16, gradient_steps=4, exploration_fraction=0.3792097857, exploration_final_eps=0.0774892213

Optuna - A2C: Najlepsze parametry

Środowisko	Parametry
CartPole	learning_rate=0.0000156660, gamma=0.9740529111, net_arch=small, optimizer_class=Adam, activation_fn=ReLU, weight_decay=0.0000019809, ent_coef=0.0000094432, vf_coef=0.6361241406, max_grad_norm=2.6224568685, gae_lambda=0.9582131557, n_steps=10
LunarLander	learning_rate=0.0033341384, gamma=0.9278734325, net_arch=tiny, optimizer_class=AdamW, activation_fn=ELU, weight_decay=0.0000034479, ent_coef=0.0001556883, vf_coef=0.2646656918, max_grad_norm=1.8644929648, gae_lambda=0.8640325836, n_steps=100
CarRacing	learning_rate=0.0000491340, gamma=0.9783137151, net_arch=medium, optimizer_class=RMSprop, activation_fn=ReLU, weight_decay=0.0000288824, ent_coef=0.0068872672, vf_coef=0.6833129830, max_grad_norm=4.2683798523, gae_lambda=0.9238527733, n_steps=5

Optuna - PPO: Najlepsze parametry

Środowisko	Parametry
CartPole	learning_rate=0.0011687224, gamma=0.9640504137, net_arch=medium, batch_size=512, optimizer_class=RMSprop, activation_fn=ReLU, weight_decay=0.0000045619, ent_coef=0.0000000143, vf_coef=0.2781731853, max_grad_norm=2.6276624082, gae_lambda=0.8823810760, n_steps=256
LunarLander	learning_rate=0.0010775910, gamma=0.9000274166, net_arch=medium, batch_size=256, optimizer_class=AdamW, activation_fn=ELU, weight_decay=0.0000041068, ent_coef=0.0644108211, vf_coef=0.2526720688, max_grad_norm=3.3079899133, gae_lambda=0.8072870163, n_steps=256
CarRacing	learning_rate=0.0001827441, gamma=0.9936572526, net_arch=tiny, batch_size=512, optimizer_class=RMSprop, activation_fn=ReLU, weight_decay=0.0000273247, ent_coef=0.0000911546, vf_coef=0.6395007879, max_grad_norm=0.7945144926, gae_lambda=0.9085099950, n_steps=128

Wyniki zależą od środowiska i długości triali. Aktualnie optymalizacja:

- wybiera parametry na podstawie **hold-out mean reward**,
- używa 100k kroków na próbę w CLI (batch: env-specific),
- ogranicza ryzyko przeuczenia do jednego seeda.

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

- **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.