Atividade Prática 1 - Inteligência Artificial

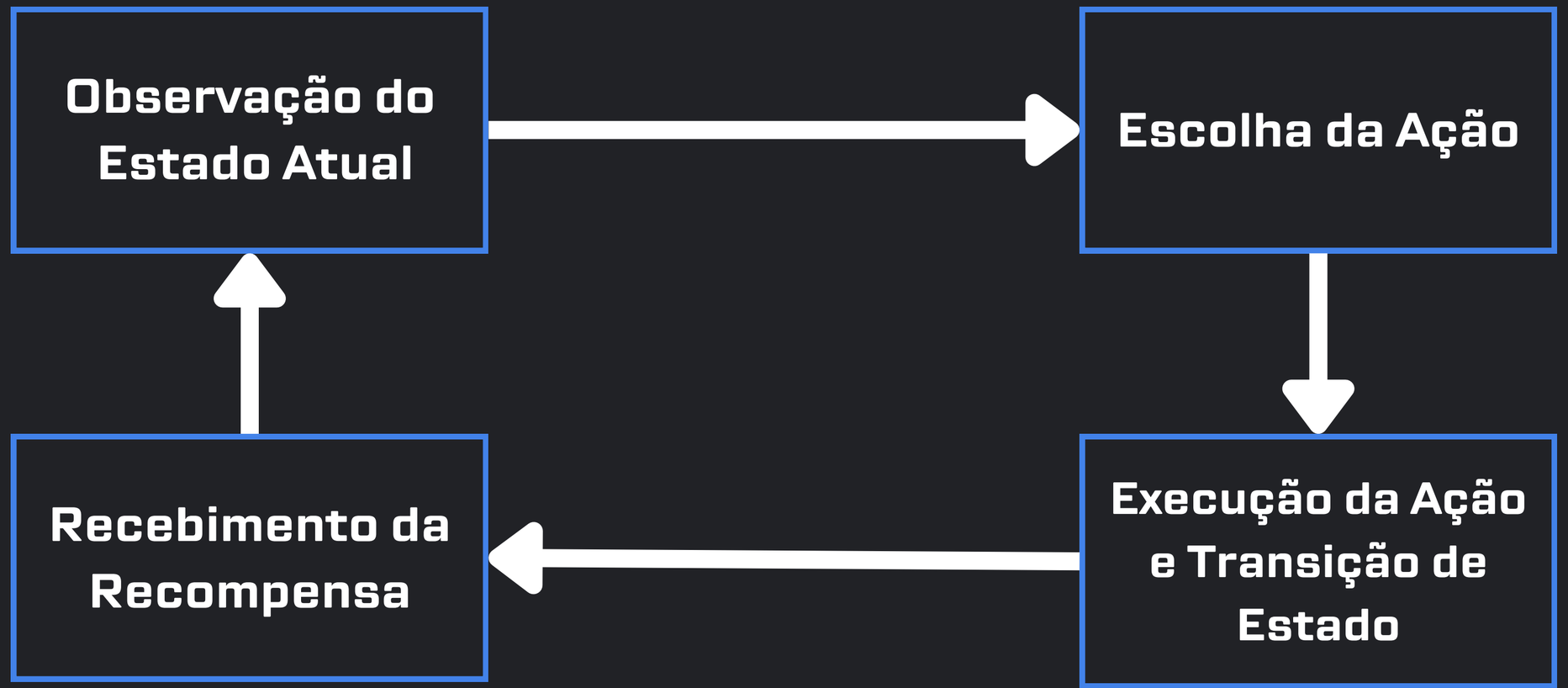
APRENDIZADO POR REFORÇO DEEP Q-NETWORKS

Bruno Innecco
Gabriel Fernandes
Giovanna Pedersoli
João Victor Teramatsu
Pedro Henrique Moreira

Aprendizado por Reforço

INTRODUÇÃO

- Agente
- Ambiente
- Estado
- Ação
- Recompensa
- Política


Observação do Estado Atual
Escolha da Ação
Execução da Ação e Transição de Estado
Recebimento da Recompensa

Q-LEARNING

Ações \ Estados	0	1	2	3	4	5	6	7	8
0	**-2.434**	**-1.32**	**2.4**	**3.2**	**0.23**	-12	**3.122**	**4.23**	**-5.23**
1	-1.434	1.92	3.46	3.9	0.54	9.88	4.8	5.6	2.33
2	-3.634	-1.82	1.1	1.9	0.77	8.9	-14.9	6.76	3.34
3	-2.434	-3.23	1.5	-3.2	1.78	3.43	-7.7	8.88	4.23
4	-3.54	-5.66	2	-4.56	2.2	0.45	-1.2	4.23	7.3
5	-4.34	2.3	3.2	1.4	1.8	-2.2	3.3	-5.4	-1.1

Q-LEARNING


ambiente
política
ação
reforço
agente
estado
+6
107
AMMO
83%
HEALTH
ARMS
176%
ARMOR
BULL 107 / 200
SHEL 24 / 50
ROKT 0 / 50
CELL 0 / 300

Q-LEARNING

$$\mathrm{Q}(e_t, a_t) \leftarrow (1-\alpha) Q(e_t, a_t) + a\,(R(e_t, a_t) + \gamma \max Q\,(e_t + 1, a))$$

e_t = estado no tempo t

a_t = ação no tempo t

α = taxa de aprendizagem

γ = fator de desconto

$\mathrm{Q}(e_t, a_t)$ = qualidade da ação a_t no estado e_t

$R(e_t, a_t)$ = valor do reforço da ação a_t no estado e_t

LIMITAÇÕES

Quais as limitações do Q-Learning?

Escalabilidade e Alta Dimensionanilidade

Convergência Lenta

Dependência de uma Política de Exploração Adequada

Sensibilidade a Hiperparâmetros

Problemas com Recompensas Esparsas

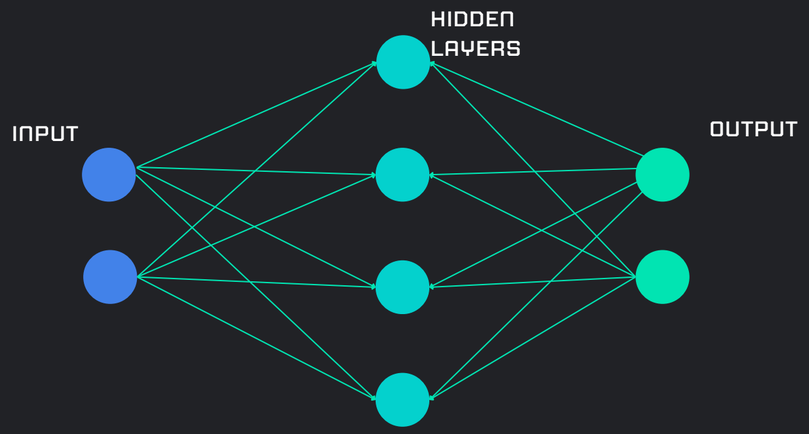
Deep Q-Networks

DQN

- Fases
- Estrutura

COLLECTION PHASE

Estado (s)



Ação (a)

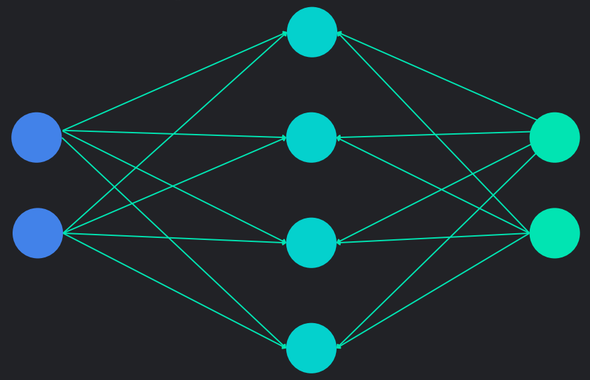
Replay Buffer

(s, a, r, s)

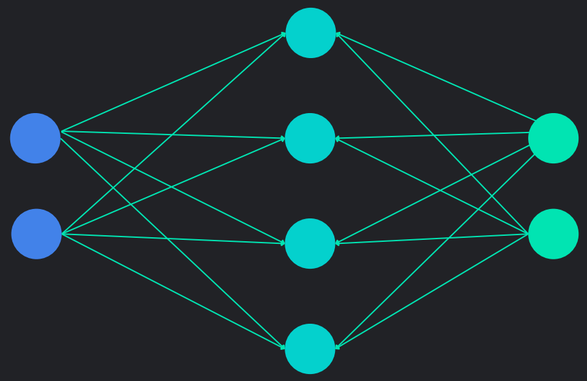
TRAINING PHASE

Q-Network

s1



s2

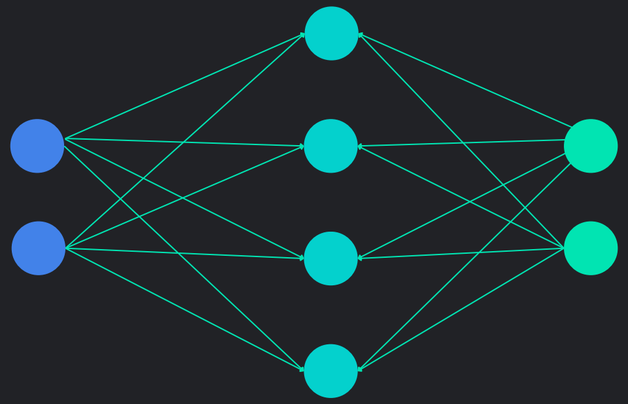


Target Network

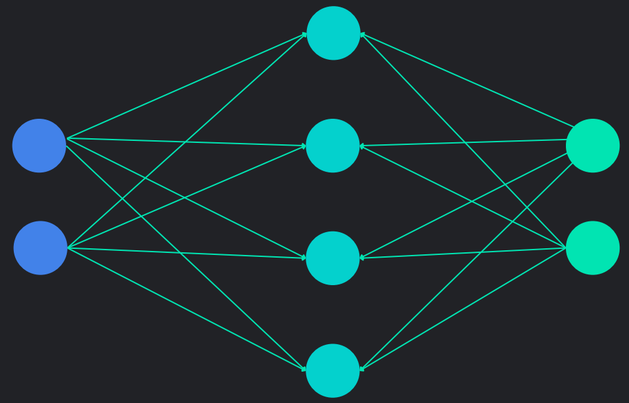
Replay Buffer

(s1, right, -1, s2)

TRAINING PHASE



Q (s1, right) -0.67
Q (s2, left) 1.36

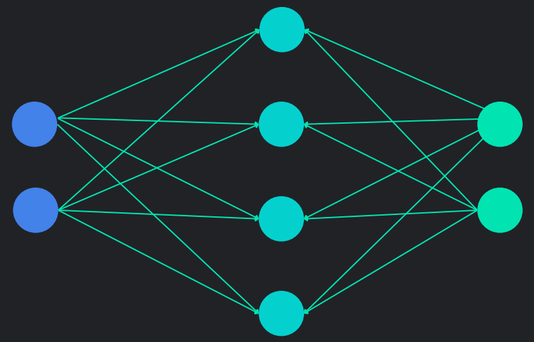


Qt (s1, right) 1.02
Qt (s2, left) 1.42

Replay Buffer

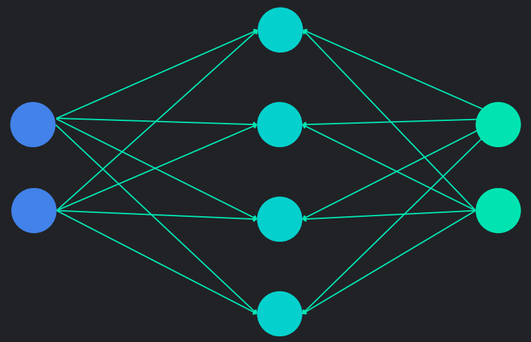
(s1, right, -1, s2)

TRAINING PHASE



Q (s1, right) -0.67

Q (s2, left) 1.36



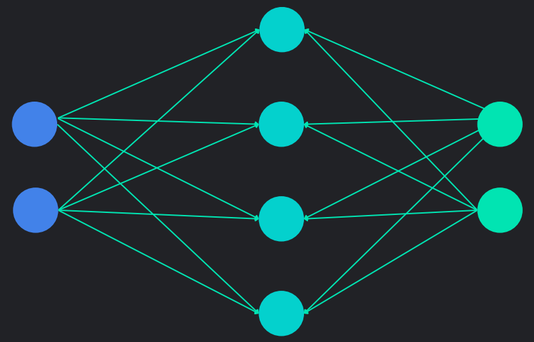
Qt (s1, right) 1.02

Qt (s2, left) 1.42 **+ (-1) = 0.42**

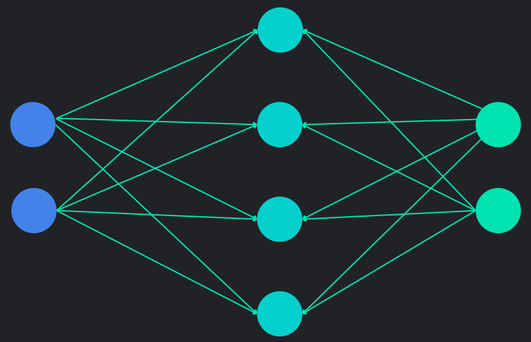
Replay Buffer

(s1, right, **-1**, s2)

TRAINING PHASE



Q (s1, right) -0.67
Q (s2, left) 1.36

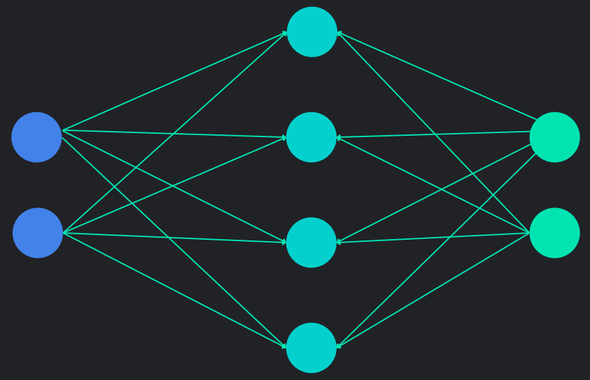


Qt (s1, right) 1.02
Qt (s2, left) 1.42 + (-1) = 0.42

Erro Quadrático Médio → 1.1881

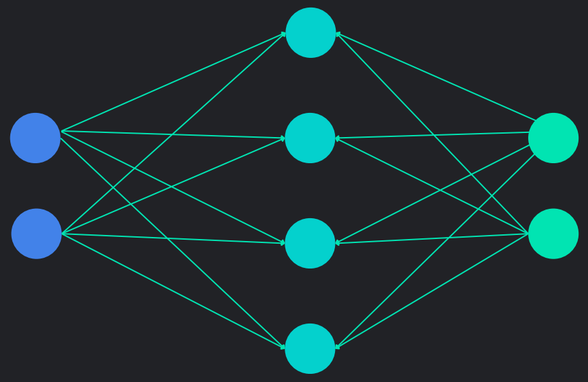
TRAINING PHASE

Q-Network



Retropropagação

1.1881



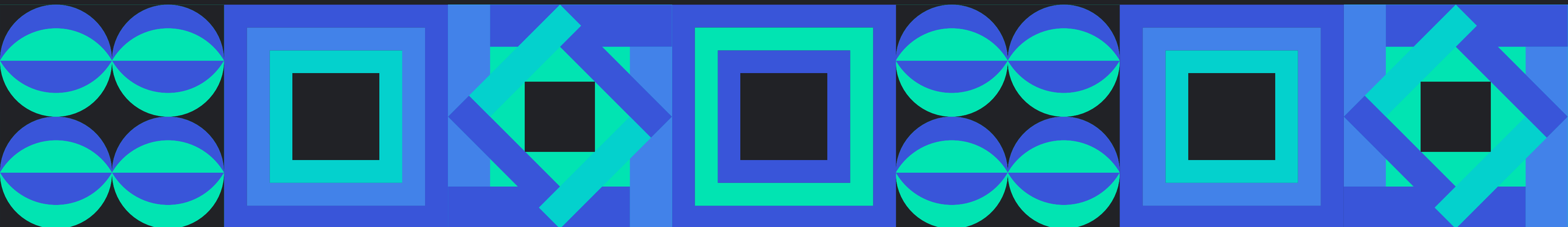
Target Network

ESTRUTURA

1 Rede Neural Profunda

2 Replay buffer
(Memória de Repetição)

3 Target Network
(Rede Alvo)



ALGORITMO

DQN (Deep Q-Network)

- Combinação de Q-Learning e Redes Neurais
- Aprendizado por Reforço
- Aproximação com Redes Neurais
- Replay Buffer
- Rede-Alvo
- Atualizações Periódicas

INICIALIZAÇÃO DOS AGENTES:

```python
class DQN(agent.Agent):
    def __init__(self,
                 environment_spec: specs.EnvironmentSpec,
                 network: snt.Module,
                 batch_size: int = 256,
                 prefetch_size: int = 4,
                 target_update_period: int = 100,
                 samples_per_insert: float = 32.0,
                 min_replay_size: int = 1000,
                 max_replay_size: int = 1000000,
                 importance_sampling_exponent: float = 0.2,
                 priority_exponent: float = 0.6,
                 n_step: int = 5,
                 epsilon: Optional[tf.Tensor] = None,
                 learning_rate: float = 1e-3,
                 discount: float = 0.99,
                 logger: loggers.Logger = None,
                 expert_data: List[Dict] = None) -> None:
```

REPLAY BUFFER :

```
replay_table = reverb.Table(
    name=adders.DEFAULT_PRIORITY_TABLE,
    sampler=reverb.selectors.Prioritized(priority_exponent),
    remover=reverb.selectors.Fifo(),
    max_size=max_replay_size,
    rate_limiter=reverb.rate_limiters.MinSize(1),
    signature=adders.NStepTransitionAdder.signature(environment_spec))
self._server = reverb.Server([replay_table], port=None)
```

DATASET E POLÍTICA:

```
replay_client = reverb.TFClient(address)
dataset = datasets.make_reverb_dataset(server_address=address,
                                       batch_size=batch_size,
                                       prefetch_size=prefetch_size)


if epsilon is None:
    epsilon = tf.Variable(0.05, trainable=False)
policy_network = snt.Sequential([
    network,
    lambda q: trfl.epsilon_greedy(q, epsilon=epsilon).sample(),
])
```

REDE-ALVO (TARGET_NETWORK):

```
target_network = copy.deepcopy(network)
tf2_utils.create_variables(network, [environment_spec.observations])
tf2_utils.create_variables(target_network, [environment_spec.observations])
actor = actors.FeedForwardActor(policy_network, adder)
```

APRENDIZ:

```
learner = learning.DQNLearner(
    network=network,
    target_network=target_network,
    discount=discount,
    importance_sampling_exponent=importance_sampling_exponent,
    learning_rate=learning_rate,
    target_update_period=target_update_period,
    dataset=dataset,
    replay_client=replay_client,
    logger=logger)
```

VANTAGENS

Deep Q-Networks

x

Q-Learning

VANTAGENS

Deep Q-Networks

x

Q-Learning

- **Escalabilidade**
- **Eficiência na Convergência**
- **Estabilidade**
- **Melhor Exploração**

VANTAGENS

Deep Q-Networks

x

Q-Learning

- Escalabilidade
- Eficiência na Convergência
- Estabilidade
- Melhor Exploração

VANTAGENS

Deep Q-Networks

x

Q-Learning

- Escalabilidade
- Eficiência na Convergência
- Estabilidade
- Melhor Exploração

VANTAGENS

Deep Q-Networks

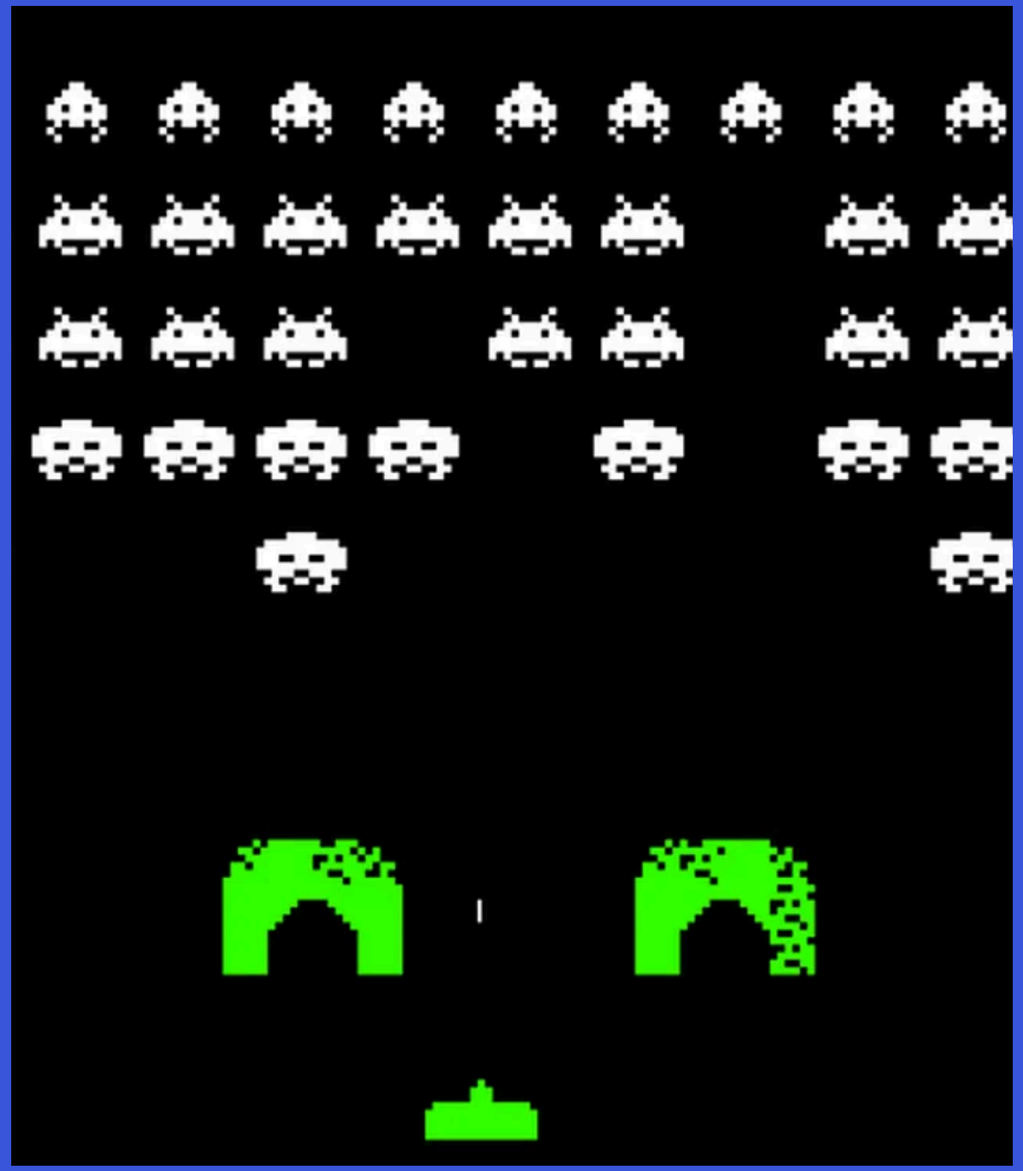
x

Q-Learning

- Escalabilidade
- Eficiência na Convergência
- Estabilidade
- Melhor Exploração

EXEMPLO DE APLICAÇÃO

Space Invaders



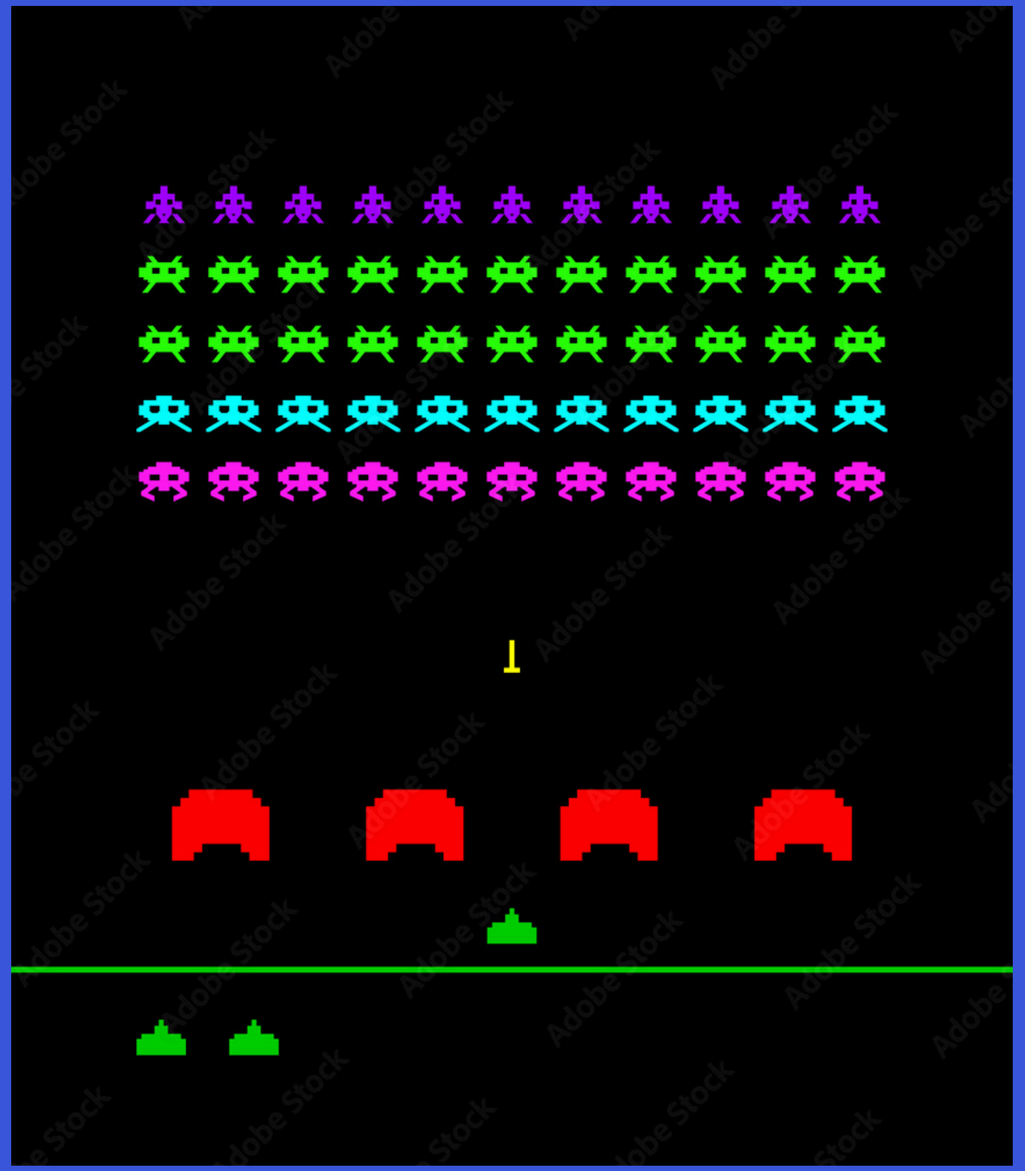
EXEMPLO DE APLICAÇÃO

Objetivos:

- Maximizar a pontuação
- Eliminar inimigos
- Evitar ser atingido.

Implementação:

- Preparação dos Dados
- Arquitetura da Rede Neural
- Replay Buffer
- Rede Alvo
- Treinamento



OBRIGADO!

Bruno Innecco
Gabriel Fernandes
Giovanna Pedersoli
João Victor Teramatsu
Pedro Henrique Moreira