# Dynamic Weights in Multi-Objective Deep Reinforcement Learning

Axel Abels [1,2] Diederik Roijers [3] Tom Lenaerts [1,2]
Ann Nowé [2] Denis Steckelmacher [2]

[1]Machine Learning Group, Université Libre de Bruxelles

[2]Artificial Intelligence Lab, Vrije Universiteit Brussel

[3]Computational Intelligence, Vrije Universiteit Amsterdam

ICML 2019

# Problem

- Multi-Objective Reinforcement Learning

# Problem

- Multi-Objective Reinforcement Learning
  - Vector-valued rewards: $\mathbf{r}$

# Problem

- Multi-Objective Reinforcement Learning
  - Vector-valued rewards: $\mathbf{r}$
  - Linear scalarization: 'Importance' of each component: $\mathbf{w}$

# Problem

- Multi-Objective Reinforcement Learning
  - Vector-valued rewards: $\mathbf{r}$
  - Linear scalarization: 'Importance' of each component: $\mathbf{w}$
  - Try to maximize weighted return:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t (\mathbf{w} \cdot \mathbf{r}_t)\right]$$

# Problem

- Dynamic Weights

# Problem

- Dynamic Weights
  - 'importance' $\mathbf{w}_t$ changes over time

# Problem

- Dynamic Weights
  - 'importance' $\mathbf{w}_t$ changes over time
  - Quick adaptation needed to maximize:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t (\mathbf{w}_t \cdot \mathbf{r}_t)\right]$$

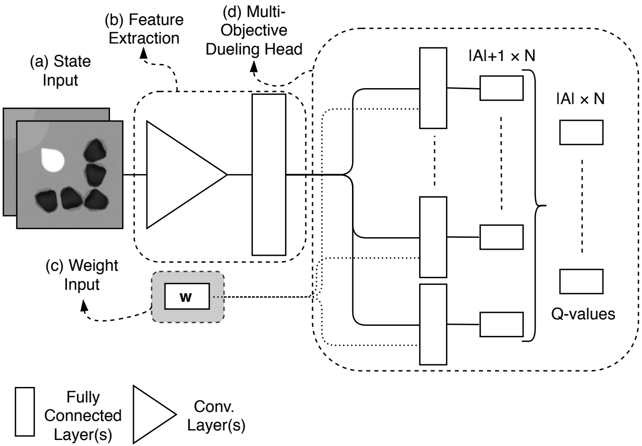
# Problem

- Dynamic Weights
  - 'importance' $\mathbf{w}_t$ changes over time
  - Quick adaptation needed to maximize:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t (\mathbf{w}_t \cdot \mathbf{r}_t)\right]$$

- Focus on high-dimensional problems

# Conditioned Network (CN)



# Updating the Conditioned Network

Considered loss functions

1. Train on current weight vector $\mathbf{w}_t$

$$LOSS_{CN-ACTIVE} = |\mathbf{y}^{(j)}_{\mathbf{w}_t} - \mathbf{Q}_{CN}(a_j, s_j; \mathbf{w}_t)|$$

# Updating the Conditioned Network

Considered loss functions

1. Train on current weight vector $\mathbf{w}_t$

$$LOSS_{CN-ACTIVE} = |\mathbf{y}^{(j)}_{\mathbf{w}_t} - \mathbf{Q}_{CN}(a_j, s_j; \mathbf{w}_t)|$$

2. Train on randomly sampled past weight vector $\mathbf{w}_j$

$$LOSS_{CN-UVFA} = |\mathbf{y}^{(j)}_{\mathbf{w}_j} - \mathbf{Q}_{CN}(a_j, s_j; \mathbf{w}_j)|$$

# Updating the Conditioned Network

Considered loss functions

1. Train on current weight vector $\mathbf{w}_t$

$$LOSS_{CN-ACTIVE} = |\mathbf{y}^{(j)}_{\mathbf{w}_t} - \mathbf{Q}_{CN}(a_j, s_j; \mathbf{w}_t)|$$

2. Train on randomly sampled past weight vector $\mathbf{w}_j$

$$LOSS_{CN-UVFA} = |\mathbf{y}^{(j)}_{\mathbf{w}_j} - \mathbf{Q}_{CN}(a_j, s_j; \mathbf{w}_j)|$$

3. Train on both

$$LOSS_{CN} = \frac{1}{2}[|\mathbf{y}^{(j)}_{\mathbf{w}_t} - \mathbf{Q}_{CN}(a_j, s_j; \mathbf{w}_t)| + |\mathbf{y}^{(j)}_{\mathbf{w}_j} - \mathbf{Q}_{CN}(a_j, s_j; \mathbf{w}_j)|]$$

# Diverse Experience Replay (DER)

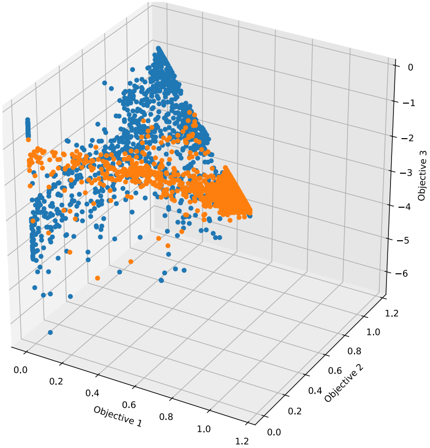
- Replay buffer bias

# Diverse Experience Replay (DER)

- Replay buffer bias: how can we counter it?

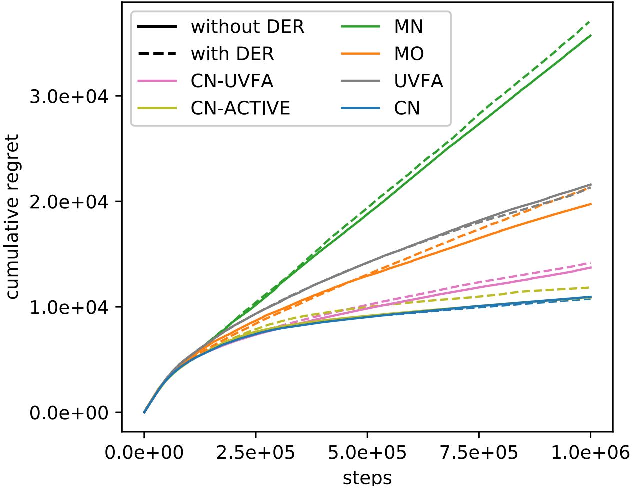
# Diverse Experience Replay (DER)

- Replay buffer bias: how can we counter it?
- By preserving diverse experiences



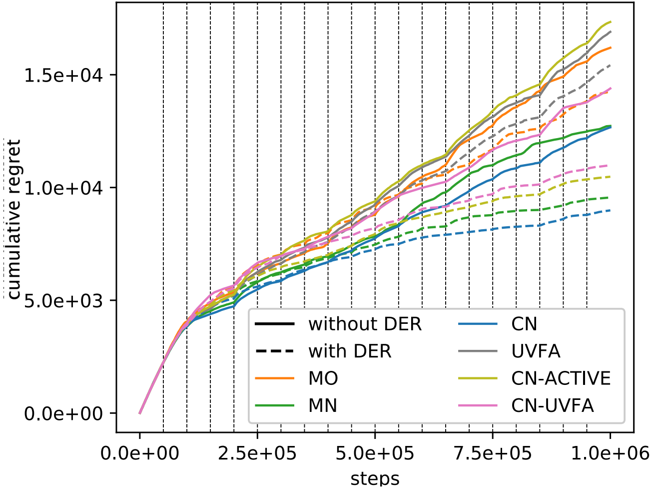

**Replay buffer diversity with and without DER.** Each dot marks a stored trajectory's 3-dimensional return.

# Our CN algorithm converges to near-optimality




Total regret when weights change regularly (lower is better)

# Diversity is crucial for large but sparse weight changes




Total regret when weights change occasionally (lower is better)

# Thank you!

- Poster #49
- 6:30pm to 9pm