Optymalizacja Agentów RL w Środowiskach Gier

Teoria i Praktyka: DQN, A2C, PPO

Jan

January 8, 2026

Plan Prezentacji

1. Definicja Problemu i Cel

Cel Projektu:
Stworzenie i optymalizacja agentów Uczenia ze Wzmocnieniem (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:** trajektorie powstają w czasie treningu.
- **Różne przestrzenie stanów:** wektor stanu (CartPole, LunarLander) vs obraz RGB 96x96 (CarRacing).
- **Powtarzalność:** seed=0 trening, seed=42 ewaluacja, seed=1000 hold-out.

3. Przygotowanie Danych

- **Preprocessing CarRacing:** grayscale + resize do 84x84 + frame stacking (4 klatki).
- **Normalizacja obrazu:** wejście jest normalizowane wewnątrz modelu (normalize_images).
- **Dostosowanie akcji:** dyskretyzacja sterowania tylko dla DQN w CarRacing.
- **Wektoryzacja środowisk:** DummyVecEnv; A2C (16 env), PPO (8 env).

4. EDA w RL

- **Metryki z TensorBoard:** `rollout/ep_rew_mean`, `train/loss` lub `train/value_loss`.
- **Eksploracja DQN:** śledzenie `rollout/exploration_rate`.
- **Wykresy porównawcze:** automatyczne generowanie `debug_out/comparison_*` z najnowszych runów.
- **Oś X:** normalizacja kroków do startu od 0.
- **Trend stabilności:** analiza nachylenia krzywych reward/loss w czasie.

Uzasadnienie Wyboru Modelu

- **DQN:** najlepszy dla dyskretnych akcji i prostszych przestrzeni stanów (CartPole, LunarLander).
- **PPO:** stabilny algorytm on-policy do akcji ciągłych i dyskretnych (CarRacing).
- **A2C:** szybka alternatywa dla PPO, testowana jako kompromis szybkość/stabilność.
- **Cel porównania:** sprawdzenie wpływu typu algorytmu na stabilność i sample efficiency.

Proces Decyzyjny Markova (MDP): Agent w stanie s_t podejmuje akcję a_t, otrzymuje nagrodę r_t i przechodzi do stanu s_{t+1}.

Cel Agenta: Maksymalizacja oczekiwanej sumy zdyskontowanych nagród:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

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

Podstawy Teoretyczne RL - Pojęcia

- **Polityka** $\pi(a|s)$**:** rozkład akcji w danym stanie (deterministyczna lub stochastyczna).
- **Funkcja wartości** $V^{\pi}(s)$**:** oczekiwany zwrot z danego stanu przy polityce π.
- **Funkcja** $Q^{\pi}(s, a)$**:** oczekiwany zwrot po wykonaniu akcji a w stanie s.
- **Eksploracja vs eksploatacja:** wybór między nowymi akcjami a tymi już opłacalnymi.
- **On-policy vs off-policy:** czy dane do uczenia pochodzą z aktualnej polityki.

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.
- Nadal ma problem przy bardzo dużej liczbie możliwych akcji, bo trzeba liczyć Q dla każdej z nich.
- Mimo ograniczeń jest skuteczny i pozwala grać nawet w skomplikowane gry 3D.

DQN - Funkcja straty

Funkcja straty (loss):

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

gdzie:

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

DQN - Problemy

- **Zapominanie doświadczeń:** nowe epizody nadpisują wcześniejsze umiejętności.
- **Korelacja doświadczeń:** agent uczy się sekwencji akcji zamiast reagować na bieżący stan.

DQN - Rozwiązania

- Rozdzielenie eksploracji i nauki przez bufor doświadczeń.
- Zapis epizodów do kolejki o maksymalnym rozmiarze.
- Losowy dobór próbek do uczenia, co redukuje korelacje i zapominanie.

DQN - Kluczowe mechanizmy

- **Równanie Bellmana:** $Q(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a')$.
- **Target Network:** osobna sieć do obliczania celu stabilizuje uczenie.
- **Eksploracja ϵ-greedy:** kontrola kompromisu eksploracja/eksploatacja.
- **Double DQN:** rozdzielenie wyboru i oceny akcji zmniejsza zawyżanie Q.

Advantage Actor-Critic (A2C)

- Hybryda podejścia z polityką i bez polityki.
- Składa się z dwóch sieci: **Aktor** steruje zachowaniem, **Krytyk** ocenia akcję.
- A2C dyskontuje nagrody na każdym kroku, co stabilizuje ocenę.

A2C - Aktualizacja krytyka

Krytyk używa funkcji wartości $V(s)$:

$$\Delta w = \beta(r_t + \gamma V(s_{t+1}) - V(s_t))\nabla_w V_w(s_t)$$

Użycie V zamiast Q poprawia stabilność uczenia.

A2C - Aktualizacja aktora

Funkcja przewagi:

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

Aktualizacja aktora:

$$\Delta\theta = \alpha[\nabla(\log \pi(s, a, \theta)) \cdot A(s_t, s_{t+1}) + c\nabla S\pi(s_t)]$$

Jest to forma analogiczna do strategii gradientowej.

A2C - Dlaczego działa

- **Błąd TD:** $r_t + \gamma V(s_{t+1}) - V(s_t)$ szybko przenosi informację o nagrodzie.
- **N-step returns:** łączy stabilność TD i informację z dłuższych trajektorii.
- **Entropia polityki:** bonus za różnorodność działań zapobiega zbyt wczesnej zbieżności.
- **Równoległe środowiska:** redukują korelacje i przyspieszają zbieranie danych.

A2C - Replay i równoległość

- Uczenie równoległe na wielu środowiskach zamiast klasycznego replay.
- Kroki: ruch w każdym środowisku, gradienty lokalne, średni gradient, aktualizacja strategii.
- Alternatywnie: dodatkowa sieć do aktualizacji wag (zysk na GPU).

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:

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

Loss krytyka:

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

Loss całkowity:

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

PPO - Stabilność i praktyka

- **Clipping** działa jak miękki ogranicznik kroku uczenia.
- **GAE (λ):** wygładza estymację przewagi i zmniejsza wariancję.
- **Minibatch SGD:** kilka epok po tych samych danych zwiększa stabilność.
- **Monitorowanie KL:** zbyt duża dywergencja oznacza zbyt agresywne aktualizacje.

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

- **Cel optymalizacji:** holdout/mean_reward.
- **Budżet:** domyślnie 30 prób, 100k kroków na próbę.
- **Różne algorytmy:** osobne przestrzenie hiperparametrów dla DQN oraz PPO/A2C.

Trening Modeli

- **Budżet treningu:** konfigurowalny w `config.json` (aktualnie 5000 kroków).
- **Podział środowisk:** trening seed=0, ewaluacja seed=42, hold-out seed=1000.
- **Early stopping:** patience=20 w CarRacing, minimalny próg kroków (500k dla CarRacing, 100k dla pozostałych).
- **Optymalizacja:** Optuna (domyślnie 30 prób, 100k kroków na próbę).

Ewaluacja Modeli

- **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.

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.

LunarLander - Porównanie Algorytmów

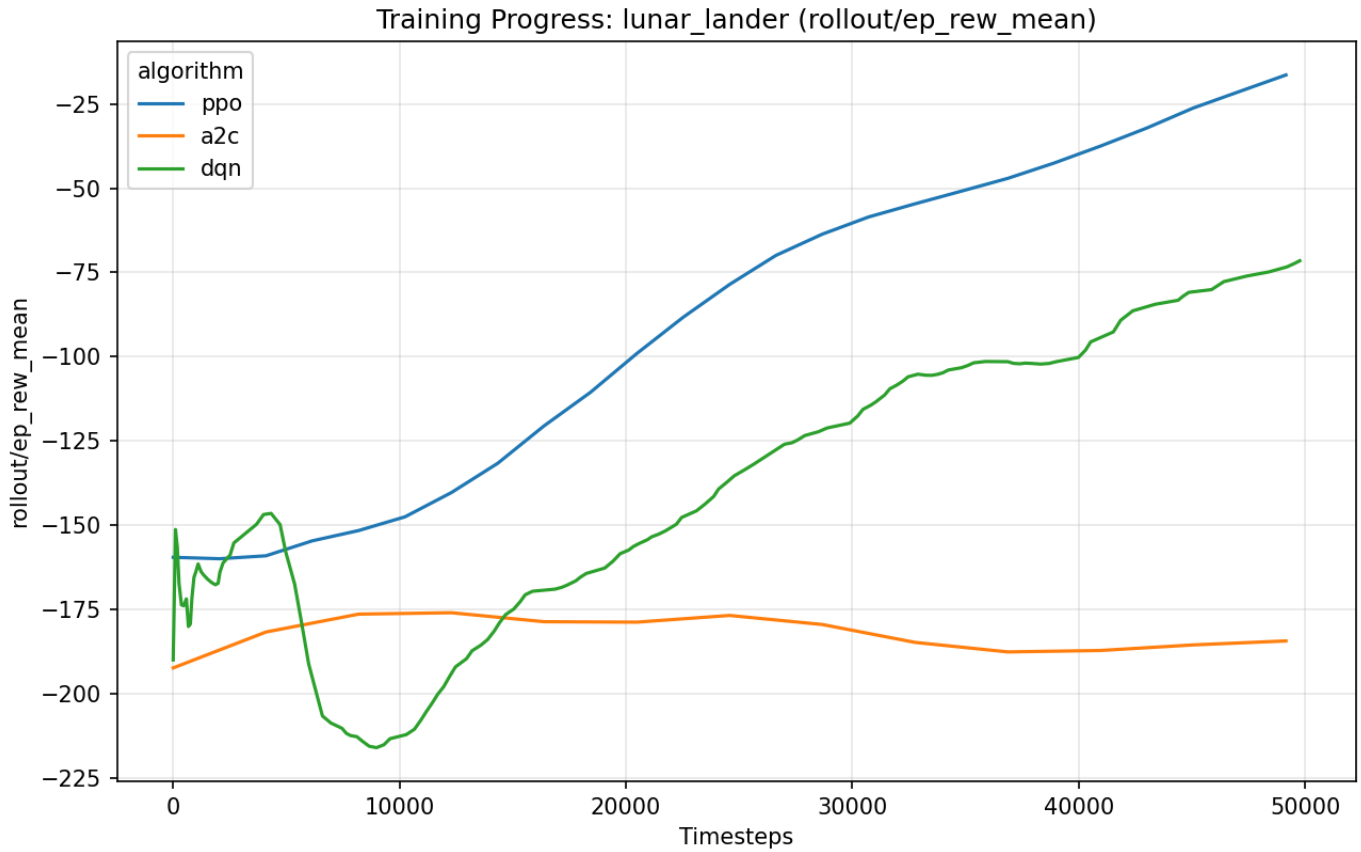
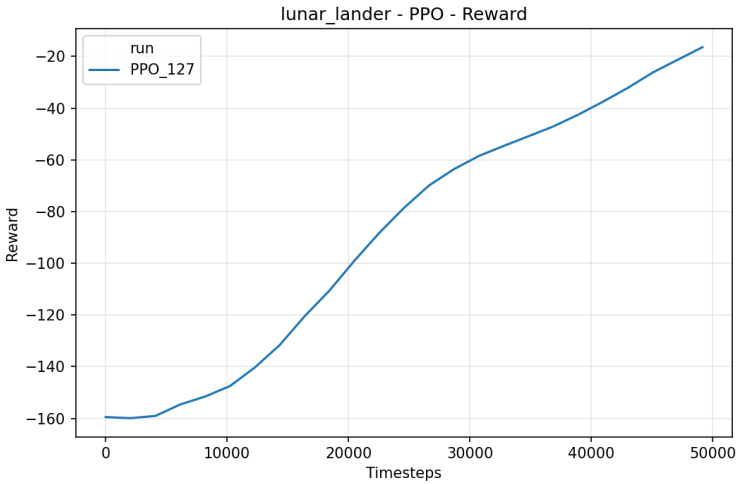
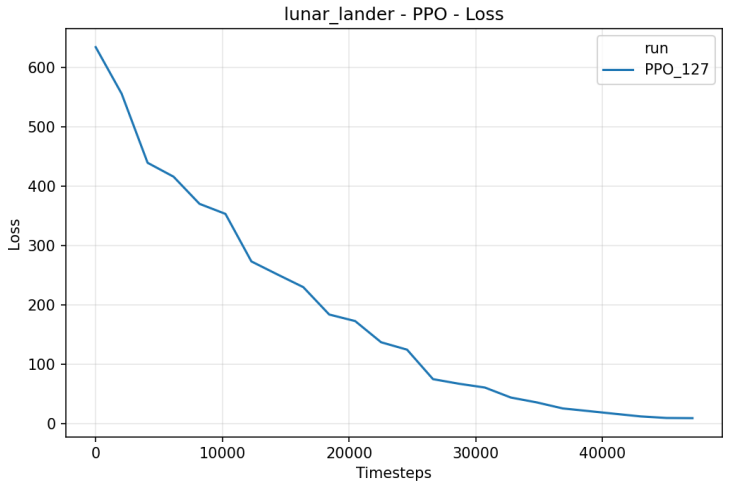


Figure: Średnia nagroda w epizodzie dla LunarLander (najnowsze runy).

LunarLander - PPO Szczegóły



(a) PPO Reward (Szybka zbieżność)



(b) PPO Loss

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

Wyniki Optymalizacji z Optuna

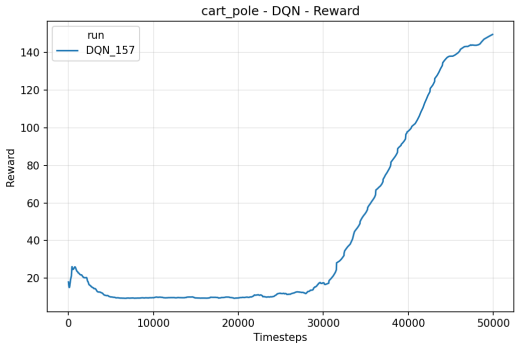
Wyniki zależą od środowiska i długości triali. Aktualnie optymalizacja:

- wybiera parametry na podstawie **hold-out mean reward**,
- używa dłuższych triali (100k kroków),
- ogranicza ryzyko przeuczenia do jednego seeda.

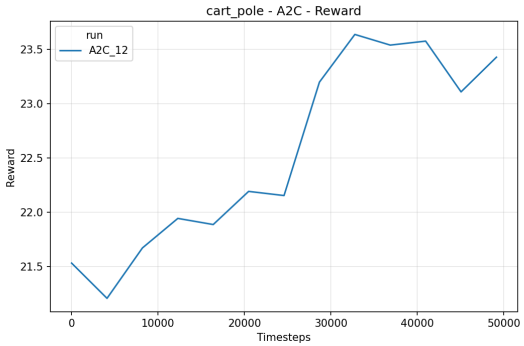
CartPole - Porównanie Algorytmów

Figure: Średnia nagroda w epizodzie (Rollout Ep Reward Mean) dla algorytmów PPO, A2C i DQN.

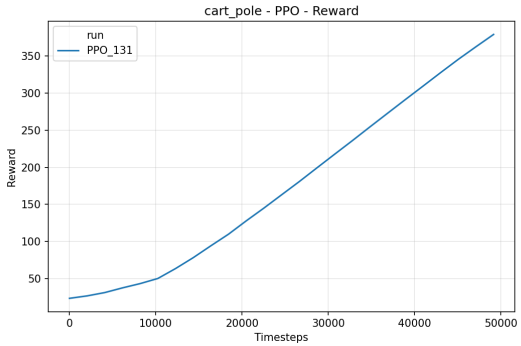
CartPole - Szczegóły Treningu



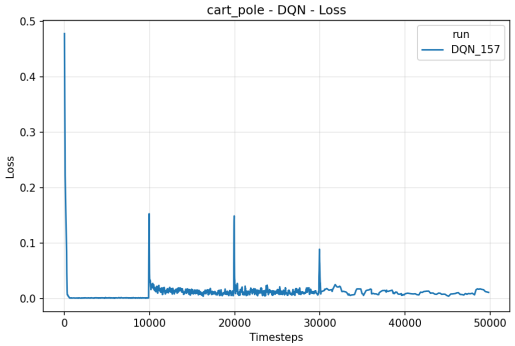
(a) DQN Reward



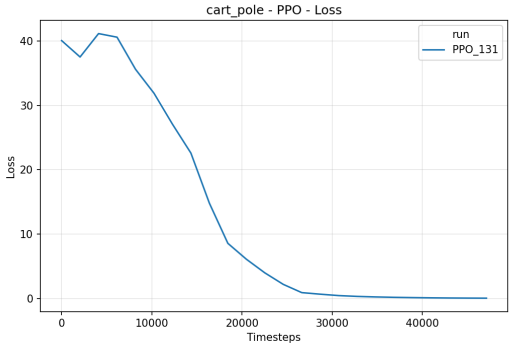
(b) A2C Reward



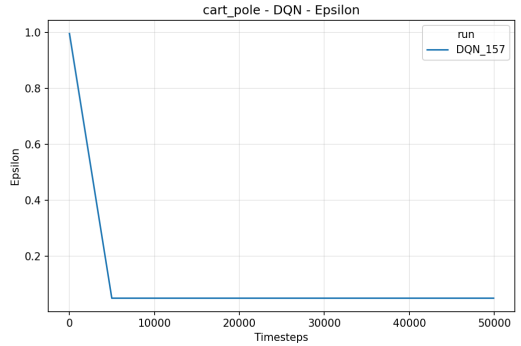
(c) PPO Reward



(d) DQN Loss

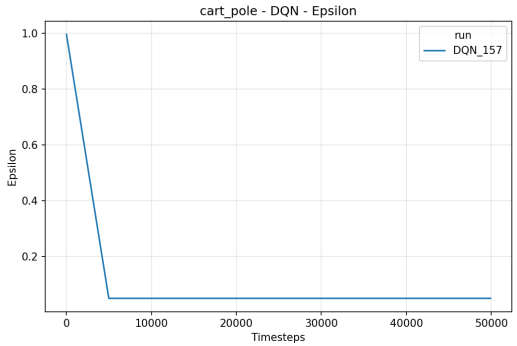


(e) PPO Loss

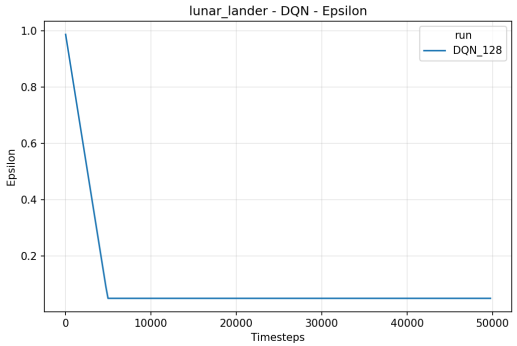


(f) DQN Epsilon

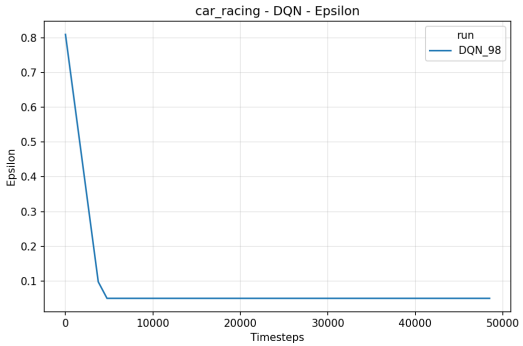
DQN - Epsilon Decay (Porownanie)



(a) CartPole



(b) LunarLander



(c) CarRacing

CarRacing - Porównanie Algorytmów

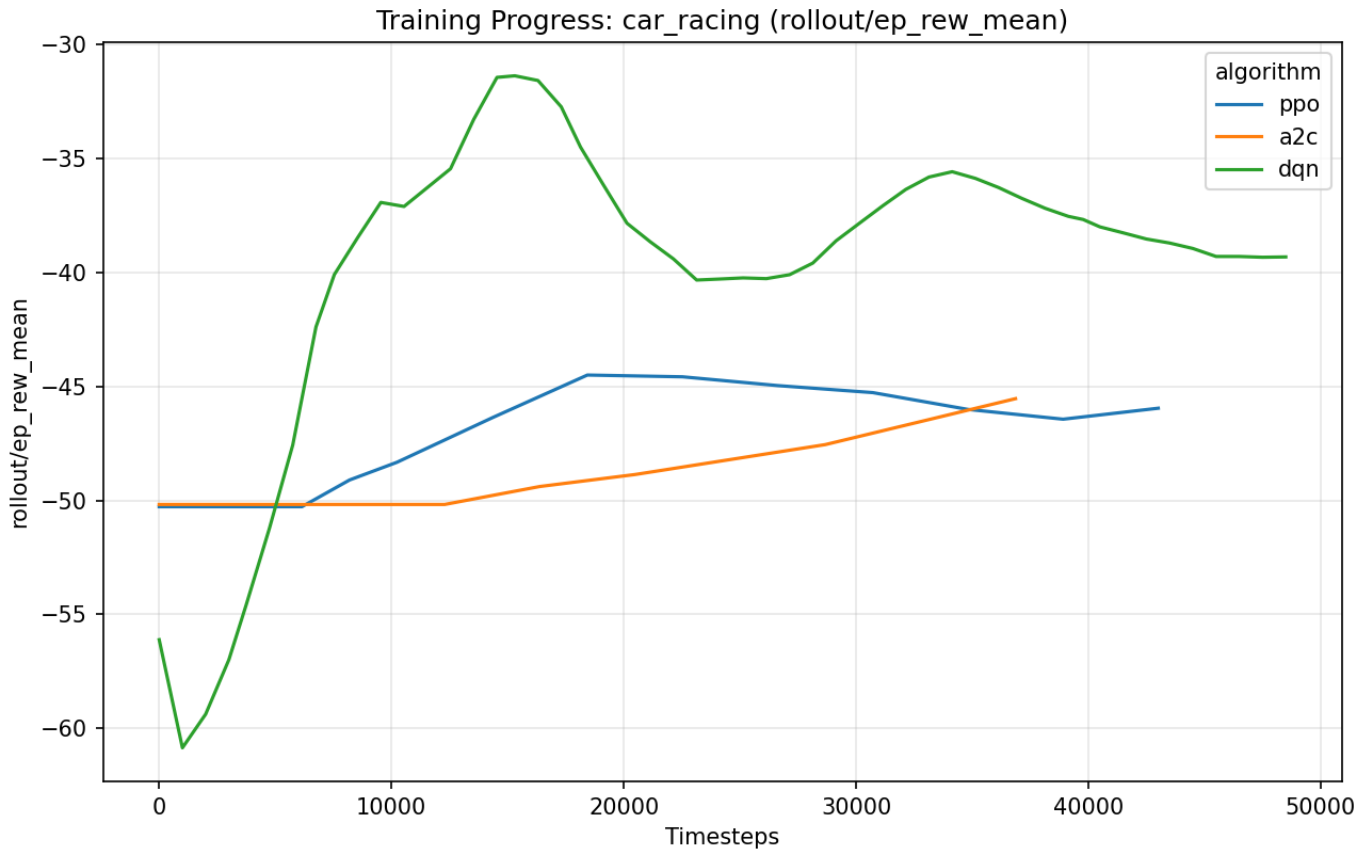
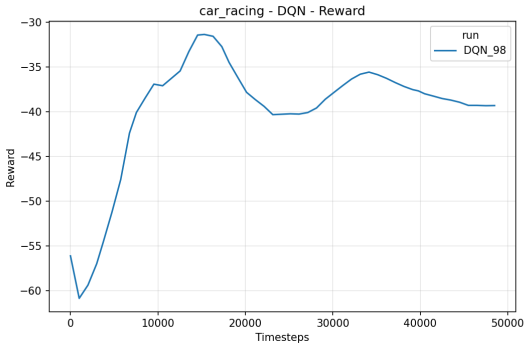
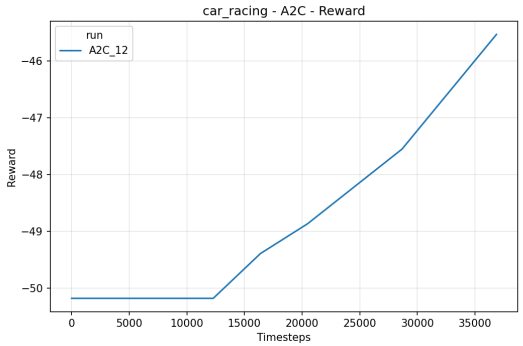


Figure: Porównanie średniej nagrody w środowisku CarRacing.

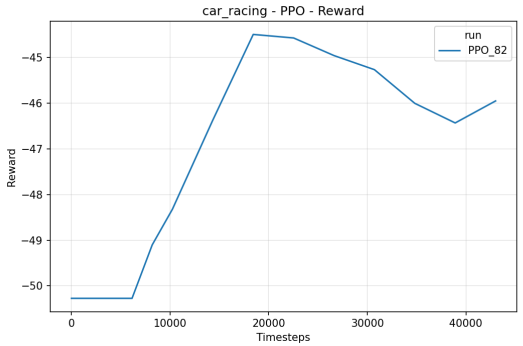
CarRacing - Szczegóły Treningu



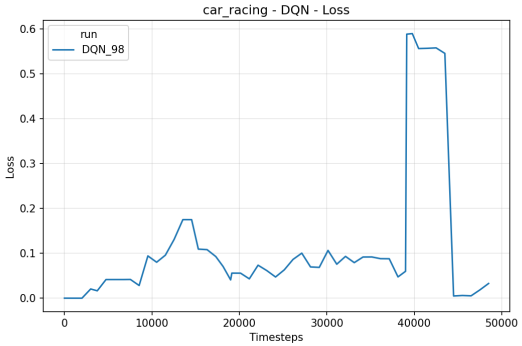
(a) DQN Reward



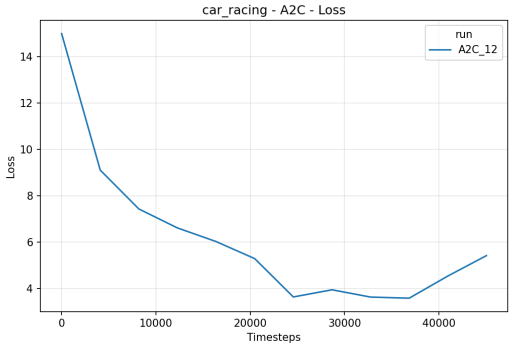
(b) A2C Reward



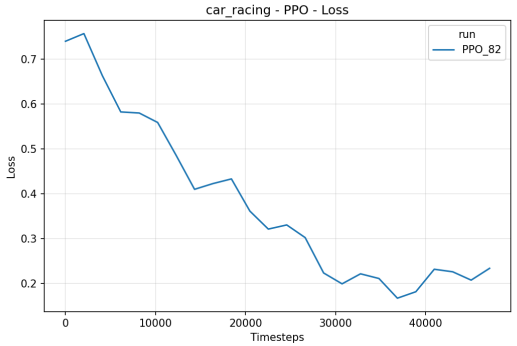
(c) PPO Reward



(d) DQN Loss



(e) A2C Loss



(f) PPO Loss

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.