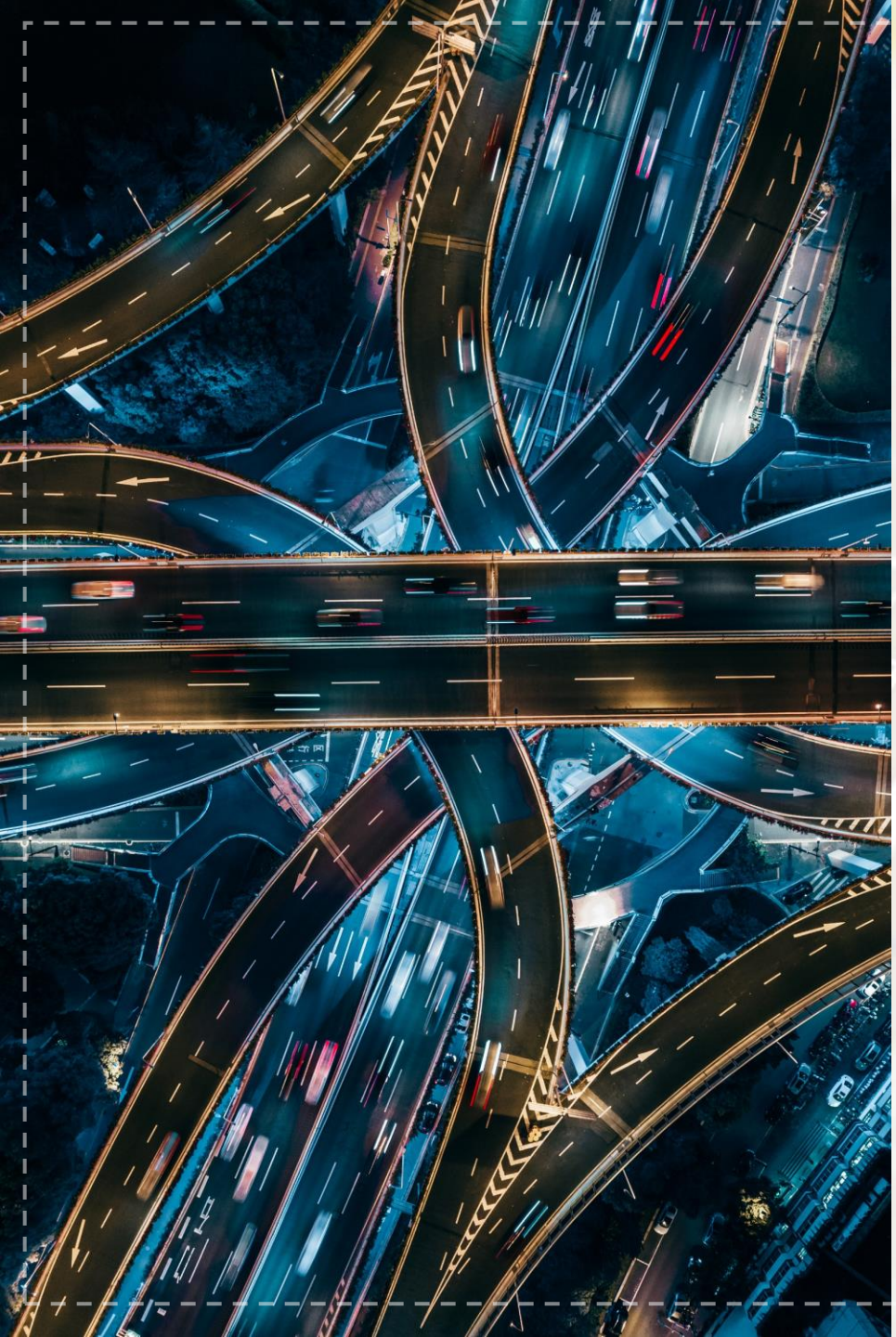


WYŻSZA SZKOŁA ZARZĄDZANIA I BANKOWOŚCI W KRAKOWIE

Wydział Nauk Stosowanych

KIERUNEK: **Informatyka**

ZAKRES KSZTAŁCENIA: **Bazy danych**

PRACA DYPLOMOWA

Robert Jan Smoter

Symulacja ruchu drogowego z zastosowaniem algorytmów optymalizacji sterowania sygnalizacją świetlną

Promotor: **dr hab. inż. Rafał Dreżewski**

Recenzent: **dr inż. Robert Marcjan**

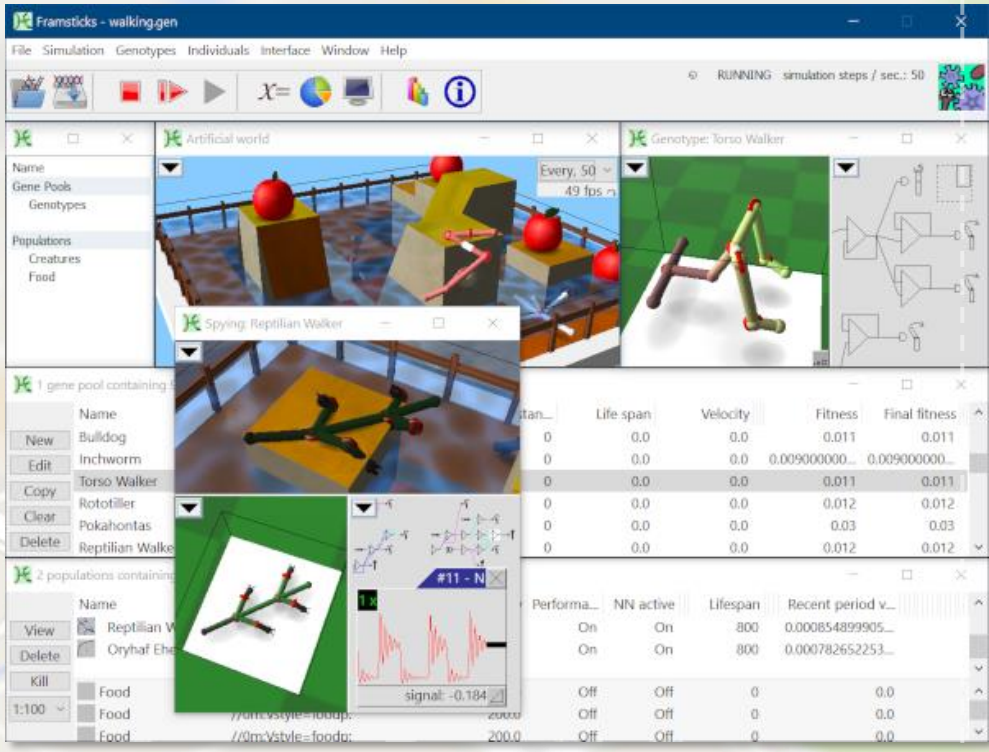
WSTĘP, MOTYWACJA, CEL I ZAKRES PRACY

MOTYWACJA
Motywacją do podjęcia tematu były zajęcia, podczas których pracowaliśmy z symulatorem Framsticks. Duże wrażenie zrobiła na mnie możliwość obserwowania na bieżąco, jak algorytm wpływa na zachowanie wirtualnych organizmów – jak stopniowo się rozwijały i przystosowywały do środowiska. Ta wizualna forma procesu uczenia zainspirowała mnie do zgłębienia tematyki algorytmów uczących się oraz środowisk symulacyjnych, w których można obserwować, jak operacje matematyczne przekładają się na konkretne działania i decyzje systemu.

CEL PRACY
Praktyczna implementacja algorytmu uczenia ze wzmocnieniem w środowisku symulacyjnym SUMO.
Weryfikacja, czy agent RL (Reinforcement Learning) może skutecznie sterować sygnalizacją świetlną i adaptować się do zmiennych warunków ruchu drogowego.

ZAKRES METODYCZNY;
- **Literatura** – analiza rozwiązań ITS i algorytmów RL.
- **Implementacja** – model Actor-Critic w Pythonie z wykorzystanie biblioteki Google Open Source, TensorFlow.
- **Integracja** – komunikacja z SUMO przez interfejs TraCI.
- **Analiza wyników** – ocena skuteczności agenta i porównanie z innymi podejściami.

OCZEKIWANE REZULTATY
Weryfikacja, czy agent RL potrafi autonomicznie dostosowywać sygnalizację do środowiska.
Ocena realnego potencjału uczenia maszynowego w sterowaniu ruchem drogowym.



INTELIGENTNE SYSTEMY TRANSPORTOWE (ITS)

ITS to nowoczesne technologie wspomagające zarządzanie ruchem drogowym, zwiększające bezpieczeństwo oraz efektywność transportu.
Systemy te integrują elementy informatyki, komunikacji i automatyki, umożliwiając dynamiczne reagowanie na zmienne warunki w czasie rzeczywistym.

Rodzaje systemów sterowania ruchem:
Scentralizowane - decyzje podejmowane są przez centralny system zarządzania, który ma dostęp do danych z wielu skrzyżowań;
Zdecentralizowane – każde skrzyżowanie działa autonomicznie, podejmując decyzje na podstawie lokalnych danych;
Stałoczasowe – działają według ustalonego harmonogramu, niezależnie od aktualnego ruchu;
Zmiennoczasowe – dostosowują długość faz na podstawie danych z lokalnych czujników;

Przykłady adaptacyjnych systemów ITS:
SCATS – australijski system adaptujący cykle świateł do natężenia ruchu (Australia)
SCOOT – system optymalizujący długość i przesunięcia cykli świetlnych (GB)
RHODES – wykorzystuje prognozy przepływu pojazdów do dynamicznego sterowania (USA)
PIACON – sterowania ruchem drogowym, opracowana w 2008 roku przez AGH

Współczesne systemy ITS korzystają z szerokiego wachlarza technologii, takich jak czujniki ruchu, detektory, kamery, stacje pogodowe oraz komunikacja między pojazdami (V2V) i infrastrukturą (V2I).
Coraz częściej wspomagane są metodami sztucznej inteligencji, które pozwalają na bardziej elastyczne, skalowalne i autonomiczne podejmowanie decyzji — bez potrzeby ręcznego programowania wszystkich możliwych scenariuszy ruchu drogowego.

PODSTAWY UCZENIA ZE WZMOCNIENIEM

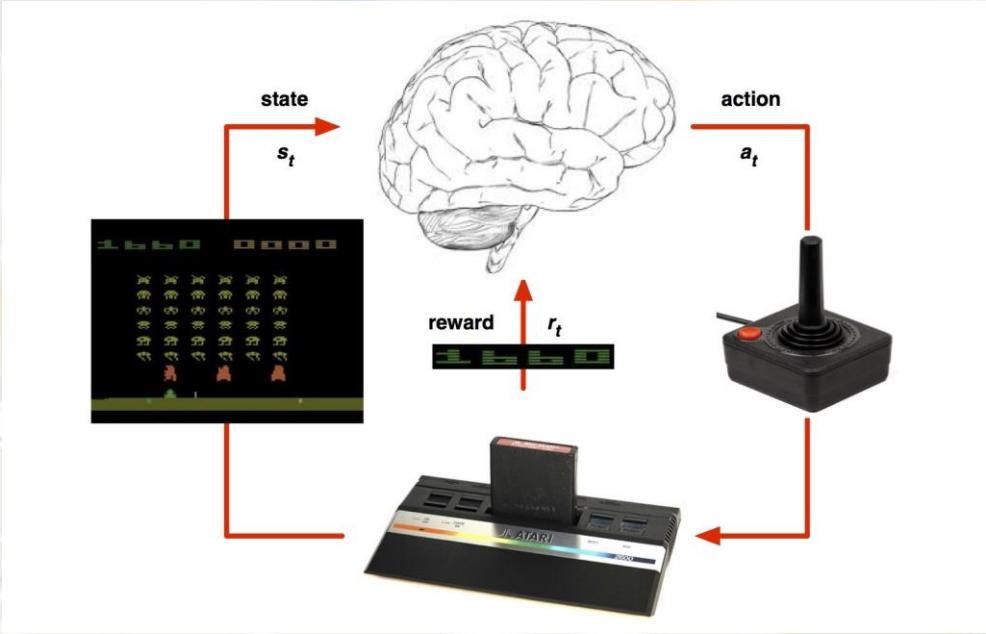
Uczenie ze wzmocnieniem (RL) to dziedzina sztucznej inteligencji, w której agent uczy się poprzez interakcję ze środowiskiem, otrzymując nagrody lub kary za swoje działania.

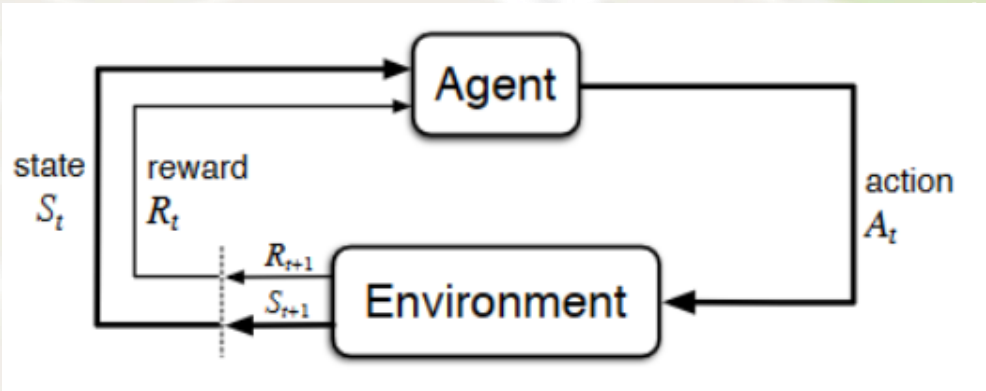
Korzenie RL sięgają lat 60 – pierwsze modele oparte na warunkowaniu i teorii decyzji. W latach 80 pojawiły się algorytmy takie jak TD-learning (Temporal Difference) i Q-learning, umożliwiające efektywną naukę bez znajomości modelu środowiska.

2015: DeepMind zaprezentowało agenta, który uczył się grać w gry Atari (np. Breakout) wyłącznie na podstawie obrazu i punktacji – bez jakiejkolwiek wiedzy eksperckiej.
2017: Projekty AlphaGo i AlphaZero pokazały, że agent może nauczyć się gry planszowej od zera i pokonać mistrzów świata (bez danych wejściowych od ludzi).

Model RL opiera się na tzw. **procesie decyzyjnym Markowa (MDP)**, który składa się z:
- **stanów (S)** – opisujących bieżącą sytuację (np. długości kolejek),
- **akcji (A)** – możliwych decyzji agenta (np. zmiana świateł),
- **nagrody (R)** – informacji zwrotnej za wykonanie danej akcji,
- **funkcji przejścia (P)** – określającej funkcję przejścia ze stanu do stanu.

Celem agenta jest **maksymalizacja skumulowanej nagrody w czasie**.
RL może być stosowane w środowiskach, gdzie model świata nie jest w pełni znany – agent sam uczy się optymalnej polityki działania.





ALGORYTM ACTOR-CRITIC

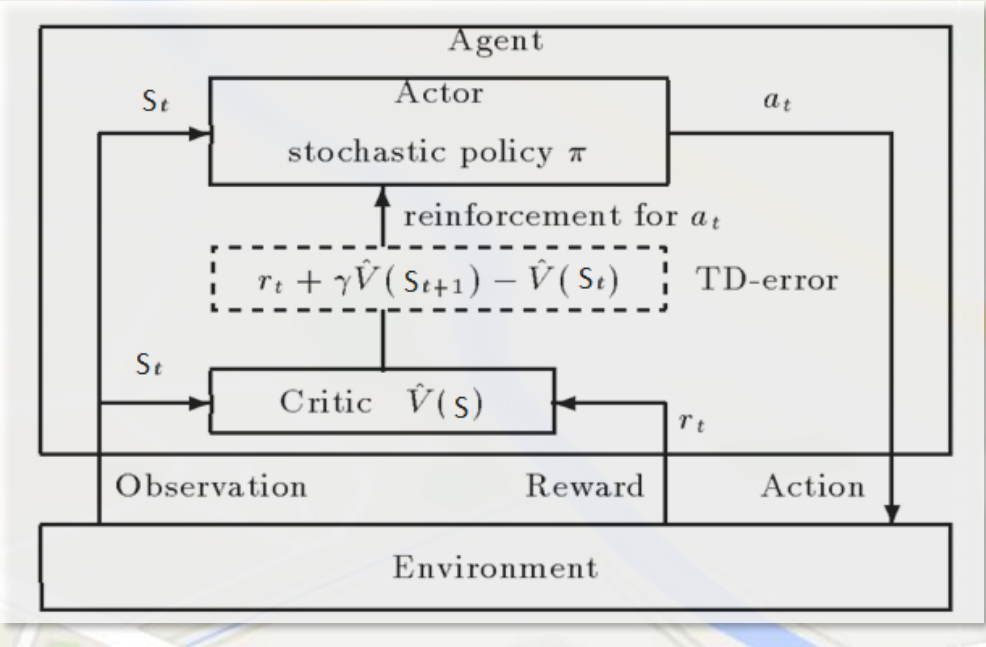
Actor-Critic to algorytm uczenia ze wzmocnieniem, który rozwija ideę Q-learningu, łącząc: szacowanie wartości stanu, uczenie się strategii działania (tzw. polityki).

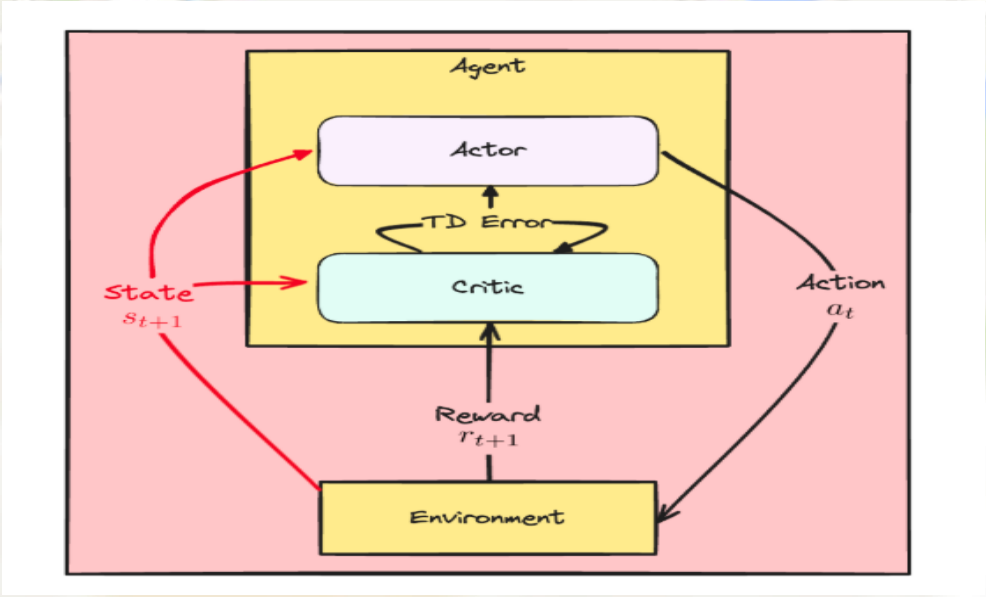
Dzięki temu agent potrafi zarówno oceniać sytuację, jak i decydować, co zrobić – co sprawia, że jest bardziej elastyczny od prostych metod opartych tylko na tablicach wartości.

Aktor (Actor) – wybiera akcję w danym stanie, ucząc się polityki działania.
Krytyk (Critic) – ocenia działania Aktora, wyznaczając błąd TD (różnica między przewidywaną a rzeczywistą nagrodą).
Aktor uczy się podejmowania lepszych decyzji na podstawie informacji zwrotnej od Krytyka, który aktualizuje swoje oceny na podstawie błędu prognozy.

Algorytm może działać w sposób deterministyczny (np. wybierając najlepszą akcję) lub stochastyczny (np. losując akcję z rozkładu).

Dlaczego Actor-Critic?
Łączy zalety metod opartych na wartości i na polityce.
Umożliwia działanie w ciągłych i dynamicznych środowiskach, jak ruch drogowy.
Daje szybszą i stabilniejszą naukę niż klasyczne algorytmy.





UCZENIE GŁĘBOKIE, APROKSYMACJA FUNKCJI W ACTOR-CRITIC

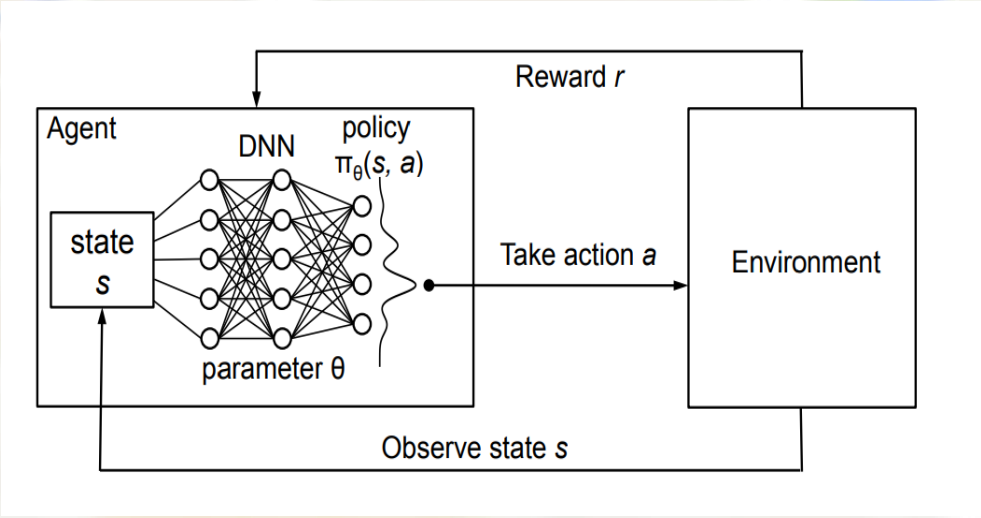
Aproksymacja funkcji w Actor-Critic
W klasycznych algorytmach uczenia ze wzmocnieniem, wartości stanów V(s) lub par Q(s, a) przechowywane są w tablicach — czyli przypisuje się każdej możliwej sytuacji konkretną liczbę.
To działa dobrze tylko w małych i dyskretnych przestrzeniach, gdzie można „zapisać wszystko". W bardziej złożonych środowiskach (np. sterowanie ruchem drogowym), takie podejście staje się niepraktyczne lub wręcz niemożliwe.

Co zastępuje aproksymacja funkcji?
Zamiast trzymać gotowe wartości w tablicach:

V(s) staje się funkcją **V(s; θ)** – aproksymowaną np. przez sieć neuronową,
πθ(a|s) – czyli polityka – również staje się funkcją, siecią wyjściową aktora.

Aktualizacja tablicy Q lub wartości V zostaje zastąpiona obliczaniem gradientu i aktualizacją parametrów funkcji.

Dzięki temu:
Nie trzeba przechowywać dużych tablic stan–akcja.
Obliczenia są szybsze i prostsze – funkcja przyjmuje stan i od razu zwraca wynik,
Algorytm zużywa mniej pamięci i zasobów, co ma znaczenie przy długim trenowaniu lub działaniu w czasie rzeczywisty



ŚRODOWISKO SYMULACYJNE SUMO + TRACI

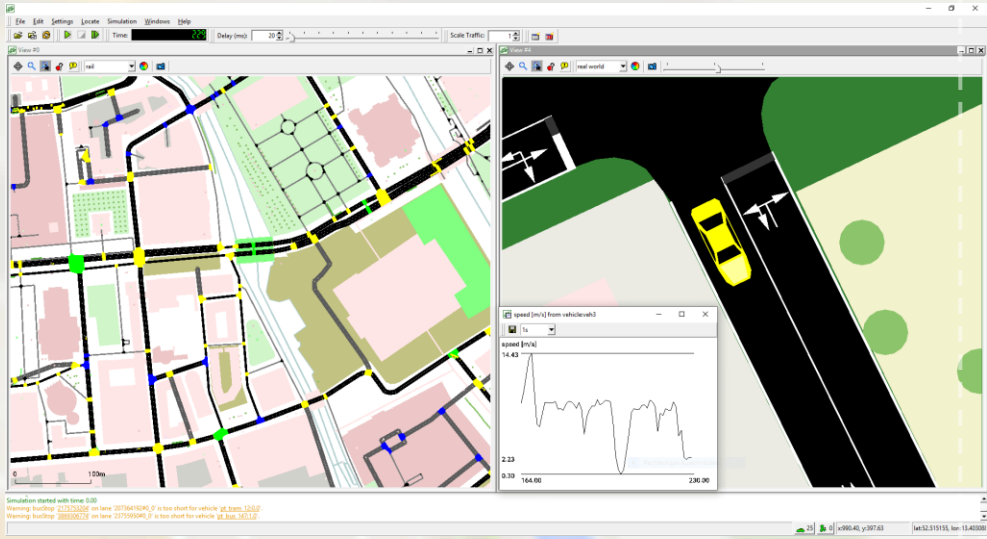
SUMO (Simulation of Urban MObility) to otwartoźródłowy mikrosymulator ruchu drogowego, który umożliwia dokładne modelowanie zachowań pojedynczych pojazdów. Pozwala tworzyć realistyczne sieci dróg, definiować skrzyżowania, sygnalizację świetlną, źródła ruchu oraz pojazdy komunikacji publicznej.

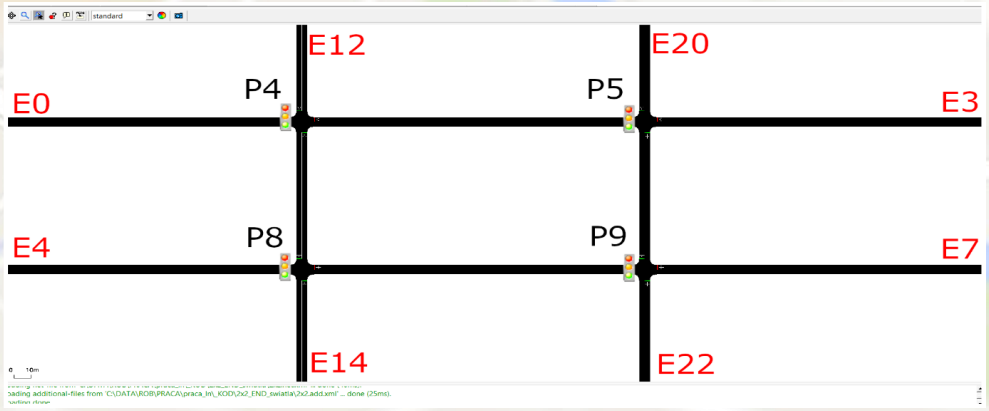
W pracy wykorzystano sieć składającą się z czterech skrzyżowań i ośmiu wlotów – zróżnicowany, realistyczny model ruchu.

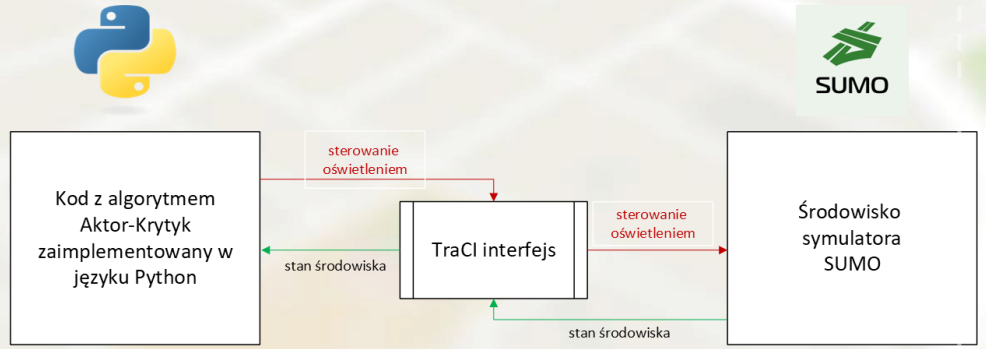
TraCI (Traffic Control Interface)
Interfejs umożliwiający dwukierunkową komunikację między Python-em a SUMO w czasie rzeczywistym.

Dzięki TraCI agent RL może:
Odczytywać dane o stanie ruchu (kolejki, czas oczekiwania, pojazdy),
sterować światłami: traci.trafficlight.setRedYellowGreenState()
wykonywać akcje co krok symulacji i na bieżąco reagować na sytuację na skrzyżowaniu.

Dlaczego SUMO + TraCI ?
Łatwa integracja z Python-em,
Niskie wymagania sprzętowe,
Możliwość pełnej kontroli środowiska do testów algorytmu RL.







IMPLEMENTACJA ALGORYTMU W JĘZYKU PYTHON

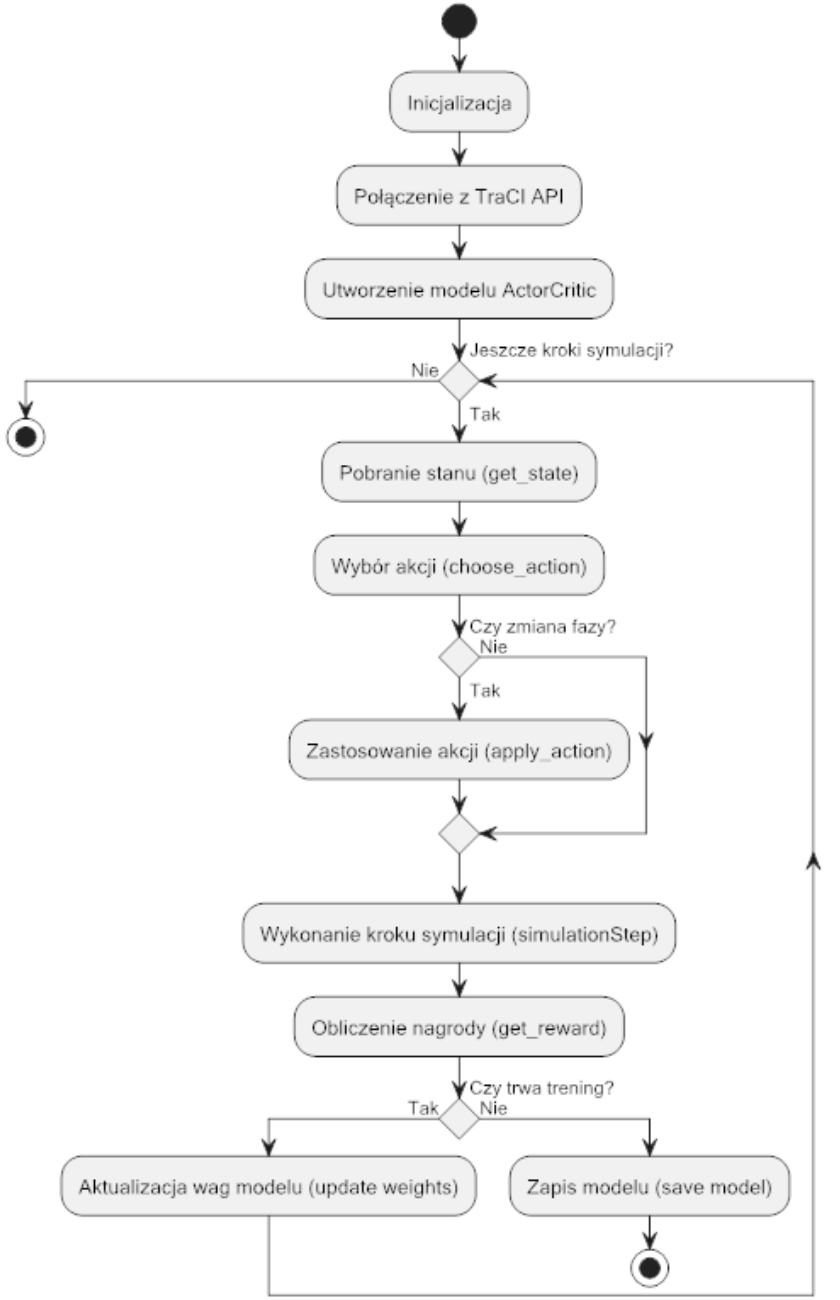
Inicjalizacja i połączenie z TraCI API
Uruchomienie symulacji SUMO oraz nawiązanie połączenia z Pythonem.

Utworzenie modelu Actor-Critic
Inicjalizacja architektury sieci neuronowej aktora i krytyka.

Pętla działania, iteracja (dla każdego kroku symulacji):
- Pobranie aktualnego stanu (get_state) – długości kolejek, ilość zatrzymanych pojazdów
- Wybór akcji (choose_action) – np. zmiana fazy świateł.
- Zastosowanie akcji (apply_action), jeśli decyzja tego wymaga.
- Wykonanie kroku symulacji (simulationStep).
- Obliczenie nagrody (get_reward) – np. na podstawie łącznego czasu oczekiwania.

Uczenie i zapis modelu:
- Jeśli trwa trening, następuje aktualizacja wag sieci (update weights).
- Po zakończeniu treningu model zostaje zapisany (save model).

```
import traci                # Interfejs komunikacji z SUMO
import numpy                # Operacje na danych wejściowych
import tensorflow           # Uczenie głębokie (sieci neuronowe)
```



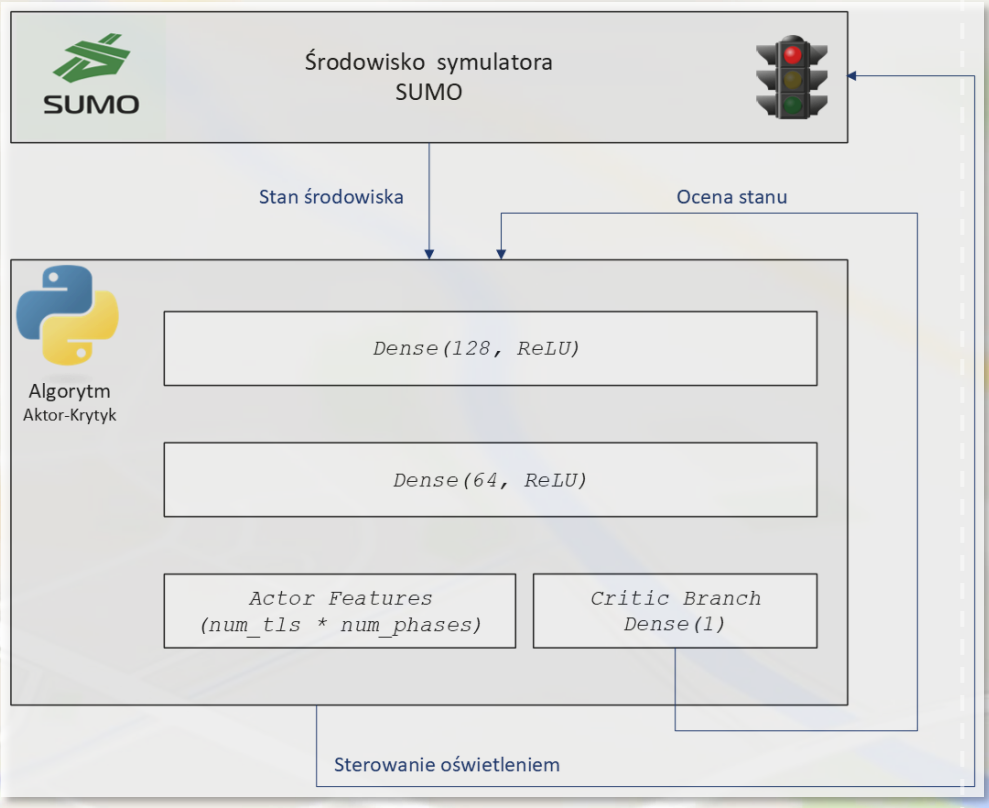
SIEĆ NEURONOWA I PROCES UCZENIA MODELU

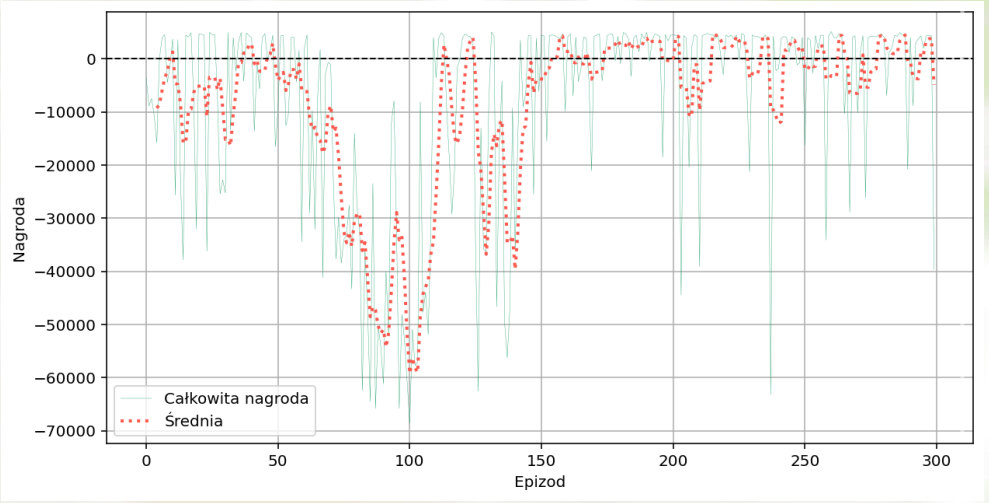
Warstwa wspólna (self.common)
Wektor wejściowy (1,8) (długość kolejki, czasy oczekiwania).
Dense(128) + ReLU
Dense(64) + ReLU

Warstwa aktora (self.actor)
Na podstawie danych o ruchu wyznacza prawdopodobieństwa dla każdej fazy świateł. Dzięki funkcji softmax sieć wybiera fazę najbardziej odpowiednią w danej sytuacji.

Warstwa krytyka (self.critic)
Ocena bieżącej wartości stanu ruchu (V(s)), czyli przewidywanie skumulowanej przyszłej nagrody. Zwraca skalarną wartość stanu.
Wartość ta służy do obliczenia błędu TD i aktualizacji polityki Aktora.

TRENING
Na początku treningu agent eksplorował losowo – widać to po bardzo niskich i silnie niestabilnych nagrodach z pojedynczymi skokami do wartości wysokich. Około 150. epizodu średnia krocząca zaczęła rosnąć, a fluktuacje się zmniejszyły, co świadczy o wypracowaniu efektywnej polityki. W późniejszych epizodach nagrody stają się coraz wyższe i bardziej stabilne





WYNIKI SYMULACJI

Przebieg kluczowych wskaźników w trakcie symulacji SUMO sterowanej modelem AI – widzimy, jak zmieniały się średnia prędkość, liczba zatrzymanych pojazdów oraz czas oczekiwania.

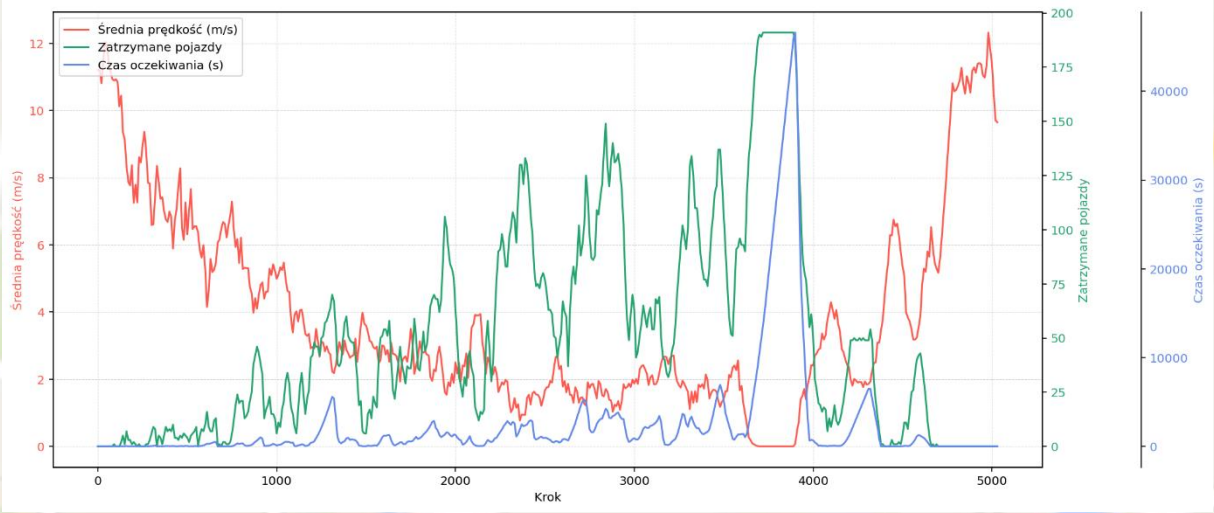
Analiza wyników
Duża zmienność i skrajne wartości wynikają z celowego przeciążenia środowiska testowego.

Średnia prędkość utrzymuje się na stosunkowo stabilnym poziomie, choć w warunkach ekstremalnego przeciążenia obserwujemy chwilowe zatrzymanie ruchu.

Liczba zatrzymanych pojazdów rośnie gwałtownie w momentach kulminacyjnego ruchu, ale model skutecznie przywraca płynność.

Czas oczekiwania osiąga wysokie wartości w szczytowych momentach, lecz po ustąpieniu zatoru szybko się stabilizuje

Model AI dostosowuje długości i kolejność faz sygnalizacji do zmieniającego się natężenia ruchu. Jednak w momentach największego przeciążenia obserwujemy kulminację zatorów. Mimo to agent potrafi szybko reagować na te szczytowe sytuacje, skracając czas oczekiwania zaraz po przejściu przez najgorsze punkty korka.



Wskaźnik	Średnia	Max. wartość	Odchylenie standardowe
Średnia prędkość (m/s)	3,86	12,80	2,94
Zatrzymane pojazdy	50,40	191,00	49,35
Czas oczekiwania wszystkich pojazdów (s)	2536,35	47 445,00	6594,54

PORÓWNANIE STRATEGII STEROWANIA RUCHEM

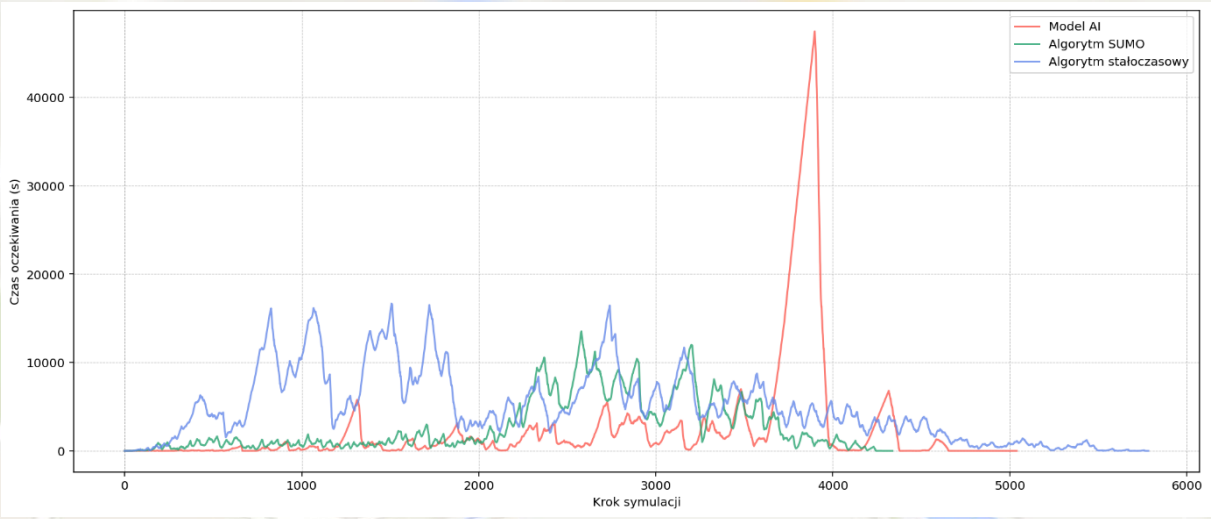
Alternatywne metody:

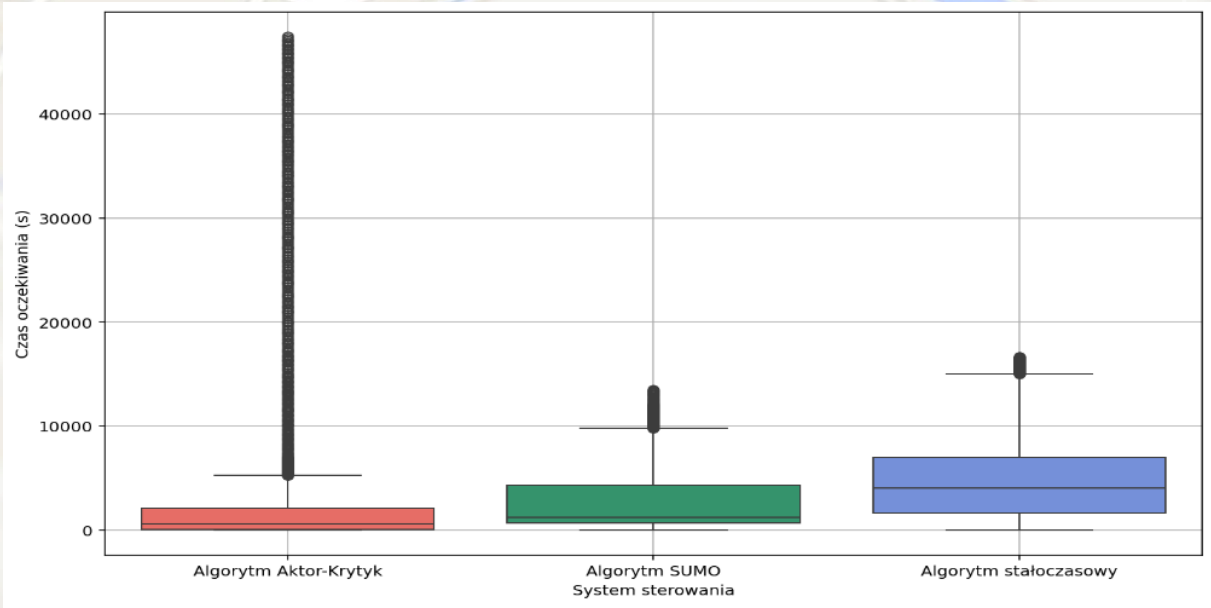
Algorytm SUMO – domyślny, adaptacyjny system zmieniający czasy i sekwencje faz w oparciu o bieżące natężenie ruchu.
Sterowanie stałoczasowe – każda faza ma jednakową, niezmienną długość, niezależnie od warunków.

Czas oczekiwania pojazdów
Model AI utrzymuje względnie stabilny poziom czasu oczekiwania, szybko reagując na pojawiające się zatory.
Algorytm SUMO radzi sobie dobrze przy umiarkowanym ruchu, ale przy skrajnych natężeniach wykazuje wydłużone przestoje.

Anomalia i adaptacja
W jednym kroku symulacji zanotowano pik ok. 47 000 s – ekstremalna wartość, po której model AI samoregulacją szybko powrócił do stabilnego poziomu.

Czas oczekiwania pojazdów (Rys. 17)
Na wykresie skrzynkowym widać, że większość pomiarów czasu oczekiwania jest najniższa dla modelu AI, nawet pomimo pojedynczych skrajnych wartości.





PORÓWNANIE STRATEGII STEROWANIA RUCHEM

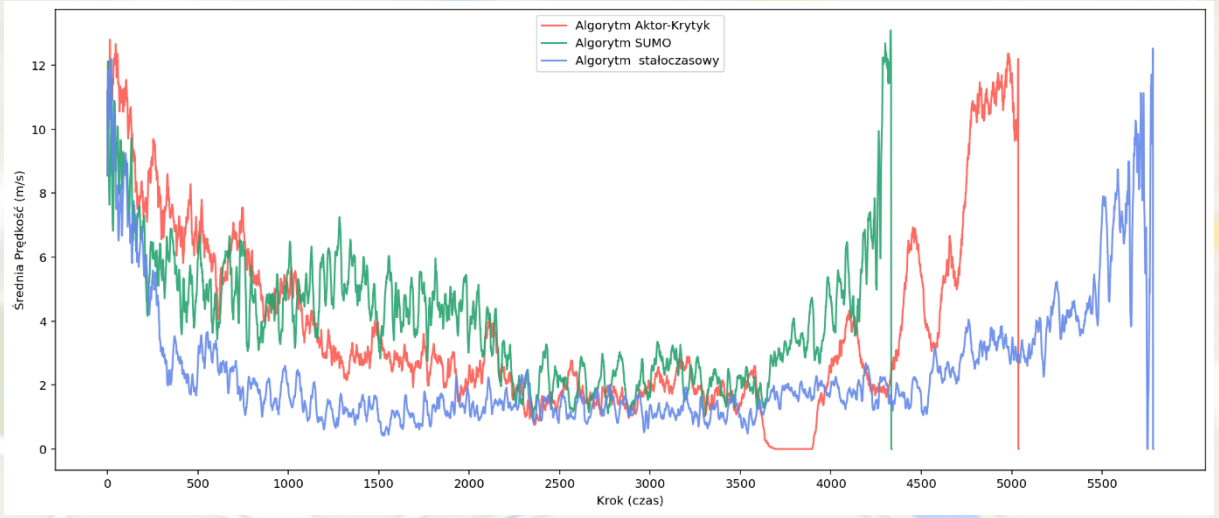
Średnia prędkość pojazdów (Rysunek 18)
Cel pomiaru: średnia prędkość ruchu drogowego – kluczowy wskaźnik płynności i efektywności sterowania.

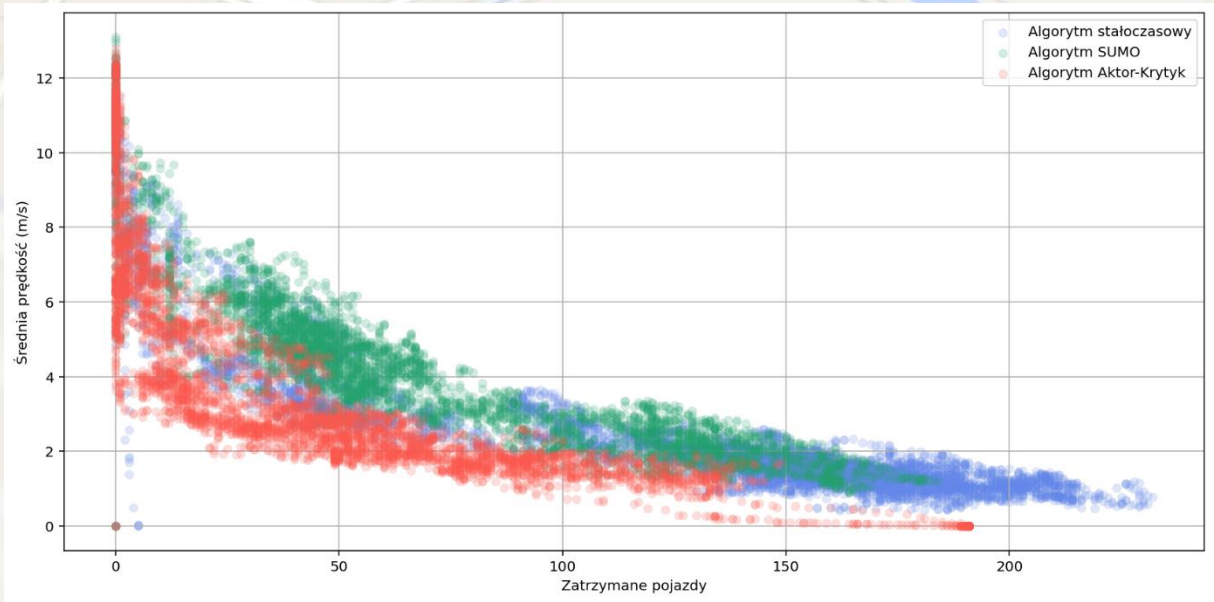
Wniosek: adaptacyjne algorytmy (SUMO, AI) znacząco przewyższają podejście stałoczasowe pod względem utrzymania wysokiej średniej prędkości.

Zależność: liczba zatrzymanych pojazdów vs. średnia prędkość
Analiza rozrzutu punktów pozwala ocenić jednoczesne wpływy dwóch wskaźników:

Actor-Critic: większość punktów w lewym górnym rogu → niskie zatrzymania + wysoka prędkość.

Wniosek: Actor-Critic najskuteczniej utrzymuje równowagę między minimalizacją zatorów a maksymalizacją prędkości, potwierdzając swoją adaptacyjność w zmiennych warunkach ruchu.



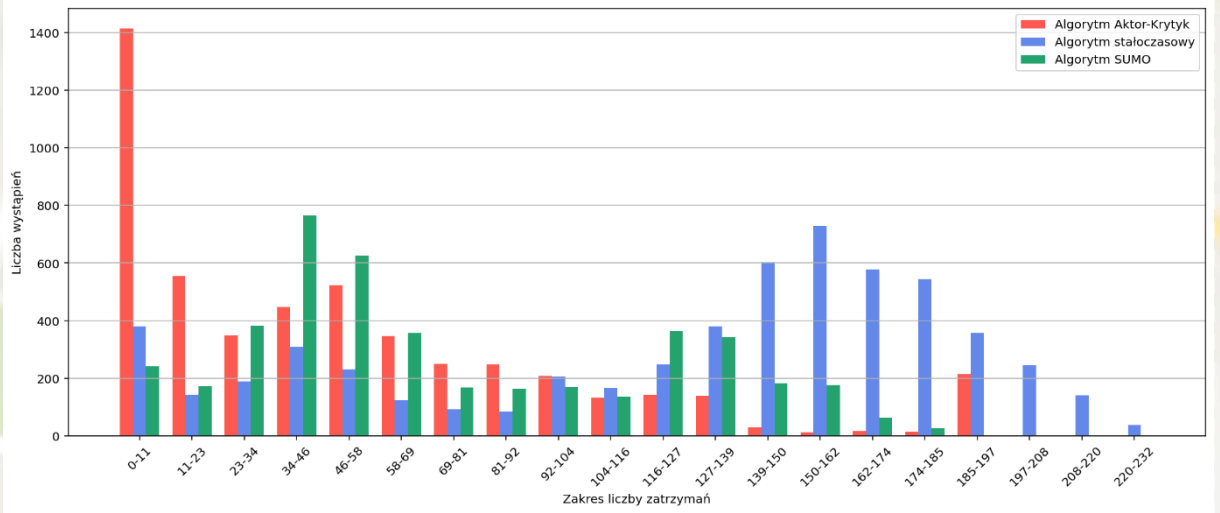


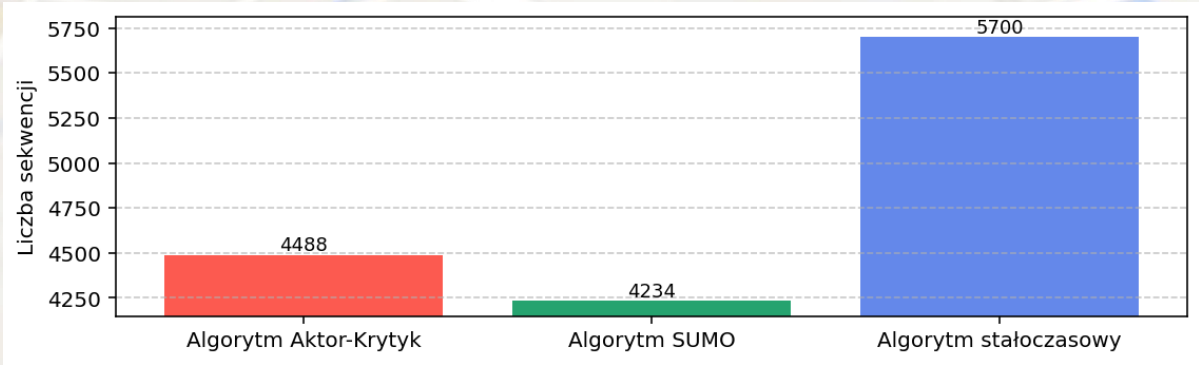
PORÓWNANIE STRATEGII STEROWANIA RUCHEM

Rozkład liczby zatrzymanych pojazdów

Actor–Critic konsekwentnie utrzymuje niską liczbę zatrzymań (dominują przedziały 0–20), co świadczy o jego zdolności do ograniczania powstawania korków. Histogram pokazuje wąski rozkład i niewiele wartości skrajnych, co oznacza, że rzadko zdarzają się długotrwałe przestoje. Jednocześnie, choć agent potrzebuje nieco więcej kroków symulacji na całkowite opróżnienie skrzyżowań niż algorytm SUMO, to i tak znacząco krócej niż w przypadku sterowania stałoczasowego , a rozkład tych czasów jest znacznie węższy niż dla trybu stałoczasowego.

Wniosek: Actor–Critic efektywnie łączy minimalizację przestojów z akceptowalnym czasem rozładowania ruchu, dzięki czemu w dynamicznym środowisku zapewnia sprawną i stabilną adaptację.





REZULTATY WNIOSKI

Actor-Critic wyraźnie przewyższył sterowanie stałoczasowe, lecz wciąż nie dorównuje dopracowanemu, bogatszemu w heurystyki algorytmowi SUMO.

Kryterium	Actor-Critic	SUMO	Stałoczasowe
Czas oczekiwania	**umiarkowany**	najlepszy	najdłuższy
Liczba zatrzymań	najmniej	umiarkowanie	najwięcej
Adaptacja do zmian	**wysoka**	bardzo wysoka	Brak

Problemy
Koszt obliczeniowy – nawet przy zwartej architekturze trening wymaga wielogodzinnej symulacji na GPU/Colab.
Lokalne minima – agent często utknął w suboptymalnych strategiach znalezienie balansu między karami a nagrodami było dużym wyzwaniem.

Wyzwania i możliwości rozwoju:
Wieloagentowe sterowanie ruchem (A3C)
Integracja z danymi zewnętrznymi – np. z pojazdów autonomicznych, sieci GSM

Wniosek: Actor-Critic potwierdził, że nawet stosunkowo prosta architektura RL potrafi utrzymać płynność ruchu w warunkach dużego obciążenia, jednak wyniki względem SUMO pokazują, że ten konkretny wariant algorytmu nie jest rozwiązaniem optymalnym.

DZIĘKUJE ZA UWAGĘ