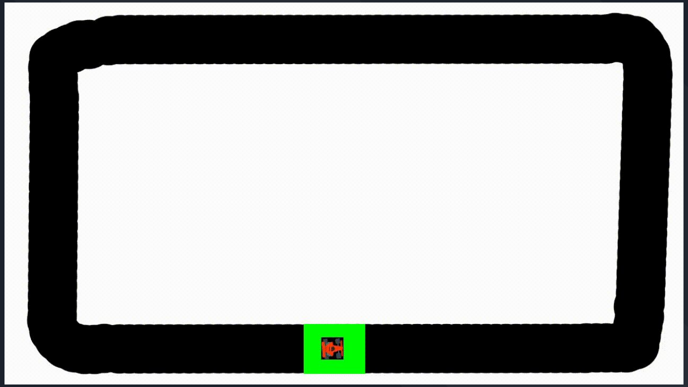
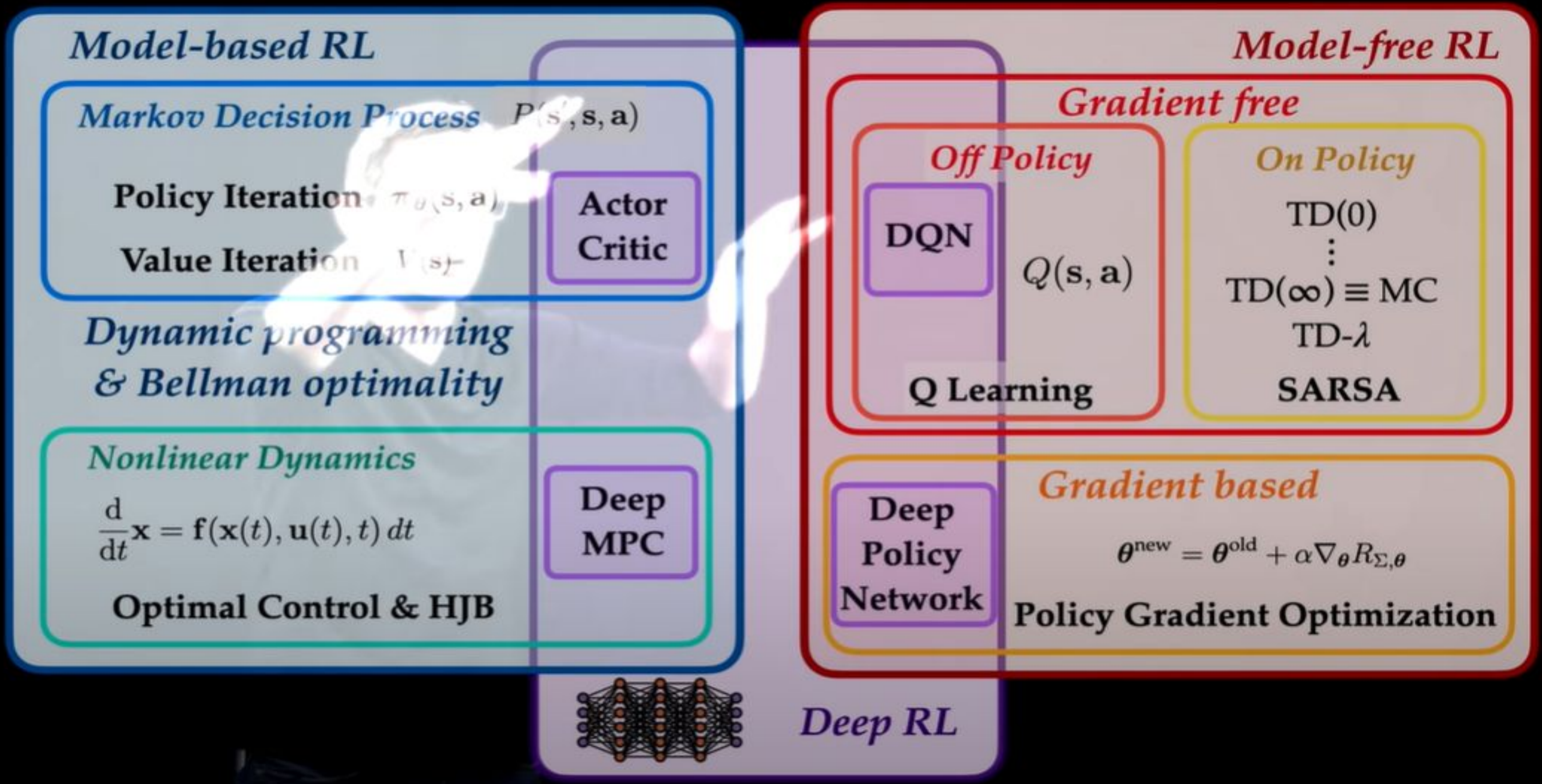
F1 - Reinforcement Learning Algorithms for Self Driving Cars



Florin-Silviu Dinu (331)
Andrei-Vlad Talpiga (331)
Adrian-Octavian Patrascu (332)

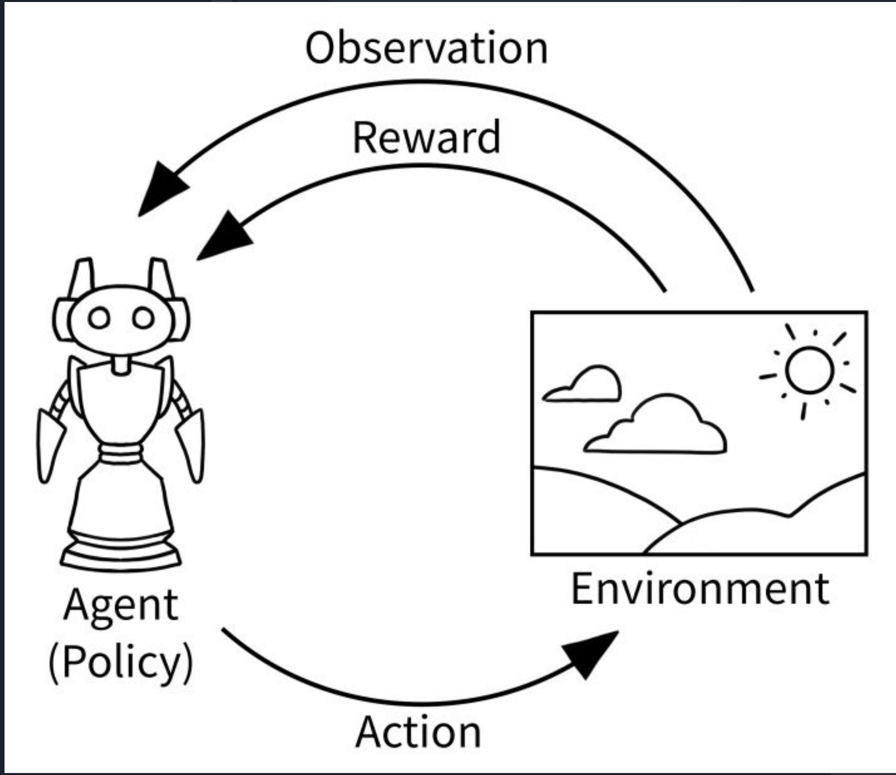
Idee

REINFORCEMENT LEARNING



Implementarea mediilor:

- Pygame si gymnasium
- Algoritmi diferiti, nevoi diferite
- Stari, observatii, recompense



Mediul 1 & 1.1 vs Mediul 2

- Aceleasi actiuni
- Observatii diferite;
- Viteza, unghi, distanta vs senzori distante

Rewards

- DQN: “be alive and fast”
- NEAT: “be a marathon runner”
- QL: “overthink and thrive!”

NEAT

Provocarile NeuroEvolutiei:

- Reprezentarea genetica semnificativa pentru crossover-ul topologiilor
- Protectia inovatiei topologice
- Minimizarea topologiilor in timpul evolutiei

Reprezentarea genetica NEAT a unei retele neurale:

- Fenotip: reteaua neurala
- Genotip: informatia pentru crearea ei
- Crossover: combinarea proprietatilor individuale ale neuronilor si sinapselor
- Mutatii
  - Structurale: adaugarea unei legaturi noi intre neuroni (cu verificarea ciclurilor), stergerea unei legaturi, adaugarea de neuroni in straturile ascunse, stergerea de astfel de neuroni
  - Non-structurale: mutatia unui neuron sau legaturi existente

Setari importante in configurare:

- num_hidden = 0
- num_inputs = 8
- num_outputs = 5
- species_fitness_func = max
- max_stagnation = 20
- species_elitism = 2
- elitism = 3
- survival_threshold = 0.2

Numarul de inputuri este dat de 8 radare si cel de outputuri de 5 miscari posibile

Traseul 1 - 30 generatii

Prima generatie:

Population's average fitness: 17.13222 stdev: 15.76406
Best fitness: 53.90000 - size: (5, 40) - species 1 - id 6
Average adjusted fitness: 0.293
Mean genetic distance 1.078, standard deviation 0.280
Population of 30 members in 1 species:

ID	age	size	fitness	adj fit	stag
1	0	30	53.9	0.293	0

Total extinctions: 0
Generation time: 4.557 sec

Ultima generatie:

Population's average fitness: 107.26556 stdev: 225.44513
Best fitness: 931.50000 - size: (5, 36) - species 1 - id 524
Average adjusted fitness: 0.112
Mean genetic distance 1.069, standard deviation 0.318
Population of 30 members in 1 species:

ID	age	size	fitness	adj fit	stag
1	29	30	931.5	0.112	10

Total extinctions: 0
Generation time: 52.959 sec (52.687 average)

Fitnessul a crescut, iar numarul de conexiuni a scazut in conformitate cu concluziile articolului

Traseul 2 - 60 generatii

Prima generatie:

Population's average fitness: 15.24444 stdev: 15.70978
Best fitness: 39.83333 - size: (5, 40) - species 1 - id 10
Average adjusted fitness: 0.344
Mean genetic distance 1.224, standard deviation 0.197
Population of 30 members in 1 species:

ID	age	size	fitness	adj fit	stag
1	0	30	39.8	0.344	0

Total extinctions: 0
Generation time: 4.635 sec

Ultima generatie:

Population's average fitness: 477.17111 stdev: 951.18665
Best fitness: 2844.53333 - size: (6, 35) - species 2 - id 823
Average adjusted fitness: 0.157
Mean genetic distance 1.503, standard deviation 0.642
Population of 30 members in 2 species:

ID	age	size	fitness	adj fit	stag
2	25	19	2844.5	0.182	24
3	4	11	2844.0	0.131	2

Total extinctions: 0
Generation time: 62.603 sec (92.954 average)

Fitnessul a crescut, iar numarul de conexiuni a scazut in conformitate cu concluziile articolului, de asemenea sunt 2 specii pentru ca sunt foarte distincte

Comparatii intre cei 2 indivizi pe traseele opuse

Distantele indivizilor pe trasee	Traseu 1	Traseu 2
Individ 1	27945	1552
Individ 2	1278	111359

Deep Q-Network

DQN - Teorie 🤔

Q-learning, dar cu Retele in loc de Q-tables 🎉

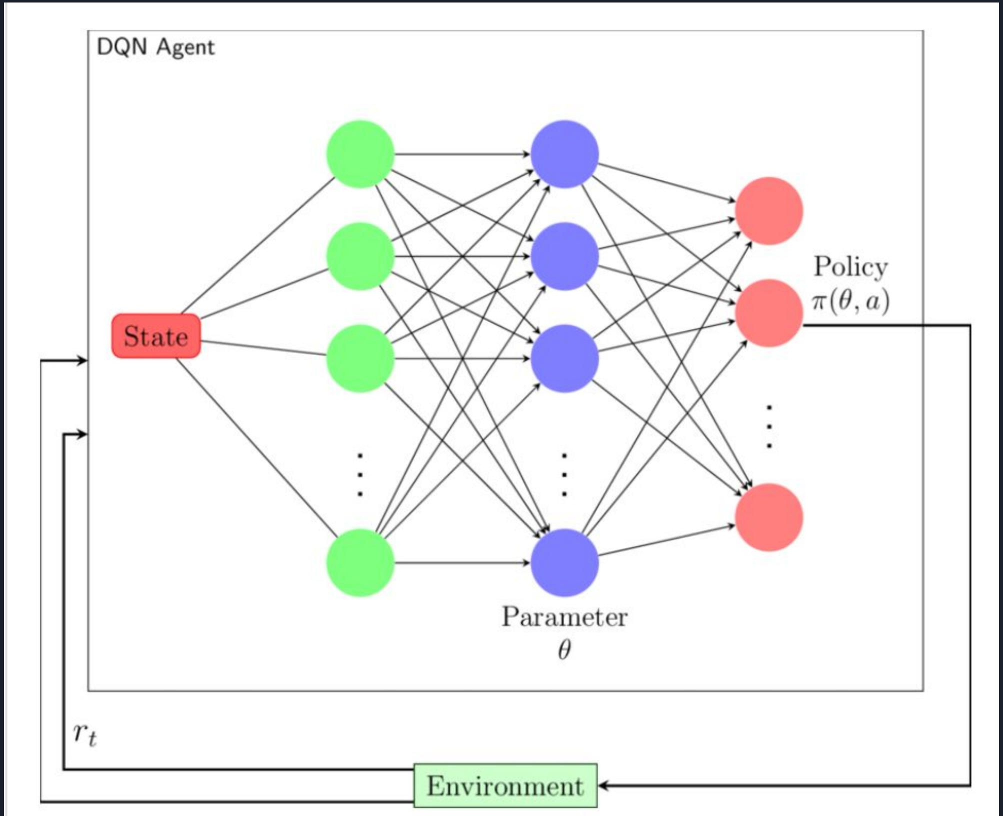
Ofera o aproximare a Q-table-ului, in loc sa le ia barbar pe rand

Avantaje:

Flexibilitate, viteza 🙂

Dezavantaje:

Flexibilitate, viteza 😔



DQN - Teorie 🤔

Action-State: $Q(s,a) = r + \gamma \max_{a'} Q(s',a')$

Target-Policy Networks: Doua NN-uri. Al doilea il actualizeaza pe primul o data la cativa pasi pentru a-l netezi

Experience Replay: Aici stocam amintirile care vor fi preluate in batch-uri pentru lucru pe NN.

Epsilon-Greedy: Exporare-Exploatare

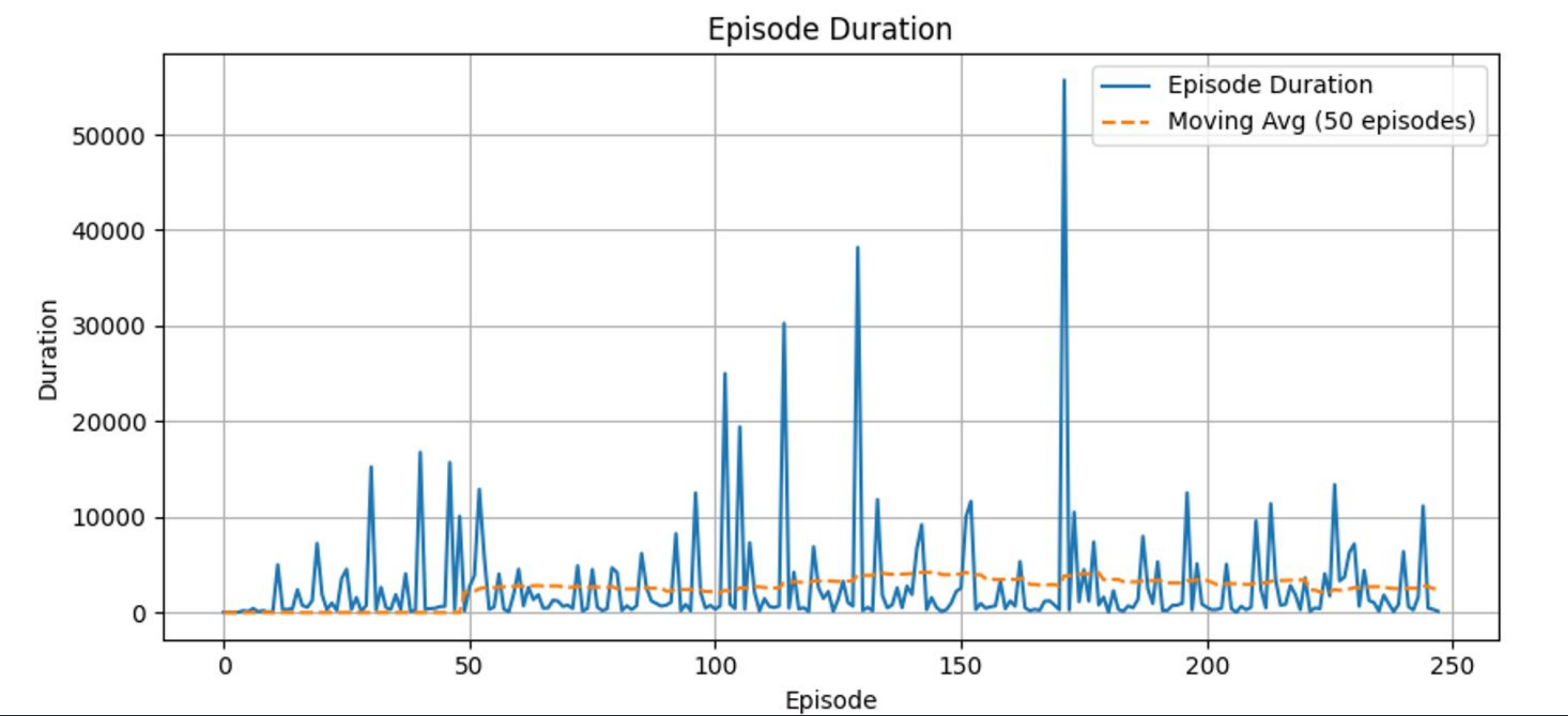

Your friend
You
@successpictures

Reteaua secreta 🤫
Parametri pe sistem🔊

```
BATCH_SIZE = 512
GAMMA = 0.90
EPS_START = 0.95
EPS_END = 0.005
EPS_DECAY = 500
TAU = 0.005
LR = 1e-4
```

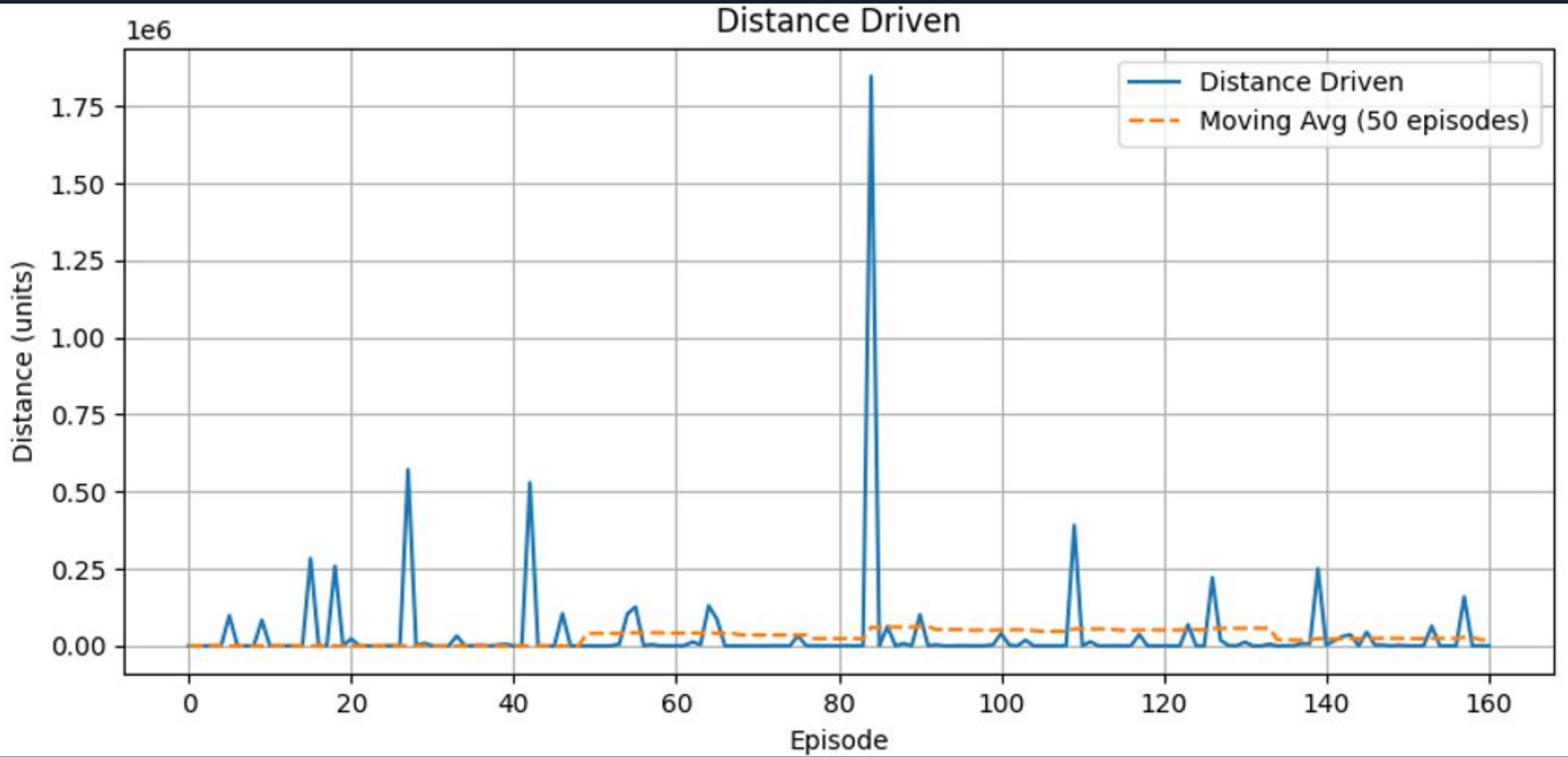
```python
class DQN(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(DQN, self).__init__()
        self.fc1 = nn.Linear(input_dim, 256)
        self.fc2 = nn.Linear(256, 128)
        self.fc3 = nn.Linear(128, 64)
        self.fc4 = nn.Linear(64, output_dim)

        self.bn1 = nn.InstanceNorm1d(256)
        self.bn2 = nn.InstanceNorm1d(128)
        self.bn3 = nn.InstanceNorm1d(64)

        self.dropout = nn.Dropout(p=0.2)

    def forward(self, x):
        x = F.leaky_relu(self.bn1(self.fc1(x)))
        x = self.dropout(x)
        x = F.leaky_relu(self.bn2(self.fc2(x)))
        x = self.dropout(x)
        x = F.leaky_relu(self.bn3(self.fc3(x)))
        x = self.fc4(x)
        return x
```

Track 1



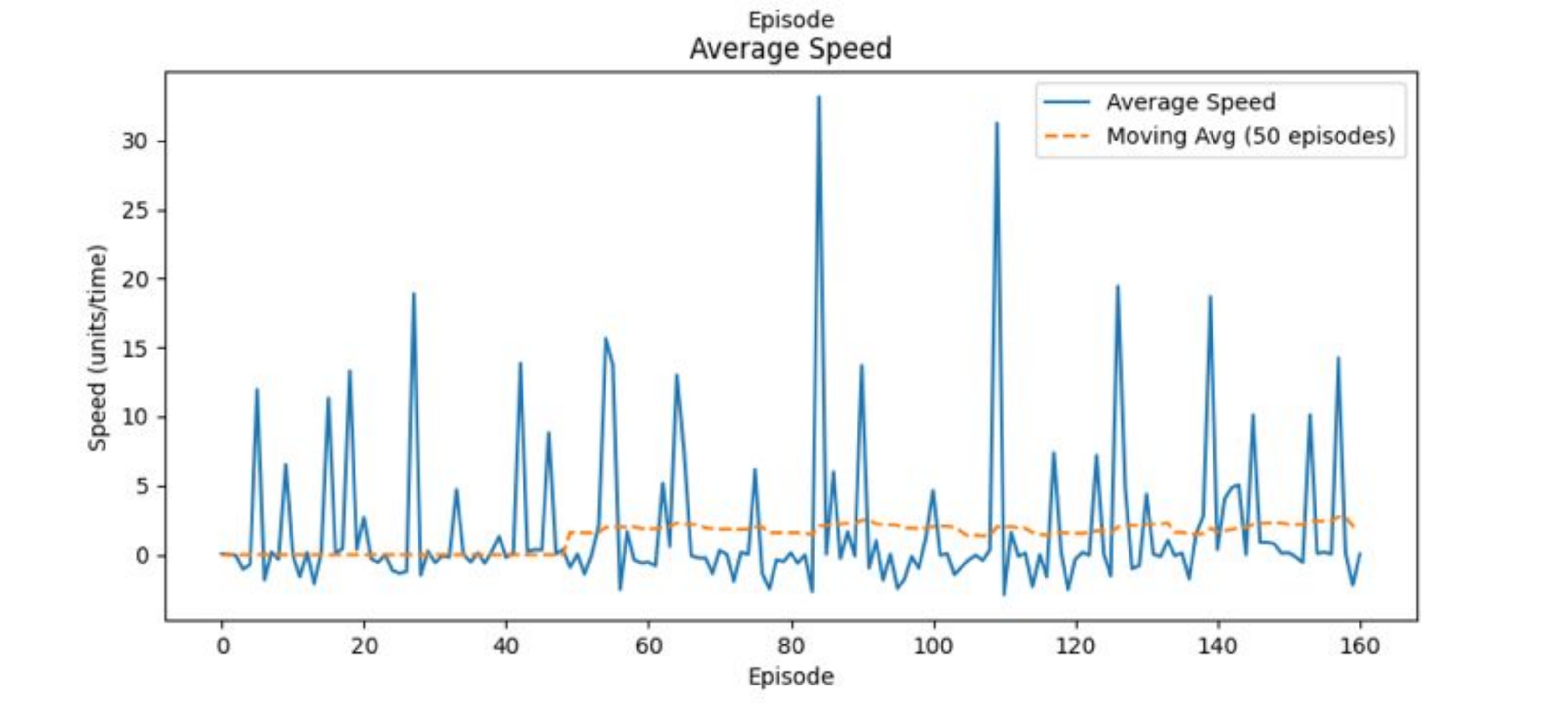
Episode Duration
Episode Duration
Moving Avg (50 episodes)
Duration
50000
40000
30000
20000
10000
0
0
50
100
150
200
250
Episode

Track 1

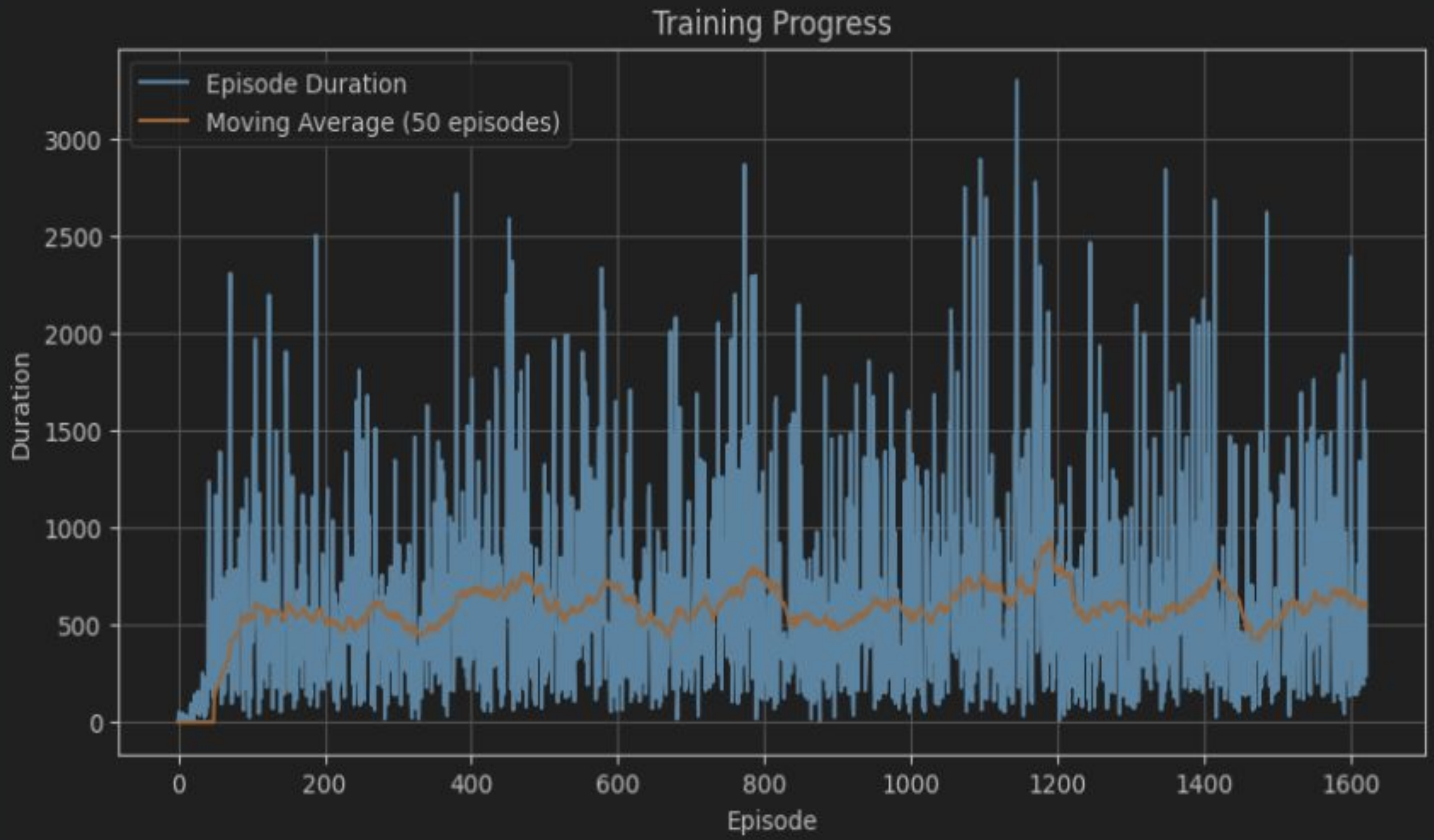


Distance Driven
1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Distance (units)
0
20
40
60
80
100
120
140
160
Episode
Distance Driven
Moving Avg (50 episodes)

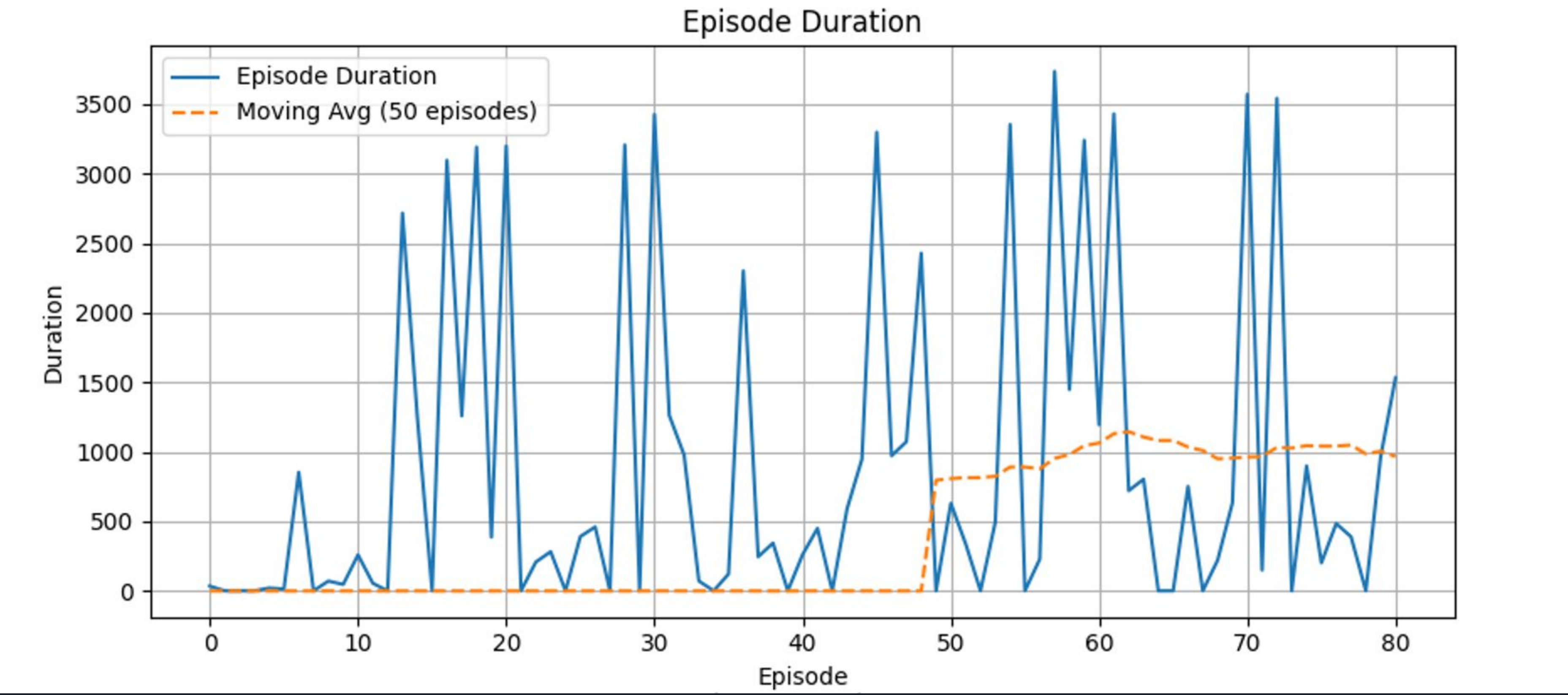
Track 1



Episode
Average Speed
Average Speed
Moving Avg (50 episodes)
Speed (units/time)
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140
160
Episode

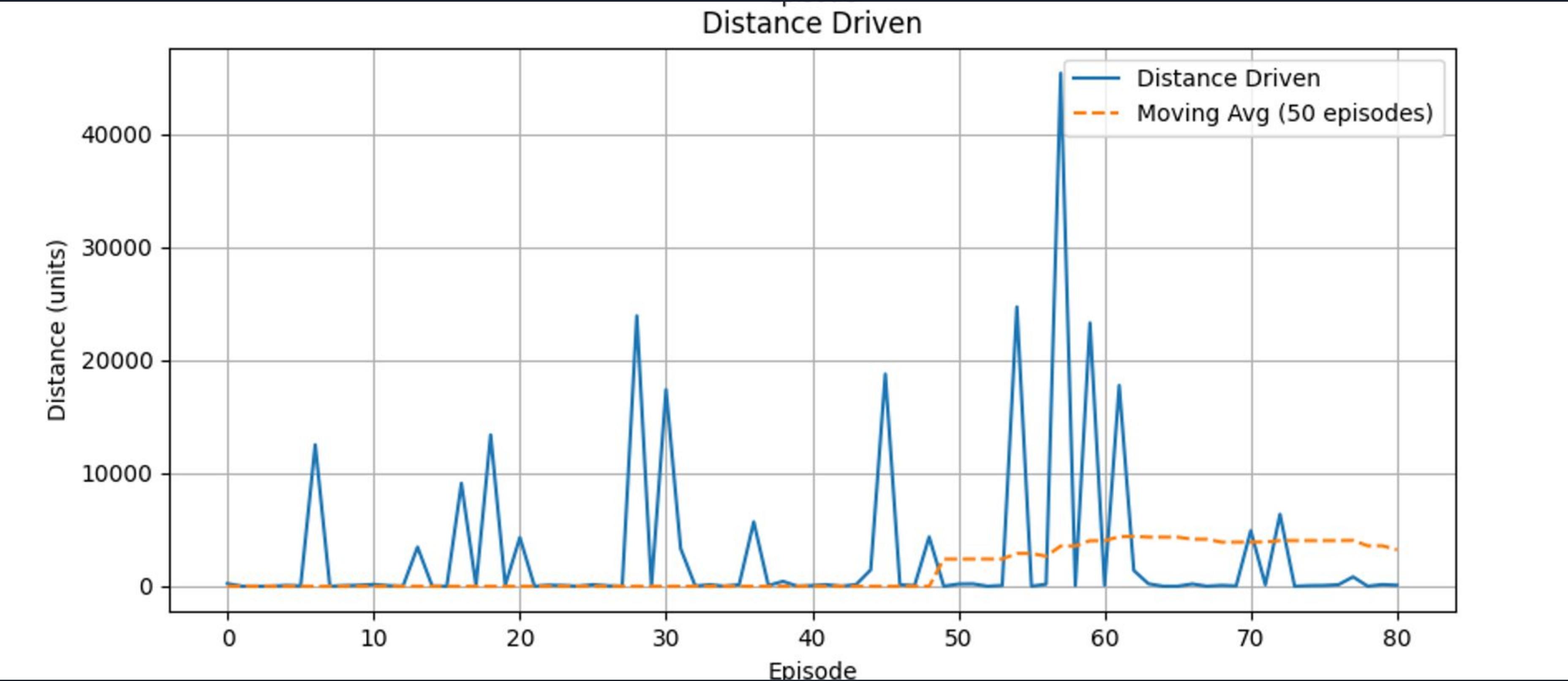

Training Progress
Episode Duration
Moving Average (50 episodes)
Duration
Episode
0
500
1000
1500
2000
2500
3000
0
200
400
600
800
1000
1200
1400
1600

Track 2



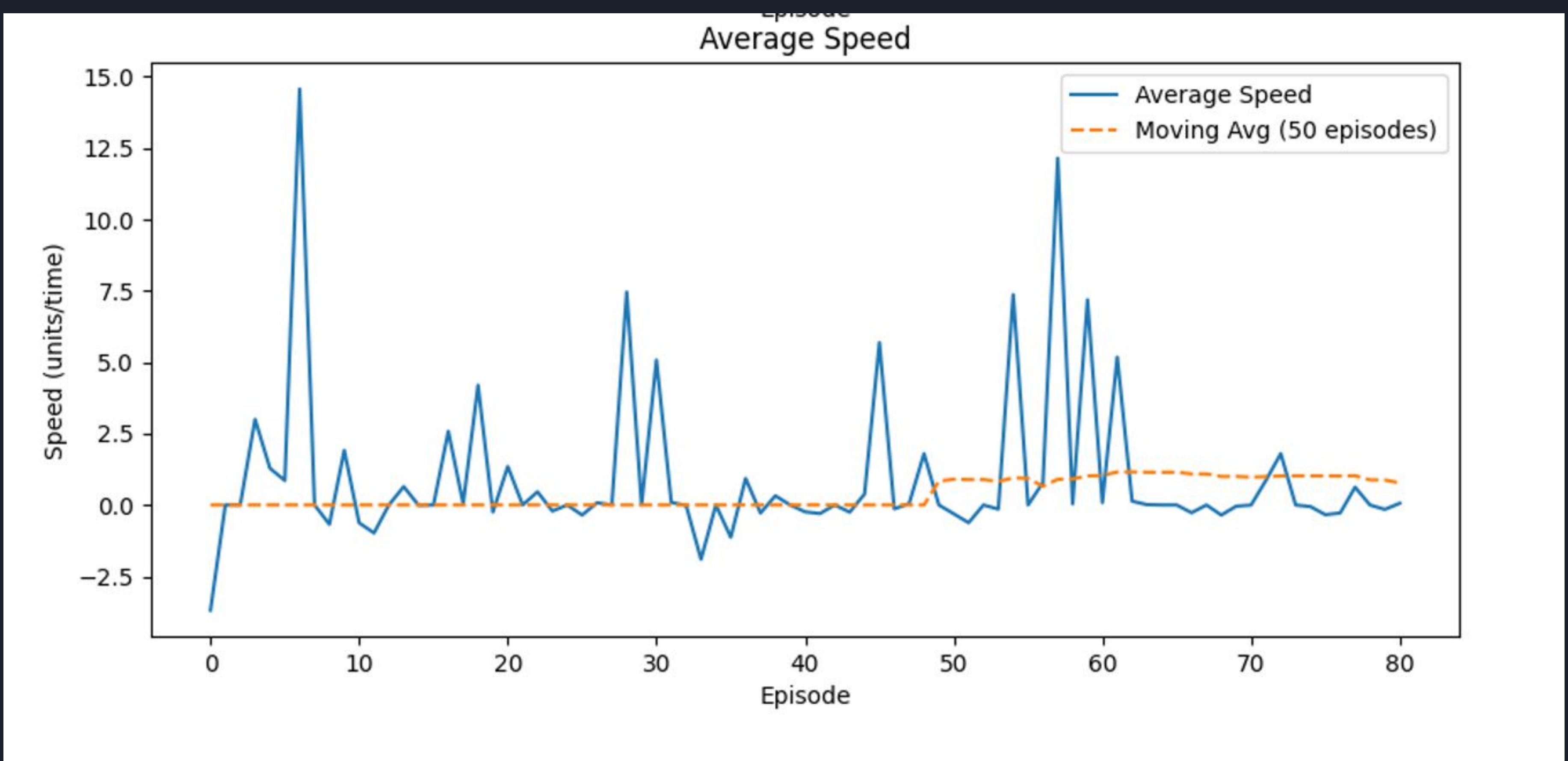
Episode Duration
Episode Duration
Moving Avg (50 episodes)
Duration
Episode
0
500
1000
1500
2000
2500
3000
3500
0
10
20
30
40
50
60
70
80

Track 2



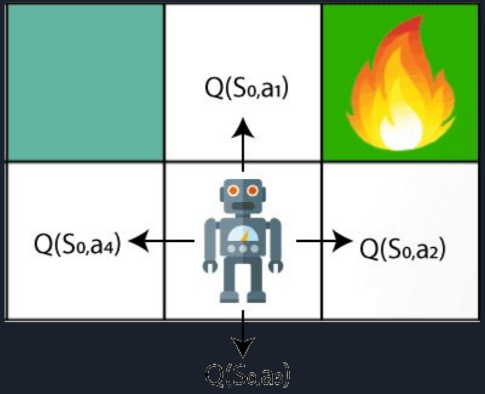
Distance Driven
Distance Driven
Moving Avg (50 episodes)
Distance (units)
0
10000
20000
30000
40000
0
10
20
30
40
50
60
70
80
Episode

Track 2


Average Speed
Average Speed
Moving Avg (50 episodes)
Speed (units/time)
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
0
10
20
30
40
50
60
70
80
Episode

Q-learning

Q-Learning Teorie



- Fara model (Model-free)
- Fara polita (Off-policy)
- Stari (pozitia curenta) si actiuni
- Agentul se antreneaza in mediu (environment) prin incercare si eroare (trial & error)
- Recompensa (reward) pentru fiecare actiune din fiecare stare
- Valorile Q ne ajuta sa alegem actiunea potrivita la o anumita stare
- Adesea aplicat problemelor modelate ca procese de decizie Markov

$$Q^{new}(S_t,A_t) \leftarrow (1-\underbrace{\alpha}_{\text{learning rate}})\cdot\underbrace{Q(S_t,A_t)}_{\text{current value}} + \underbrace{\alpha}_{\text{learning rate}}\cdot\bigg(\underbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}}\cdot\underbrace{\max_a Q(S_{t+1},a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}}\bigg)$$

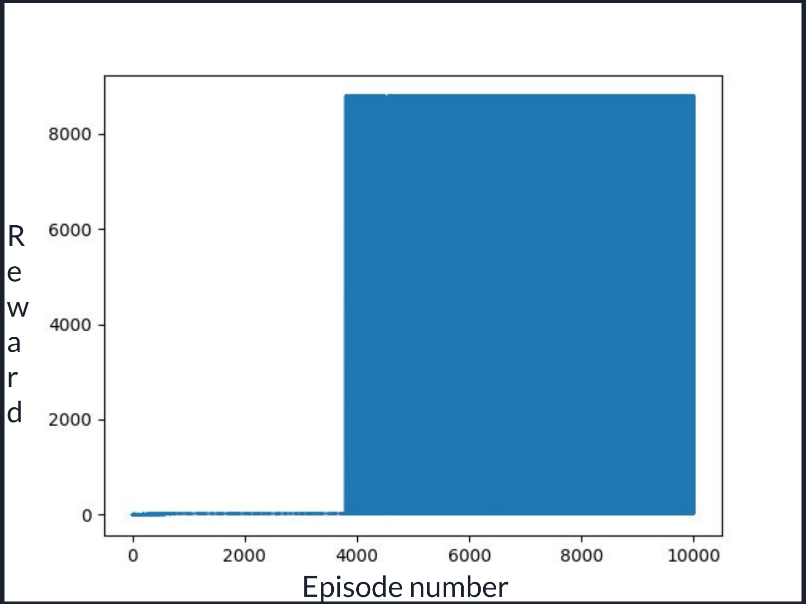
Q-Learning Configuratie

```
epsilon = 1.0
epsilon_decay = 0.00013
lr = 1.0 # alpha
lr_decay = 0.00013
gamma = 0.8

CarAgent.epsilon = max(CarAgent.epsilon
- CarAgent.epsilon_decay, 0.01)

CarAgent.lr = max(CarAgent.lr -
CarAgent.lr_decay, 0.01)
```

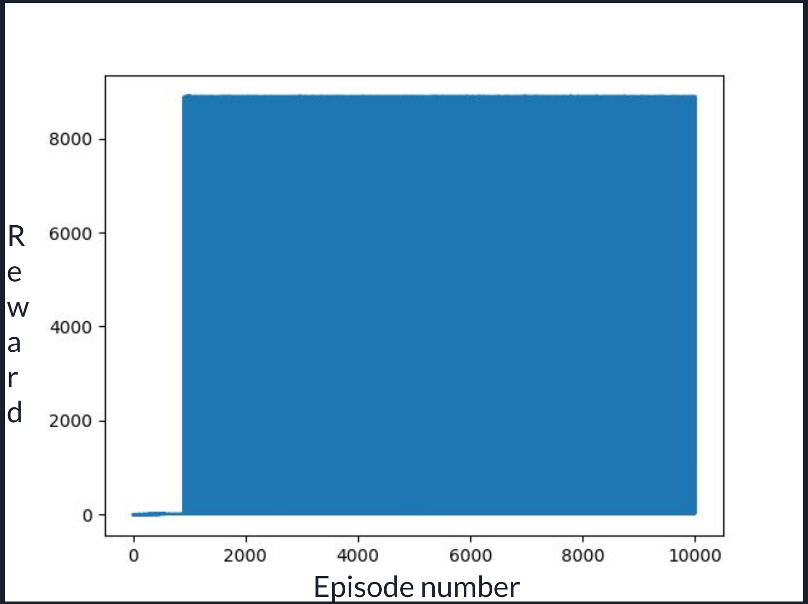
Hiperparametrii alesi permit:

- la inceputul algoritmului: alegerea unor actiuni mai random, favorizand EXPLORAREA
- in final: selectia actiunii care asigura o recompensa mai mare, favorizand EXPLOATAREA

Q-Learning Rezultate



Track 01



Track 02

Comparatii

	DQN	Q-learning	NEAT	Tracks
Distance in 30 seconds / until first accident	~80000	1228	27945	**Track01**
	~3000	1121	111359	**Track02**
Average speed	~3.52	779.65	9.43	**Track01**
	~1.25	582.86	29.84	**Track02**
Episodes/Generations	240	1382	30	**Track01**
	80	913	60	**Track02**

Concluzii

Atat DQN cat si NEAT pot fi folositi cu succes pentru rezolvarea problemei in timp util.

Bibliografie 📚

1. DQN for PyTorch (reinforcement_q_learning.ipynb - Colaboratory (google.com))
2. CONTINUOUS CONTROL WITH DL (1509.02971.pdf (arxiv.org))
3. DQN Explained (DQN Explained | Papers With Code)
4. Car Env Gym (Car Racing - Gymnasium Documentation (farama.org))
5. AI car simulation: (AI Car Simulator - Github)
6. Efficient Evolution of Neural Network Topologies (Paper)
7. NEAT Algorithm from Scratch (NEAT algorithm from scratch (it was hard))
8. Cursurile si laboratoarele de RL ❤️😉