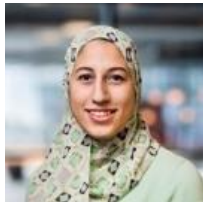




# Dynamic Sparse Training for Deep Reinforcement Learning



**Ghada Sokar**<sup>1</sup>



**Elena Mocanu**<sup>2</sup>



**Decebal Mocanu**<sup>1,2</sup>



**Mykola Pechenizkiy**<sup>1</sup>



**Peter Stone**<sup>3</sup>

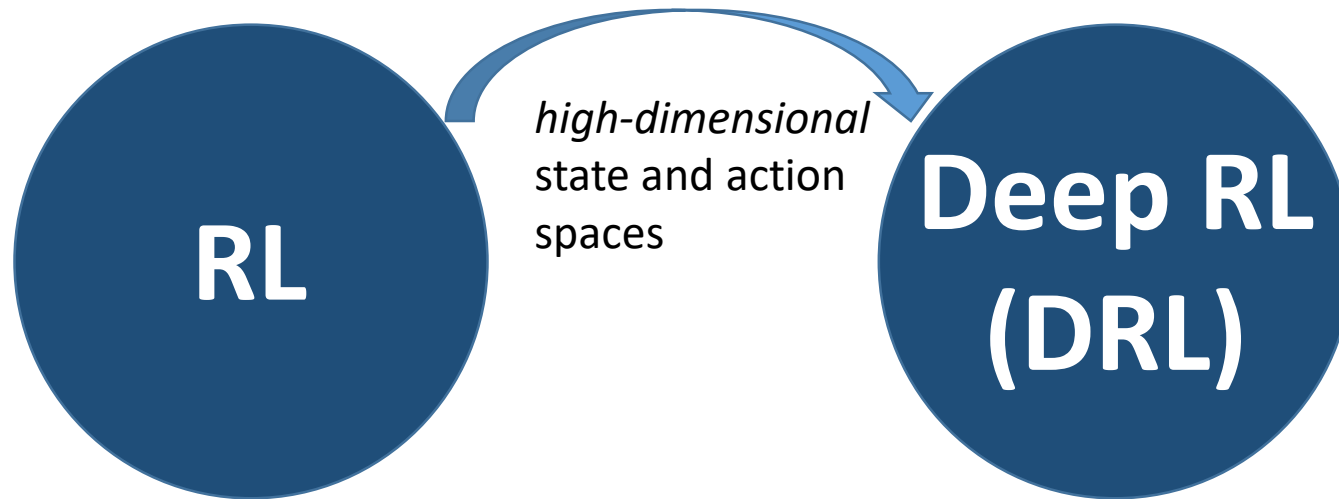
<sup>1</sup> *Eindhoven University of Technology, The Netherlands*

<sup>2</sup> *University of Twente, The Netherlands*

<sup>3</sup> *University of Texas at Austin, Sony AI*

**Email: [g.a.z.n.sokar@tue.nl](mailto:g.a.z.n.sokar@tue.nl)**

# Reinforcement Learning (RL)



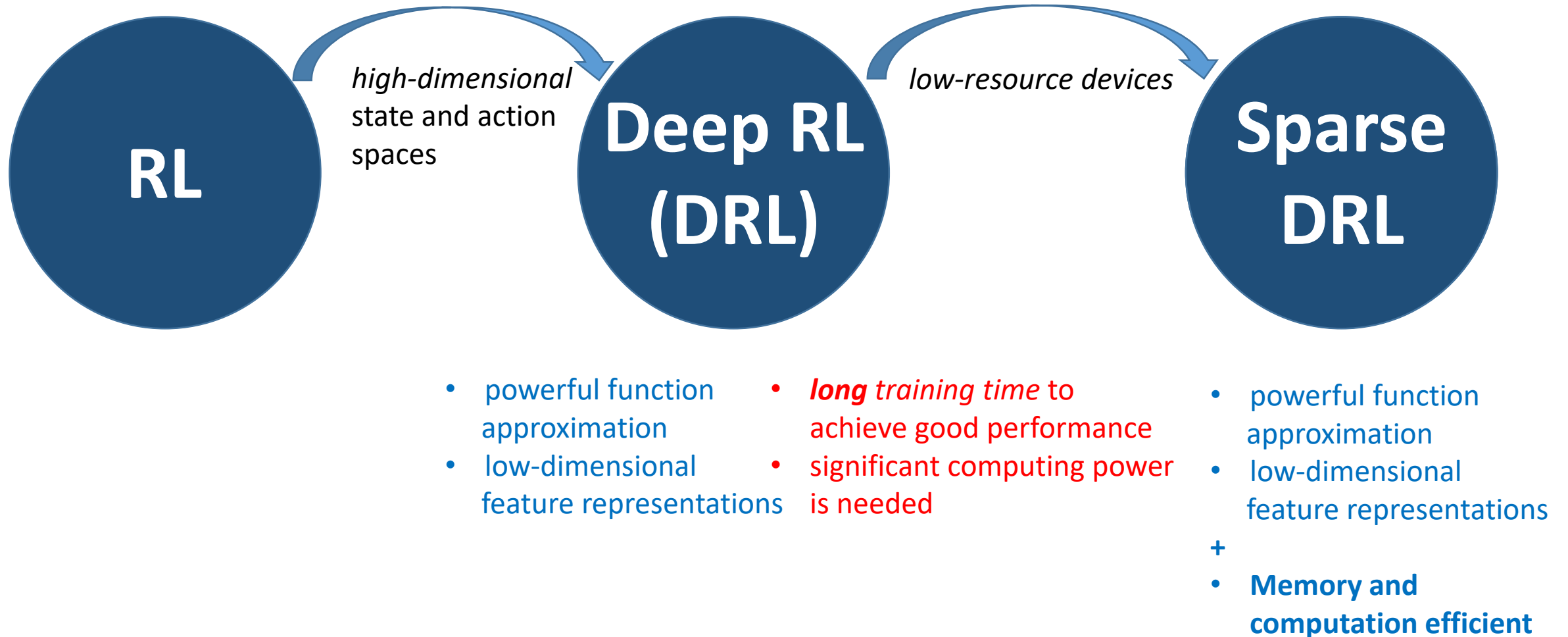
*Can we do better?*  
Can we make DRL models practically appealing to low-resource devices?

- powerful function approximation
- low-dimensional feature representations
- **long training time** to achieve good performance
- **significant computing power is needed**

# New Research Question

***How to train DRL models with strong performance while being memory and computation efficient?***

# Reinforcement Learning (RL)



# Sparse DRL

- We are aware of very few recent studies that introduce sparsity in DRL via pruning and knowledge distillation [1, 2].
- *Limitation:* they still use **dense** neural networks to obtain the compressed versions which still limits their applicability to low-resource devices.

[1] Livne, D., Cohen, K.: Pops: Policy pruning and shrinking for deep reinforcement learning. IEEE Journal of Selected Topics in Signal Processing 14(4), 789–801 (2020)

[2] Zhang, H., He, Z., Li, J.: Accelerating the deep reinforcement learning with neural network compression. In: 2019 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2019)

# Proposed Method

## “Dynamic Sparse Training for Deep Reinforcement Learning”



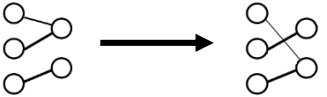

# Dynamic Sparse Training

A research direction in which deep neural networks with *sparse topology* are trained from *scratch*. During training *both* the sparse topology and the weights are optimized.

# Dynamic Sparse Training

A research direction in which deep neural networks with *sparse topology* are trained from *scratch*. During training *both* the sparse topology and the weights are optimized.

Test accuracy% (top-1) of Resnet-50 trained on Imagenet [2]

		Density level (# Parameters)	20% (7.3M)	10% (5.1M)	100% (25.6M)
Dynamic Sparse Training		SET [1]	72.6	70.4	74.9
		Dynamic sparse [2]	73.3	71.6	
Static Sparse neural network		Static sparse	71.6	67.8	

[1] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science." Nature communications 9.1 (2018): 1-12.

[2] Hesham Mostafa, and Xin Wang. "Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization." International Conference on Machine Learning. PMLR, 2019.



# Dynamic Sparse Training

A research direction in which deep neural networks with *sparse topology* are trained from *scratch*. During training *both* the sparse topology and the weights are optimized.

## ➤ Success in other fields

- Feature selection in very high dimensional spaces [1]
- Continual lifelong learning tasks [2]
- Text classification and language modeling tasks [3]
- Federated learning [4]

[1] Zahra Atashgahi, Ghada Sokar, Tim van der Lee, Elena Mocanu, Decebal Constantin Mocanu, Veldhuis, R., Mykola Pechenizkiy,: Quick and robust feature selection: the strength of energy-efficient sparse training for autoencoders. arXiv preprint arXiv:2012.00560 (2020)

[2] Ghada Sokar, Decebal Constantin Mocanu, Mykola Pechenizkiy: Spacenet: Make free space for continual learning. Neurocomputing 439, 1–11 (2021)

[3] Shiwei Liu, Iftitahu Ni'mah, Vlado Menkovski, Decebal Constantin Mocanu, Mykola Pechenizkiy: Efficient and effective training of sparse recurrent neural networks. Neural Computing and Applications pp. 1–12 (2021)

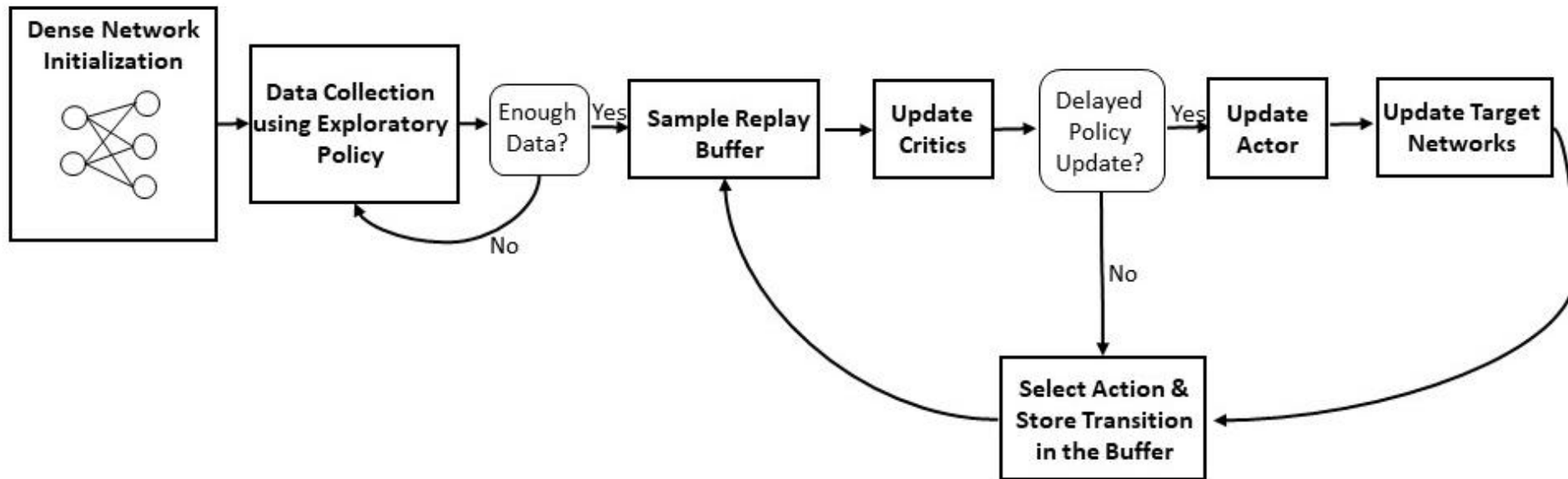
[4] Hangyu Zhu, and Yaochu Jin: Multi-objective evolutionary federated learning. IEEE Transactions on Neural Networks and Learning Systems 31(4), 1310–1322 (2019)

# Contributions

- Introduce for the first time *dynamic sparse training* to deep RL.
- Merge our proposed training method with the Twin Delayed Deep Deterministic policy gradient (TD3) algorithm [1] and introduce **Dynamic Sparse TD3 (DS-TD3)**.
- We show that DS-TD3 outperforms TD3 on five challenging continuous control tasks while reducing the memory and computation costs substantially.
- This work is a step towards enabling DRL agents to be trained on low-resource devices.

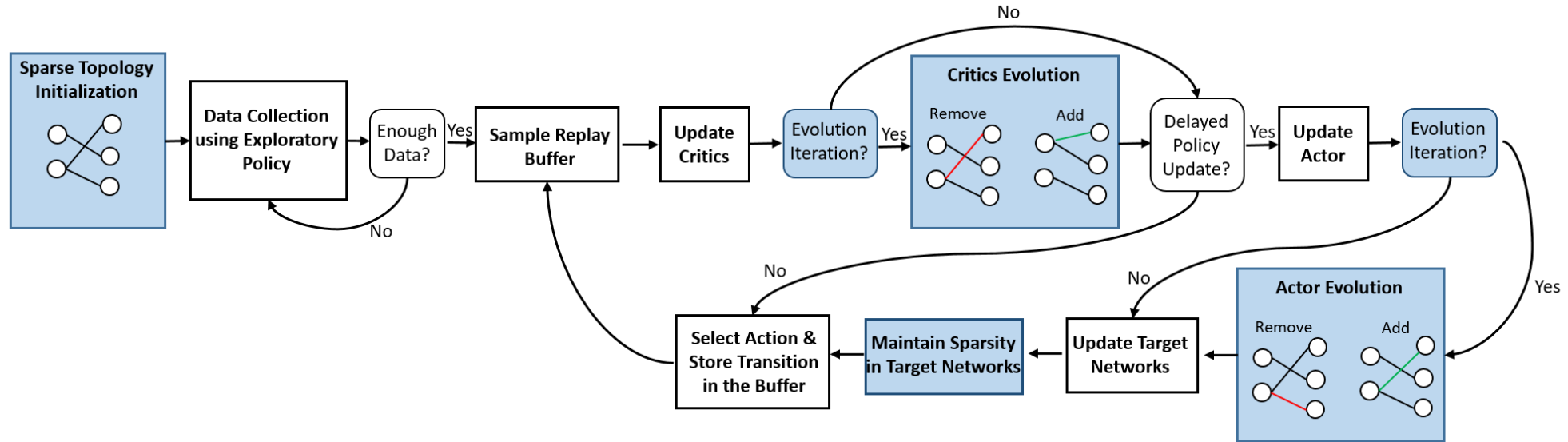
## Twin Delayed Deep Deterministic policy gradient (TD3)

- Actor-critic method (policy  $\pi_\phi$ , value function  $Q_\theta$ )
- Improve stability by:
  - limiting the overestimation bias using pair of critics  $Q_{\theta_1}$  and  $Q_{\theta_2}$ . It takes the smallest value of the two critic networks as an estimation for the Q value.
  - proposing a delayed update of the actor and target networks.



# Proposed Method (DS-TD3)

## Dynamic Sparse training for TD3



# Proposed Method (DS-TD3)

- Four simple modifications

## I Sparse Topology Initialization

## II Evolution Schedule

## III Evolution Process

## IV Maintain Sparsity in Target Networks

### Algorithm 1 DS-TD3

---

```

1: Requires:  $\lambda^l, \eta, e$ 
2: Initialize critic networks  $Q_{\theta_1}, Q_{\theta_2}$ , and actor network  $\pi_\phi$  with sparse parameters  $\theta_1, \theta_2, \phi$  according to Erdős-Rényi graph with the sparsity level defined by  $\lambda^l$ :
3:  $\theta_1^l \leftarrow \theta_1^l \odot M_{\theta_1}^l, \theta_2^l \leftarrow \theta_2^l \odot M_{\theta_2}^l, \phi^l \leftarrow \phi^l \odot M_\phi^l$ 
4: Initialize target networks  $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$ 
5: Initialize replay buffer  $\mathcal{B}$ 
6: for  $t = 1$  to  $T$  do
7:   Select action with exploration noise  $a \sim \pi_\phi(s) + \epsilon$ ,
8:    $\epsilon \sim \mathcal{N}(0, \sigma)$  and observe reward  $r$  and new state  $s'$ 
9:   Store transition tuple  $(s, a, r, s')$  in  $\mathcal{B}$ 
10:  Sample mini-batch of  $N$  transitions  $(s, a, r, s')$  from  $\mathcal{B}$ 
11:   $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$ 
12:   $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \tilde{a})$ 
13:  Update critics  $\theta_i \leftarrow \argmin_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$ 
14:  if  $t \bmod e$  then
15:    Evolve the critic networks  $\theta_i$ :
16:     $\theta_i^l \leftarrow \theta_i^l \odot (M_{\theta_i}^l - R_{\theta_i}^l)$ 
17:     $M_{\theta_i}^l \leftarrow M_{\theta_i}^l - R_{\theta_i}^l$ 
18:     $M_{\theta_i}^l \leftarrow M_{\theta_i}^l + A_{\theta_i}^l$ 
19:  end if
20:  if  $t \bmod d$  then
21:    Update  $\phi$  by the deterministic policy gradient:
22:     $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$ 
23:    if  $t \bmod e$  then
24:      Evolve the actor network  $\phi$ :
25:       $\phi^l \leftarrow \phi^l \odot (M_\phi^l - R_\phi^l)$ 
26:       $M_\phi^l \leftarrow M_\phi^l - R_\phi^l$ 
27:       $M_\phi^l \leftarrow M_\phi^l + A_\phi^l$ 
28:    end if
29:    Update target networks:
30:     $\theta_i' \leftarrow \tau \theta_i + (1 - \tau) \theta_i'$ 
31:     $\phi' \leftarrow \tau \phi + (1 - \tau) \phi'$ 
32:    Prune the smallest weights from  $|\theta_i'|$  and  $|\phi'|$  to maintain the initial sparsity level
33:  end if
34: end for

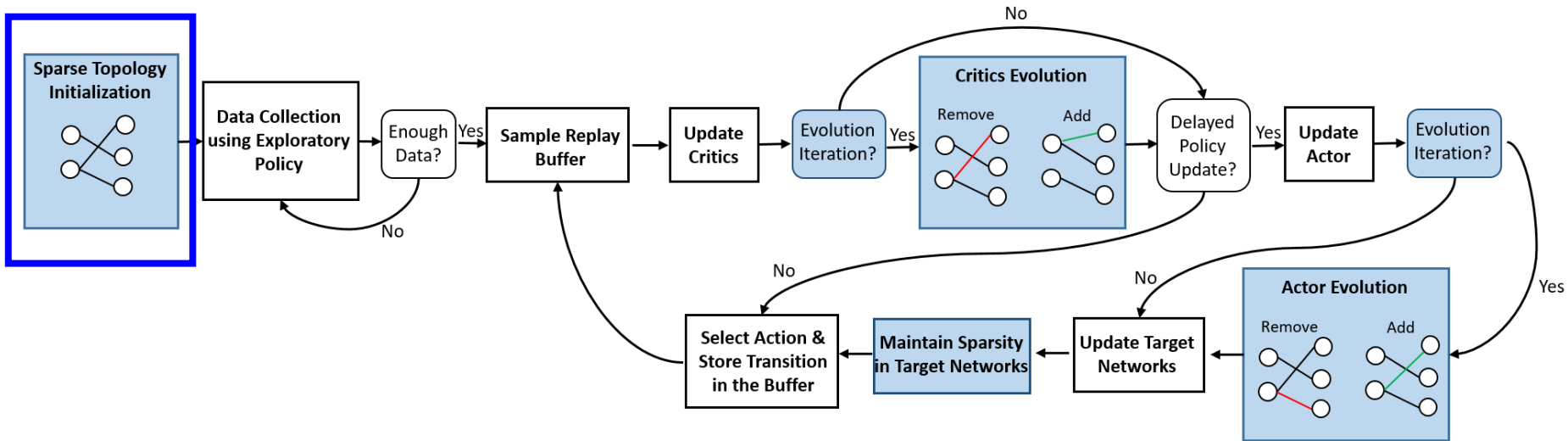
```

---

# Proposed Method (DS-TD3)

## I Sparse Topology Initialization

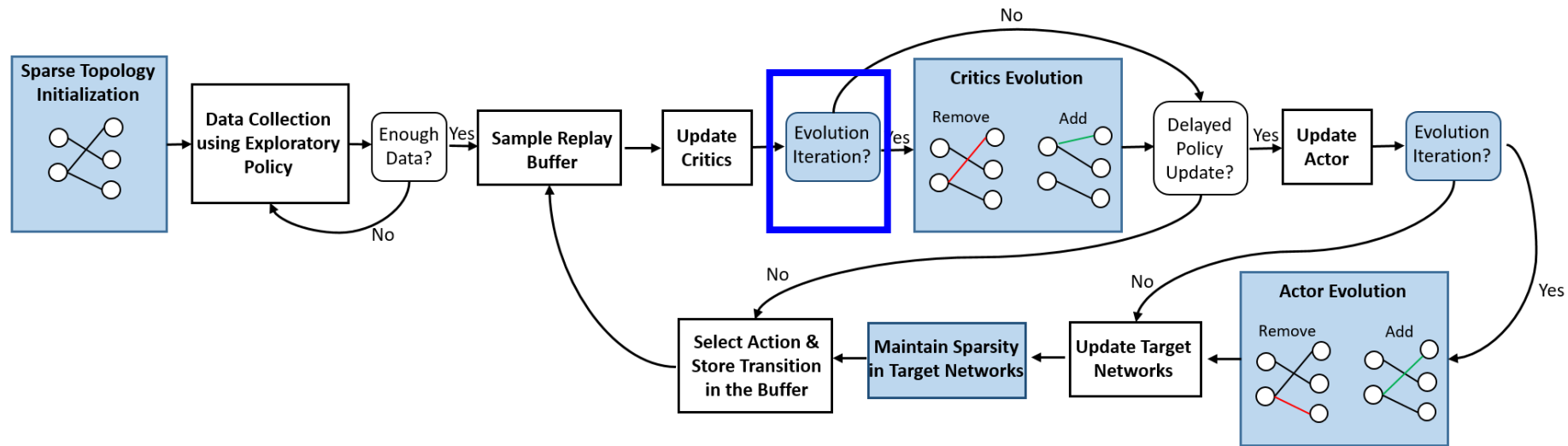
The actor and critics networks along with the corresponding target networks are initialized with sparse topologies.



# Proposed Method (DS-TD3)

## II Evolution Schedule

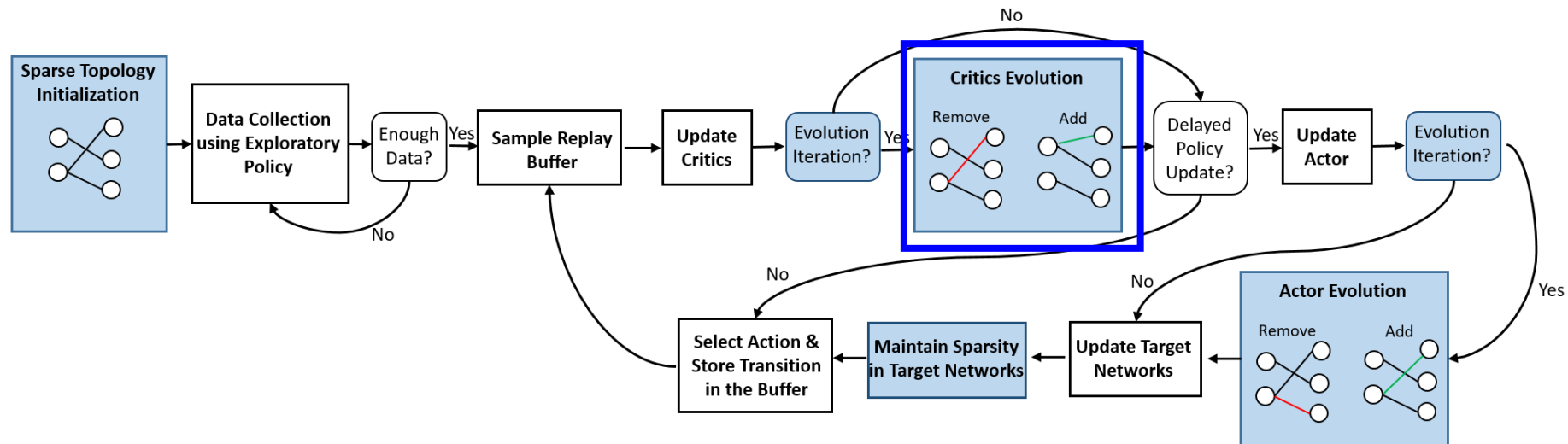
Delay the evolution process and perform it every  $e$  iterations. This would allow the newly added connections from the previous evolution process to grow and learn in this online setting.



# Proposed Method (DS-TD3)

## III Evolution Process

Every  $e$  steps, the topology of the critics and the actor are evolved using the SET algorithm [1].





# Proposed Method (DS-TD3)

## III Evolution Process

Every  $e$  steps, the topology of the critics and the actor are evolved using the SET algorithm [1].

### *SET algorithm*

❖ *Remove and Add phases*

- *Remove phase:*

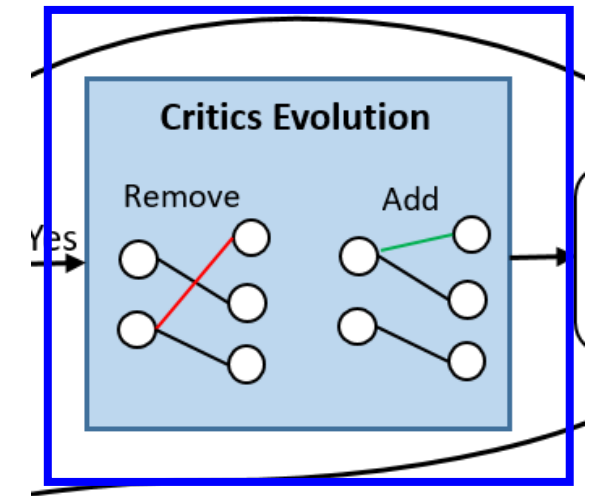
Remove a fraction of the least important connections from each layer.

The *least important weights* are the ones closest to zero

(i.e. a subset of the smallest positive weights and the largest negative weights)

- *Add phase:*

Add the same fraction of removed connections in random locations in each layer.



# Proposed Method (DS-TD3)

## IV Maintain Sparsity in Target Networks

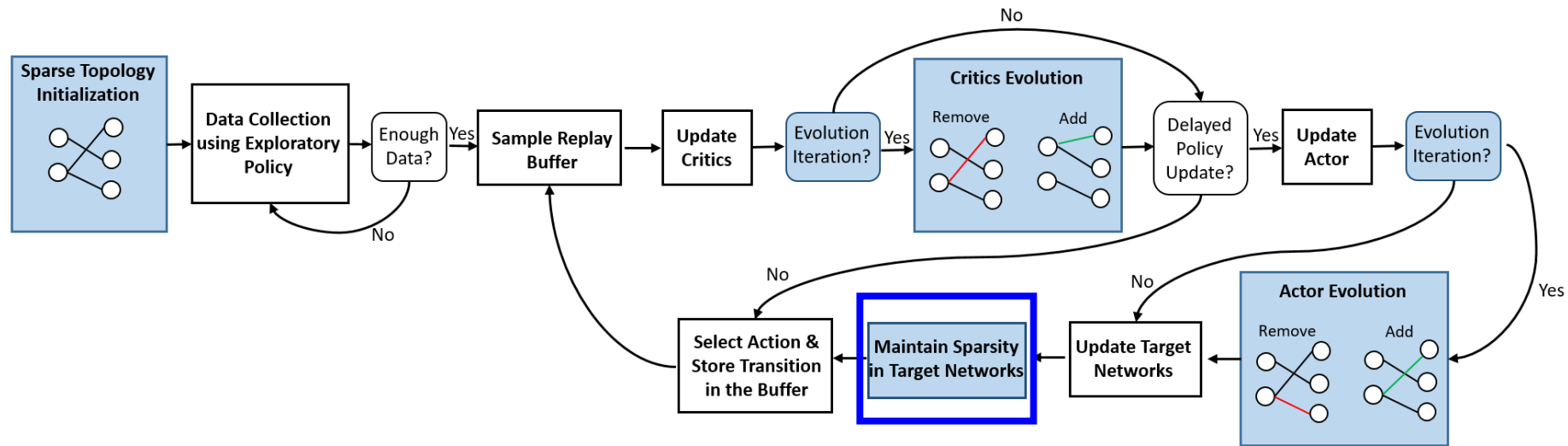
- Since the target networks (parametrized by  $\phi'$ ,  $\Theta_1'$ ,  $\Theta_2'$ ) are slowly updated by the current networks ( $\phi$ ,  $\Theta_1$ ,  $\Theta_2$ ) by proportion  $\tau$ , the sparsity level could decrease.

$$\phi' = \tau \phi + (1 - \tau) \phi'$$

$$\Theta_1' = \tau \Theta_1' + (1 - \tau) \Theta_1$$

$$\Theta_2' = \tau \Theta_2' + (1 - \tau) \Theta_2$$

- We prune the extra weights based on their magnitude (i.e. the least absolute value).



# Experiments & Results

# Experimental Setup

## Baselines:

1. TD3
2. Static-TD3
3. DS-TD3 (ours)

## Environments:

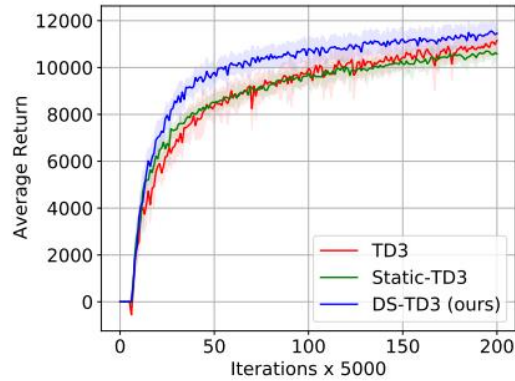
- We performed our experiments on MuJoCo continuous control tasks interfaced through OpenAI Gym
- We tested on five environments (HalfCheetah-v3, Hopper-v3, Walker2d-v3, Ant-v3, and Humanoid-v3)

## We study:

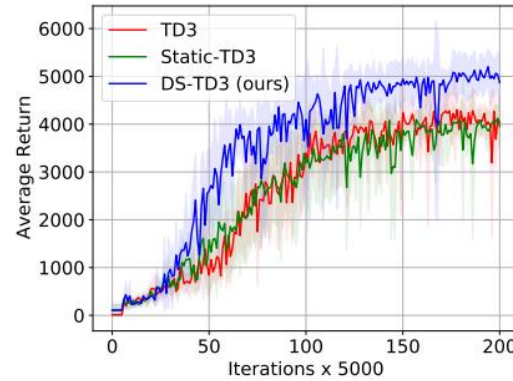
1. The learning speed and the final performance
2. Memory and computation costs
3. The effect of evolution schedule
4. The effect of sparsity level

# Results

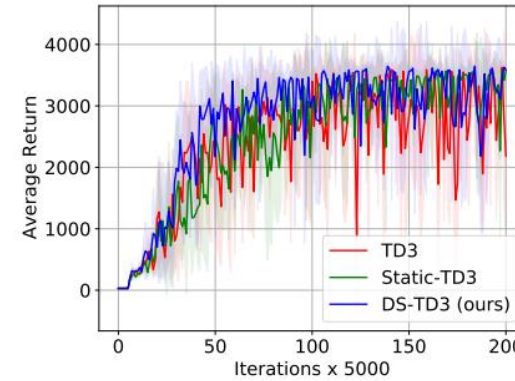
## 1. Learning speed and the final performance



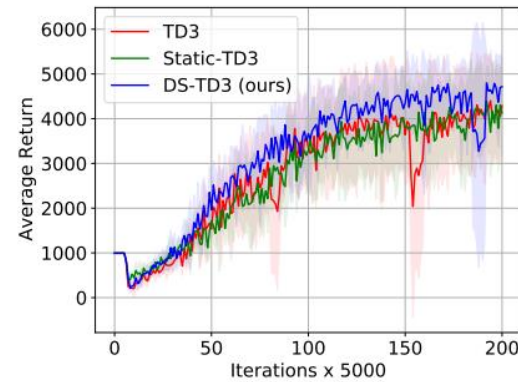
(a) HalfCheetah-v3.



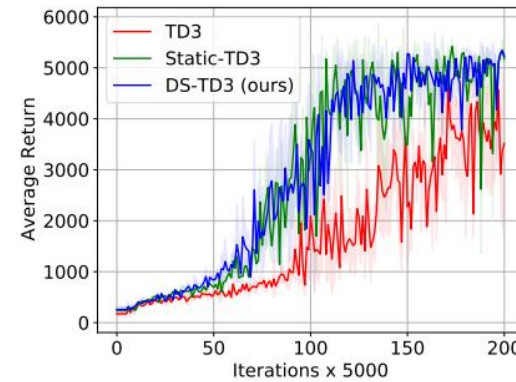
(b) Walker2d-v3.



(c) Hopper-v3.



(d) Ant-v3.



(e) Humanoid-v3.

Learning curves of the studied algorithms on different continuous control tasks. The shaded region represents the standard deviation of the average evaluation over 5 runs.

# Results

## 2. Memory and computation costs

Performance and costs of training the actor and critics networks on HalfCheetah-v3 using different methods.

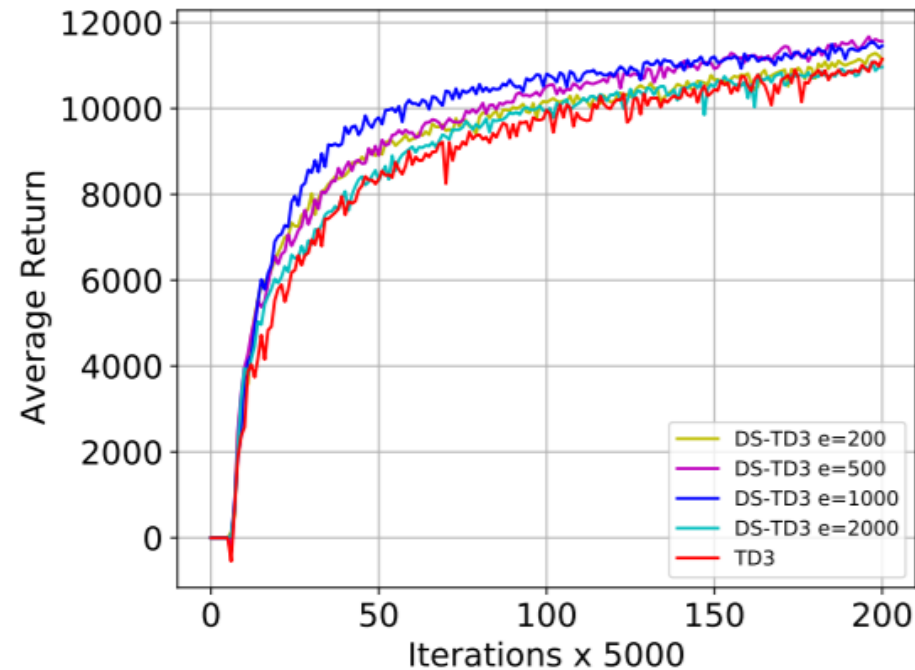
Method	FLOPs *	Networks Size (# connections)	Average Return
TD3	$1 \times (1.07e14)$	214784	$11153.48 \pm 473.29$
Static-TD3	$0.49 \times$	106169	$10583.84 \pm 307.03$
DS-TD3 (ours)	$0.49 \times$	106169	<b><math>11459.88 \pm 482.55</math></b>

\* Floating-point operations (FLOPs) are calculated using the method described in [1]

[1] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. "Rigging the lottery: Making all tickets winners." In International Conference on Machine Learning, pp. 2943-2952. PMLR, 2020.

# Results

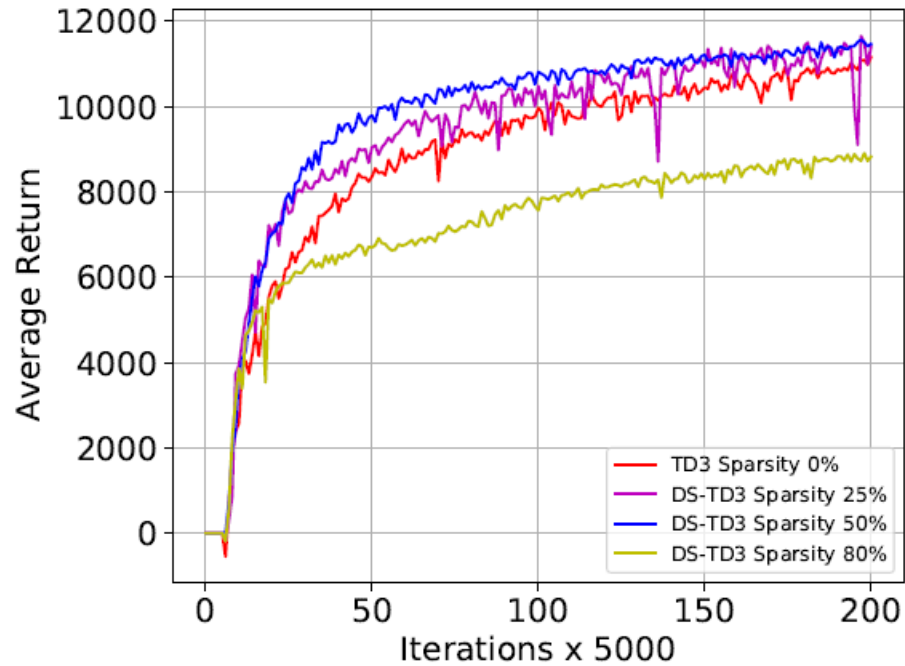
## 3. The effect of evolution schedule



The learning curves of our proposed DS-TD3 algorithm on HalfCheetah-v3 using different evolution schedules along with the learning curve of TD3 using dense networks.

# Results

## 4. The effect of sparsity level



The learning curves of our proposed DS-TD3 on HalfCheetah-v3 using different sparsity levels in the actor and critics networks along with the learning curve of TD3 using dense networks.



# Conclusion & Future Works

# Conclusion

- We proposed DS-TD3, a dynamic sparse training algorithm applied to the TD3 method.
- Our proposed method outperforms the TD3 method on 5 OpenAI gym continuous control tasks.
- DS-TD3 is able to find a smaller sparse topology that learns efficient representation with at least a *50% reduction* in the parameters.
- The performance gain is accompanied by speeding up the agent learning process. DS-TD3 reaches the same performance achieved by the dense network with a 40-50% reduction in the number of training steps.

# Future Works

- The performance of very high sparse models is not yet satisfactory.
  - study the possibility of increasing the sparsity level, while keeping the same performance.
- Study different evolution schedules.
- Try/Design other dynamic sparse training algorithms for DRL paradigm.
- Evaluate our proposed training algorithm on other DRL algorithms.

# Thank You!

Feel free to reach out!  
[g.a.z.n.sokar@tue.nl](mailto:g.a.z.n.sokar@tue.nl)