MushroomRL Benchmark Documentation

Release 1.0.0

Carlo D'Eramo, Davide Tateo

Apr 07, 2021

Benchmarks Results:

CHAPTER 1

Reinforcement Learning python library

MushroomRL Benchmark is a benchmarking tool for the Mushroom RL library. The focus of this benchmarking tool is to benchmark the results of deep reinforcement learning algorithms, in particular Deep Actor-Critic. The idea behind MushroomRL Benchmarking is to have a complete platform to run batch comparisons of Deep RL algorithms implemented in MushroomRL under a set of standard benchmark tasks.

With MushroomRL Benchmarking you can:

- Run the benchmarks in a local machine, both sequentially and in parallel fashion
- Run experiments on a SLURM-based cluster.

CHAPTER 2

Basic run example

CHAPTER 3

Download and installation

MushroomRL Benchmark can be downloaded from the GitHub repository. Installation can be done running

```
cd mushroom-rl-benchmark
pip install -e .[all]
```

To compile the documentation:

```
cd mushroom-rl-benchmark/docs
make html
```

or to compile the pdf version:

```
cd mushroom-rl-benchmark/docs
make latexpdf
```

3.1 Actor-Critic Benchmarks

We provide the benchmarks for the following Deep Actor-Critic algorithms:

- **A2C**
- **PPO**
- **TRPO**
- **SAC**
- **DDPG**
- **TD3**

We consider the following environments in the benchmark

3.1.1 Gym Environments Benchmarks

Run Parameters	
n_runs	25
n_epochs	10
n_steps	30000
n_episodes_test	10

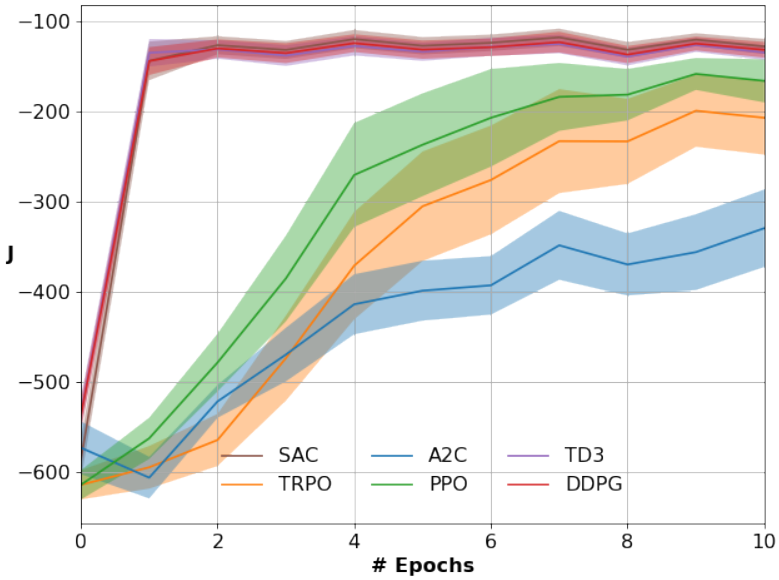
Pendulum-v0

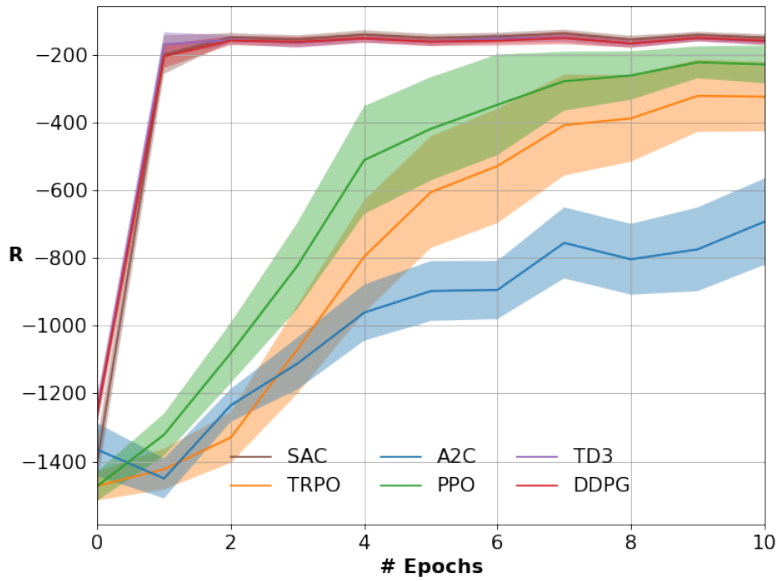
```
A2C:
  actor_lr: 0.0007
  batch_size: 64
  critic_lr: 0.0007
  critic_network: A2CNetwork
  ent_coeff: 0.01
  eps_actor: 0.003
  eps_critic: 1.0e-05
  max_grad_norm: 0.5
  n_features: 64
  preprocessors: null
DDPG:
  actor_lr: 0.0001
  actor_network: DDPGActorNetwork
  batch_size: 64
  critic_lr: 0.001
  critic_network: DDPGCriticNetwork
  initial_replay_size: 128
  max_replay_size: 1000000
  n_features:
  - 64
  - 64
  tau: 0.001
PPO:
  actor_lr: 0.0003
  batch_size: 64
  critic_fit_params: null
  critic_lr: 0.0003
  critic_network: TRPONetwork
  eps: 0.2
  lam: 0.95
  n_epochs_policy: 4
  n_features: 32
  n_steps_per_fit: 3000
  preprocessors: null
SAC:
  actor_lr: 0.0001
  actor_network: SACActorNetwork
  batch_size: 64
  critic_lr: 0.0003
  critic_network: SACCriticNetwork
  initial_replay_size: 128
  lr_alpha: 0.0003
  max_replay_size: 500000
  n_features: 64
```

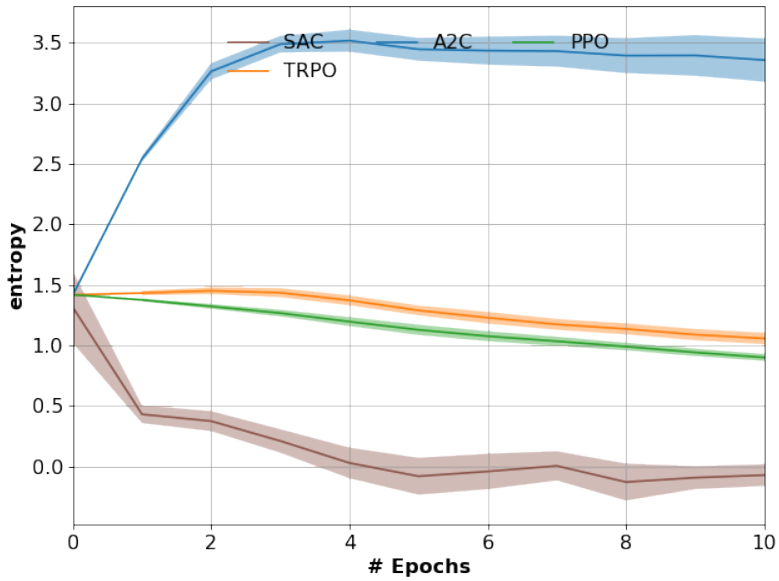
(continues on next page)

(continued from previous page)

```
  preprocessors: null
  target_entropy: null
  tau: 0.001
  warmup_transitions: 128
TD3:
  actor_lr: 0.0001
  actor_network: TD3ActorNetwork
  batch_size: 64
  critic_lr: 0.001
  critic_network: TD3CriticNetwork
  initial_replay_size: 128
  max_replay_size: 1000000
  n_features:
  - 64
  - 64
  tau: 0.001
TRPO:
  batch_size: 64
  cg_damping: 0.01
  cg_residual_tol: 1.0e-10
  critic_fit_params: null
  critic_lr: 0.0003
  critic_network: TRPONetwork
  ent_coeff: 0.0
  lam: 0.95
  max_kl: 0.01
  n_epochs_cg: 100
  n_epochs_line_search: 10
  n_features: 32
  n_steps_per_fit: 3000
  preprocessors: null
```







LunarLanderContinuous-v2

```
A2C:
  actor_lr: 0.0007
  batch_size: 64
  critic_lr: 0.0007
  critic_network: A2CNetwork
  ent_coeff: 0.01
  eps_actor: 0.003
  eps_critic: 1.0e-05
  max_grad_norm: 0.5
  n_features: 64
  preprocessors: StandardizationPreprocessor
DDPG:
```

(continues on next page)

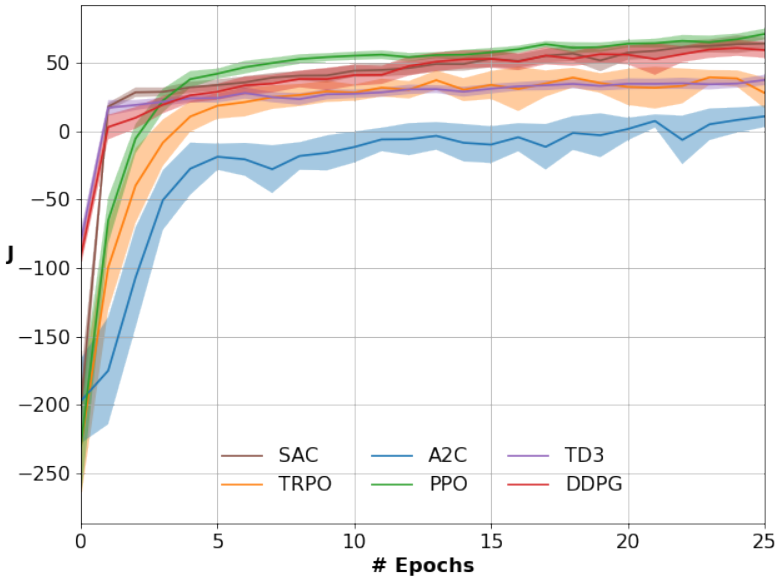
(continued from previous page)

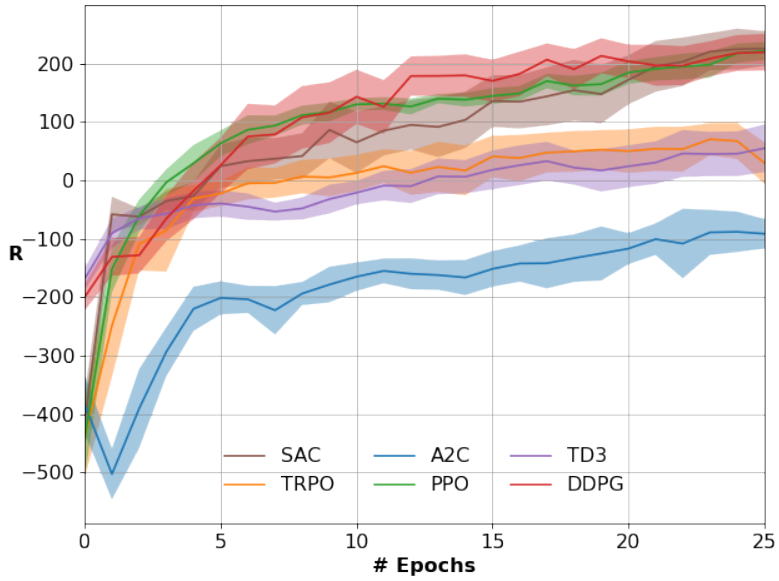
```
  actor_lr: 0.0001
  actor_network: DDPGActorNetwork
  batch_size: 64
  critic_lr: 0.001
  critic_network: DDPGCriticNetwork
  initial_replay_size: 128
  max_replay_size: 1000000
  n_features:
  - 64
  - 64
  tau: 0.001
PPO:
  actor_lr: 0.0003
  batch_size: 64
  critic_fit_params: null
  critic_lr: 0.003
  critic_network: TRPONetwork
  eps: 0.2
  lam: 0.95
  n_epochs_policy: 4
  n_features: 32
  n_steps_per_fit: 3000
  preprocessors: StandardizationPreprocessor
SAC:
  actor_lr: 0.0001
  actor_network: SACActorNetwork
  batch_size: 64
  critic_lr: 0.0003
  critic_network: SACCriticNetwork
  initial_replay_size: 128
  lr_alpha: 0.0003
  max_replay_size: 500000
  n_features: 64
  preprocessors: null
  target_entropy: null
  tau: 0.005
  warmup_transitions: 128
TD3:
  actor_lr: 0.0001
  actor_network: TD3ActorNetwork
  batch_size: 64
  critic_lr: 0.001
  critic_network: TD3CriticNetwork
  initial_replay_size: 128
  max_replay_size: 1000000
  n_features:
  - 64
  - 64
  tau: 0.001
TRPO:
  batch_size: 64
  cg_damping: 0.01
  cg_residual_tol: 1.0e-10
  critic_fit_params: null
  critic_lr: 0.03
  critic_network: TRPONetwork
  ent_coeff: 0.0
```

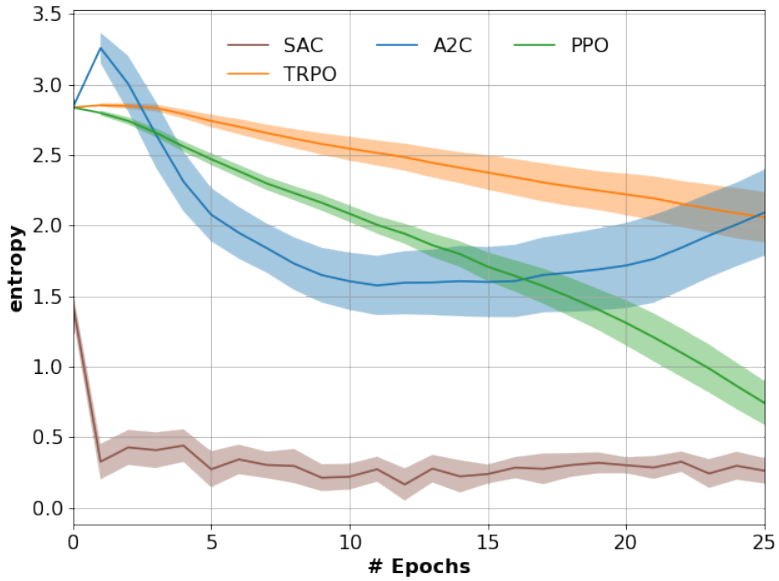
(continues on next page)

(continued from previous page)

```
  lam: 0.95
  max_kl: 0.01
  n_epochs_cg: 100
  n_epochs_line_search: 10
  n_features: 32
  n_steps_per_fit: 3000
  preprocessors: StandardizationPreprocessor
```







3.1.2 Mujoco Environments Benchmarks

Run Parameters	
n_runs	25
n_epochs	50
n_steps	30000
n_episodes_test	10

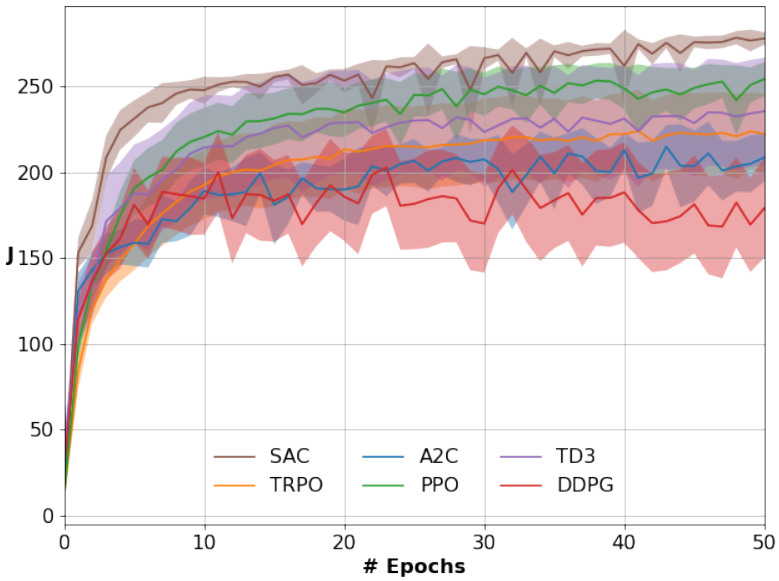
Hopper-v3

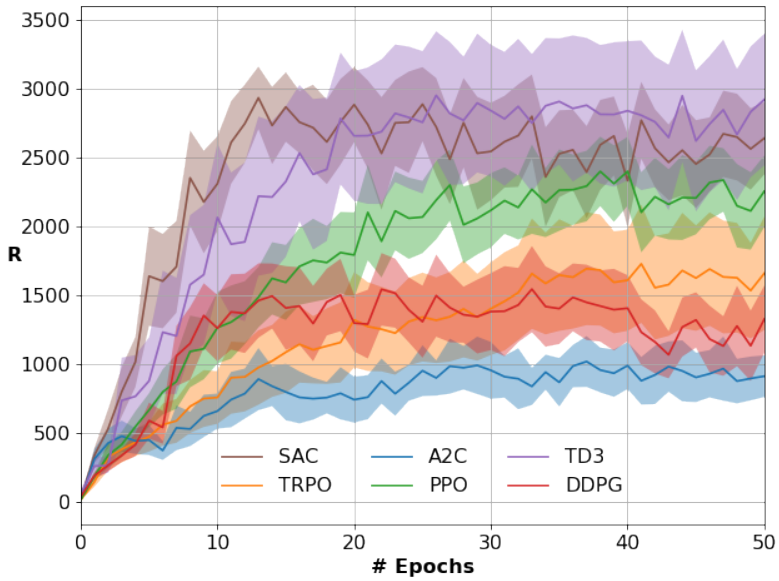
```
A2C:
  actor_lr: 0.0007
  batch_size: 64
  critic_lr: 0.0007
  critic_network: A2CNetwork
  ent_coeff: 0.01
  eps_actor: 0.003
  eps_critic: 1.0e-05
  max_grad_norm: 0.5
  n_features: 64
  preprocessors: StandardizationPreprocessor
DDPG:
  actor_lr: 0.0001
  actor_network: DDPGActorNetwork
  batch_size: 128
  critic_lr: 0.001
  critic_network: DDPGCriticNetwork
  initial_replay_size: 5000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
```

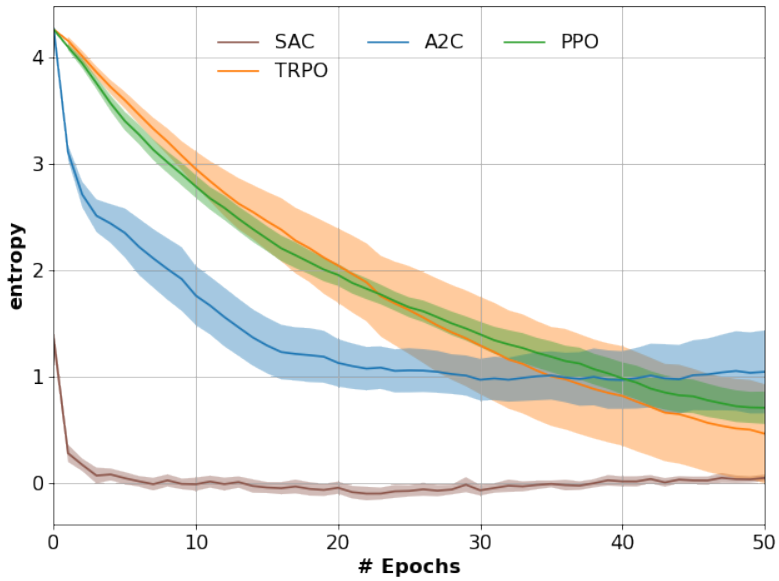
(continues on next page)

(continued from previous page)

```
  tau: 0.001
PPO:
  actor_lr: 0.0003
  batch_size: 32
  critic_fit_params: null
  critic_lr: 0.0003
  critic_network: TRPONetwork
  eps: 0.2
  lam: 0.95
  n_epochs_policy: 10
  n_features: 32
  n_steps_per_fit: 2000
  preprocessors: StandardizationPreprocessor
SAC:
  actor_lr: 0.0001
  actor_network: SACActorNetwork
  batch_size: 256
  critic_lr: 0.0003
  critic_network: SACCriticNetwork
  initial_replay_size: 5000
  lr_alpha: 0.0003
  max_replay_size: 500000
  n_features: 256
  preprocessors: null
  target_entropy: null
  tau: 0.005
  warmup_transitions: 10000
TD3:
  actor_lr: 0.001
  actor_network: TD3ActorNetwork
  batch_size: 100
  critic_lr: 0.001
  critic_network: TD3CriticNetwork
  initial_replay_size: 1000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
  tau: 0.005
TRPO:
  batch_size: 64
  cg_damping: 0.01
  cg_residual_tol: 1.0e-10
  critic_fit_params: null
  critic_lr: 0.001
  critic_network: TRPONetwork
  ent_coeff: 0.0
  lam: 0.95
  max_kl: 0.01
  n_epochs_cg: 100
  n_epochs_line_search: 10
  n_features: 32
  n_steps_per_fit: 1000
  preprocessors: StandardizationPreprocessor
```


J
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG


R
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG



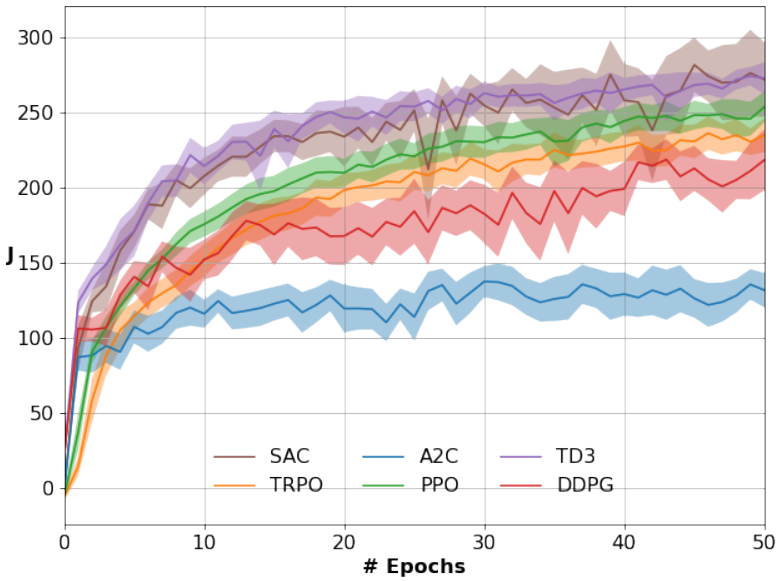
Walker2d-v3

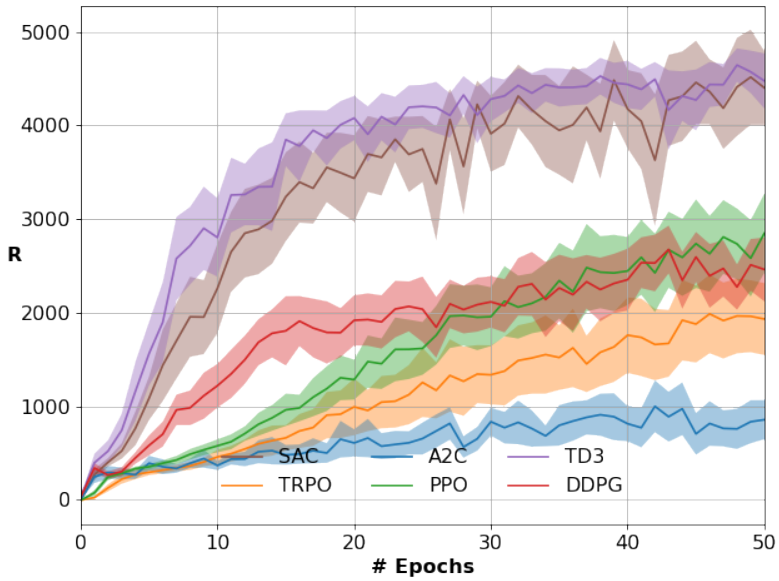
```yaml
A2C:
  actor_lr: 0.0007
  batch_size: 64
  critic_lr: 0.0007
  critic_network: A2CNetwork
  ent_coeff: 0.01
  eps_actor: 0.003
  eps_critic: 1.0e-05
  max_grad_norm: 0.5
  n_features: 64
  preprocessors: StandardizationPreprocessor
DDPG:
  actor_lr: 0.0001
  actor_network: DDPGActorNetwork
  batch_size: 128
  critic_lr: 0.001
  critic_network: DDPGCriticNetwork
  initial_replay_size: 5000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
  tau: 0.001
PPO:
  actor_lr: 0.0003
  batch_size: 32
  critic_fit_params: null
  critic_lr: 0.0003
  critic_network: TRPONetwork
  eps: 0.2
  lam: 0.95
  n_epochs_policy: 10
  n_features: 32
```

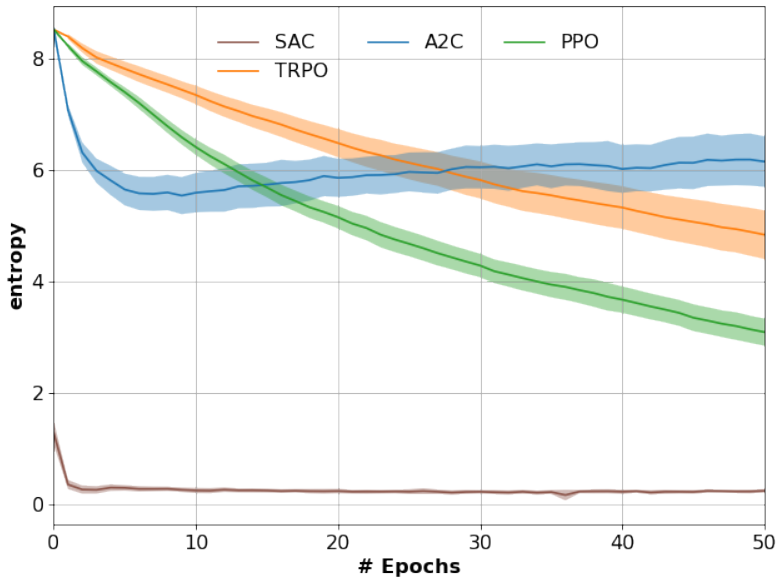
(continues on next page)

(continued from previous page)

```
  n_steps_per_fit: 2000
  preprocessors: StandardizationPreprocessor
SAC:
  actor_lr: 0.0001
  actor_network: SACActorNetwork
  batch_size: 256
  critic_lr: 0.0003
  critic_network: SACCriticNetwork
  initial_replay_size: 5000
  lr_alpha: 0.0003
  max_replay_size: 500000
  n_features: 256
  preprocessors: null
  target_entropy: null
  tau: 0.005
  warmup_transitions: 10000
TD3:
  actor_lr: 0.001
  actor_network: TD3ActorNetwork
  batch_size: 100
  critic_lr: 0.001
  critic_network: TD3CriticNetwork
  initial_replay_size: 1000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
  tau: 0.005
TRPO:
  batch_size: 64
  cg_damping: 0.01
  cg_residual_tol: 1.0e-10
  critic_fit_params: null
  critic_lr: 0.001
  critic_network: TRPONetwork
  ent_coeff: 0.0
  lam: 0.95
  max_kl: 0.01
  n_epochs_cg: 100
  n_epochs_line_search: 10
  n_features: 32
  n_steps_per_fit: 1000
  preprocessors: StandardizationPreprocessor
```


J
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG


R
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG



HalfCheetah-v3

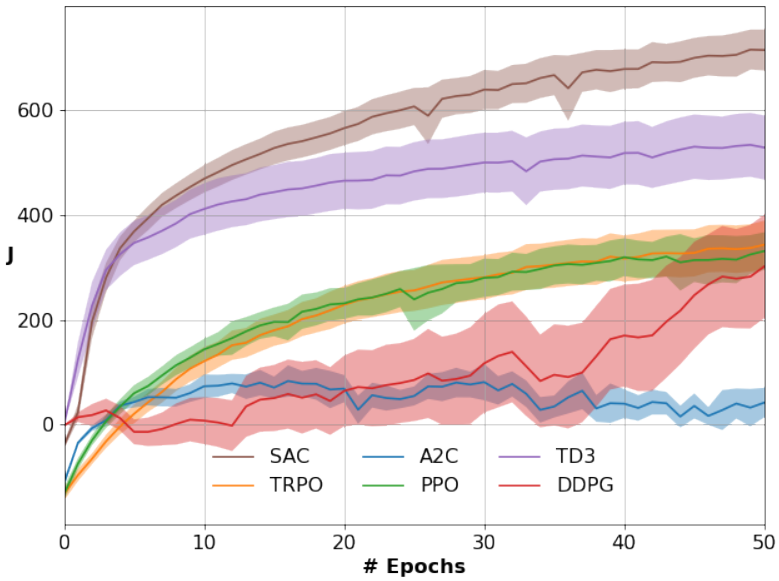
```yaml
A2C:
  actor_lr: 0.0007
  batch_size: 64
  critic_lr: 0.0007
  critic_network: A2CNetwork
  ent_coeff: 0.01
  eps_actor: 0.003
  eps_critic: 1.0e-05
  max_grad_norm: 0.5
  n_features: 64
  preprocessors: StandardizationPreprocessor
DDPG:
  actor_lr: 0.0001
  actor_network: DDPGActorNetwork
  batch_size: 128
  critic_lr: 0.001
  critic_network: DDPGCriticNetwork
  initial_replay_size: 5000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
  tau: 0.001
PPO:
  actor_lr: 0.0003
  batch_size: 32
  critic_fit_params: null
  critic_lr: 0.0003
  critic_network: TRPONetwork
  eps: 0.2
  lam: 0.95
  n_epochs_policy: 10
  n_features: 32
```

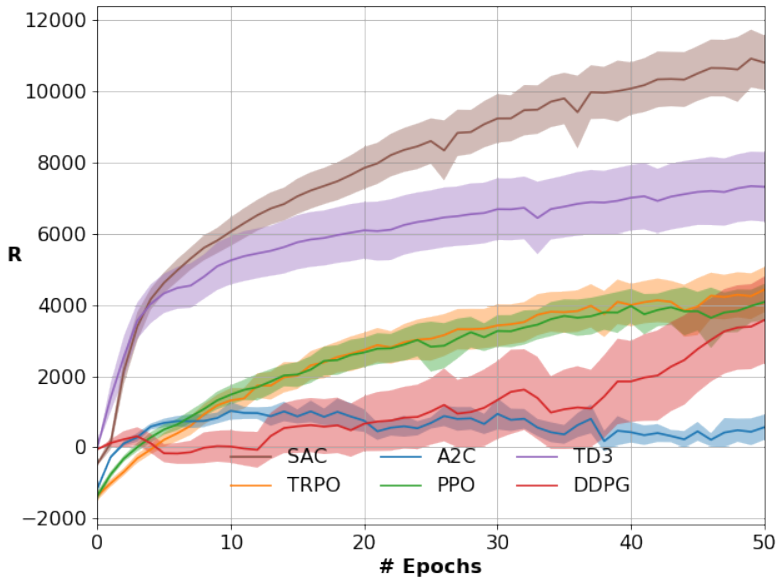
(continues on next page)

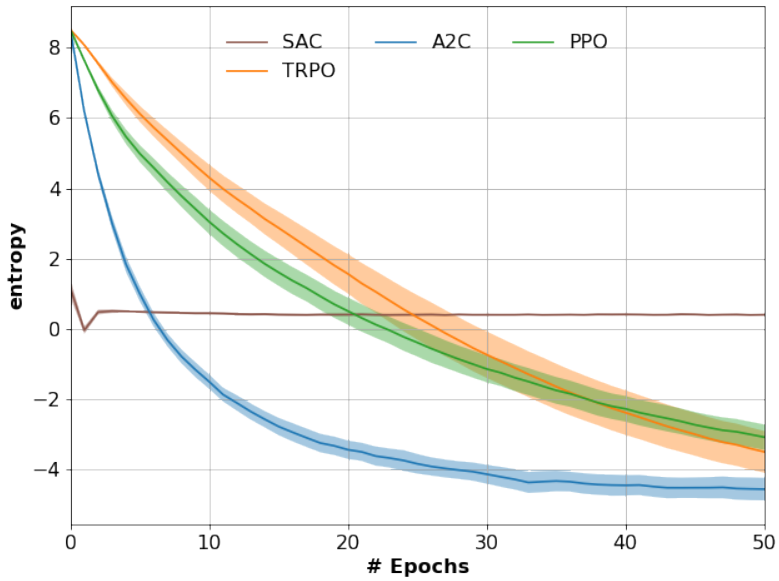
(continued from previous page)

```
  n_steps_per_fit: 2000
  preprocessors: StandardizationPreprocessor
SAC:
  actor_lr: 0.0001
  actor_network: SACActorNetwork
  batch_size: 256
  critic_lr: 0.0003
  critic_network: SACCriticNetwork
  initial_replay_size: 5000
  lr_alpha: 0.0003
  max_replay_size: 500000
  n_features: 256
  preprocessors: null
  target_entropy: null
  tau: 0.005
  warmup_transitions: 10000
TD3:
  actor_lr: 0.001
  actor_network: TD3ActorNetwork
  batch_size: 100
  critic_lr: 0.001
  critic_network: TD3CriticNetwork
  initial_replay_size: 10000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
  tau: 0.005

TRPO:
  batch_size: 64
  cg_damping: 0.01
  cg_residual_tol: 1.0e-10
  critic_fit_params: null
  critic_lr: 0.001
  critic_network: TRPONetwork
  ent_coeff: 0.0
  lam: 0.95
  max_kl: 0.01
  n_epochs_cg: 100
  n_epochs_line_search: 10
  n_features: 32
  n_steps_per_fit: 1000
  preprocessors: StandardizationPreprocessor
```


J
600
400
200
0
0
10
20
30
40
50
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG


R
12000
10000
8000
6000
4000
2000
0
−2000
0
10
20
30
40
50
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG



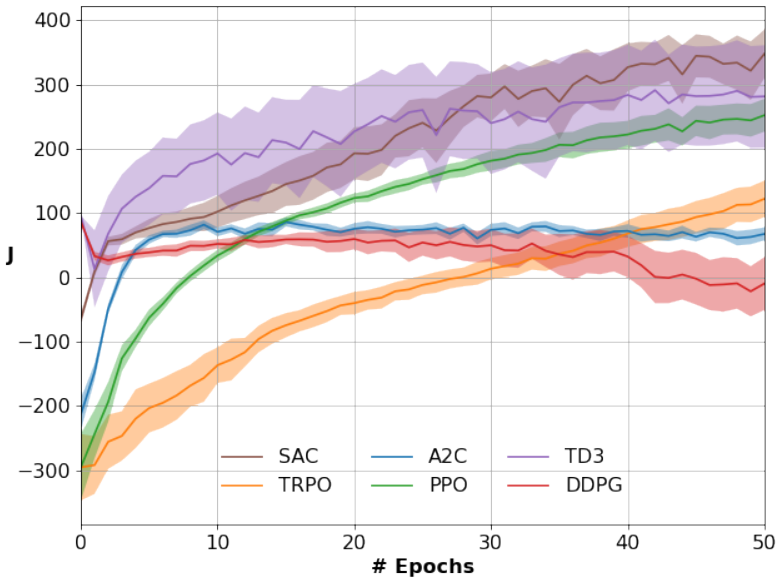
Ant-v3

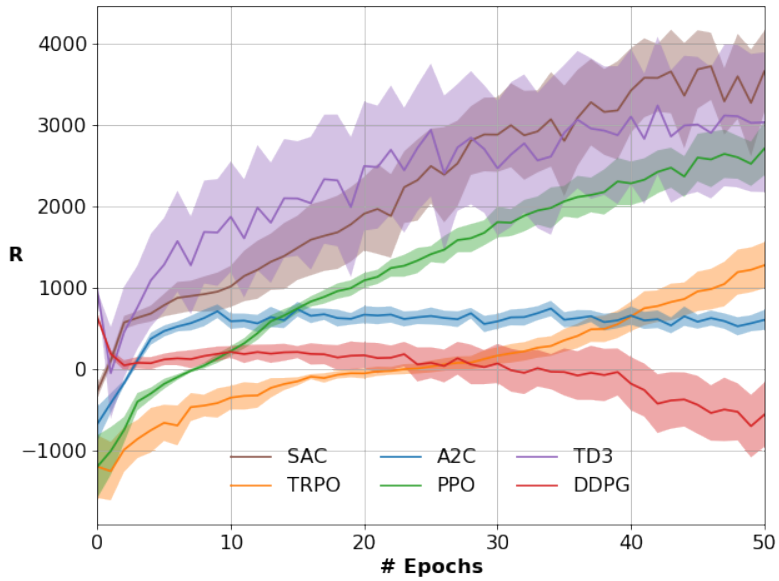
```yaml
A2C:
  actor_lr: 0.0007
  batch_size: 64
  critic_lr: 0.0007
  critic_network: A2CNetwork
  ent_coeff: 0.01
  eps_actor: 0.003
  eps_critic: 1.0e-05
  max_grad_norm: 0.5
  n_features: 64
  preprocessors: StandardizationPreprocessor
DDPG:
  actor_lr: 0.0001
  actor_network: DDPGActorNetwork
  batch_size: 128
  critic_lr: 0.001
  critic_network: DDPGCriticNetwork
  initial_replay_size: 5000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
  tau: 0.001
PPO:
  actor_lr: 0.0003
  batch_size: 32
  critic_fit_params: null
  critic_lr: 0.0003
  critic_network: TRPONetwork
  eps: 0.2
  lam: 0.95
  n_epochs_policy: 10
  n_features: 32
```

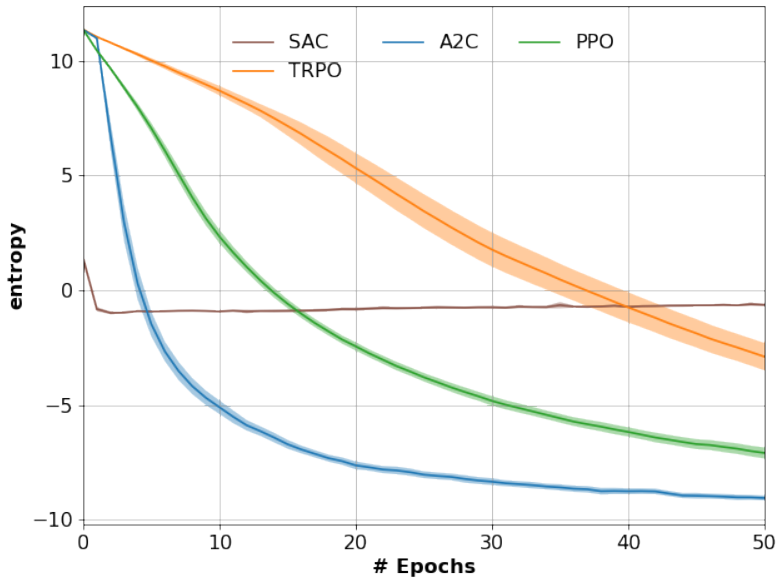
(continues on next page)

(continued from previous page)

```
  n_steps_per_fit: 2000
  preprocessors: StandardizationPreprocessor
SAC:
  actor_lr: 0.0001
  actor_network: SACActorNetwork
  batch_size: 256
  critic_lr: 0.0003
  critic_network: SACCriticNetwork
  initial_replay_size: 5000
  lr_alpha: 0.0003
  max_replay_size: 500000
  n_features: 256
  preprocessors: null
  target_entropy: null
  tau: 0.005
  warmup_transitions: 10000
TD3:
  actor_lr: 0.001
  actor_network: TD3ActorNetwork
  batch_size: 100
  critic_lr: 0.001
  critic_network: TD3CriticNetwork
  initial_replay_size: 10000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
  tau: 0.005
TRPO:
  batch_size: 64
  cg_damping: 0.01
  cg_residual_tol: 1.0e-10
  critic_fit_params: null
  critic_lr: 0.001
  critic_network: TRPONetwork
  ent_coeff: 0.0
  lam: 0.95
  max_kl: 0.01
  n_epochs_cg: 100
  n_epochs_line_search: 10
  n_features: 32
  n_steps_per_fit: 1000
  preprocessors: StandardizationPreprocessor
```


J
400
300
200
100
0
−100
−200
−300
0
10
20
30
40
50
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG


R
4000
3000
2000
1000
0
−1000
0
10
20
30
40
50
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG



3.1.3 Bullet Environments Benchmarks

Run Parameters	
n_runs	25
n_epochs	50
n_steps	30000
n_episodes_test	10

HopperBulletEnv-v0

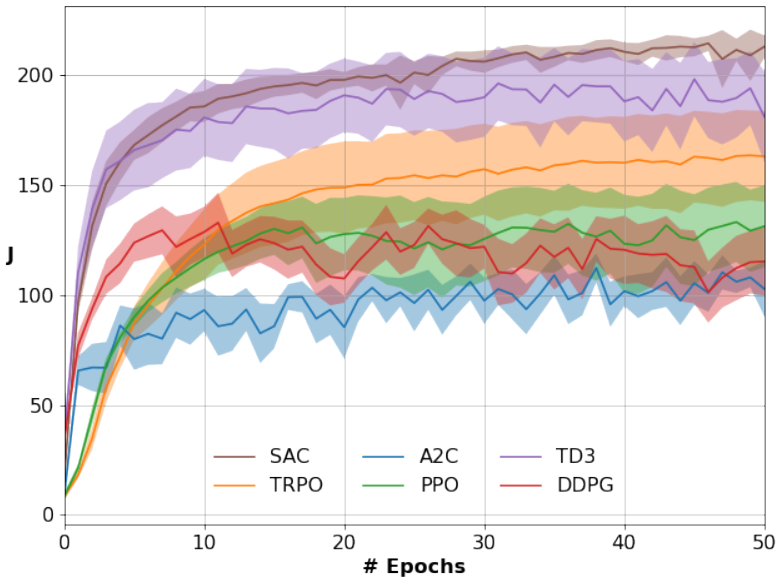
```
A2C:
  actor_lr: 0.0007
  batch_size: 64
  critic_lr: 0.0007
  critic_network: A2CNetwork
  ent_coeff: 0.01
  eps_actor: 0.003
  eps_critic: 1.0e-05
  max_grad_norm: 0.5
  n_features: 64
  preprocessors: null
DDPG:
  actor_lr: 0.0001
  actor_network: DDPGActorNetwork
  batch_size: 128
  critic_lr: 0.001
  critic_network: DDPGCriticNetwork
  initial_replay_size: 5000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
```

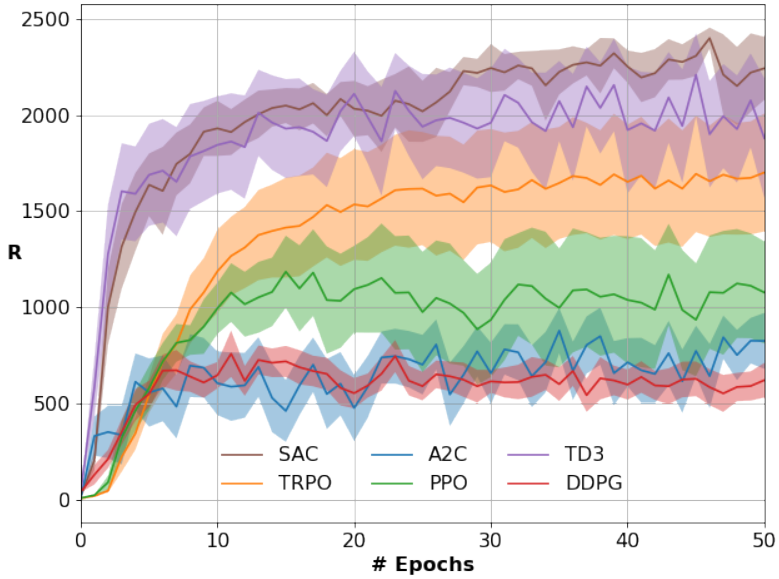
(continues on next page)

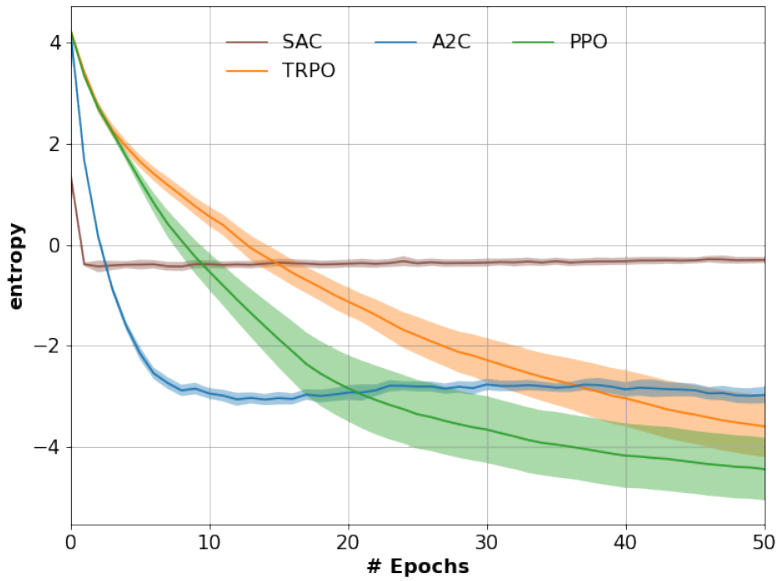
(continued from previous page)

```
  tau: 0.001
PPO:
  actor_lr: 0.0003
  batch_size: 64
  critic_fit_params: null
  critic_lr: 0.0003
  critic_network: TRPONetwork
  eps: 0.2
  lam: 0.95
  n_epochs_policy: 4
  n_features: 32
  n_steps_per_fit: 3000
  preprocessors: null
SAC:
  actor_lr: 0.0001
  actor_network: SACActorNetwork
  batch_size: 256
  critic_lr: 0.0003
  critic_network: SACCriticNetwork
  initial_replay_size: 5000
  lr_alpha: 0.0003
  max_replay_size: 500000
  n_features: 256
  preprocessors: null
  target_entropy: null
  tau: 0.005
  warmup_transitions: 10000
TD3:
  actor_lr: 0.001
  actor_network: TD3ActorNetwork
  batch_size: 100
  critic_lr: 0.001
  critic_network: TD3CriticNetwork
  initial_replay_size: 1000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
  tau: 0.005

TRPO:
  batch_size: 64
  cg_damping: 0.01
  cg_residual_tol: 1.0e-10
  critic_fit_params: null
  critic_lr: 0.003
  critic_network: TRPONetwork
  ent_coeff: 0.0
  lam: 0.95
  max_kl: 0.01
  n_epochs_cg: 100
  n_epochs_line_search: 10
  n_features: 32
  n_steps_per_fit: 3000
  preprocessors: null
```


J
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG


R
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG



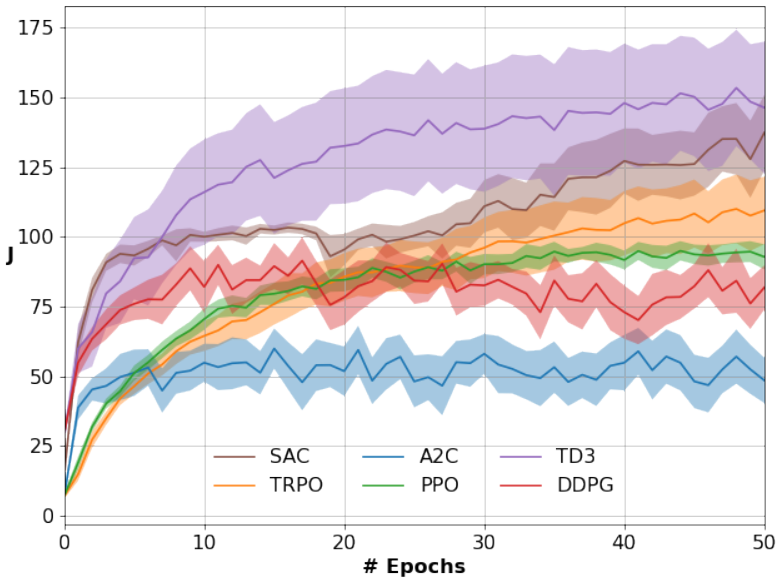
Walker2DBulletEnv-v0

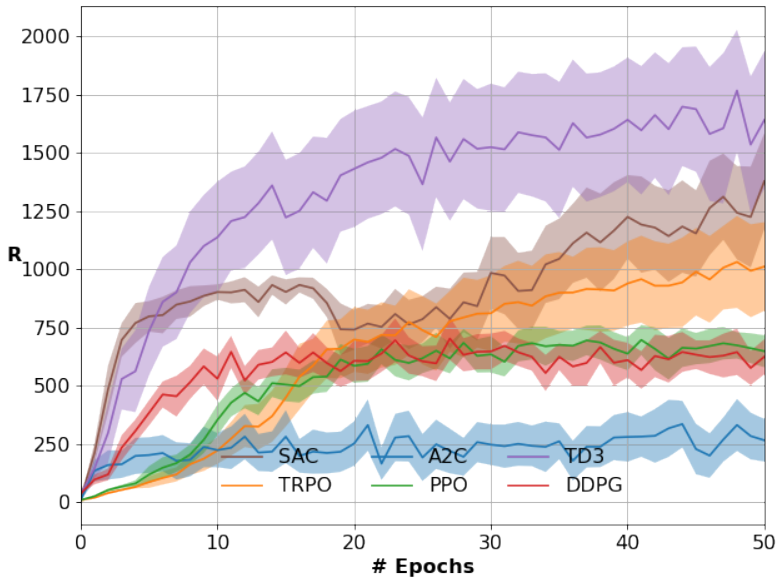
```yaml
A2C:
  actor_lr: 0.0007
  batch_size: 64
  critic_lr: 0.0007
  critic_network: A2CNetwork
  ent_coeff: 0.01
  eps_actor: 0.003
  eps_critic: 1.0e-05
  max_grad_norm: 0.5
  n_features: 64
  preprocessors: null
DDPG:
  actor_lr: 0.0001
  actor_network: DDPGActorNetwork
  batch_size: 128
  critic_lr: 0.001
  critic_network: DDPGCriticNetwork
  initial_replay_size: 5000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
  tau: 0.001
PPO:
  actor_lr: 0.0003
  batch_size: 64
  critic_fit_params: null
  critic_lr: 0.0003
  critic_network: TRPONetwork
  eps: 0.2
  lam: 0.95
  n_epochs_policy: 4
  n_features: 32
```

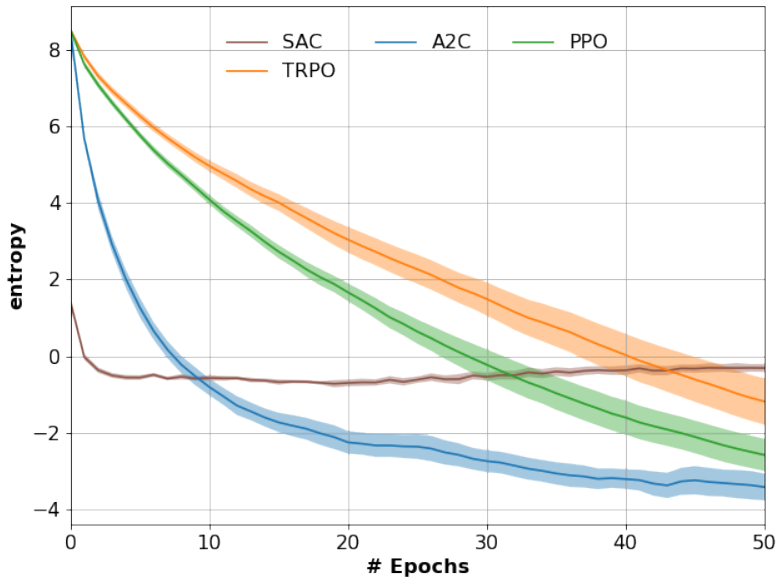
(continues on next page)

(continued from previous page)

```
  n_steps_per_fit: 3000
  preprocessors: null
SAC:
  actor_lr: 0.0001
  actor_network: SACActorNetwork
  batch_size: 256
  critic_lr: 0.0003
  critic_network: SACCriticNetwork
  initial_replay_size: 5000
  lr_alpha: 0.0003
  max_replay_size: 500000
  n_features: 256
  preprocessors: null
  target_entropy: null
  tau: 0.005
  warmup_transitions: 10000
TD3:
  actor_lr: 0.001
  actor_network: TD3ActorNetwork
  batch_size: 100
  critic_lr: 0.001
  critic_network: TD3CriticNetwork
  initial_replay_size: 1000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
  tau: 0.005
TRPO:
  batch_size: 64
  cg_damping: 0.01
  cg_residual_tol: 1.0e-10
  critic_fit_params: null
  critic_lr: 0.003
  critic_network: TRPONetwork
  ent_coeff: 0.0
  lam: 0.95
  max_kl: 0.01
  n_epochs_cg: 100
  n_epochs_line_search: 10
  n_features: 32
  n_steps_per_fit: 3000
  preprocessors: null
```


J
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG


R
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG



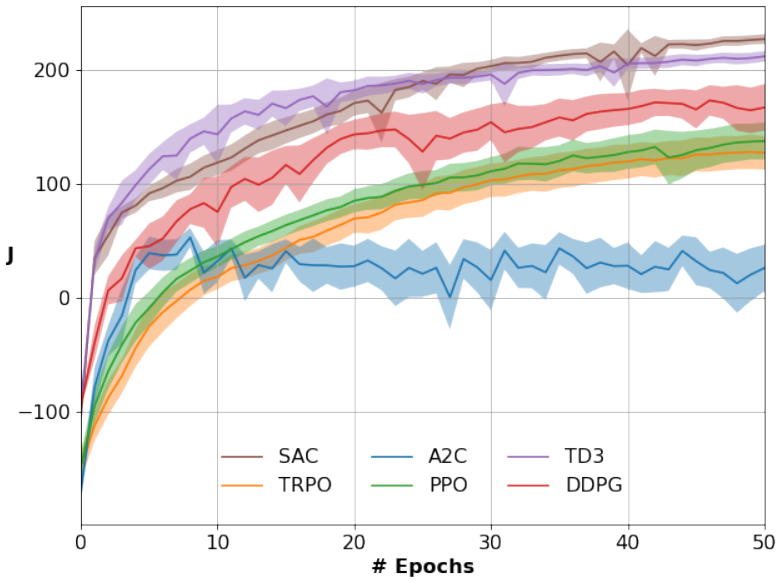
HalfCheetahBulletEnv-v0

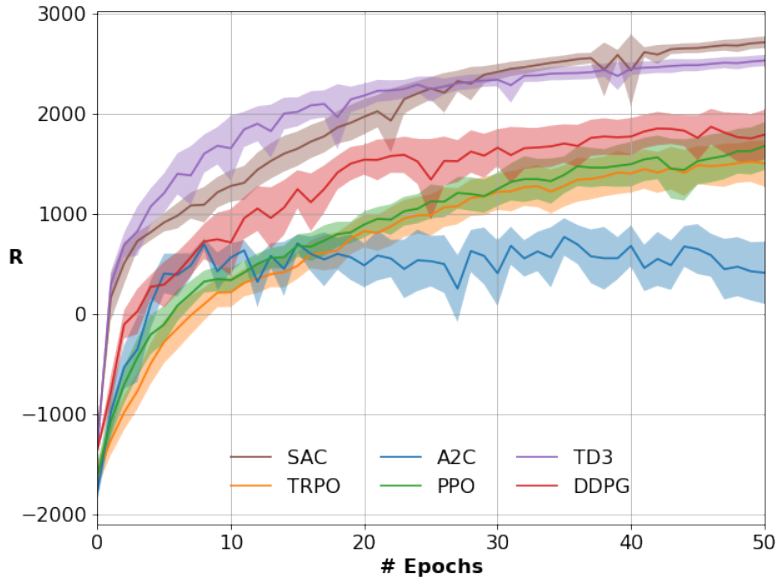
```yaml
A2C:
  actor_lr: 0.0007
  batch_size: 64
  critic_lr: 0.0007
  critic_network: A2CNetwork
  ent_coeff: 0.01
  eps_actor: 0.003
  eps_critic: 1.0e-05
  max_grad_norm: 0.5
  n_features: 64
  preprocessors: null
DDPG:
  actor_lr: 0.0001
  actor_network: DDPGActorNetwork
  batch_size: 128
  critic_lr: 0.001
  critic_network: DDPGCriticNetwork
  initial_replay_size: 5000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
  tau: 0.001
PPO:
  actor_lr: 0.0003
  batch_size: 64
  critic_fit_params: null
  critic_lr: 0.0003
  critic_network: TRPONetwork
  eps: 0.2
  lam: 0.95
  n_epochs_policy: 4
  n_features: 32
```

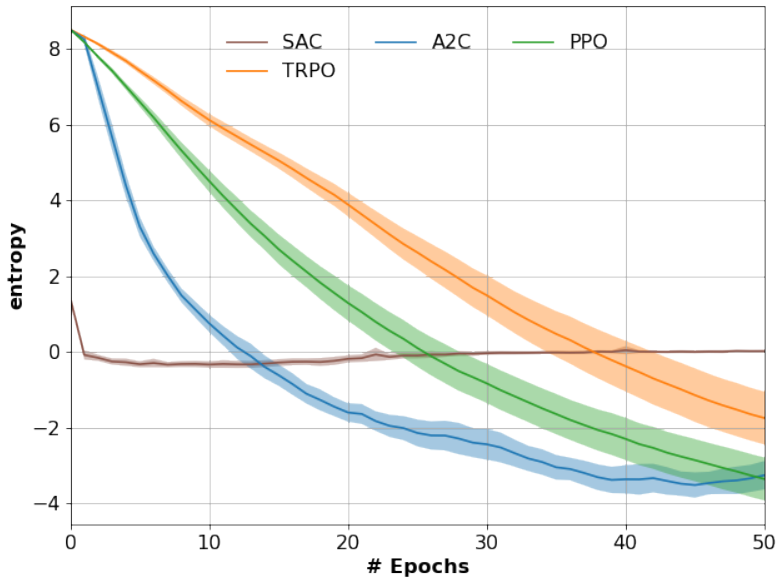
(continues on next page)

(continued from previous page)

```
  n_steps_per_fit: 3000
  preprocessors: null
SAC:
  actor_lr: 0.0001
  actor_network: SACActorNetwork
  batch_size: 256
  critic_lr: 0.0003
  critic_network: SACCriticNetwork
  initial_replay_size: 5000
  lr_alpha: 0.0003
  max_replay_size: 500000
  n_features: 256
  preprocessors: null
  target_entropy: null
  tau: 0.005
  warmup_transitions: 10000
TD3:
  actor_lr: 0.001
  actor_network: TD3ActorNetwork
  batch_size: 100
  critic_lr: 0.001
  critic_network: TD3CriticNetwork
  initial_replay_size: 10000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
  tau: 0.005
TRPO:
  batch_size: 64
  cg_damping: 0.01
  cg_residual_tol: 1.0e-10
  critic_fit_params: null
  critic_lr: 0.003
  critic_network: TRPONetwork
  ent_coeff: 0.0
  lam: 0.95
  max_kl: 0.01
  n_epochs_cg: 100
  n_epochs_line_search: 10
  n_features: 32
  n_steps_per_fit: 3000
  preprocessors: null
```


J
200
100
0
−100
0
10
20
30
40
50
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG


R
3000
2000
1000
0
−1000
−2000
0
10
20
30
40
50
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG



AntBulletEnv-v0

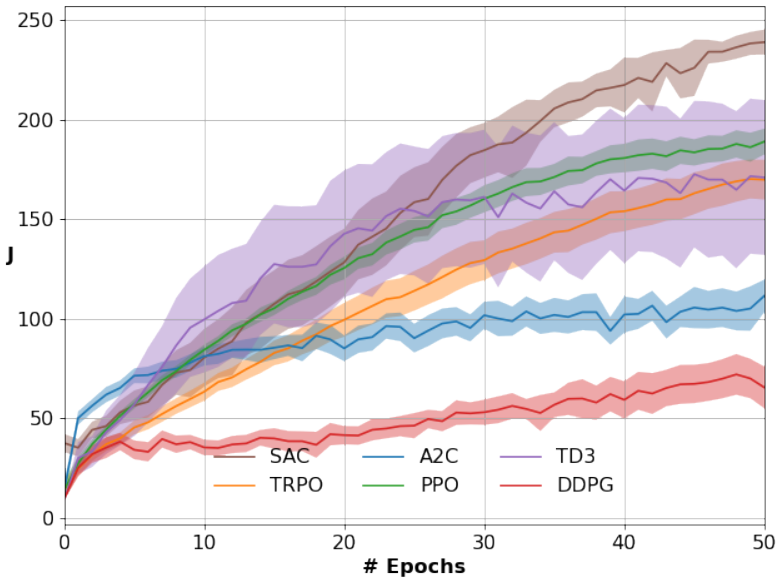
```yaml
A2C:
  actor_lr: 0.0007
  batch_size: 64
  critic_lr: 0.0007
  critic_network: A2CNetwork
  ent_coeff: 0.01
  eps_actor: 0.003
  eps_critic: 1.0e-05
  max_grad_norm: 0.5
  n_features: 64
  preprocessors: null
DDPG:
  actor_lr: 0.0001
  actor_network: DDPGActorNetwork
  batch_size: 128
  critic_lr: 0.001
  critic_network: DDPGCriticNetwork
  initial_replay_size: 5000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
  tau: 0.001
PPO:
  actor_lr: 0.0003
  batch_size: 64
  critic_fit_params: null
  critic_lr: 0.0003
  critic_network: TRPONetwork
  eps: 0.2
  lam: 0.95
  n_epochs_policy: 4
  n_features: 32
```

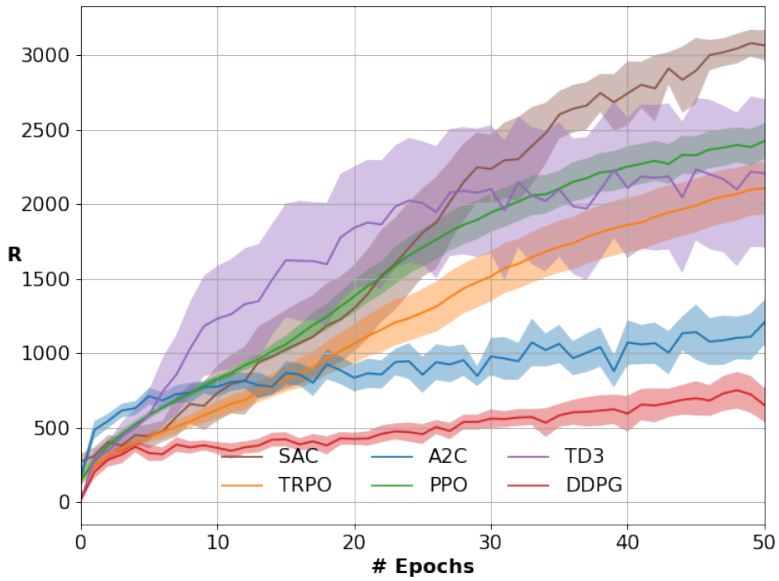
(continues on next page)

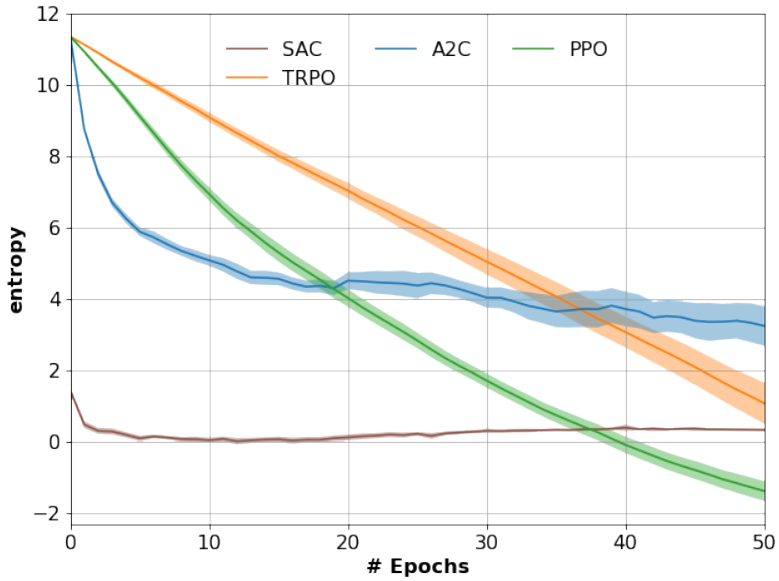
(continued from previous page)

```
  n_steps_per_fit: 3000
  preprocessors: null
SAC:
  actor_lr: 0.0001
  actor_network: SACActorNetwork
  batch_size: 256
  critic_lr: 0.0003
  critic_network: SACCriticNetwork
  initial_replay_size: 5000
  lr_alpha: 0.0003
  max_replay_size: 500000
  n_features: 256
  preprocessors: null
  target_entropy: null
  tau: 0.005
  warmup_transitions: 10000
TD3:
  actor_lr: 0.001
  actor_network: TD3ActorNetwork
  batch_size: 100
  critic_lr: 0.001
  critic_network: TD3CriticNetwork
  initial_replay_size: 10000
  max_replay_size: 1000000
  n_features:
  - 400
  - 300
  tau: 0.005

TRPO:
  batch_size: 64
  cg_damping: 0.01
  cg_residual_tol: 1.0e-10
  critic_fit_params: null
  critic_lr: 0.003
  critic_network: TRPONetwork
  ent_coeff: 0.0
  lam: 0.95
  max_kl: 0.01
  n_epochs_cg: 100
  n_epochs_line_search: 10
  n_features: 32
  n_steps_per_fit: 3000
  preprocessors: null
```


J
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG


R
Epochs
SAC
TRPO
A2C
PPO
TD3
DDPG



3.2 Value-Based Benchmarks

We provide the benchmarks for the following DQN algorithms:

- **DQN**
- **PrioritiziedDQN**
- **DoubleDQN**
- **AveragedDQN**
- **DuelingDQN**
- **MaxminDQN**
- **CategoricalDQN**
- **NoisyDQN**

We consider the following environments in the benchmark

3.2.1 Breakout Environment Benchmark

Run Parameters	
n_runs	5
n_epochs	200
n_steps	250000
n_episodes_test	125000

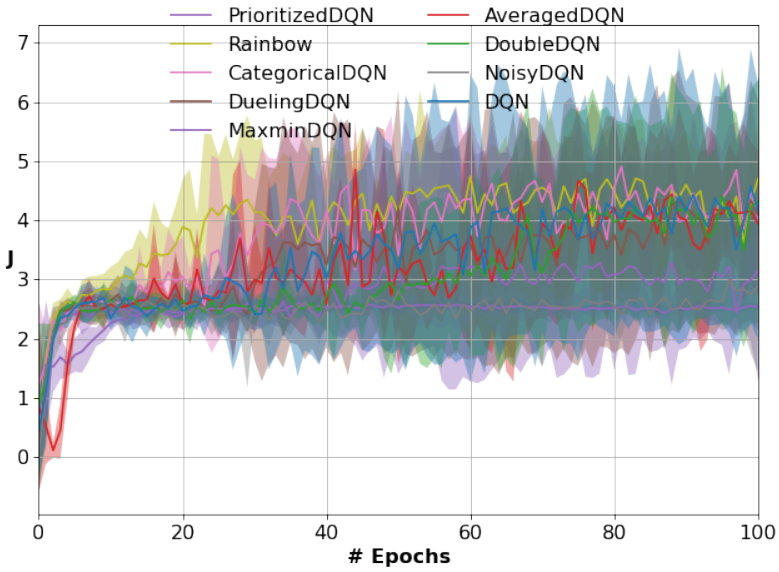
BreakoutDeterministic-v4

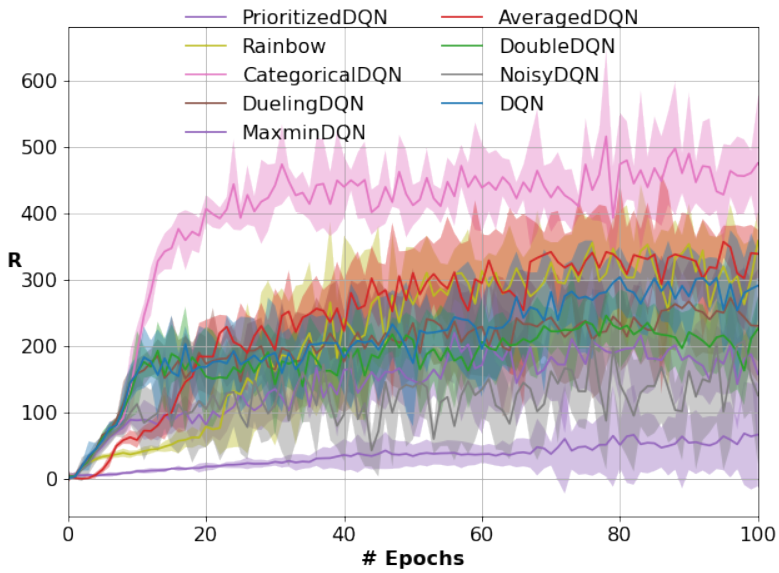
```
AveragedDQN:
  batch_size: 32
  initial_replay_size: 50000
  lr: 0.0001
  max_replay_size: 1000000
  n_approximators: 10
  n_steps_per_fit: 4
  network: DQNNetwork
  target_update_frequency: 2500
CategoricalDQN:
  batch_size: 32
  initial_replay_size: 50000
  lr: 0.0001
  max_replay_size: 1000000
  n_atoms: 51
  n_features: 512
  n_steps_per_fit: 4
  network: DQNFeatureNetwork
  target_update_frequency: 2500
  v_max: 10
  v_min: -10
DQN:
  batch_size: 32
  initial_replay_size: 50000
  lr: 0.0001
  max_replay_size: 1000000
  n_steps_per_fit: 4
  network: DQNNetwork
  target_update_frequency: 2500
DoubleDQN:
  batch_size: 32
  initial_replay_size: 50000
  lr: 0.0001
  max_replay_size: 1000000
  n_steps_per_fit: 4
  network: DQNNetwork
  target_update_frequency: 2500
DuelingDQN:
  batch_size: 32
  initial_replay_size: 50000
  lr: 0.0001
  max_replay_size: 1000000
  n_features: 512
  n_steps_per_fit: 4
  network: DQNFeatureNetwork
  target_update_frequency: 2500
MaxminDQN:
  batch_size: 32
  initial_replay_size: 50000
  lr: 0.0001
  max_replay_size: 1000000
  n_approximators: 3
  n_steps_per_fit: 4
  network: DQNNetwork
  target_update_frequency: 2500
NoisyDQN:
  batch_size: 32
```

(continues on next page)

(continued from previous page)

```
  initial_replay_size: 50000
  lr: 0.0001
  max_replay_size: 1000000
  n_features: 512
  n_steps_per_fit: 4
  network: DQNFeatureNetwork
  target_update_frequency: 2500
PrioritizedDQN:
  batch_size: 32
  initial_replay_size: 50000
  lr: 0.0001
  max_replay_size: 1000000
  n_steps_per_fit: 4
  network: DQNNetwork
  target_update_frequency: 2500
```





3.3 Core functionality

3.3.1 Suite

class mushroom_rl_benchmark.core.suite.**BenchmarkSuite**(*log_dir=None*, *log_id=None*, *use_timestamp=True*, *parallel=None*, *slurm=None*)

Bases: object

Class to orchestrate the execution of multiple experiments.

__init__(*log_dir=None*, *log_id=None*, *use_timestamp=True*, *parallel=None*, *slurm=None*)
Constructor.

Parameters

- **log_dir** (*str*) – path to the log directory (Default: ./logs or /work/scratch/$USER)
- **log_id** (*str*) – log id (Default: benchmark[_YYYY-mm-dd-HH-MM-SS])
- **use_timestamp** (*bool*) – select if a timestamp should be appended to the log id
- **parallel** (*dict, None*) – parameters that are passed to the run_parallel method of the experiment
- **slurm** (*dict, None*) – parameters that are passed to the run_slurm method of the experiment

add_experiments(*environment_name*, *environment_builder_params*, *agent_names_list*, *agent_builders_params*, ***run_params*)
Add a set of experiments for the same environment to the suite.

Parameters

- **environment_name** (*str*) – name of the environment for the experiment (E.g. Gym.Pendulum-v0);
- **environment_builder_params** (*dict*) – parameters for the environment builder;
- **agent_names_list** (*list*) – list of names of the agents for the experiments;
- **agent_builders_params** (*list*) – list of dictionaries containing the parameters for the agent builder;
- **run_params** – Parameters that are passed to the run method of the experiment.

add_environment(*environment_name*, *environment_builder_params*, ***run_params*)
Add an environment to the benchmarking suite.

Parameters

- **environment_name** (*str*) – name of the environment for the experiment (E.g. Gym.Pendulum-v0);
- **environment_builder_params** (*dict*) – parameters for the environment builder;
- **run_params** – Parameters that are passed to the run method of the experiment.

add_agent(*environment_name*, *agent_name*, *agent_params*)
Add an agent to the benchmarking suite.

Parameters

- **environment_name** (*str*) – name of the environment for the experiment (E.g. Gym.Pendulum-v0);

- **agent_name** (*str*) – name of the agent for the experiments;
- **agent_params** (*list*) – dictionary containing the parameters for the agent builder.

run(*exec_type='sequential'*)
Run all experiments in the suite.

print_experiments()
Print the experiments in the suite.

save_parameters()
Save the experiment parameters in yaml files inside the parameters folder

save_plots(***plot_params*)
Save the result plots to the log directory.

Parameters ****plot_params** – parameters to be passed to the suite visualizer.

show_plots(***plot_params*)
Display the result plots.

Parameters ****plot_params** – parameters to be passed to the suite visualizer.

3.3.2 Experiment

class mushroom_rl_benchmark.core.experiment.**BenchmarkExperiment**(*agent_builder*, *env_builder*, *logger*)

Bases: object

Class to create and run an experiment using MushroomRL

__init__(*agent_builder*, *env_builder*, *logger*)
Constructor.

Parameters

- **agent_builder** (AgentBuilder) – instance of a specific agent builder;
- **env_builder** (EnvironmentBuilder) – instance of an environment builder;
- **logger** (BenchmarkLogger) – instance of a benchmark logger.

run(*exec_type='sequential'*, ***run_params*)
Execute the experiment.

Parameters

- **exec_type** (*str, 'sequential'*) – type of executing the experiment [sequential|parallel|slurm];
- ****run_params** – parameters for the selected execution type.

run_sequential(*n_runs*, *n_runs_completed=0*, *save_plot=True*, ***run_params*)
Execute the experiment sequential.

Parameters

- **n_runs** (*int*) – number of total runs of the experiment;
- **n_runs_completed** (*int, 0*) – number of completed runs of the experiment;
- **save_plot** (*bool, True*) – select if a plot of the experiment should be saved to the log directory;

- ****run_params** – parameters for executing a benchmark run.

run_parallel(*n_runs*, *n_runs_completed=0*, *threading=False*, *save_plot=True*, *max_concurrent_runs=None*, ***run_params*)
Execute the experiment in parallel threads.

Parameters

- **n_runs** (*int*) – number of total runs of the experiment;
- **n_runs_completed** (*int, 0*) – number of completed runs of the experiment;
- **threading** (*bool, False*) – select to use threads instead of processes;
- **save_plot** (*bool, True*) – select if a plot of the experiment should be saved to the log directory;
- **max_concurrent_runs** (*int, -1*) – maximum number of concurrent runs. By default it uses the number of cores;
- ****run_params** – parameters for executing a benchmark run.

run_slurm(*n_runs*, *n_runs_completed=0*, *aggregation_job=True*, *aggregate_hours=3*, *aggregate_minutes=0*, *aggregate_seconds=0*, *only_print=False*, ***run_params*)
Execute the experiment with SLURM.

Parameters

- **n_runs** (*int*) – number of total runs of the experiment;
- **n_runs_completed** (*int, 0*) – number of completed runs of the experiment;
- **aggregation_job** (*bool, True*) – select if an aggregation job should be scheduled;
- **aggregate_hours** (*int, 3*) – maximum number of hours for the aggregation job;
- **aggregate_minutes** (*int, 0*) – maximum number of minutes for the aggregation job;
- **aggregate_seconds** (*int, 0*) – maximum number of seconds for the aggregation job;
- **only_print** (*bool, False*) – if True, don't launch the benchmarks, only print the submitted commands to the terminal;
- ****run_params** – parameters for executing a benchmark run.

reset()
Reset the internal state of the experiment.

resume(*logger*)
Resume an experiment from disk

start_timer()
Start the timer.

stop_timer()
Stop the timer.

save_builders()
Save agent and environment builder to the log directory.

extend_and_save_J(*J*)
Extend J with another datapoint and save the current state to the log directory.

extend_and_save_R(*R*)
Extend R with another datapoint and save the current state to the log directory.

extend_and_save_V(*V*)
Extend V with another datapoint and save the current state to the log directory.

extend_and_save_entropy(*entropy*)
Extend entropy with another datapoint and save the current state to the log directory.

set_and_save_config(***settings*)
Save the experiment configuration to the log directory.

set_and_save_stats(***info*)
Save the run statistics to the log directory.

save_plot()
Save the result plot to the log directory.

show_plot()
Display the result plot.

3.3.3 Logger

class mushroom_rl_benchmark.core.logger.**BenchmarkLogger**(*log_dir=None*, *log_id=None*, *use_timestamp=True*)
Bases: mushroom_rl.core.logger.console_logger.ConsoleLogger

Class to handle all interactions with the log directory.

__init__(*log_dir=None*, *log_id=None*, *use_timestamp=True*)
Constructor.

Parameters

- **log_dir** (*str, None*) – path to the log directory, if not specified defaults to ./logs or to /work/scratch/$USER if the second directory exists;
- **log_id** (*str, None*) – log id, if not specified defaults to: benchmark[_YY-mm-ddTHH:MM:SS.zzz]);
- **use_timestamp** (*bool, True*) – select if a timestamp should be appended to the log id.

set_log_dir(*log_dir*)

get_log_dir()

set_log_id(*log_id*, *use_timestamp=True*)

get_log_id()

get_path(*filename=''*)

get_params_path(*filename=''*)

get_figure_path(*filename=''*, *subfolder=None*)

save_J(*J*)

load_J()

save_R(*R*)

load_R()

save_V(*V*)

load_V()

save_entropy(*entropy*)

load_entropy()

exists_policy_entropy()

save_best_agent(*agent*)

save_last_agent(*agent*)

exists_best_agent()

load_best_agent()

load_last_agent()

save_environment_builder(*env_builder*)

load_environment_builder()

save_agent_builder(*agent_builder*)

load_agent_builder()

save_config(*config*)

load_config()

exists_stats()

save_stats(*stats*)

load_stats()

save_params(*env*, *params*)

save_figure(*figure*, *figname*, *subfolder=None*, *as_pdf=False*, *transparent=True*)

classmethod from_path(*path*)
Method to create a BenchmarkLogger from a path.

3.3.4 Visualizer

class mushroom_rl_benchmark.core.visualizer.**BenchmarkVisualizer**(*logger*, *data=None*, *has_entropy=None*, *id=1*)

Bases: object

Class to handle all visualizations of the experiment.

plot_counter = 0

__init__(*logger*, *data=None*, *has_entropy=None*, *id=1*)
Constructor.

Parameters

- **logger** (BenchmarkLogger) – logger to be used;
- **data** (*dict, None*) – dictionary with data points for visualization;
- **has_entropy** (*bool, None*) – select if entropy is available for the algorithm.

is_data_persisted
Check if data was passed as dictionary or should be read from log directory.

get_J()
Get J from dictionary or log directory.

get_R()
Get R from dictionary or log directory.

get_V()
Get V from dictionary or log directory.

get_entropy()
Get entropy from dictionary or log directory.

get_report()
Create report plot with matplotlib.

save_report(*file_name='report_plot'*)
Method to save an image of a report of the training metrics from a performend experiment.

show_report()
Method to show a report of the training metrics from a performend experiment.

show_agent(*episodes=5*, *mdp_render=False*)
Method to run and visualize the best builders in the environment.

classmethod from_path(*path*)
Method to create a BenchmarkVisualizer from a path.

3.4 Builders

class mushroom_rl_benchmark.builders.environment_builder.**EnvironmentBuilder**(*env_name*, *env_params*)

Bases: object

Class to spawn instances of a MushroomRL environment

__init__(*env_name*, *env_params*)
Constructor

Parameters

- **env_name** – name of the environment to build;
- **env_params** – required parameters to build the specified environment.

build()
Build and return an environment

static set_eval_mode(*env*, *eval*)
Make changes to the environment for evaluation mode.

Parameters

- **env** (*Environment*) – the environment to change;
- **eval** (*bool*) – flag for activating evaluation mode.

copy()
Create a deepcopy of the environment_builder and return it

class mushroom_rl_benchmark.builders.agent_builder.**AgentBuilder**(*n_steps_per_fit*, *compute_policy_entropy=True*, *compute_entropy_with_states=False*, *preprocessors=None*)

Bases: object

Base class to spawn instances of a MushroomRL agent

__init__(*n_steps_per_fit*, *compute_policy_entropy=True*, *compute_entropy_with_states=False*, *preprocessors=None*)
Initialize AgentBuilder

set_n_steps_per_fit(*n_steps_per_fit*)
Set n_steps_per_fit for the specific AgentBuilder

Parameters **n_steps_per_fit** – number of steps per fit.

get_n_steps_per_fit()
Get n_steps_per_fit for the specific AgentBuilder

set_preprocessors(*preprocessors*)
Set preprocessor for the specific AgentBuilder

Parameters **preprocessors** – list of preprocessor classes.

get_preprocessors()
Get preprocessors for the specific AgentBuilder

copy()
Create a deepcopy of the AgentBuilder and return it

build(*mdp_info*)
Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

compute_Q(*agent*, *states*)
Compute the Q Value for an AgentBuilder

Parameters

- **agent** (*Agent*) – the considered agent;
- **states** (*np.ndarray*) – the set of states over which we need to compute the Q function.

set_eval_mode(*agent*, *eval*)
Set the eval mode for the agent. This function can be overwritten by any agent builder to setup specific evaluation mode for the agent.

Parameters

- **agent** (*Agent*) – the considered agent;
- **eval** (*bool*) – whether to set eval mode (true) or learn mode.

classmethod default(*get_default_dict=False*, ***kwargs*)
Create a default initialization for the specific AgentBuilder and return it

3.4.1 Value Based Builders

class mushroom_rl_benchmark.builders.value.dqn.**DQNBuilder**(*policy*, *approximator*, *approximator_params*, *alg_params*, *n_steps_per_fit=1*)

Bases: *mushroom_rl_benchmark.builders.agent_builder.AgentBuilder*

AgentBuilder for Deep Q-Network (DQN).

__init__(*policy*, *approximator*, *approximator_params*, *alg_params*, *n_steps_per_fit=1*)

Constructor.

Parameters

- **policy** (*Policy*) – policy class;
- **approximator** (*dict*) – Q-function approximator;
- **approximator_params** (*dict*) – parameters of the Q-function approximator;
- **alg_params** (*dict*) – parameters for the algorithm;
- **n_steps_per_fit** (*int, 1*) – number of steps per fit.

build(*mdp_info*)

Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

compute_Q(*agent*, *states*)

Compute the Q Value for an AgentBuilder

Parameters

- **agent** (*Agent*) – the considered agent;
- **states** (*np.ndarray*) – the set of states over which we need to compute the Q function.

set_eval_mode(*agent*, *eval*)

Set the eval mode for the agent. This function can be overwritten by any agent builder to setup specific evaluation mode for the agent.

Parameters

- **agent** (*Agent*) – the considered agent;
- **eval** (*bool*) – whether to set eval mode (true) or learn mode.

classmethod default(*lr=0.0001*, *network=<class 'mushroom_rl_benchmark.builders.network.dqn_network.DQNNetwor*
initial_replay_size=50000, *max_replay_size=1000000*, *batch_size=32*, *target_update_frequency=2500*, *n_steps_per_fit=1*, *use_cuda=False*, *get_default_dict=False*)

Create a default initialization for the specific AgentBuilder and return it

class mushroom_rl_benchmark.builders.value.double_dqn.**DoubleDQNBuilder**(*policy*, *approximator*, *approximator_params*, *alg_params*, *n_steps_per_fit=1*)

Bases: *mushroom_rl_benchmark.builders.value.dqn.DQNBuilder*

build(*mdp_info*)

Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

class mushroom_rl_benchmark.builders.value.averaged_dqn.**AveragedDQNBuilder**(*policy*, *approximator*, *approximator_params*, *alg_params*, *n_steps_per_fit=1*)

Bases: *mushroom_rl_benchmark.builders.value.dqn.DQNBuilder*

build(*mdp_info*)

Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

classmethod **default**(*lr=0.0001*, *network=<class 'mushroom_rl_benchmark.builders.network.dqn_network.DQNNetwor*
initial_replay_size=50000, *max_replay_size=1000000*, *batch_size=32*, *target_update_frequency=2500*, *n_steps_per_fit=1*, *n_approximators=10*, *use_cuda=False*, *get_default_dict=False*)

Create a default initialization for the specific AgentBuilder and return it

class mushroom_rl_benchmark.builders.value.prioritized_dqn.**PrioritizedDQNBuilder**(*policy*, *approximator*, *approximator_params*, *alg_params*, *n_steps_per_f*

Bases: *mushroom_rl_benchmark.builders.value.dqn.DQNBuilder*

build(*mdp_info*)
Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

classmethod default(*lr=0.0001*, *network=<class 'mushroom_rl_benchmark.builders.network.dqn_network.DQNNetwor*
initial_replay_size=50000, *max_replay_size=1000000*, *batch_size=32*,
target_update_frequency=2500, *n_steps_per_fit=1*, *use_cuda=False*,
get_default_dict=False)
Create a default initialization for the specific AgentBuilder and return it

class mushroom_rl_benchmark.builders.value.dueling_dqn.**DuelingDQNBuilder**(*policy*, *approximator*, *approximator_params*, *alg_params*, *n_steps_per_fit=1*)
Bases: *mushroom_rl_benchmark.builders.value.dqn.DQNBuilder*

build(*mdp_info*)
Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

classmethod default(*lr=0.0001*, *network=<class 'mushroom_rl_benchmark.builders.network.dqn_network.DQNFeature*
initial_replay_size=50000, *max_replay_size=1000000*, *batch_size=32*,
target_update_frequency=2500, *n_features=512*, *n_steps_per_fit=1*,
use_cuda=False, *get_default_dict=False*)
Create a default initialization for the specific AgentBuilder and return it

class mushroom_rl_benchmark.builders.value.maxmin_dqn.**MaxminDQNBuilder**(*policy*, *approximator*, *approximator_params*, *alg_params*, *n_steps_per_fit=1*)
Bases: *mushroom_rl_benchmark.builders.value.dqn.DQNBuilder*

build(*mdp_info*)
Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

classmethod default(*lr=0.0001*, *network=<class 'mushroom_rl_benchmark.builders.network.dqn_network.DQNNetwor*
initial_replay_size=50000, *max_replay_size=1000000*, *batch_size=32*, *target_update_frequency=2500*, *n_steps_per_fit=1*, *n_approximators=3*, *use_cuda=False*, *get_default_dict=False*)
Create a default initialization for the specific AgentBuilder and return it

class mushroom_rl_benchmark.builders.value.noisy_dqn.**NoisyDQNBuilder**(*policy*, *approximator*, *approximator_params*, *alg_params*, *n_steps_per_fit=1*)

Bases: *mushroom_rl_benchmark.builders.value.dqn.DQNBuilder*

build(*mdp_info*)

Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

classmethod default(*lr=0.0001*, *network=<class 'mushroom_rl_benchmark.builders.network.dqn_network.DQNFeature*, *initial_replay_size=50000*, *max_replay_size=1000000*, *batch_size=32*, *target_update_frequency=2500*, *n_features=512*, *n_steps_per_fit=1*, *use_cuda=False*, *get_default_dict=False*)

Create a default initialization for the specific AgentBuilder and return it

class mushroom_rl_benchmark.builders.value.categorical_dqn.**CategoricalDQNBuilder**(*policy*, *approximator*, *approximator_params*, *alg_params*, *n_steps_per_f*

Bases: *mushroom_rl_benchmark.builders.value.dqn.DQNBuilder*

build(*mdp_info*)

Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

classmethod default(*lr=0.0001*, *network=<class 'mushroom_rl_benchmark.builders.network.dqn_network.DQNFeature*, *initial_replay_size=50000*, *max_replay_size=1000000*, *batch_size=32*, *target_update_frequency=2500*, *n_features=512*, *n_steps_per_fit=1*, *v_min=-10*, *v_max=10*, *n_atoms=51*, *use_cuda=False*, *get_default_dict=False*)

Create a default initialization for the specific AgentBuilder and return it

3.4.2 Actor Critic Builders

class mushroom_rl_benchmark.builders.actor_critic.a2c.**A2CBuilder**(*policy_params*, *actor_optimizer*, *critic_params*, *alg_params*, *n_steps_per_fit=5*, *preprocessors=None*)

Bases: *mushroom_rl_benchmark.builders.agent_builder.AgentBuilder*

AgentBuilder for Advantage Actor Critic algorithm (A2C)

__init__(*policy_params*, *actor_optimizer*, *critic_params*, *alg_params*, *n_steps_per_fit=5*, *preprocessors=None*)

Constructor.

Parameters

- **policy_params** (*dict*) – parameters for the policy;
- **actor_optimizer** (*dict*) – parameters for the actor optimizer;
- **critic_params** (*dict*) – parameters for the critic;
- **alg_params** (*dict*) – parameters for the algorithm;
- **n_steps_per_fit** (*int, 5*) – number of steps per fit;
- **preprocessors** (*list, None*) – list of preprocessors.

build(*mdp_info*)

Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

compute_Q(*agent*, *states*)

Compute the Q Value for an AgentBuilder

Parameters

- **agent** (*Agent*) – the considered agent;
- **states** (*np.ndarray*) – the set of states over which we need to compute the Q function.

classmethod default(*actor_lr=0.0007*, *critic_lr=0.0007*, *eps_actor=0.003*, *eps_critic=1e-05*, *batch_size=64*, *max_grad_norm=0.5*, *ent_coeff=0.01*, *critic_network=<class 'mushroom_rl_benchmark.builders.network.a2c_network.A2CNetwork'>*, *n_features=64*, *preprocessors=None*, *use_cuda=False*, *get_default_dict=False*)

Create a default initialization for the specific AgentBuilder and return it

class mushroom_rl_benchmark.builders.actor_critic.ddpg.**DDPGBuilder**(*policy_class*, *policy_params*, *actor_params*, *actor_optimizer*, *critic_params*, *alg_params*, *n_steps_per_fit=1*)

Bases: *mushroom_rl_benchmark.builders.agent_builder.AgentBuilder*

AgentBuilder for Deep Deterministic Policy Gradient algorithm (DDPG)

__init__(*policy_class*, *policy_params*, *actor_params*, *actor_optimizer*, *critic_params*, *alg_params*, *n_steps_per_fit=1*)

Constructor.

Parameters

- **policy_class** (*Policy*) – policy class;
- **policy_params** (*dict*) – parameters for the policy;
- **actor_params** (*dict*) – parameters for the actor;
- **actor_optimizer** (*dict*) – parameters for the actor optimizer;
- **critic_params** (*dict*) – parameters for the critic;
- **alg_params** (*dict*) – parameters for the algorithm;
- **n_steps_per_fit** (*int, 1*) – number of steps per fit.

build(*mdp_info*)

Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

compute_Q(*agent*, *states*)

Compute the Q Value for an AgentBuilder

Parameters

- **agent** (*Agent*) – the considered agent;
- **states** (*np.ndarray*) – the set of states over which we need to compute the Q function.

classmethod default(*actor_lr=0.0001, actor_network=<class 'mushroom_rl_benchmark.builders.network.ddpg_network.DDPGActorNetwork'>, critic_lr=0.001, critic_network=<class 'mushroom_rl_benchmark.builders.network.ddpg_network.DDPGCriticNetwork'>, initial_replay_size=500, max_replay_size=50000, batch_size=64, n_features=[80, 80], tau=0.001, use_cuda=False, get_default_dict=False*)

Create a default initialization for the specific AgentBuilder and return it

class mushroom_rl_benchmark.builders.actor_critic.ppo.**PPOBuilder**(*policy_params*, *actor_optimizer*, *critic_params*, *alg_params*, *n_steps_per_fit=3000*, *preprocessors=None*)

Bases: *mushroom_rl_benchmark.builders.agent_builder.AgentBuilder*

AgentBuilder for Proximal Policy Optimization algorithm (PPO)

__init__(*policy_params*, *actor_optimizer*, *critic_params*, *alg_params*, *n_steps_per_fit=3000*, *preprocessors=None*)

Constructor.

Parameters

- **policy_params** (*dict*) – parameters for the policy;
- **actor_optimizer** (*dict*) – parameters for the actor optimizer;
- **critic_params** (*dict*) – parameters for the critic;
- **alg_params** (*dict*) – parameters for the algorithm;
- **n_steps_per_fit** (*int, 3000*) – number of steps per fit;
- **preprocessors** (*list, None*) – list of preprocessors.

build(*mdp_info*)

Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

compute_Q(*agent*, *states*)

Compute the Q Value for an AgentBuilder

Parameters

- **agent** (*Agent*) – the considered agent;
- **states** (*np.ndarray*) – the set of states over which we need to compute the Q function.

classmethod default(*eps=0.2*, *n_epochs_policy=4*, *actor_lr=0.0003*, *critic_lr=0.0003*, *critic_fit_params=None*, *critic_network=<class 'mushroom_rl_benchmark.builders.network.trpo_network.TRPONetwork'>*, *lam=0.95*, *batch_size=64*, *n_features=32*, *n_steps_per_fit=3000*, *preprocessors=None*, *use_cuda=False*, *get_default_dict=False*)

Create a default initialization for the specific AgentBuilder and return it

class mushroom_rl_benchmark.builders.actor_critic.sac.**SACBuilder**(*actor_mu_params*, *actor_sigma_params*, *actor_optimizer*, *critic_params*, *alg_params*, *n_q_samples=100*, *n_steps_per_fit=1*, *preprocessors=None*)

Bases: *mushroom_rl_benchmark.builders.agent_builder.AgentBuilder*

AgentBuilder Soft Actor-Critic algorithm (SAC)

__init__(*actor_mu_params*, *actor_sigma_params*, *actor_optimizer*, *critic_params*, *alg_params*, *n_q_samples=100*, *n_steps_per_fit=1*, *preprocessors=None*)

Constructor.

Parameters

- **actor_mu_params** (*dict*) – parameters for actor mu;
- **actor_sigma_params** (*dict*) – parameters for actor sigma;
- **actor_optimizer** (*dict*) – parameters for the actor optimizer;
- **critic_params** (*dict*) – parameters for the critic;
- **alg_params** (*dict*) – parameters for the algorithm;
- **n_q_samples** (*int, 100*) – number of samples to compute value function;
- **n_steps_per_fit** (*int, 1*) – number of steps per fit;
- **preprocessors** (*list, None*) – list of preprocessors.

build(*mdp_info*)

Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

compute_Q(*agent*, *states*)

Compute the Q Value for an AgentBuilder

Parameters

- **agent** (*Agent*) – the considered agent;
- **states** (*np.ndarray*) – the set of states over which we need to compute the Q function.

classmethod default(*actor_lr=0.0003*, *actor_network=<class 'mushroom_rl_benchmark.builders.network.sac_network.SACActorNetwork'>*, *critic_lr=0.0003*, *critic_network=<class 'mushroom_rl_benchmark.builders.network.sac_network.SACCriticNetwork'>*, *initial_replay_size=64*, *max_replay_size=50000*, *n_features=64*, *warmup_transitions=100*, *batch_size=64*, *tau=0.005*, *lr_alpha=0.003*, *preprocessors=None*, *target_entropy=None*, *use_cuda=False*, *get_default_dict=False*)

Create a default initialization for the specific AgentBuilder and return it

class mushroom_rl_benchmark.builders.actor_critic.td3.**TD3Builder**(*policy_class*, *policy_params*, *actor_params*, *actor_optimizer*, *critic_params*, *alg_params*, *n_steps_per_fit=1*)

Bases: *mushroom_rl_benchmark.builders.agent_builder.AgentBuilder*

AgentBuilder for Twin Delayed DDPG algorithm (TD3)

__init__(*policy_class*, *policy_params*, *actor_params*, *actor_optimizer*, *critic_params*, *alg_params*, *n_steps_per_fit=1*)
Constructor.

Parameters

- **policy_class** (*Policy*) – policy class;
- **policy_params** (*dict*) – parameters for the policy;
- **actor_params** (*dict*) – parameters for the actor;
- **actor_optimizer** (*dict*) – parameters for the actor optimizer;
- **critic_params** (*dict*) – parameters for the critic;
- **alg_params** (*dict*) – parameters for the algorithm;
- **n_steps_per_fit** (*int, 1*) – number of steps per fit.

build(*mdp_info*)
Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

compute_Q(*agent*, *states*)
Compute the Q Value for an AgentBuilder

Parameters

- **agent** (*Agent*) – the considered agent;
- **states** (*np.ndarray*) – the set of states over which we need to compute the Q function.

classmethod default(*actor_lr=0.0001, actor_network=<class 'mushroom_rl_benchmark.builders.network.td3_network.TD3ActorNetwork'>, critic_lr=0.001, critic_network=<class 'mushroom_rl_benchmark.builders.network.td3_network.TD3CriticNetwork'>, initial_replay_size=500, max_replay_size=50000, batch_size=64, n_features=[80, 80], tau=0.001, use_cuda=False, get_default_dict=False*)
Create a default initialization for the specific AgentBuilder and return it

class mushroom_rl_benchmark.builders.actor_critic.trpo.**TRPOBuilder**(*policy_params*, *critic_params*, *alg_params*, *n_steps_per_fit=3000*, *preprocessors=None*)
Bases: *mushroom_rl_benchmark.builders.agent_builder.AgentBuilder*

AgentBuilder for Trust Region Policy optimization algorithm (TRPO)

__init__(*policy_params*, *critic_params*, *alg_params*, *n_steps_per_fit=3000*, *preprocessors=None*)
Constructor.

Parameters

- **policy_params** (*dict*) – parameters for the policy;
- **critic_params** (*dict*) – parameters for the critic;
- **alg_params** (*dict*) – parameters for the algorithm;
- **n_steps_per_fit** (*int, 3000*) – number of steps per fit;
- **preprocessors** (*list, None*) – list of preprocessors.

build(*mdp_info*)
Build and return the AgentBuilder

Parameters **mdp_info** (*MDPInfo*) – information about the environment.

compute_Q(*agent*, *states*)
Compute the Q Value for an AgentBuilder

Parameters

- **agent** (*Agent*) – the considered agent;
- **states** (*np.ndarray*) – the set of states over which we need to compute the Q function.

classmethod default(*critic_lr=0.0003*, *critic_network=<class 'mushroom_rl_benchmark.builders.network.trpo_network.TRPONetwork'>*, *max_kl=0.01*, *ent_coeff=0.0*, *lam=0.95*, *batch_size=64*, *n_features=32*, *critic_fit_params=None*, *n_steps_per_fit=3000*, *n_epochs_line_search=10*, *n_epochs_cg=100*, *cg_damping=0.01*, *cg_residual_tol=1e-10*, *preprocessors=None*, *use_cuda=False*, *get_default_dict=False*)
Create a default initialization for the specific AgentBuilder and return it

3.5 Networks

class mushroom_rl_benchmark.builders.network.a2c_network.**A2CNetwork**(*input_shape*, *output_shape*, *n_features*, ***kwargs*)

Bases: torch.nn.modules.module.Module

__init__(*input_shape*, *output_shape*, *n_features*, ***kwargs*)
Initializes internal Module state, shared by both nn.Module and ScriptModule.

forward(*state*, ***kwargs*)
Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

class mushroom_rl_benchmark.builders.network.ddpg_network.**DDPGCriticNetwork**(*input_shape*, *output_shape*, *n_features*, ***kwargs*)

Bases: torch.nn.modules.module.Module

__init__(*input_shape*, *output_shape*, *n_features*, ***kwargs*)
Initializes internal Module state, shared by both nn.Module and ScriptModule.

forward(*state*, *action*)
Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

class mushroom_rl_benchmark.builders.network.ddpg_network.**DDPGActorNetwork**(*input_shape*, *output_shape*, ***kwargs*)

Bases: `torch.nn.modules.module.Module`

__init__(*input_shape*, *output_shape*, ***kwargs*)
Initializes internal Module state, shared by both nn.Module and ScriptModule.

forward(*state*)
Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

class mushroom_rl_benchmark.builders.network.sac_network.**SACCriticNetwork**(*input_shape*, *output_shape*, *n_features*, ***kwargs*)

Bases: `torch.nn.modules.module.Module`

__init__(*input_shape*, *output_shape*, *n_features*, ***kwargs*)
Initializes internal Module state, shared by both nn.Module and ScriptModule.

forward(*state*, *action*)
Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

class mushroom_rl_benchmark.builders.network.sac_network.**SACActorNetwork**(*input_shape*, *output_shape*, *n_features*, ***kwargs*)

Bases: `torch.nn.modules.module.Module`

__init__(*input_shape*, *output_shape*, *n_features*, ***kwargs*)
Initializes internal Module state, shared by both nn.Module and ScriptModule.

forward(*state*)
Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

class mushroom_rl_benchmark.builders.network.td3_network.**TD3CriticNetwork**(*input_shape*, *output_shape*, *n_features*, ***kwargs*)

Bases: `torch.nn.modules.module.Module`

__init__(*input_shape*, *output_shape*, *n_features*, ***kwargs*)
Initializes internal Module state, shared by both nn.Module and ScriptModule.

forward(*state*, *action*)
Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

class mushroom_rl_benchmark.builders.network.td3_network.**TD3ActorNetwork**(*input_shape*, *output_shape*, *n_features*, ***kwargs*)

Bases: `torch.nn.modules.module.Module`

__init__(*input_shape*, *output_shape*, *n_features*, ***kwargs*)
Initializes internal Module state, shared by both nn.Module and ScriptModule.

forward(*state*)
Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

class mushroom_rl_benchmark.builders.network.trpo_network.**TRPONetwork**(*input_shape*, *output_shape*, *n_features*, ***kwargs*)

Bases: `torch.nn.modules.module.Module`

__init__(*input_shape*, *output_shape*, *n_features*, ***kwargs*)
Initializes internal Module state, shared by both nn.Module and ScriptModule.

forward(*state*, ***kwargs*)
Defines the computation performed at every call.

Should be overridden by all subclasses.

Note: Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

3.6 Experiment

mushroom_rl_benchmark.experiment.run.**exec_run**(*agent_builder*, *env_builder*, *n_epochs*, *n_steps*, *n_steps_test=None*, *n_episodes_test=None*, *seed=None*, *save_agent=False*, *quiet=True*, ***kwargs*)

Function that handles the execution of an experiment run.

Parameters

- **agent_builder** (AgentBuilder) – agent builder to spawn an agent;
- **env_builder** (EnvironmentBuilder) – environment builder to spawn an environment;
- **n_epochs** (*int*) – number of epochs;
- **n_steps** (*int*) – number of steps per epoch;
- **n_steps_test** (*int, None*) – number of steps for testing;
- **n_episodes_test** (*int, None*) – number of episodes for testing;
- **seed** (*int, None*) – the seed;
- **save_agent** (*bool, False*) – select if the agent should be logged or not;
- **quiet** (*bool, True*) – select if run should print execution information.

mushroom_rl_benchmark.experiment.run.**compute_metrics**(*core*, *eval_params*, *agent_builder*, *env_builder*, *cmp_E*)

Function to compute the metrics.

Parameters

- **eval_params** (*dict*) – parameters for running the evaluation;
- **agent_builder** (AgentBuilder) – the agent builder;
- **env_builder** (EnvironmentBuilder) – environment builder to spawn an environment;
- **cmp_E** (*bool*) – select if policy entropy should be computed.

mushroom_rl_benchmark.experiment.run.**print_metrics**(*logger*, *epoch*, *J*, *R*, *V*, *E*)

Function that pretty prints the metrics on the standard output.

Parameters

- **logger** (*Logger*) – MushroomRL logger object;
- **epoch** (*int*) – the current epoch;
- **J** (*float*) – the current value of J;

- **R** (*float*) – the current value of R;
- **V** (*float*) – the current value of V;
- **E** (*float*) – the current value of E (Set None if not defined).

3.6.1 Slurm utilities

mushroom_rl_benchmark.experiment.slurm.aggregate_results.**aggregate_results**(*res_dir*, *res_id*, *console_log_dir=None*)

Function to aggregate the benchmark results from running in SLURM mode.

Parameters

- **res_dir** (*str*) – path to the result directory;
- **res_id** (*str*) – log id of the result directory;
- **console_log_dir** (*str, None*) – path to be used to log console.

mushroom_rl_benchmark.experiment.slurm.arguments.**make_arguments**(***params*)

Create a script argument string from dictionary

mushroom_rl_benchmark.experiment.slurm.arguments.**read_arguments_run**(*arg_string=None*)

Parse the arguments for the run script.

Parameters **arg_string** (*str, None*) – pass the argument string.

mushroom_rl_benchmark.experiment.slurm.arguments.**read_arguments_aggregate**(*arg_string=None*)

Parse the arguments for the aggregate script.

Parameters **arg_string** (*str, None*) – pass the argument string.

mushroom_rl_benchmark.experiment.slurm.slurm_script.**create_slurm_script**(*slurm_path*, *slurm_script_name='slurm*, ***slurm_params*)

Function to create a slurm script in a specific directory

Parameters

- **slurm_path** (*str*) – path to locate the slurm script;
- **slurm_script_name** (*str, slurm.sh*) – name of the slurm script;
- ****slurm_params** – parameters for generating the slurm file content.

Returns The path to the slurm script.

mushroom_rl_benchmark.experiment.slurm.slurm_script.**generate_slurm**(*exp_name*, *exp_dir_slurm*, *python_file*, *gres=None*, *project_name=None*, *n_exp=1*, *max_concurrent_runs=None*, *memory=2000*, *hours=24*, *minutes=0*, *seconds=0*)

Function to generate the slurm file content.

Parameters

- **exp_name** (*str*) – name of the experiment;
- **exp_dir_slurm** (*str*) – directory where the slurm log files are located;
- **python_file** (*str*) – path to the python file that should be executed;
- **gres** (*str, None*) – request cluster resources. E.g. to add a GPU in the IAS cluster specify gres='gpu:rtx2080:1';
- **project_name** (*str, None*) – name of the slurm project;
- **n_exp** (*int, 1*) – number of experiments in the slurm array;
- **max_concurrent_runs** (*int, None*) – maximum number of runs that should be executed in parallel on the SLURM cluster;
- **memory** (*int, 2000*) – memory limit in mega bytes (MB) for the slurm jobs;
- **hours** (*int, 24*) – maximum number of execution hours for the slurm jobs;
- **minutes** (*int, 0*) – maximum number of execution minutes for the slurm jobs;
- **seconds** (*int, 0*) – maximum number of execution seconds for the slurm jobs.

Returns The slurm script as string.

mushroom_rl_benchmark.experiment.slurm.slurm_script.**to_duration**(*hours*, *minutes*, *seconds*)

3.7 Utils

mushroom_rl_benchmark.utils.utils.**get_init_states**(*dataset*)

Get the initial states of a MushroomRL dataset

Parameters **dataset** (*Dataset*) – a MushroomRL dataset.

mushroom_rl_benchmark.utils.utils.**extract_arguments**(*args*, *method*)

Extract the arguments from a dictionary that fit to a methods parameters.

Parameters

- **args** (*dict*) – dictionary of arguments;
- **method** (*function*) – method for which the arguments should be extracted.

mushroom_rl_benchmark.utils.primitive.**object_to_primitive**(*obj*)
Converts an object into a string using the class name

Parameters **obj** – the object to convert.

Returns A string representing the object.

mushroom_rl_benchmark.utils.primitive.**dictionary_to_primitive**(*data*)
Function that converts a dictionary by transforming any objects inside into strings

Parameters **data** (*dict*) – the dictionary to convert.

Returns The converted dictionary.

mushroom_rl_benchmark.utils.plot.**get_mean_and_confidence**(*data*)
Compute the mean and 95% confidence interval

Parameters **data** (*np.ndarray*) – Array of experiment data of shape (n_runs, n_epochs).

Returns The mean of the dataset at each epoch along with the confidence interval.

mushroom_rl_benchmark.utils.plot.**plot_mean_conf**(*data*, *ax*, *color='blue'*, *facecolor=None*, *alpha=0.4*, *label=None*)
Method to plot mean and confidence interval for data on pyplot axes.

Python Module Index

m

Index

Symbols

A

B

C

D

E

F

G

T