
MushroomRL Benchmark Documentation

Release 1.1.0

Carlo D'Eramo, Davide Tateo

Jun 10, 2021

Benchmarks Results:

1	Reinforcement Learning python library	1
2	Download and installation	3
3	Benchmarks	5
3.1	Policy Search Benchmarks	5
3.2	Actor-Critic Benchmarks	8
3.3	Value-Based Benchmarks	40
3.4	Core functionality	46
3.5	Builders	55
3.6	Networks	71
3.7	Experiment	73
3.8	Utils	75
	Python Module Index	77
	Index	79

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

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 Policy Search Benchmarks

We provide the benchmarks for the Policy Gradient algorithms:

- **REINFORCE**
- **GPOMDP**
- **eNAC**

We provide the benchmarks for the following Black-Box optimization algorithms:

- **RWR**
- **REPS**
- **PGPE**
- **ConstrainedREPS**

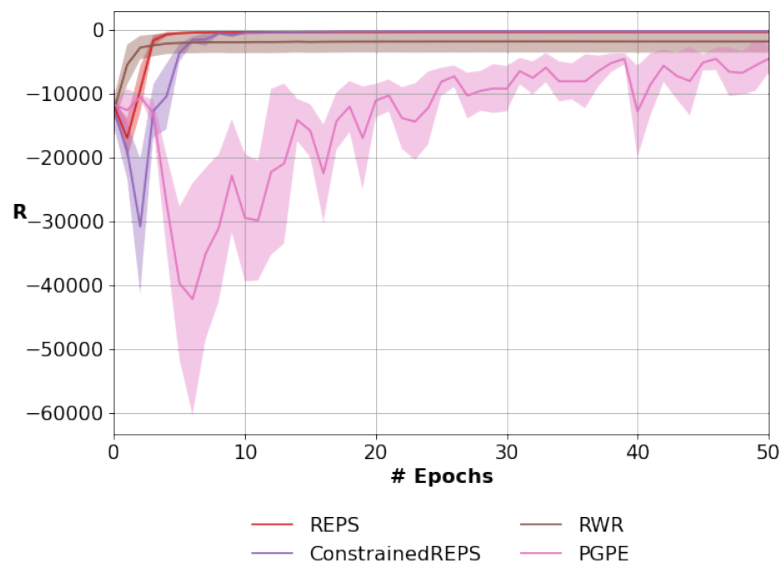
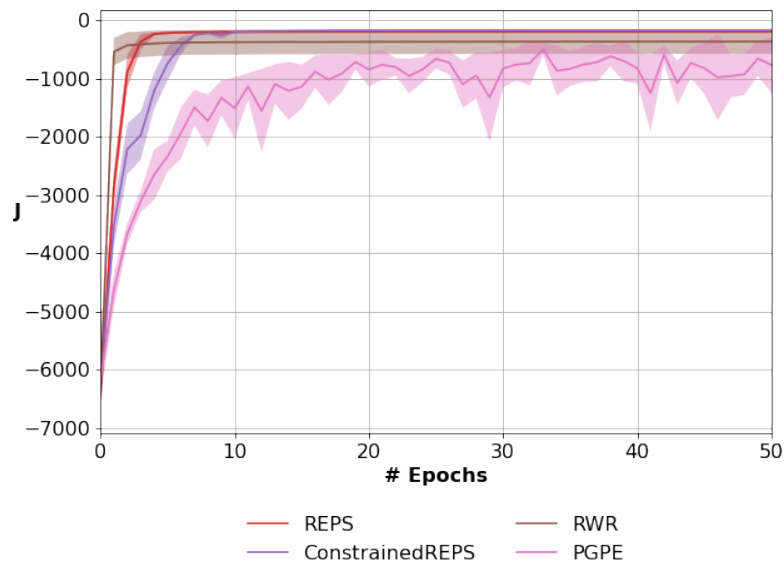
We consider the following environments in the benchmark

3.1.1 Classic Control Environments Benchmarks

Segway

Run Parameters	
n_runs	25
n_epochs	50
n_episodes	100
n_episodes_test	10

```
ConstrainedREPS:
  eps: 0.5
  kappa: 0.1
  n_episodes_per_fit: 25
PGPE:
  alpha: 0.3
  n_episodes_per_fit: 25
REPS:
  eps: 0.5
  n_episodes_per_fit: 25
RWR:
  beta: 0.01
  n_episodes_per_fit: 25
```



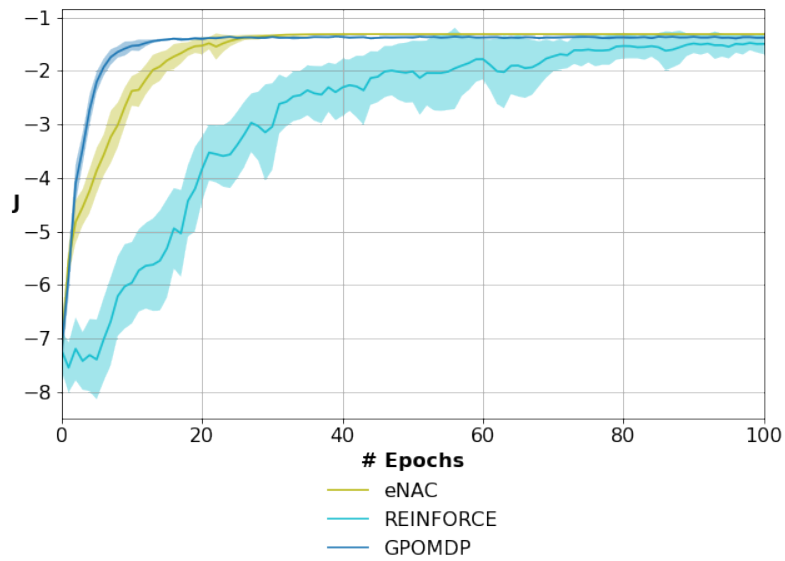
LQR

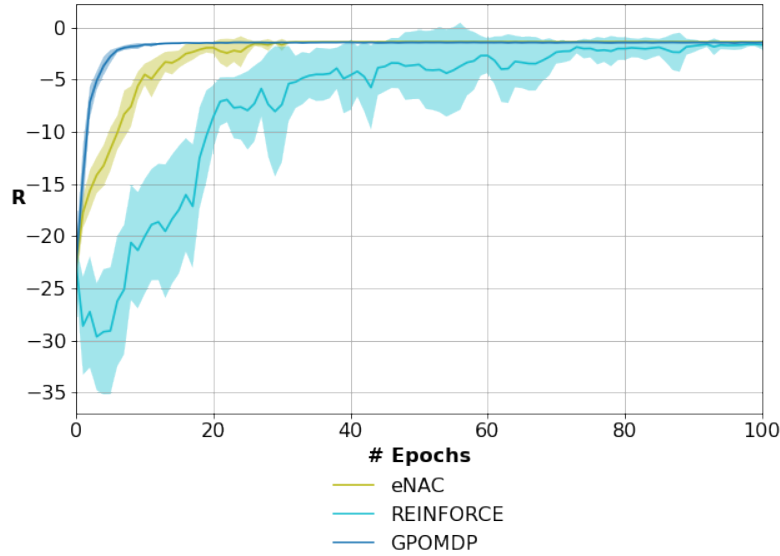
Run Parameters	
n_runs	25
n_epochs	100
n_episodes	100
n_episodes_test	10

```

GPOMDP:
  alpha: 0.01
  n_episodes_per_fit: 25
REINFORCE:
  alpha: 0.01
  n_episodes_per_fit: 25
eNAC:
  alpha: 0.01
  n_episodes_per_fit: 25

```





3.2 Actor-Critic Benchmarks

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

- **StochasticAC**
- **COPDAC_Q**

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

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

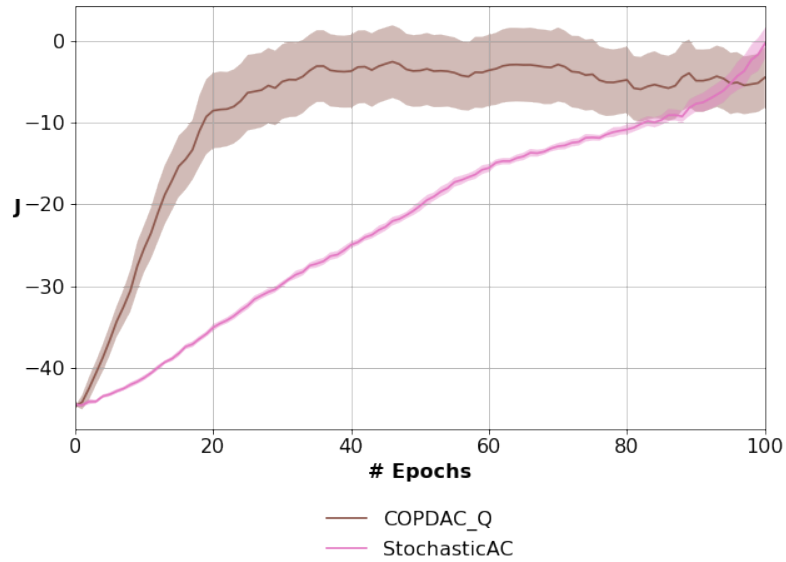
We consider the following environments in the benchmark

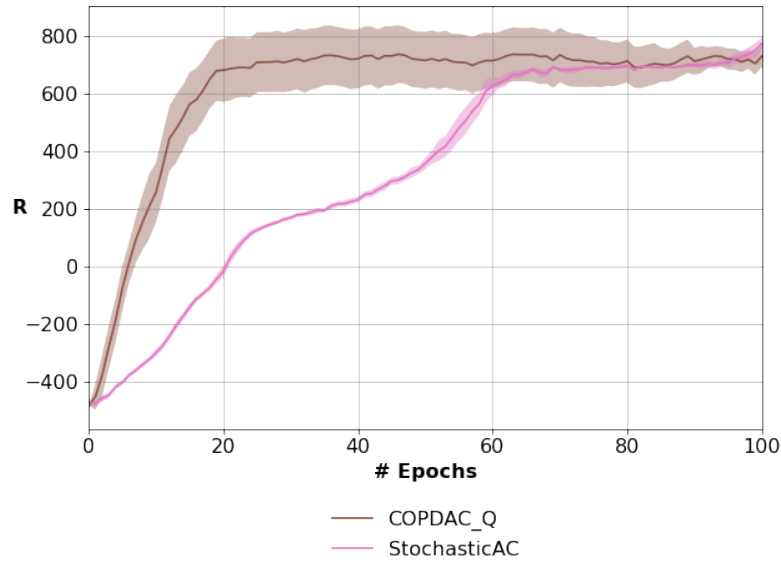
3.2.1 Classic Control Environments Benchmarks

Run Parameters	
n_runs	25
n_epochs	100
n_episodes	10
n_episodes_test	5

InvertedPendulum

```
COPDAC_Q:
  alpha_omega: 0.5
  alpha_theta: 0.005
  alpha_v: 0.5
  n_tiles: 11
  n_tilings: 10
  std_eval: 0.001
  std_exp: 0.1
StochasticAC:
  alpha_theta: 0.001
  alpha_v: 0.1
  lambda_par: 0.9
  n_tiles: 11
  n_tilings: 10
  std_0: 1.0
```





3.2.2 Gym Control 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:

```

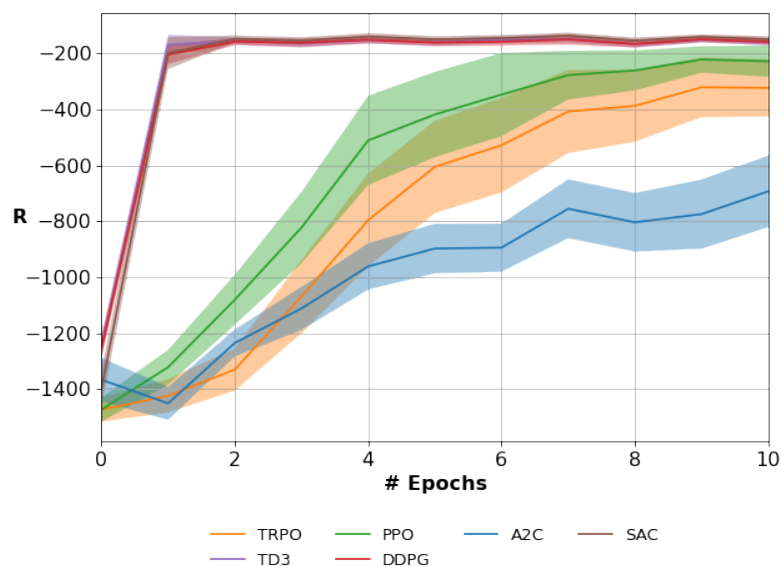
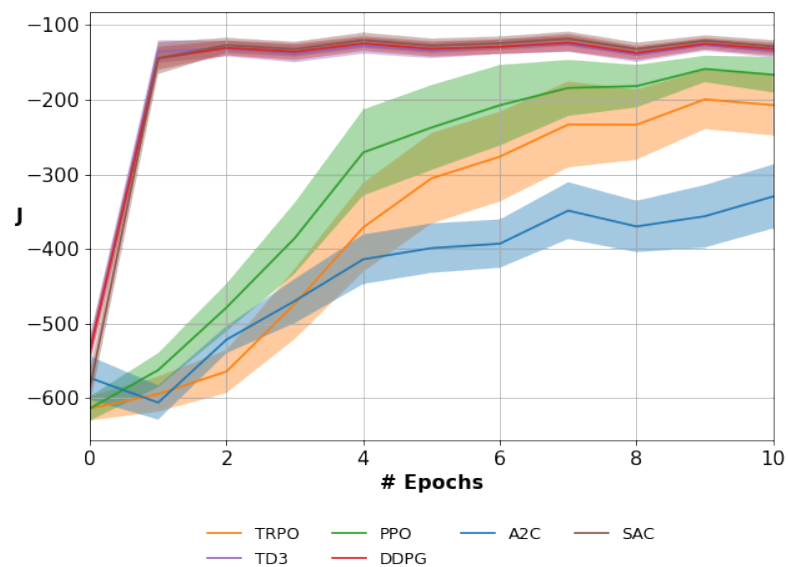
actor_lr: 0.0001
actor_network: DDPGActorNetwork
batch_size: 64
critic_lr: 0.001
critic_network: DDPGCriticNetwork
initial_replay_size: 128
max_replay_size: 1000000
n_features:
- 64
- 64

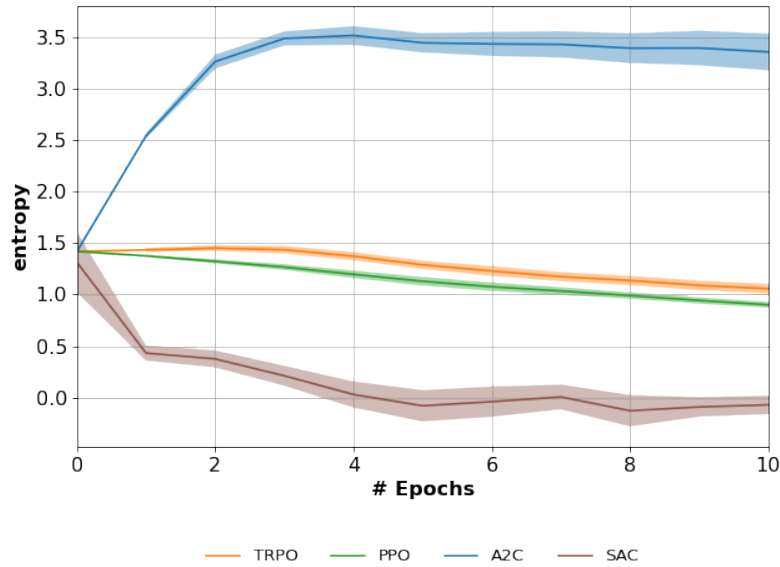
```

(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: SACTActorNetwork
  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: 0.001
  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:

```

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:

```

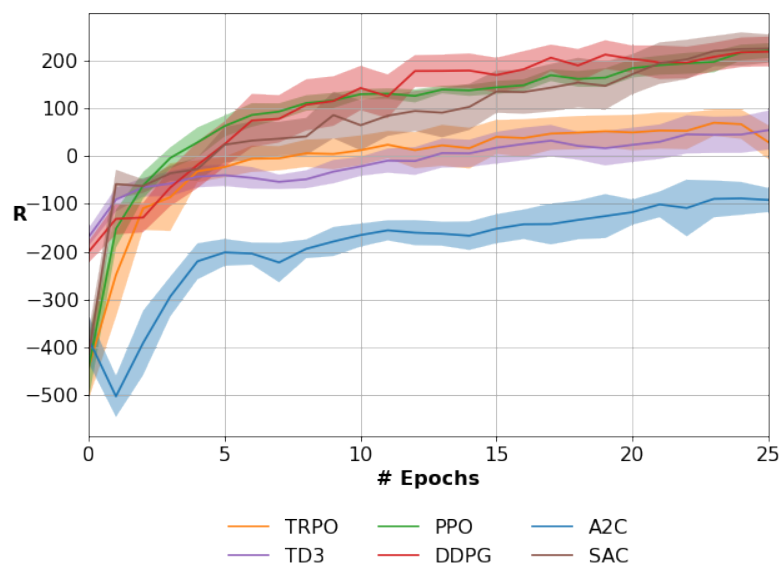
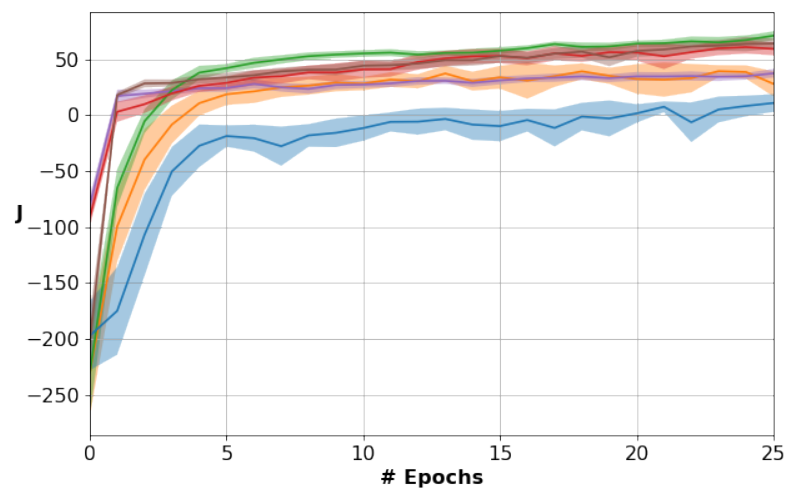
actor_lr: 0.0003
batch_size: 64
critic_fit_params: null
critic_lr: 0.003
critic_network: TRPONetwork
eps: 0.2
lam: 0.95
n_epochs_policy: 4
n_features: 32

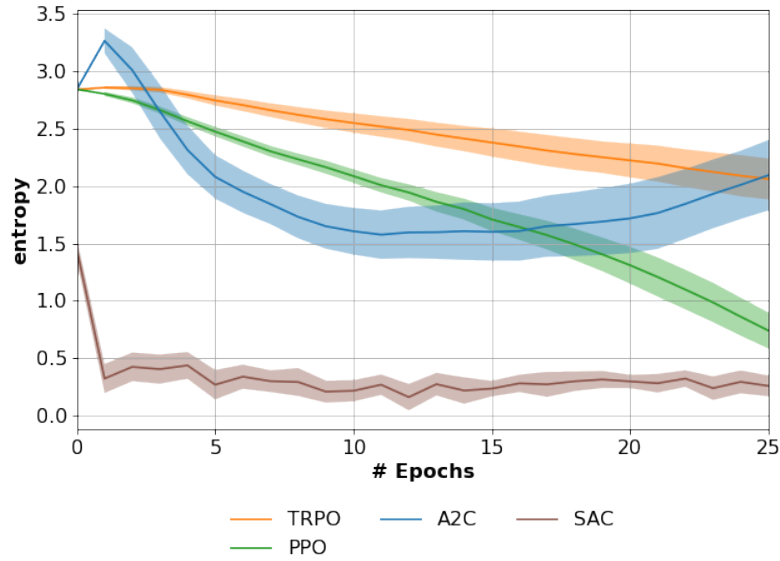
```

(continues on next page)

(continued from previous page)

```
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
  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
```





3.2.3 Mujoco Environments Benchmarks

Run Parameters	
n_runs	25
n_epochs	50
n_steps	30000
n_episodes_test	10

Hopper-v3

```

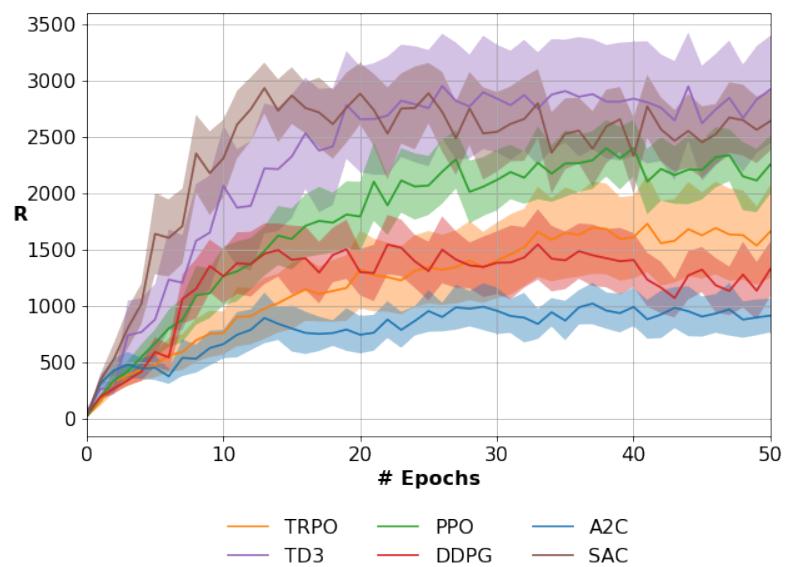
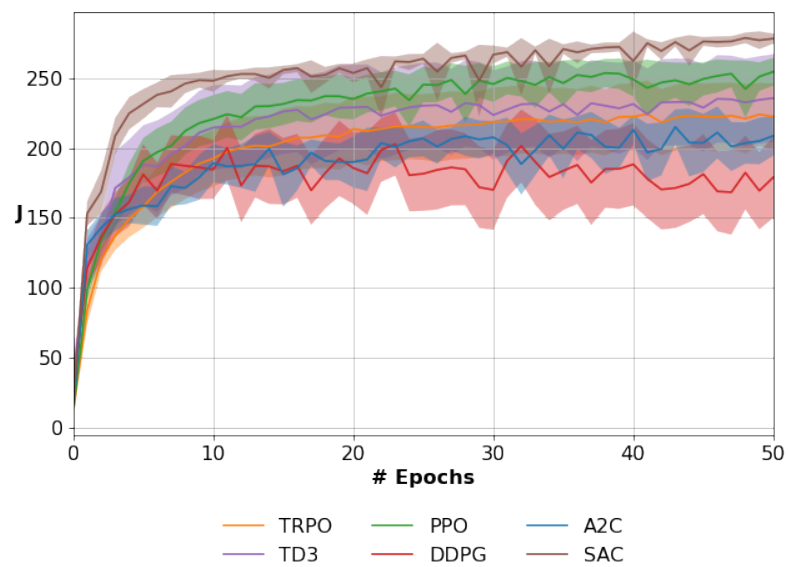
A2C:
  actor_lr: 0.0007
  batch_size: 64
  critic_lr: 0.0007
  critic_network: A2CNetwork
  ent_coeff: 0.01
  eps_actor: 0.003
  eps_critic: 1.0e-05
  max_grad_norm: 0.5
  n_features: 64
  preprocessors: StandardizationPreprocessor
DDPG:
  actor_lr: 0.0001
  actor_network: DDPGActorNetwork
  batch_size: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

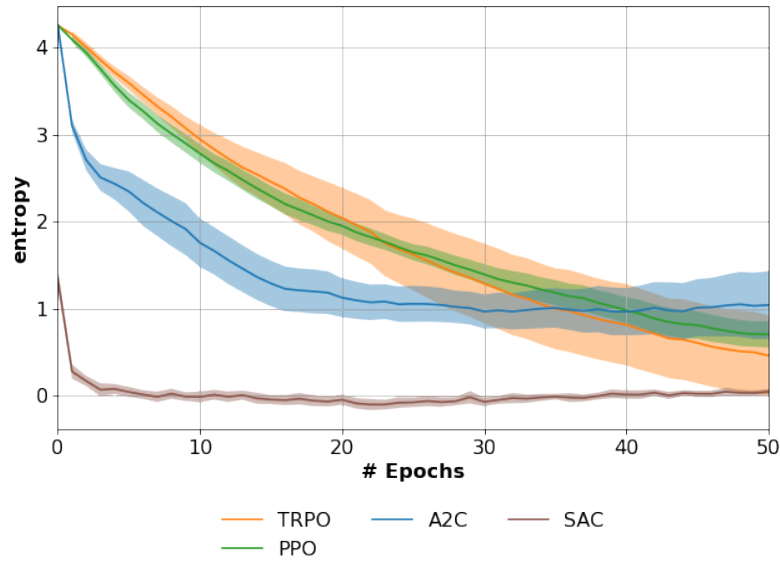
```

(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: SACTActorNetwork
  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
```





Walker2d-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
tau: 0.001

```

PPO:

```

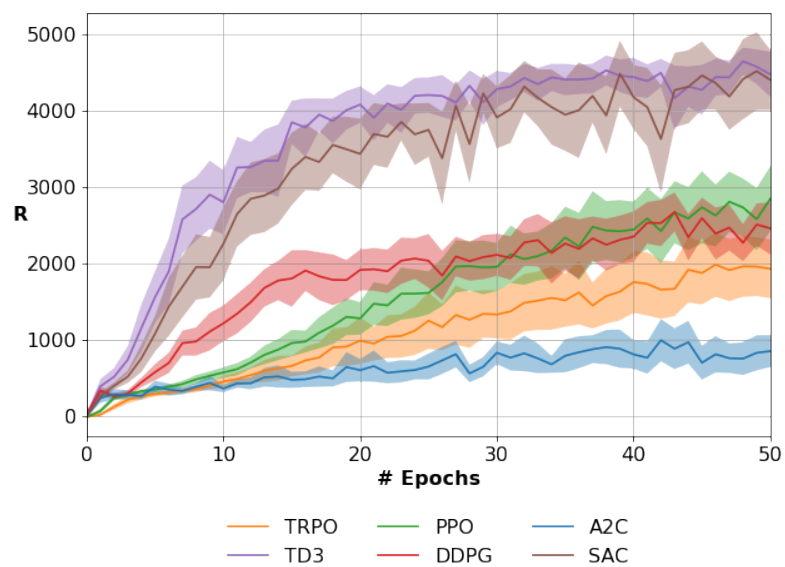
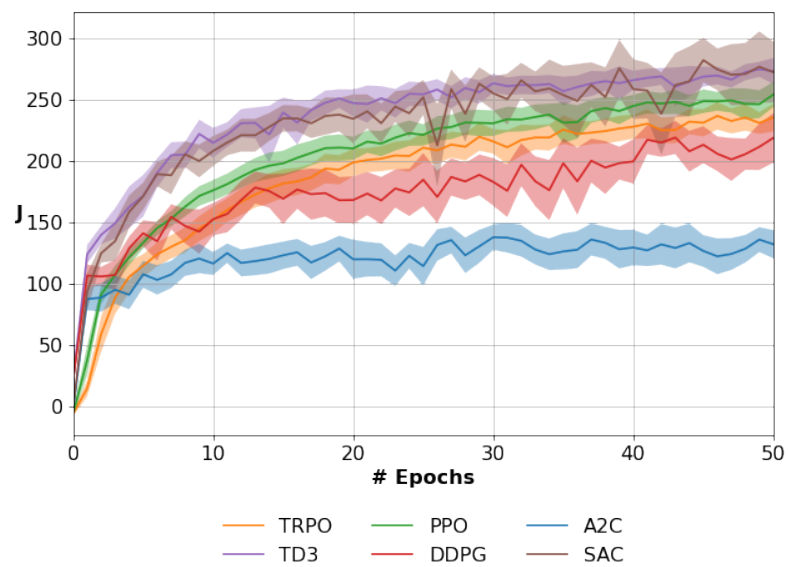
actor_lr: 0.0003
batch_size: 32
critic_fit_params: null
critic_lr: 0.0003
critic_network: TRPONetwork
eps: 0.2
lam: 0.95
n_epochs_policy: 10
n_features: 32

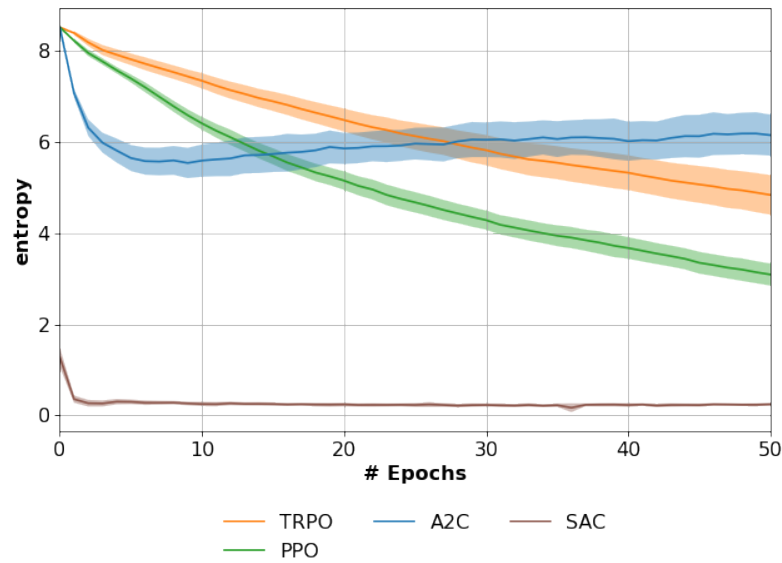
```

(continues on next page)

(continued from previous page)

```
n_steps_per_fit: 2000
preprocessors: StandardizationPreprocessor
SAC:
  actor_lr: 0.0001
  actor_network: SACTActorNetwork
  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
```



HalfCheetah-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
tau: 0.001

```

PPO:

```

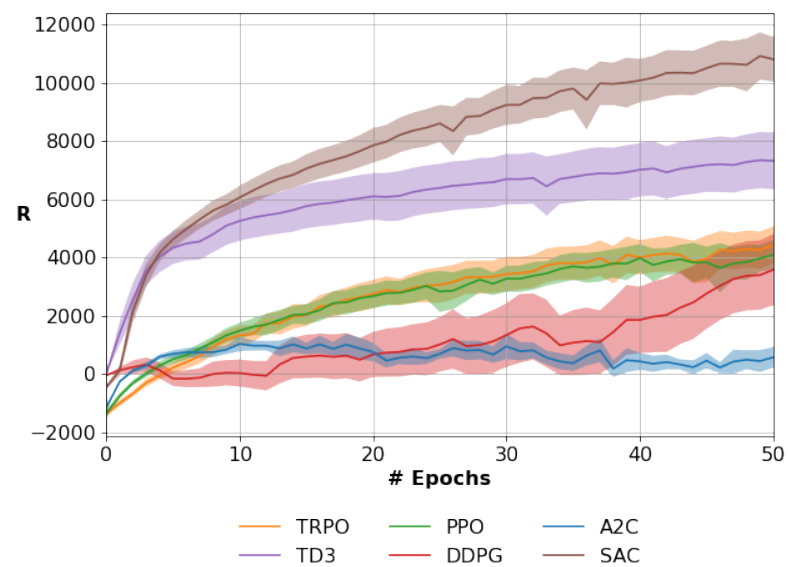
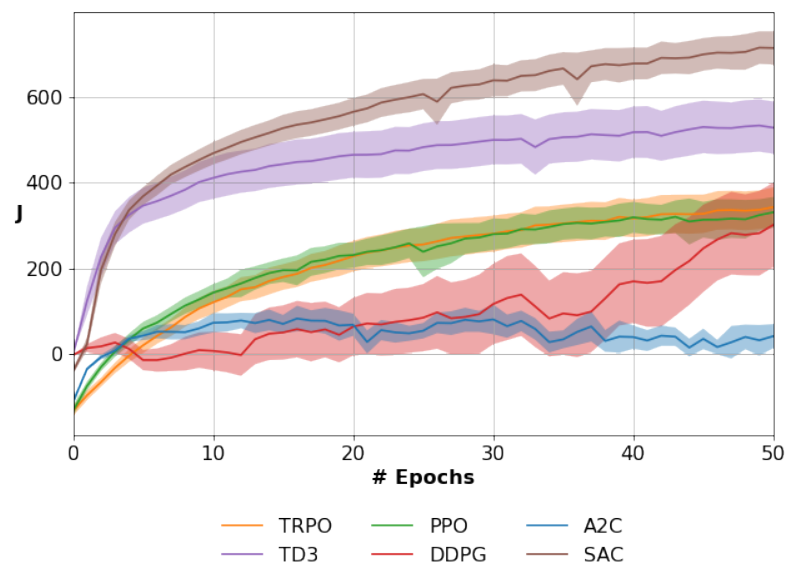
actor_lr: 0.0003
batch_size: 32
critic_fit_params: null
critic_lr: 0.0003
critic_network: TRPONetwork
eps: 0.2
lam: 0.95
n_epochs_policy: 10
n_features: 32

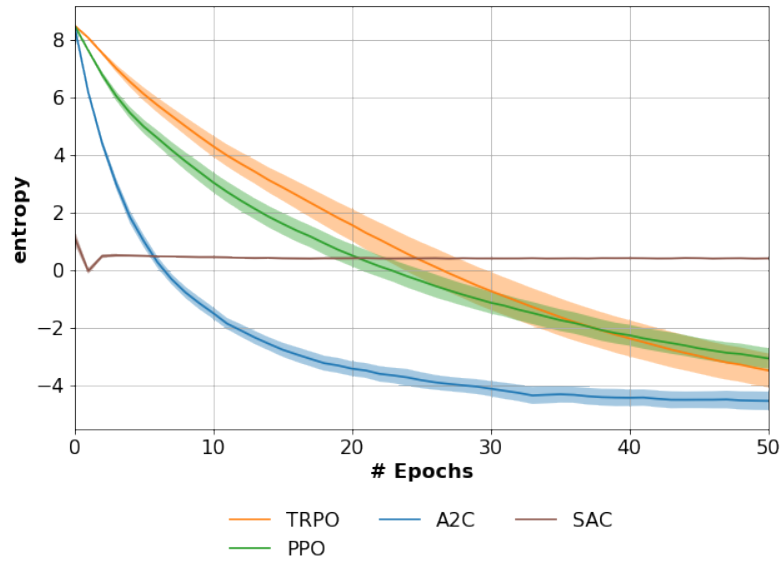
```

(continues on next page)

(continued from previous page)

```
n_steps_per_fit: 2000
preprocessors: StandardizationPreprocessor
SAC:
  actor_lr: 0.0001
  actor_network: SACTActorNetwork
  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
```





Ant-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
tau: 0.001

```

PPO:

```

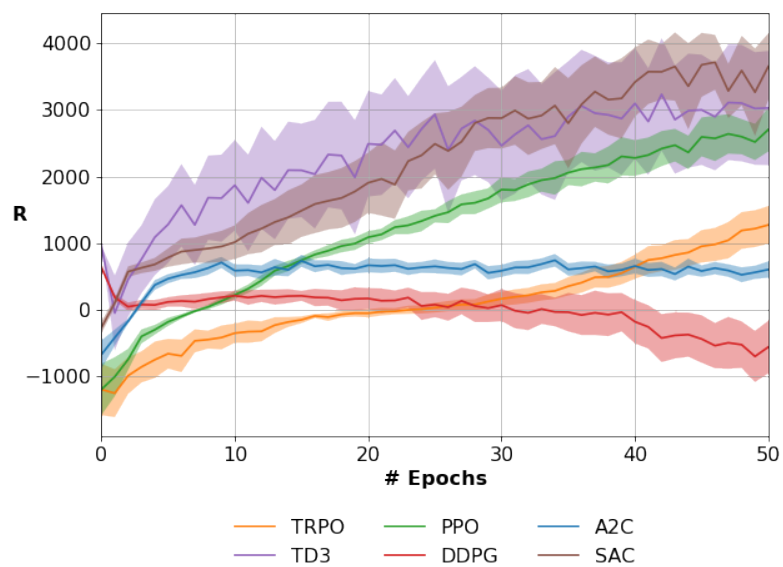
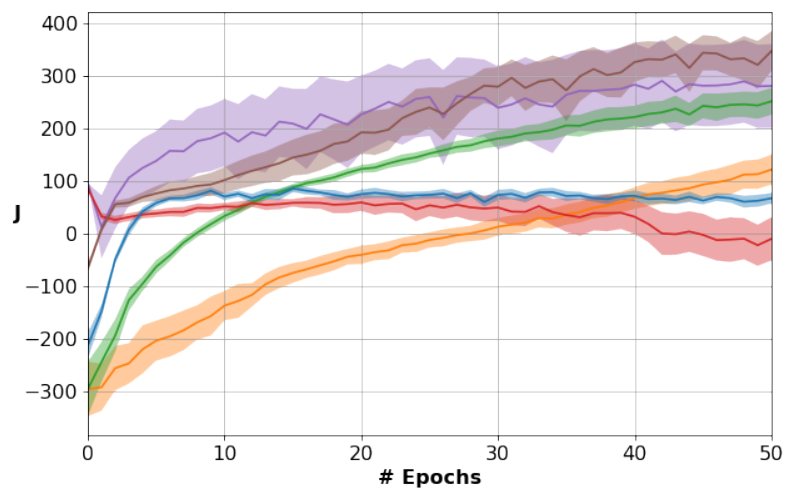
actor_lr: 0.0003
batch_size: 32
critic_fit_params: null
critic_lr: 0.0003
critic_network: TRPONetwork
eps: 0.2
lam: 0.95
n_epochs_policy: 10
n_features: 32

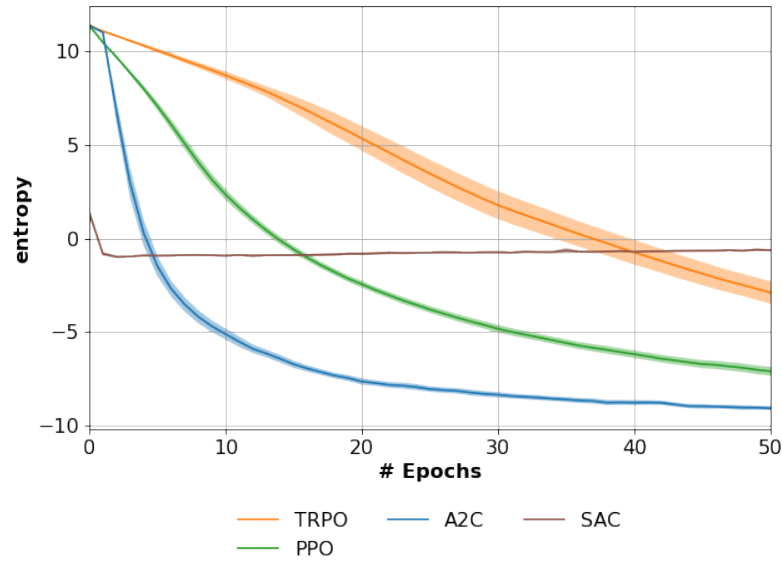
```

(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
```





3.2.4 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

```

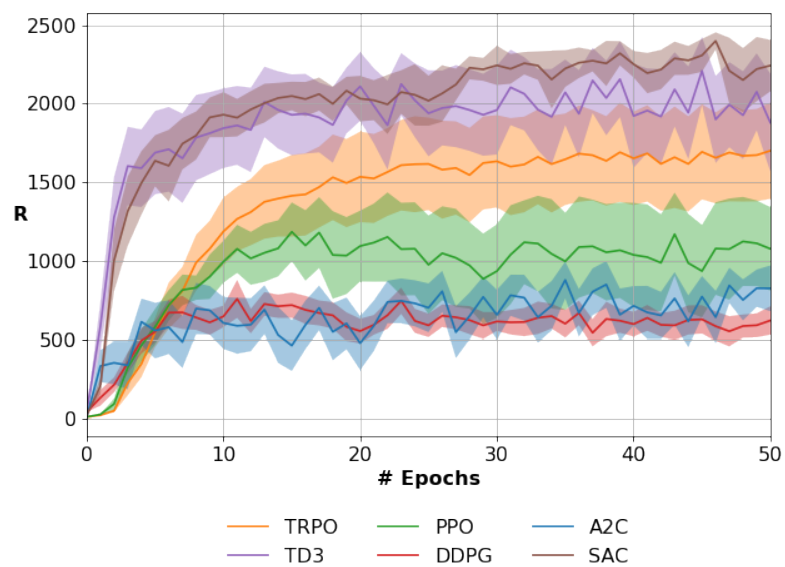
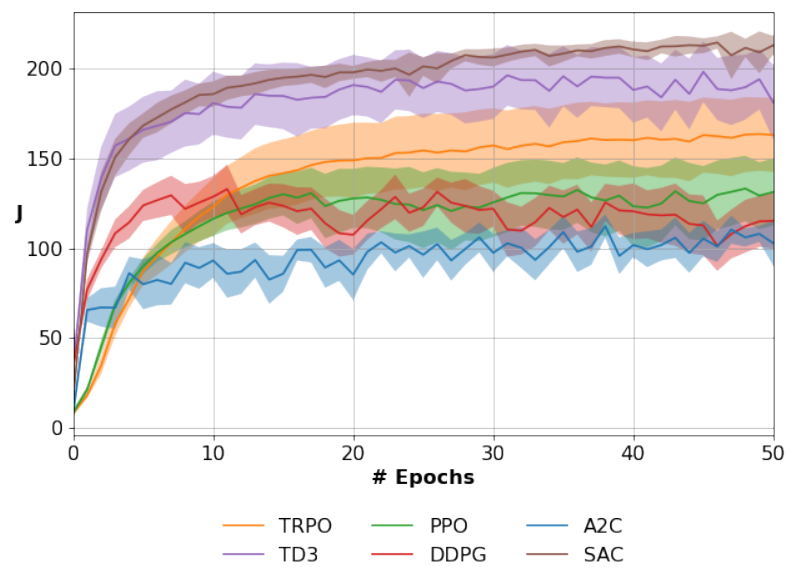
(continues on next page)

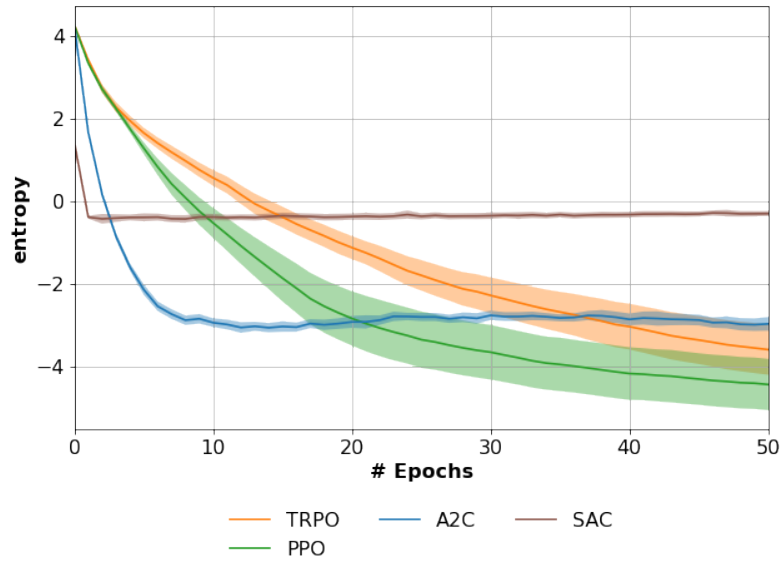
(continued from previous page)

```

    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: SACTActorNetwork
    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

```





Walker2DBulletEnv-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
tau: 0.001
```

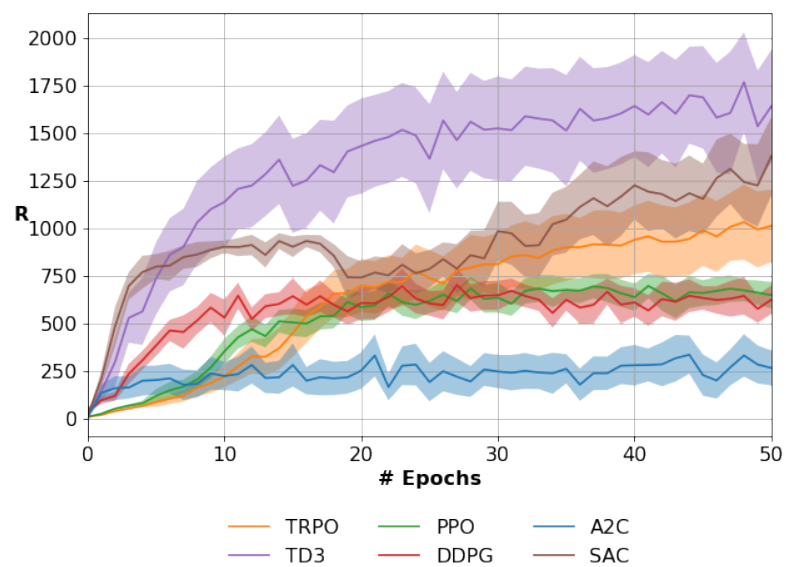
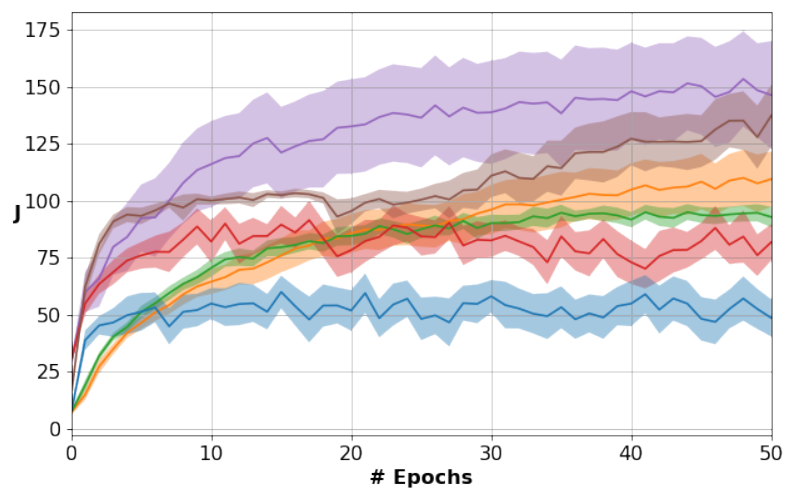
PPO:

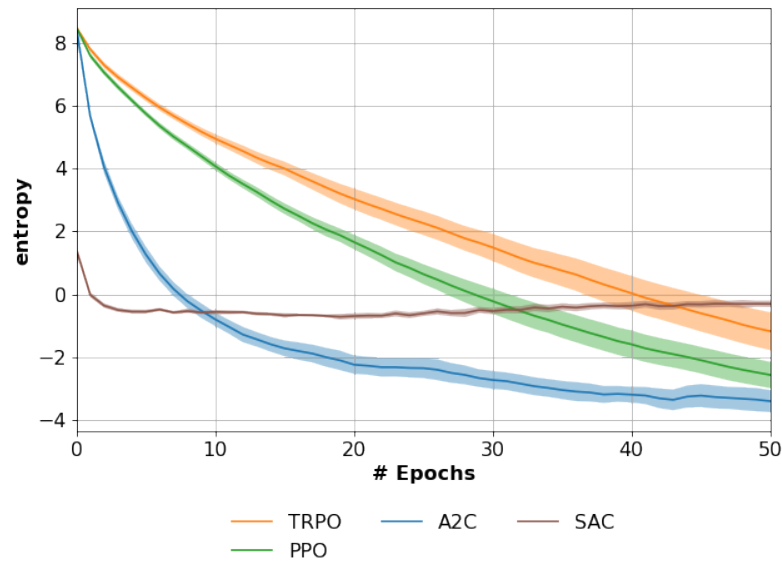
```
actor_lr: 0.0003
batch_size: 64
critic_fit_params: null
critic_lr: 0.0003
critic_network: TRPONetwork
eps: 0.2
lam: 0.95
n_epochs_policy: 4
n_features: 32
```

(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
```





HalfCheetahBulletEnv-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
tau: 0.001

```

PPO:

```

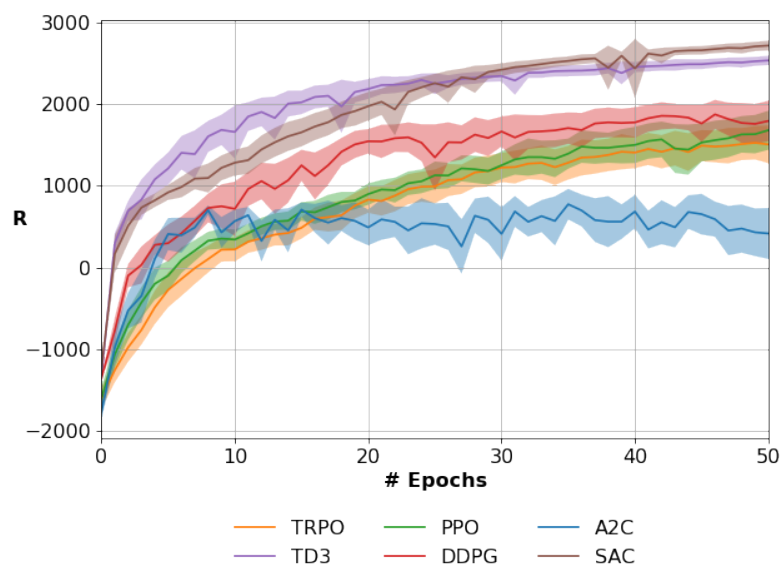
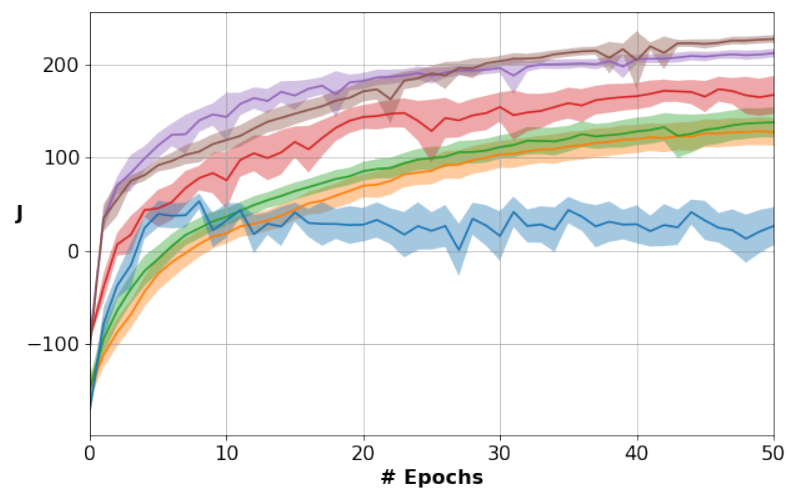
actor_lr: 0.0003
batch_size: 64
critic_fit_params: null
critic_lr: 0.0003
critic_network: TRPONetwork
eps: 0.2
lam: 0.95
n_epochs_policy: 4
n_features: 32

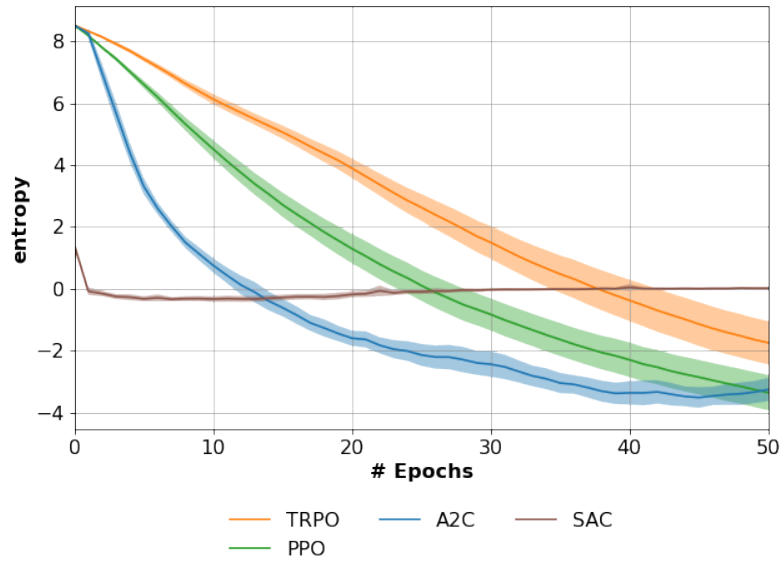
```

(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
```





AntBulletEnv-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
tau: 0.001
```

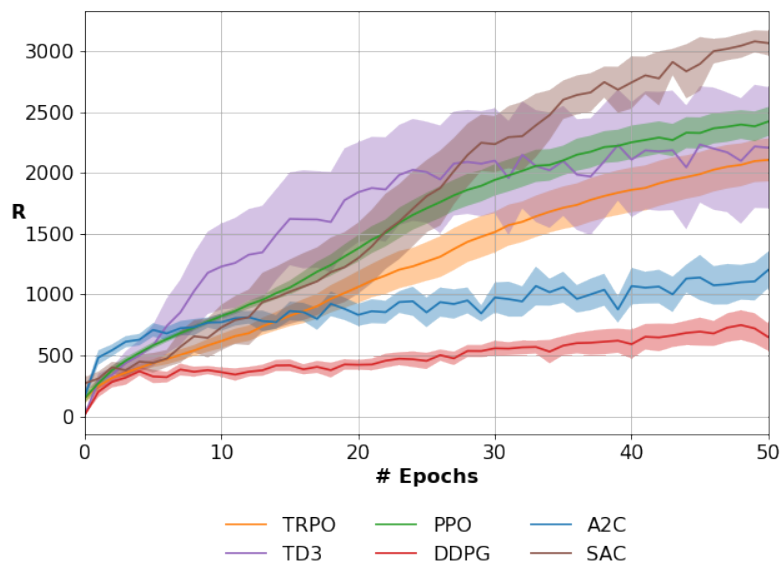
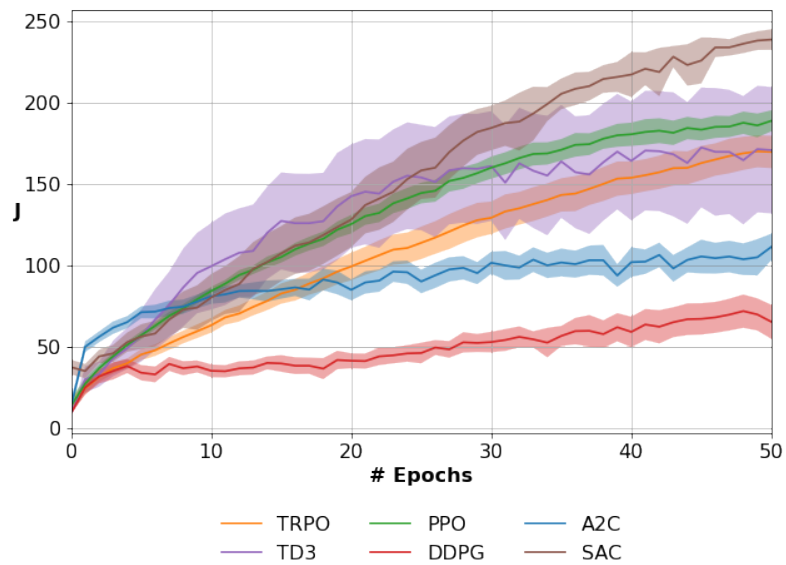
PPO:

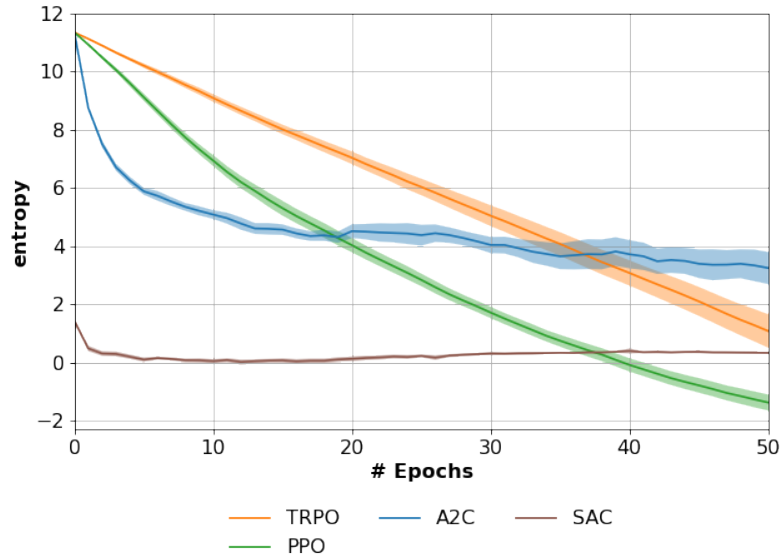
```
actor_lr: 0.0003
batch_size: 64
critic_fit_params: null
critic_lr: 0.0003
critic_network: TRPONetwork
eps: 0.2
lam: 0.95
n_epochs_policy: 4
n_features: 32
```

(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
```





3.3 Value-Based Benchmarks

We provide the benchmarks for the following Finite Temporal-Difference algorithms:

- **SARSA**
- **QLearning**
- **SpeedyQLearning**
- **WeightedQLearning**
- **DoubleQLearning**
- **SARSALambda**
- **QLambda**

We provide the benchmarks for the following Continuous state Temporal-Difference algorithms:

- **SARSALambdaContinuous**
- **TrueOnlineSARSALambda**

We provide the benchmarks for the following DQN algorithms:

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

We consider the following environments in the benchmark

3.3.1 Finite State Environment Benchmark

Run Parameters	
n_runs	25
n_epochs	100
n_steps	100
n_steps_test	1000

GridWorld

```

DoubleQLearning:
  decay_eps: 0.5
  decay_lr: 0.8
  epsilon: ExponentialParameter
  epsilon_test: 0.0
  learning_rate: ExponentialParameter
QLambda:
  decay_eps: 0.5
  decay_lr: 0.8
  epsilon: ExponentialParameter
  epsilon_test: 0.0
  lambda_coeff: 0.9
  learning_rate: ExponentialParameter
  trace: replacing
QLearning:
  decay_eps: 0.5
  decay_lr: 0.8
  epsilon: ExponentialParameter
  epsilon_test: 0.0
  learning_rate: ExponentialParameter
SARSA:
  decay_eps: 0.5
  decay_lr: 0.8
  epsilon: ExponentialParameter
  epsilon_test: 0.0
  learning_rate: ExponentialParameter
SARSLambda:
  decay_eps: 0.5
  decay_lr: 0.8
  epsilon: ExponentialParameter
  epsilon_test: 0.0
  lambda_coeff: 0.9
  learning_rate: ExponentialParameter
  trace: replacing
SpeedyQLearning:
  decay_eps: 0.5
  decay_lr: 0.8
  epsilon: ExponentialParameter
  epsilon_test: 0.0
  learning_rate: ExponentialParameter
WeightedQLearning:
  decay_eps: 0.5

```

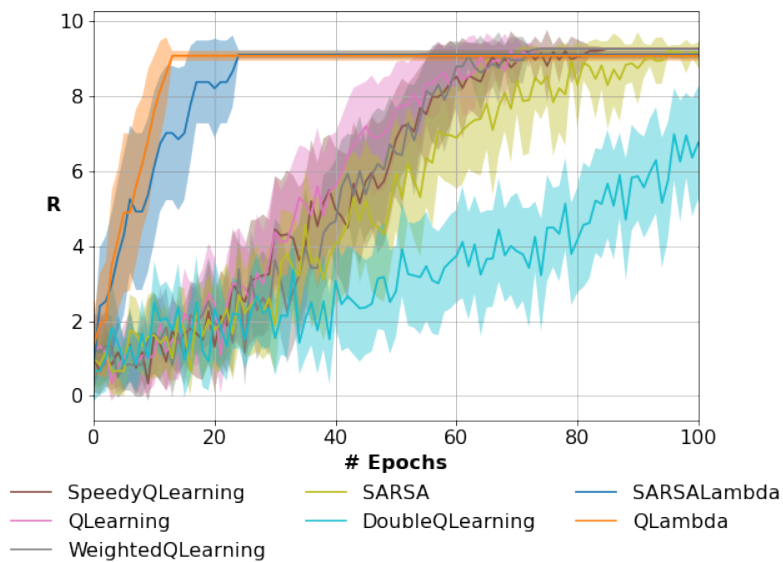
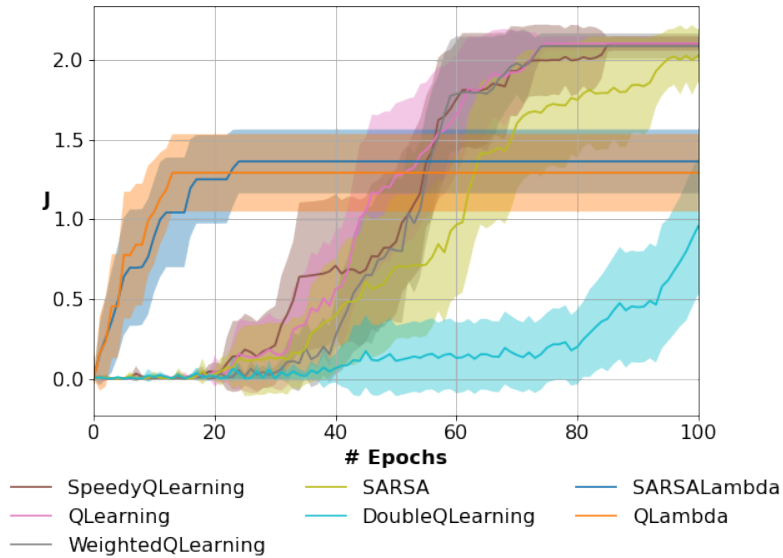
(continues on next page)

(continued from previous page)

```

decay_lr: 0.8
epsilon: ExponentialParameter
epsilon_test: 0.0
learning_rate: ExponentialParameter
precision: 1000
sampling: true

```



3.3.2 Gym Environments Benchmarks

Run Parameters	
n_runs	25
n_epochs	100
n_steps	1000
n_episodes_test	10

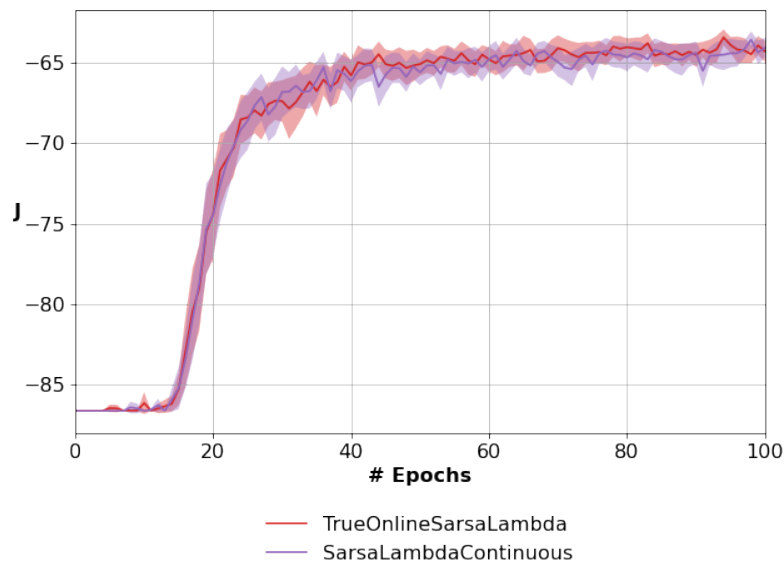
MountainCar-v0

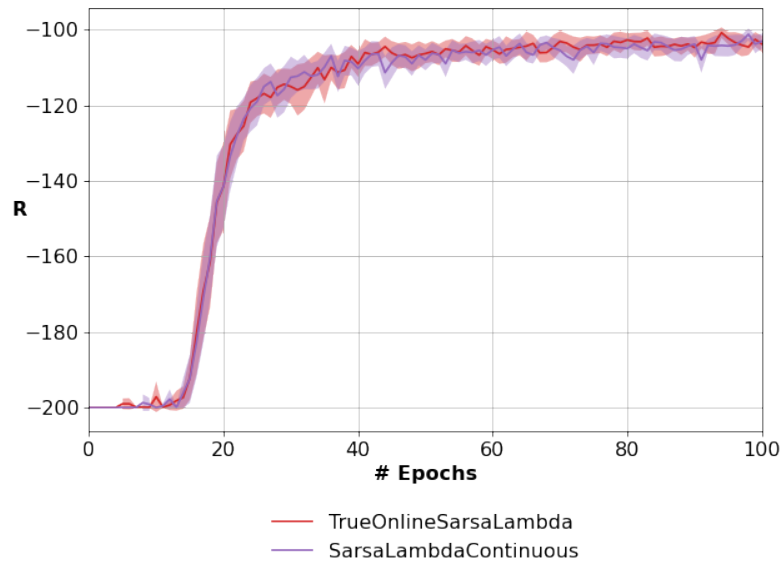
SarsaLambdaContinuous:

```
alpha: 0.1
epsilon: 0
epsilon_test: 0.0
lambda_coeff: 0.9
n_tiles: 10
n_tilings: 10
```

TrueOnlineSarsaLambda:

```
alpha: 0.1
epsilon: 0
epsilon_test: 0.0
lambda_coeff: 0.9
n_tiles: 10
n_tilings: 10
```





3.3.3 Atari Environment Benchmark

Run Parameters	
n_runs	5
n_epochs	200
n_steps	250000
n_episodes_test	125000

BreakoutDeterministic-v4

```

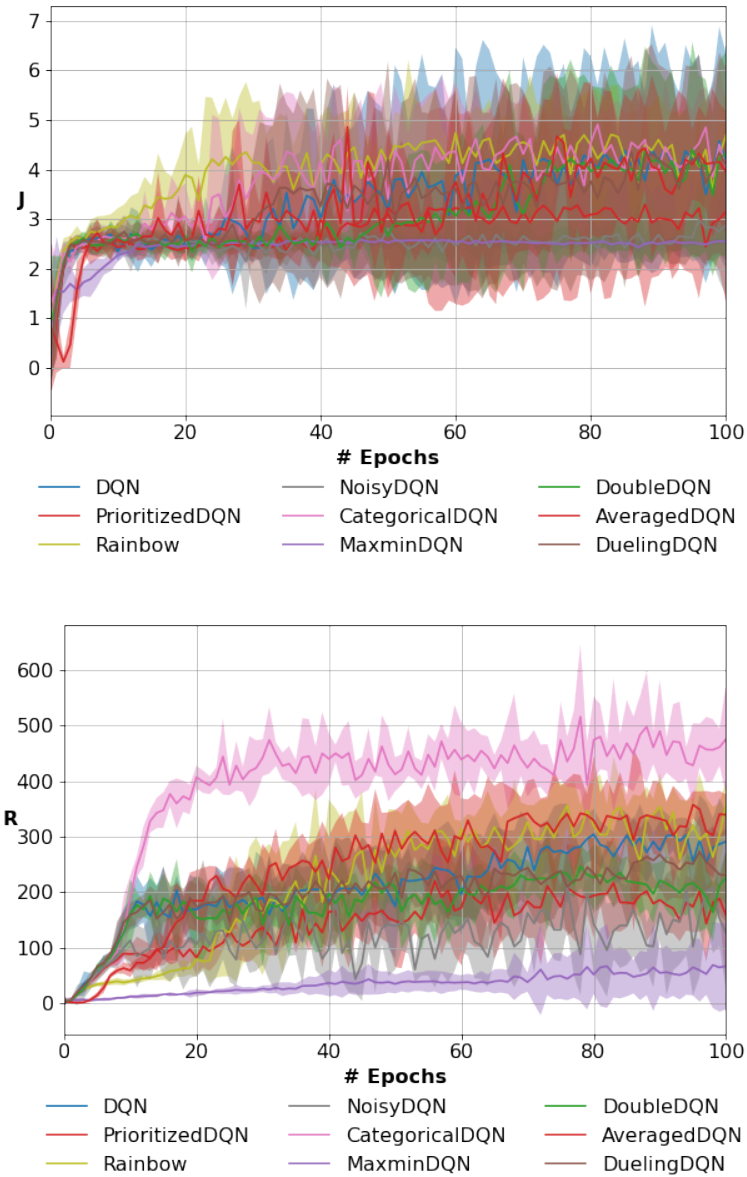
AveragedDQN:
  batch_size: 32
  initial_replay_size: 50000
  lr: 0.0001
  max_replay_size: 1000000
  n_approximators: 10
  n_steps_per_fit: 4
  network: DQNNetwork
  target_update_frequency: 2500
CategoricalDQN:
  batch_size: 32
  initial_replay_size: 50000
  lr: 0.0001
  max_replay_size: 1000000
  n_atoms: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

```

(continues on next page)

(continued from previous page)

```
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
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
```



3.4 Core functionality

3.4.1 Suite

class 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_experiments_sweeps

environment_name, *environment_builder_params*, *agent_names_list*, *agent_builders_params*, *sweeps_list*, ****run_params** Add a set of experiments sweeps 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;
- **sweeps_list** (*list*) – list of dictionaries containing the parameter sweep to be executed;
- **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.

add_sweep

environment_name, agent_name, agent_params, sweep_dict Add an agent sweep 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;
- **sweep_dict** (*dict*) – dictionary with the sweep configurations.

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.

3.4.2 Experiment

class 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.

3.4.3 Logger

class 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 `/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 Set the directory for logging.

Parameters **log_dir** (*str*) – path of the directory.

get_log_dir

Returns The path of the logging directory.

set_log_id

log_id, use_timestamp=True Set the id of the logged folder.

Parameters

- **log_id** (*str*) – id of the logged folder;
- **use_timestamp** (*bool*, *True*) – whether to use the timestamp or not.

get_log_id

Returns The id of the logged folder.

get_path

filename="" Get the path of the given file. If no filename is given, it returns the path of the logging folder.

Parameters **filename** (*str*, *' '*) – the name of the file.

Returns The complete path of the logged file.

get_params_path

filename="" Get the path of the parameters of the given file. If no filename is given, it returns the path of the parameters folder.

Parameters **filename** (*str*, *' '*) – the name of the file.

Returns The complete path of the logged file.

get_figure_path

filename="", subfolder=None Get the path of the figures of the given file. If no filename is given, it returns the path of the figures folder.

Parameters

- **filename** (*str*, *' '*) – the name of the file;
- **subfolder** (*None*) – the name of a subfolder to add.

Returns The complete path of the logged file.

save_J

J Save the log of the cumulative discounted reward.

load_J

Returns The log of the cumulative discounted reward.

save_R

R Save the log of the cumulative reward.

load_R

Returns The log of the cumulative reward.

save_V

V Save the log of the value function.

load_V

Returns The log of the value function.

save_entropy

entropy Save the log of the entropy function.

load_entropy

Returns The log of the entropy function.

exists_policy_entropy

Returns True if the log of the entropy exists, False otherwise.

exists_value_function

Returns True if the log of the value function exists, False otherwise.

save_best_agent

agent Save the best agent in the respective path.

Parameters **agent** (*object*) – the agent to save.

save_last_agent

agent Save the last agent in the respective path.

Parameters **agent** (*object*) – the agent to save.

exists_best_agent

Returns True if the entropy file exists, False otherwise.

load_best_agent

Returns The best agent.

load_last_agent

Returns The last agent.

save_environment_builder

env_builder Save the environment builder using the respective path.

Parameters **env_builder** (*str*) – the environment builder to save.

load_environment_builder

Returns The environment builder.

save_agent_builder

agent_builder Save the agent builder using the respective path.

Parameters **agent_builder** (*str*) – the agent builder to save.

load_agent_builder

Returns The agent builder.

save_config

config Save the config file using the respective path.

Parameters **config** (*str*) – the config file to save.

load_config

Returns The config file.

exists_stats

Returns True if the entropy file exists, False otherwise.

save_stats

stats Save the statistic file using the respective path.

Parameters **stats** (*str*) – the statistics file to save.

load_stats

Returns The statistics file.

save_params

env, params Save the parameters file.

Parameters

- **env** (*str*) – the environment used;
- **params** (*str*) – the parameters file to save.

save_figure

figure, filename, subfolder=None, as_pdf=False, transparent=True Save the figure file using the respective path.

Parameters

- **figure** (*object*) – the figure to save;
- **filename** (*str*) – the name of the figure;
- **subfolder** (*str, None*) – optional subfolder where to save the figure;
- **as_pdf** (*bool, False*) – whether to save the figure in PDF or not;
- **transparent** (*bool, True*) – whether the figure should be transparent or not.

classmethod from_path

path Method to create a BenchmarkLogger from a path.

3.4.4 Visualizer

class BenchmarkVisualizer

logger, data=None, has_entropy=None, has_value=None, id=1 Bases: `object`

Class to handle all visualizations of the experiment.

plot_counter = 0

__init__

logger, data=None, has_entropy=None, has_value=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 performed experiment.

show_report

Method to show a report of the training metrics from a performed 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.

class BenchmarkSuiteVisualizer

logger, is_sweep, color_cycle=None, y_limit=None, legend=None Bases: `object`

Class to handle visualization of a benchmark suite.

plot_counter = 0

__init__

logger, is_sweep, color_cycle=None, y_limit=None, legend=None Constructor.

Parameters

- **logger** (`BenchmarkLogger`) – logger to be used;
- **is_sweep** (`bool`) – whether the benchmark is a parameter sweep.
- **color_cycle** (`dict, None`) – dictionary with colors to be used for each algorithm;
- **y_limit** (`dict, None`) – dictionary with environment specific plot limits.
- **legend** (`dict, None`) – dictionary with environment specific legend parameters.

get_report

env, data_type, selected_alg=None Create report plot with matplotlib.

get_boxplot

env, metric_type, data_type, selected_alg=None Create boxplot with matplotlib for a given metric.

Parameters

- **env** (`str`) – The environment name;
- **metric_type** (`str`) – The metric to compute.

Returns A figure with the desired boxplot of the given metric.

save_reports

as_pdf=True, transparent=True, alg_sweep=False Method to save an image of a report of the training metrics from a performed experiment.

Parameters

- **as_pdf** (`bool, True`) – whether to save the reports as pdf files or png;
- **transparent** (`bool, True`) – If true, the figure background is transparent and not white;
- **alg_sweep** (`bool, False`) – If true, the method will generate a separate figure for each algorithm sweep.

save_boxplots

as_pdf=True, transparent=True, alg_sweep=False Method to save an image of a report of the training metrics from a performed experiment.

Parameters

- **as_pdf** (*bool, True*) – whether to save the reports as pdf files or png;
- **transparent** (*bool, True*) – If true, the figure background is transparent and not white;
- **alg_sweep** (*bool, False*) – If true, thw method will generate a separate figure for each algorithm sweep.

show_reports

boxplots=True, alg_sweep=False Method to show a report of the training metrics from a performend experiment.

Parameters **alg_sweep** (*bool, False*) – If true, thw method will generate a separate figure for each algorithm sweep.

3.5 Builders

class 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 AgentBuilder

n_steps_per_fit=None, n_episodes_per_fit=None, compute_policy_entropy=True, compute_entropy_with_states=False, compute_value_function=True, preprocessors=None Bases: object

Base class to spawn instances of a MushroomRL agent

__init__

n_steps_per_fit=None, n_episodes_per_fit=None, compute_policy_entropy=True, compute_entropy_with_states=False, compute_value_function=True, preprocessors=None Initialize AgentBuilder

get_fit_params

Get `n_steps_per_fit` and `n_episodes_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

3.5.1 Policy Search Builders

Policy Gradient

class PolicyGradientBuilder

*n_episodes_per_fit, optimizer, **kwargs* Bases: [mushroom_rl_benchmark.builders.agent_builder.AgentBuilder](#)

AgentBuilder for Policy Gradient Methods. The current builder uses a state dependant gaussian with diagonal standard deviation and linear mean.

__init__

*n_episodes_per_fit, optimizer, **kwargs* Constructor.

Parameters

- **optimizer** (*Optimizer*) – optimizer to be used by the policy gradient algorithm;
- ****kwargs** – others algorithms parameters.

build*mdp_info* Build and return the AgentBuilder**Parameters** *mdp_info* (*MDPInfo*) – information about the environment.**classmethod default***n_episodes_per_fit=25, alpha=0.01, get_default_dict=False* Create a default initialization for the specific AgentBuilder and return it**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.

class REINFORCEBuilder*n_episodes_per_fit, optimizer, **kwargs* Bases: *mushroom_rl_benchmark.builders.policy_search.policy_gradient.PolicyGradientBuilder***alg_class**alias of *mushroom_rl.algorithms.policy_search.policy_gradient.reinforce.REINFORCE***class GPOMDPBuilder***n_episodes_per_fit, optimizer, **kwargs* Bases: *mushroom_rl_benchmark.builders.policy_search.policy_gradient.PolicyGradientBuilder***alg_class**alias of *mushroom_rl.algorithms.policy_search.policy_gradient.gpomdp.GPOMDP***class eNACBuilder***n_episodes_per_fit, optimizer, **kwargs* Bases: *mushroom_rl_benchmark.builders.policy_search.policy_gradient.PolicyGradientBuilder***alg_class**alias of *mushroom_rl.algorithms.policy_search.policy_gradient.enac.eNAC***Black-Box optimization****class BBOBuilder***n_episodes_per_fit, **kwargs* Bases: *mushroom_rl_benchmark.builders.agent_builder.AgentBuilder*

AgentBuilder for Black Box optimization methods. The current builder uses a simple deterministic linear policy and gaussian Diagonal distribution.

__init__*n_episodes_per_fit, **kwargs* Constructor.**Parameters**

- **optimizer** (*Optimizer*) – optimizer to be used by the policy gradient algorithm;
- ****kwargs** – others algorithms parameters.

build*mdp_info* Build and return the AgentBuilder

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

classmethod default

n_episodes_per_fit=25, alpha=0.01, get_default_dict=False Create a default initialization for the specific AgentBuilder and return it

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.

class PGPEBuilder

n_episodes_per_fit, optimizer Bases: *mushroom_rl_benchmark.builders.policy_search.black_box_optimization.BBOBuilder*

alg_class

alias of *mushroom_rl.algorithms.policy_search.black_box_optimization.pgpe.PGPE*

__init__

n_episodes_per_fit, optimizer Constructor.

Parameters

- **optimizer** (*Optimizer*) – optimizer to be used by the policy gradient algorithm;
- ****kwargs** – others algorithms parameters.

classmethod default

n_episodes_per_fit=25, alpha=0.3, get_default_dict=False Create a default initialization for the specific AgentBuilder and return it

class RWRBuilder

n_episodes_per_fit, beta Bases: *mushroom_rl_benchmark.builders.policy_search.black_box_optimization.BBOBuilder*

alg_class

alias of *mushroom_rl.algorithms.policy_search.black_box_optimization.rwr.RWR*

__init__

n_episodes_per_fit, beta Constructor.

Parameters

- **optimizer** (*Optimizer*) – optimizer to be used by the policy gradient algorithm;
- ****kwargs** – others algorithms parameters.

classmethod default

n_episodes_per_fit=25, beta=0.01, get_default_dict=False Create a default initialization for the specific AgentBuilder and return it

class REPSBuilder

n_episodes_per_fit, eps Bases: *mushroom_rl_benchmark.builders.policy_search.black_box_optimization.BBOBuilder*

alg_class

alias of `mushroom_rl.algorithms.policy_search.black_box_optimization.reps.REPS`

__init__

n_episodes_per_fit, eps Constructor.

Parameters

- **optimizer** (*Optimizer*) – optimizer to be used by the policy gradient algorithm;
- ****kwargs** – others algorithms parameters.

classmethod default

n_episodes_per_fit=25, eps=0.05, get_default_dict=False Create a default initialization for the specific AgentBuilder and return it

class ConstrainedREPSBuilder

n_episodes_per_fit, eps, kappa Bases: `mushroom_rl_benchmark.builders.policy_search.black_box_optimization.BBOBuilder`

alg_class

alias of `mushroom_rl.algorithms.policy_search.black_box_optimization.constrained_reps.ConstrainedREPS`

__init__

n_episodes_per_fit, eps, kappa Constructor.

Parameters

- **optimizer** (*Optimizer*) – optimizer to be used by the policy gradient algorithm;
- ****kwargs** – others algorithms parameters.

classmethod default

n_episodes_per_fit=25, eps=0.05, kappa=0.01, get_default_dict=False Create a default initialization for the specific AgentBuilder and return it

3.5.2 Value Based Builders

Temporal Difference

class TDFiniteBuilder

*learning_rate, epsilon, epsilon_test, **alg_params* Bases: `mushroom_rl_benchmark.builders.agent_builder.AgentBuilder`

AgentBuilder for a generic TD algorithm (for finite states).

__init__

*learning_rate, epsilon, epsilon_test, **alg_params* Constructor.

Parameters

- **epsilon** (*Parameter*) – exploration coefficient for learning;
- **epsilon_test** (*Parameter*) – exploration coefficient for test.

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

learning_rate=0.9, epsilon=0.1, decay_lr=0.0, decay_eps=0.0, epsilon_test=0.0, get_default_dict=False
Create a default initialization for the specific AgentBuilder and return it

class QLearningBuilder

learning_rate, epsilon, epsilon_test Bases: *mushroom_rl_benchmark.builders.value.td.td_finite.TDFiniteBuilder*

alg_class

alias of *mushroom_rl.algorithms.value.td.q_learning.QLearning*

__init__

learning_rate, epsilon, epsilon_test Constructor.

Parameters

- **epsilon** (*Parameter*) – exploration coefficient for learning;
- **epsilon_test** (*Parameter*) – exploration coefficient for test.

class SARSABuilder

learning_rate, epsilon, epsilon_test Bases: *mushroom_rl_benchmark.builders.value.td.td_finite.TDFiniteBuilder*

alg_class

alias of *mushroom_rl.algorithms.value.td.sarsa.SARSA*

__init__

learning_rate, epsilon, epsilon_test Constructor.

Parameters

- **epsilon** (*Parameter*) – exploration coefficient for learning;
- **epsilon_test** (*Parameter*) – exploration coefficient for test.

class SpeedyQLearningBuilder

learning_rate, epsilon, epsilon_test Bases: *mushroom_rl_benchmark.builders.value.td.td_finite.TDFiniteBuilder*

alg_class

alias of *mushroom_rl.algorithms.value.td.speedy_q_learning.SpeedyQLearning*

__init__

learning_rate, epsilon, epsilon_test Constructor.

Parameters

- **epsilon** (*Parameter*) – exploration coefficient for learning;
- **epsilon_test** (*Parameter*) – exploration coefficient for test.

class DoubleQLearningBuilder

learning_rate, epsilon, epsilon_test Bases: *mushroom_rl_benchmark.builders.value.td.td_finite.TDFiniteBuilder*

alg_class

alias of *mushroom_rl.algorithms.value.td.double_q_learning.DoubleQLearning*

__init__

learning_rate, epsilon, epsilon_test Constructor.

Parameters

- **epsilon** (*Parameter*) – exploration coefficient for learning;
- **epsilon_test** (*Parameter*) – exploration coefficient for test.

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.

class WeightedQLearningBuilder

learning_rate, epsilon, epsilon_test, sampling, precision Bases: *mushroom_rl_benchmark.builders.value.td.td_finite.TDFiniteBuilder*

alg_class

alias of *mushroom_rl.algorithms.value.td.weighted_q_learning.WeightedQLearning*

__init__

learning_rate, epsilon, epsilon_test, sampling, precision Constructor.

Parameters

- **sampling** (*bool*, *True*) – use the approximated version to speed up the computation;
- **precision** (*int*, *1000*) – number of samples to use in the approximated version.

classmethod default

learning_rate=0.9, epsilon=0.1, decay_lr=0.0, decay_eps=0.0, epsilon_test=0.0, sampling=True, precision=1000, get_default_dict=False Create a default initialization for the specific AgentBuilder and return it

class TDTraceBuilder

learning_rate, epsilon, epsilon_test, lambda_coeff, trace Bases: *mushroom_rl_benchmark.builders.value.td.td_finite.TDFiniteBuilder*

Builder for TD algorithms with eligibility traces and finite states.

__init__

learning_rate, epsilon, epsilon_test, lambda_coeff, trace Constructor.

lambda_coeff ([float, *Parameter*]): eligibility trace coefficient; *trace* (str): type of eligibility trace to use.

classmethod default

learning_rate=0.9, epsilon=0.1, decay_lr=0.0, decay_eps=0.0, epsilon_test=0.0, lambda_coeff=0.9, trace='replacing', get_default_dict=False Create a default initialization for the specific AgentBuilder and return it

class SarsaLambdaBuilder

learning_rate, epsilon, epsilon_test, lambda_coeff, trace Bases: `mushroom_rl_benchmark.builders.value.td.td_trace.TDTraceBuilder`

alg_class

alias of `mushroom_rl.algorithms.value.td.sarsa_lambda.SarsaLambda`

class QLambdaBuilder

learning_rate, epsilon, epsilon_test, lambda_coeff, trace Bases: `mushroom_rl_benchmark.builders.value.td.td_trace.TDTraceBuilder`

alg_class

alias of `mushroom_rl.algorithms.value.td.q_lambda.QLambda`

class SarsaLambdaContinuousBuilder

policy, approximator, learning_rate, lambda_coeff, epsilon, epsilon_test, n_tilings, n_tiles Bases: `mushroom_rl_benchmark.builders.value.td.td_continuous.TDContinuousBuilder`

AgentBuilder for Sarsa(Lambda) Continuous. Using tiles as function approximator.

__init__

policy, approximator, learning_rate, lambda_coeff, epsilon, epsilon_test, n_tilings, n_tiles Constructor.

Parameters *approximator* (*class*) – Q-function approximator.

build

mdp_info Build and return the AgentBuilder

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

classmethod default

alpha=0.1, lambda_coeff=0.9, epsilon=0.0, decay_eps=0.0, epsilon_test=0.0, n_tilings=10, n_tiles=10, get_default_dict=False Create a default initialization for the specific AgentBuilder and return it

class TrueOnlineSarsaLambdaBuilder

policy, learning_rate, lambda_coeff, epsilon, epsilon_test, n_tilings, n_tiles Bases: `mushroom_rl_benchmark.builders.value.td.td_continuous.TDContinuousBuilder`

AgentBuilder for True Online Sarsa(Lambda) Continuous. Using tiles as function approximator.

__init__

policy, learning_rate, lambda_coeff, epsilon, epsilon_test, n_tilings, n_tiles Constructor.

build

mdp_info Build and return the AgentBuilder

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

classmethod default

alpha=0.1, lambda_coeff=0.9, epsilon=0.0, decay_eps=0.0, epsilon_test=0.0, n_tilings=10, n_tiles=10, get_default_dict=False Create a default initialization for the specific AgentBuilder and return it

DQN

class 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.DQNNetwork'>, 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 DoubleDQNBuilder`

policy, approximator, approximator_params, alg_params, n_steps_per_fit=1 Bases: [mushroom_rl_benchmark.builders.value.dqn.dqn.DQNBuilder](#)

`build`

mdp_info Build and return the AgentBuilder

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

`class AveragedDQNBuilder`

policy, approximator, approximator_params, alg_params, n_steps_per_fit=1 Bases: [mushroom_rl_benchmark.builders.value.dqn.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.DQNNetwork'>, 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 PrioritizedDQNBuilder

policy, approximator, approximator_params, alg_params, n_steps_per_fit=1 Bases: [*mushroom_rl_benchmark.builders.value.dqn.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.DQNNetwork'>, 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 DuelingDQNBuilder

policy, approximator, approximator_params, alg_params, n_steps_per_fit=1 Bases: [*mushroom_rl_benchmark.builders.value.dqn.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.DQNFeatureNetwork'>, 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 MaxminDQNBuilder

policy, approximator, approximator_params, alg_params, n_steps_per_fit=1 Bases: [*mushroom_rl_benchmark.builders.value.dqn.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.DQNNetwork'>, 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 NoisyDQNBuilder

policy, approximator, approximator_params, alg_params, n_steps_per_fit=1 Bases: [*mushroom_rl_benchmark.builders.value.dqn.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.DQNFeatureNetwork'>,

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 CategoricalDQNBuilder

policy, approximator, approximator_params, alg_params, n_steps_per_fit=1 Bases: *mushroom_rl_benchmark.builders.value.dqn.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.DQNFeatureNetwork'>, 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

3.5.3 Actor Critic Builders

Classic AC

class StochasticACBuilder

*std_0, alpha_theta, alpha_v, lambda_par, n_tilings, n_tiles, **kwargs* Bases: *mushroom_rl_benchmark.builders.agent_builder.AgentBuilder*

Builder for the stochastic actor critic algorithm. Using linear approximator with tiles for mean, standard deviation and value function approximator. The value function approximator also uses a bias term.

__init__

*std_0, alpha_theta, alpha_v, lambda_par, n_tilings, n_tiles, **kwargs* Constructor.

Parameters

- **std_0** (*float*) – initial standard deviation;
- **alpha_theta** (*Parameter*) – Learning rate for the policy;
- **alpha_v** (*Parameter*) – Learning rate for the value function;
- **n_tilings** (*int*) – number of tilings to be used as approximator;
- **n_tiles** (*int*) – number of tiles for each state space dimension.

build

mdp_info Build and return the AgentBuilder

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

classmethod default

std_0=1.0, alpha_theta=0.001, alpha_v=0.1, lambda_par=0.9, n_tilings=10, n_tiles=11, get_default_dict=False Create a default initialization for the specific AgentBuilder and return it

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.

class COPDAC_QBuilder

*std_exp, std_eval, alpha_theta, alpha_omega, alpha_v, n_tilings, n_tiles, **kwargs* Bases: *mushroom_rl_benchmark.builders.agent_builder.AgentBuilder*

Builder for the COPDAQ_Q actor critic algorithm. Using linear approximator with tiles for the mean and value function approximator.

__init__

*std_exp, std_eval, alpha_theta, alpha_omega, alpha_v, n_tilings, n_tiles, **kwargs* Constructor.

Parameters

- **std_exp** (*float*) – exploration standard deviation;
- **std_eval** (*float*) – evaluation standard deviation;
- **alpha_theta** (*Parameter*) – Learning rate for the policy;
- **alpha_omega** (*Parameter*) – Learning rate for the
- **alpha_v** (*Parameter*) – Learning rate for the value function;
- **n_tilings** (*int*) – number of tilings to be used as approximator;
- **n_tiles** (*int*) – number of tiles for each state space dimension.

build

mdp_info Build and return the AgentBuilder

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

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

std_exp=0.1, std_eval=0.001, alpha_theta=0.005, alpha_omega=0.5, alpha_v=0.5, n_tilings=10, n_tiles=11, get_default_dict=False Create a default initialization for the specific AgentBuilder and return it

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.

Deep AC

class 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 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 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, ent_coeff=0.0, 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, std_0=1.0, preprocessors=None,

use_cuda=False, get_default_dict=False Create a default initialization for the specific AgentBuilder and return it

class 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.SACriticNetwork'>, 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 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 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, std_0=1.0, preprocessors=None, use_cuda=False, get_default_dict=False
 Create a default initialization for the specific AgentBuilder and return it

3.6 Networks

class 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 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 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 SACriticNetwork

*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 SAActorNetwork

*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 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 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 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.

3.7 Experiment

exec_run

*agent_builder, env_builder, n_epochs, n_steps=None, n_episodes=None, 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, None`) – number of steps per epoch;
- **n_episodes** (`int, None`) – number of episodes 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.

compute_metrics

core, eval_params, agent_builder, env_builder 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;

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).

3.7.1 Slurm utilities

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.

make_arguments

***params* Create a script argument string from dictionary

read_arguments_run

arg_string=None Parse the arguments for the run script.

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

read_arguments_aggregate

arg_string=None Parse the arguments for the aggregate script.

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

create_slurm_script

*slurm_path, slurm_script_name='slurm.sh', **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.

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.

to_duration

hours, minutes, seconds

3.8 Utils

get_init_states

dataset Get the initial states of a MushroomRL dataset

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

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.

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.

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.

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.

plot_mean_conf

data, ax, color='blue', line='-', facecolor=None, alpha=0.4, label=None Method to plot mean and confidence interval for data on pyplot axes.

build_sweep_list

algs, sweep_conf, base_name='c_' Build the sweep list, from a compact dictionary specification, for every considered algorithm.

Parameters

- **algs** (*list*) – list of algorithms to be considered;
- **sweep_conf** (*dict*) – dictionary with a compact sweep configuration for every algorithm;
- **base_name** (*str*, 'c_') – base name for the sweep configuration.

Returns The sweep list to be used with the suite.

build_sweep_dict

*base_name='c_', **kwargs* Build the sweep dictionary, from a set of variable specifications.

Parameters

- **base_name** (*str*, 'c_') – base name for the sweep configuration;
- ****kwargs** – the parameter specifications for the sweep.

Returns The sweep dictionary, where the key is the sweep name and the value is a dictionary with the sweep parameters.

generate_sweep

*base_name='c_', **kwargs* Generator that returns tuples with sweep name and parameters

Parameters

- **base_name** (*str*, 'c_') – base name for the sweep configuration;
- ****kwargs** – the parameter specifications for the sweep.

generate_sweep_params

***kwargs* Generator that returns sweep parameters

Parameters ****kwargs** – the parameter specifications for the sweep.

m

`mushroom_rl_benchmark.builders.actor_critic.classic_actor_critic.copdac_g`, 66
`mushroom_rl_benchmark.builders.actor_critic.classic_actor_critic.stochastic_ac`, 65
`mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.a2c`, 66
`mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.ddpg`, 67
`mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.ppo`, 68
`mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.sac`, 69
`mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.td3`, 69
`mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.trpo`, 70
`mushroom_rl_benchmark.builders.agent_builder`, 55
`mushroom_rl_benchmark.builders.environment_builder`, 55
`mushroom_rl_benchmark.builders.network.a2c_network`, 71
`mushroom_rl_benchmark.builders.network.ddpg_network`, 71
`mushroom_rl_benchmark.builders.network.sac_network`, 72
`mushroom_rl_benchmark.builders.network.td3_network`, 72
`mushroom_rl_benchmark.builders.network.trpo_network`, 73
`mushroom_rl_benchmark.builders.policy_search.black_box_optimization`, 57
`mushroom_rl_benchmark.builders.policy_search.policy_gradient`, 56
`mushroom_rl_benchmark.builders.value.dqn.averaged_dqn`, 63
`mushroom_rl_benchmark.builders.value.dqn.categorical_dqn`, 65
`mushroom_rl_benchmark.builders.value.dqn.double_dqn`, 63
`mushroom_rl_benchmark.builders.value.dqn.dqn`, 62
`mushroom_rl_benchmark.builders.value.dqn.dueling_dqn`, 64
`mushroom_rl_benchmark.builders.value.dqn.maxmin_dqn`, 64
`mushroom_rl_benchmark.builders.value.dqn.noisy_dqn`, 64
`mushroom_rl_benchmark.builders.value.dqn.prioritized_dqn`, 64
`mushroom_rl_benchmark.builders.value.td.sarsa_lambda_continuous`, 62
`mushroom_rl_benchmark.builders.value.td.td_finite`, 59
`mushroom_rl_benchmark.builders.value.td.td_trace`, 61
`mushroom_rl_benchmark.builders.value.td.true_online_sarsa_lambda`, 62
`mushroom_rl_benchmark.core.experiment`, 48
`mushroom_rl_benchmark.core.logger`, 50
`mushroom_rl_benchmark.core.suite`, 46
`mushroom_rl_benchmark.core.suite_visualizer`, 54

`mushroom_rl_benchmark.core.visualizer`, [53](#)
`mushroom_rl_benchmark.experiment.run`, [73](#)
`mushroom_rl_benchmark.experiment.slurm.aggregate_results`, [74](#)
`mushroom_rl_benchmark.experiment.slurm.arguments`, [74](#)
`mushroom_rl_benchmark.experiment.slurm.run_script`, [74](#)
`mushroom_rl_benchmark.experiment.slurm.slurm_script`, [74](#)
`mushroom_rl_benchmark.utils.plot`, [76](#)
`mushroom_rl_benchmark.utils.primitive`, [75](#)
`mushroom_rl_benchmark.utils.sweep`, [76](#)
`mushroom_rl_benchmark.utils.utils`, [75](#)

Symbols

`__init__` () (*A2CBuilder method*), 67
`__init__` () (*A2CNetwork method*), 71
`__init__` () (*AgentBuilder method*), 55
`__init__` () (*BBOBuilder method*), 57
`__init__` () (*BenchmarkExperiment method*), 48
`__init__` () (*BenchmarkLogger method*), 50
`__init__` () (*BenchmarkSuite method*), 46
`__init__` () (*BenchmarkSuiteVisualizer method*), 54
`__init__` () (*BenchmarkVisualizer method*), 53
`__init__` () (*COPDAC_QBuilder method*), 66
`__init__` () (*ConstrainedREPSBuilder method*), 59
`__init__` () (*DDPGActorNetwork method*), 71
`__init__` () (*DDPGBuilder method*), 67
`__init__` () (*DDPGCriticNetwork method*), 71
`__init__` () (*DQNBuilder method*), 63
`__init__` () (*DoubleQLearningBuilder method*), 61
`__init__` () (*EnvironmentBuilder method*), 55
`__init__` () (*PGPEBuilder method*), 58
`__init__` () (*PPOBuilder method*), 68
`__init__` () (*PolicyGradientBuilder method*), 56
`__init__` () (*QLearningBuilder method*), 60
`__init__` () (*REPSBuilder method*), 59
`__init__` () (*RWRBuilder method*), 58
`__init__` () (*SACActorNetwork method*), 72
`__init__` () (*SACBuilder method*), 69
`__init__` () (*SACCriticNetwork method*), 72
`__init__` () (*SARSABuilder method*), 60
`__init__` () (*SarsaLambdaContinuousBuilder method*), 62
`__init__` () (*SpeedyQLearningBuilder method*), 60
`__init__` () (*StochasticACBuilder method*), 65
`__init__` () (*TD3ActorNetwork method*), 73
`__init__` () (*TD3Builder method*), 69
`__init__` () (*TD3CriticNetwork method*), 72
`__init__` () (*TDFiniteBuilder method*), 59
`__init__` () (*TDTraceBuilder method*), 61
`__init__` () (*TRPOBuilder method*), 70
`__init__` () (*TRPONetwork method*), 73
`__init__` () (*TrueOnlineSarsaLambdaBuilder method*), 62
`__init__` () (*WeightedQLearningBuilder method*), 61

A

A2CBuilder (*class in mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.a2c*), 66
A2CNetwork (*class in mushroom_rl_benchmark.builders.network.a2c_network*), 71

`add_agent()` (*BenchmarkSuite* method), 47
`add_environment()` (*BenchmarkSuite* method), 47
`add_experiments()` (*BenchmarkSuite* method), 47
`add_experiments_sweeps()` (*BenchmarkSuite* method), 47
`add_sweep()` (*BenchmarkSuite* method), 48
`AgentBuilder` (class in *mushroom_rl_benchmark.builders.agent_builder*), 55
`aggregate_results()` (in module *mushroom_rl_benchmark.experiment.slurm.aggregate_results*), 74
`alg_class` (*ConstrainedREPSBuilder* attribute), 59
`alg_class` (*DoubleQLearningBuilder* attribute), 61
`alg_class` (*eNACBuilder* attribute), 57
`alg_class` (*GPOMDPBuilder* attribute), 57
`alg_class` (*PGPEBuilder* attribute), 58
`alg_class` (*QLambdaBuilder* attribute), 62
`alg_class` (*QLearningBuilder* attribute), 60
`alg_class` (*REINFORCEBuilder* attribute), 57
`alg_class` (*REPSBuilder* attribute), 58
`alg_class` (*RWRBuilder* attribute), 58
`alg_class` (*SARSABuilder* attribute), 60
`alg_class` (*SARSALambdaBuilder* attribute), 62
`alg_class` (*SpeedyQLearningBuilder* attribute), 60
`alg_class` (*WeightedQLearningBuilder* attribute), 61
`AveragedDQNBuilder` (class in *mushroom_rl_benchmark.builders.value.dqn.averaged_dqn*), 63

B

`BBOBuilder` (class in *mushroom_rl_benchmark.builders.policy_search.black_box_optimization*), 57
`BenchmarkExperiment` (class in *mushroom_rl_benchmark.core.experiment*), 48
`BenchmarkLogger` (class in *mushroom_rl_benchmark.core.logger*), 50
`BenchmarkSuite` (class in *mushroom_rl_benchmark.core.suite*), 46
`BenchmarkSuiteVisualizer` (class in *mushroom_rl_benchmark.core.suite_visualizer*), 54
`BenchmarkVisualizer` (class in *mushroom_rl_benchmark.core.visualizer*), 53
`build()` (*A2CBuilder* method), 67
`build()` (*AgentBuilder* method), 56
`build()` (*AveragedDQNBuilder* method), 63
`build()` (*BBOBuilder* method), 57
`build()` (*CategoricalDQNBuilder* method), 65
`build()` (*COPDAC_QBuilder* method), 66
`build()` (*DDPGBuilder* method), 67
`build()` (*DoubleDQNBuilder* method), 63
`build()` (*DQNBuilder* method), 63
`build()` (*DuelingDQNBuilder* method), 64
`build()` (*EnvironmentBuilder* method), 55
`build()` (*MaxminDQNBuilder* method), 64
`build()` (*NoisyDQNBuilder* method), 64
`build()` (*PolicyGradientBuilder* method), 56
`build()` (*PPOBuilder* method), 68
`build()` (*PrioritizedDQNBuilder* method), 64
`build()` (*SACBuilder* method), 69
`build()` (*SarsaLambdaContinuousBuilder* method), 62
`build()` (*StochasticACBuilder* method), 65
`build()` (*TD3Builder* method), 70
`build()` (*TDFiniteBuilder* method), 59
`build()` (*TRPOBuilder* method), 70
`build()` (*TrueOnlineSarsaLambdaBuilder* method), 62
`build_sweep_dict()` (in module *mushroom_rl_benchmark.utils.sweep*), 76
`build_sweep_list()` (in module *mushroom_rl_benchmark.utils.sweep*), 76

C

`CategoricalDQNBuilder` (class in *mushroom_rl_benchmark.builders.value.dqn.categorical_dqn*), 65
`compute_metrics()` (in module *mushroom_rl_benchmark.experiment.run*), 74
`compute_Q()` (*A2CBuilder* method), 67
`compute_Q()` (*AgentBuilder* method), 56
`compute_Q()` (*BBOBuilder* method), 58
`compute_Q()` (*COPDAC_QBuilder* method), 66
`compute_Q()` (*DDPGBuilder* method), 68
`compute_Q()` (*DoubleQLearningBuilder* method), 61
`compute_Q()` (*DQNBuilder* method), 63
`compute_Q()` (*PolicyGradientBuilder* method), 57

`compute_Q()` (*PPOBuilder method*), 68
`compute_Q()` (*SACBuilder method*), 69
`compute_Q()` (*StochasticACBuilder method*), 65
`compute_Q()` (*TD3Builder method*), 70
`compute_Q()` (*TDFiniteBuilder method*), 59
`compute_Q()` (*TRPOBuilder method*), 70
`ConstrainedREPSBuilder` (*class in mushroom_rl_benchmark.builders.policy_search.black_box_optimization*), 59
`COPDAC_QBuilder` (*class in mushroom_rl_benchmark.builders.actor_critic.classic_actor_critic.copdac_q*), 66
`copy()` (*AgentBuilder method*), 56
`copy()` (*EnvironmentBuilder method*), 55
`create_slurm_script()` (*in module mushroom_rl_benchmark.experiment.slurm.slurm_script*), 74

D

`DDPGActorNetwork` (*class in mushroom_rl_benchmark.builders.network.ddpg_network*), 71
`DDPGBuilder` (*class in mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.ddpg*), 67
`DDPGCriticNetwork` (*class in mushroom_rl_benchmark.builders.network.ddpg_network*), 71
`default()` (*mushroom_rl_benchmark.builders.actor_critic.classic_actor_critic.copdac_q.COPDAC_QBuilder class method*), 66
`default()` (*mushroom_rl_benchmark.builders.actor_critic.classic_actor_critic.stochastic_ac.StochasticACBuilder class method*), 65
`default()` (*mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.a2c.A2CBuilder class method*), 67
`default()` (*mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.ddpg.DDPGBuilder class method*), 68
`default()` (*mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.ppo.PPOBuilder class method*), 68
`default()` (*mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.sac.SACBuilder class method*), 69
`default()` (*mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.td3.TD3Builder class method*), 70
`default()` (*mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.trpo.TRPOBuilder class method*), 71
`default()` (*mushroom_rl_benchmark.builders.agent_builder.AgentBuilder class method*), 56
`default()` (*mushroom_rl_benchmark.builders.policy_search.black_box_optimization.BBOBuilder class method*), 58
`default()` (*mushroom_rl_benchmark.builders.policy_search.black_box_optimization.ConstrainedREPSBuilder class method*), 59
`default()` (*mushroom_rl_benchmark.builders.policy_search.black_box_optimization.PGPEBuilder class method*), 58
`default()` (*mushroom_rl_benchmark.builders.policy_search.black_box_optimization.REPSBuilder class method*), 59
`default()` (*mushroom_rl_benchmark.builders.policy_search.black_box_optimization.RWRBuilder class method*), 58
`default()` (*mushroom_rl_benchmark.builders.policy_search.policy_gradient.PolicyGradientBuilder class method*), 57
`default()` (*mushroom_rl_benchmark.builders.value.dqn.averaged_dqn.AveragedDQNBuilder class method*), 63
`default()` (*mushroom_rl_benchmark.builders.value.dqn.categorical_dqn.CategoricalDQNBuilder class method*), 65
`default()` (*mushroom_rl_benchmark.builders.value.dqn.dqn.DQNBuilder class method*), 63
`default()` (*mushroom_rl_benchmark.builders.value.dqn.dueling_dqn.DuelingDQNBuilder class method*), 64
`default()` (*mushroom_rl_benchmark.builders.value.dqn.maxmin_dqn.MaxminDQNBuilder class method*), 64
`default()` (*mushroom_rl_benchmark.builders.value.dqn.noisy_dqn.NoisyDQNBuilder class method*), 64
`default()` (*mushroom_rl_benchmark.builders.value.dqn.prioritized_dqn.PrioritizedDQNBuilder class method*), 64
`default()` (*mushroom_rl_benchmark.builders.value.td.sarsa_lambda_continuous.SarsaLambdaContinuousBuilder class method*), 62
`default()` (*mushroom_rl_benchmark.builders.value.td.td_finite.TDFiniteBuilder class method*), 60
`default()` (*mushroom_rl_benchmark.builders.value.td.td_finite.WeightedQLearningBuilder class method*), 61
`default()` (*mushroom_rl_benchmark.builders.value.td.td_trace.TDTraceBuilder class method*), 61
`default()` (*mushroom_rl_benchmark.builders.value.td.true_online_sarsa_lambda.TrueOnlineSarsaLambdaBuilder class method*), 62
`dictionary_to_primitive()` (*in module mushroom_rl_benchmark.utils.primitive*), 76
`DoubleDQNBuilder` (*class in mushroom_rl_benchmark.builders.value.dqn.double_dqn*), 63
`DoubleQLearningBuilder` (*class in mushroom_rl_benchmark.builders.value.td.td_finite*), 61
`DQNBuilder` (*class in mushroom_rl_benchmark.builders.value.dqn.dqn*), 62
`DuelingDQNBuilder` (*class in mushroom_rl_benchmark.builders.value.dqn.dueling_dqn*), 64

E

`eNACBuilder` (*class in mushroom_rl_benchmark.builders.policy_search.policy_gradient*), 57
`EnvironmentBuilder` (*class in mushroom_rl_benchmark.builders.environment_builder*), 55
`exec_run()` (*in module mushroom_rl_benchmark.experiment.run*), 73
`exists_best_agent()` (*BenchmarkLogger method*), 52
`exists_policy_entropy()` (*BenchmarkLogger method*), 52
`exists_stats()` (*BenchmarkLogger method*), 52
`exists_value_function()` (*BenchmarkLogger method*), 52
`extend_and_save_entropy()` (*BenchmarkExperiment method*), 50
`extend_and_save_J()` (*BenchmarkExperiment method*), 50
`extend_and_save_R()` (*BenchmarkExperiment method*), 50
`extend_and_save_V()` (*BenchmarkExperiment method*), 50
`extract_arguments()` (*in module mushroom_rl_benchmark.utils.utils*), 75

F

`forward()` (*A2CNetwork method*), 71
`forward()` (*DDPGActorNetwork method*), 71

`forward()` (*DDPGCriticNetwork method*), 71
`forward()` (*SACActorNetwork method*), 72
`forward()` (*SACCriticNetwork method*), 72
`forward()` (*TD3ActorNetwork method*), 73
`forward()` (*TD3CriticNetwork method*), 72
`forward()` (*TRPONetwork method*), 73
`from_path()` (*mushroom_rl_benchmark.core.logger.BenchmarkLogger class method*), 53
`from_path()` (*mushroom_rl_benchmark.core.visualizer.BenchmarkVisualizer class method*), 54

G

`generate_slurm()` (*in module mushroom_rl_benchmark.experiment.slurm.slurm_script*), 75
`generate_sweep()` (*in module mushroom_rl_benchmark.utils.sweep*), 76
`generate_sweep_params()` (*in module mushroom_rl_benchmark.utils.sweep*), 76
`get_boxplot()` (*BenchmarkSuiteVisualizer method*), 54
`get_entropy()` (*BenchmarkVisualizer method*), 53
`get_figure_path()` (*BenchmarkLogger method*), 51
`get_fit_params()` (*AgentBuilder method*), 55
`get_init_states()` (*in module mushroom_rl_benchmark.utils.utils*), 75
`get_J()` (*BenchmarkVisualizer method*), 53
`get_log_dir()` (*BenchmarkLogger method*), 50
`get_log_id()` (*BenchmarkLogger method*), 51
`get_mean_and_confidence()` (*in module mushroom_rl_benchmark.utils.plot*), 76
`get_params_path()` (*BenchmarkLogger method*), 51
`get_path()` (*BenchmarkLogger method*), 51
`get_preprocessors()` (*AgentBuilder method*), 56
`get_R()` (*BenchmarkVisualizer method*), 53
`get_report()` (*BenchmarkSuiteVisualizer method*), 54
`get_report()` (*BenchmarkVisualizer method*), 54
`get_V()` (*BenchmarkVisualizer method*), 53
`GPOMDPBuilder` (*class in mushroom_rl_benchmark.builders.policy_search.policy_gradient*), 57

I

`is_data_persisted` (*BenchmarkVisualizer attribute*), 53

L

`load_agent_builder()` (*BenchmarkLogger method*), 52
`load_best_agent()` (*BenchmarkLogger method*), 52
`load_config()` (*BenchmarkLogger method*), 52
`load_entropy()` (*BenchmarkLogger method*), 51
`load_environment_builder()` (*BenchmarkLogger method*), 52
`load_J()` (*BenchmarkLogger method*), 51
`load_last_agent()` (*BenchmarkLogger method*), 52
`load_R()` (*BenchmarkLogger method*), 51
`load_stats()` (*BenchmarkLogger method*), 52
`load_V()` (*BenchmarkLogger method*), 51

M

`make_arguments()` (*in module mushroom_rl_benchmark.experiment.slurm.arguments*), 74
`MaxminDQNBuilder` (*class in mushroom_rl_benchmark.builders.value.dqn.maxmin_dqn*), 64
`mushroom_rl_benchmark.builders.actor_critic.classic_actor_critic.copdac_q` (*module*), 66
`mushroom_rl_benchmark.builders.actor_critic.classic_actor_critic.stochastic_ac` (*module*), 65
`mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.a2c` (*module*), 66
`mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.ddpg` (*module*), 67
`mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.ppo` (*module*), 68
`mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.sac` (*module*), 69
`mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.td3` (*module*), 69
`mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.trpo` (*module*), 70
`mushroom_rl_benchmark.builders.agent_builder` (*module*), 55
`mushroom_rl_benchmark.builders.environment_builder` (*module*), 55
`mushroom_rl_benchmark.builders.network.a2c_network` (*module*), 71
`mushroom_rl_benchmark.builders.network.ddpg_network` (*module*), 71
`mushroom_rl_benchmark.builders.network.sac_network` (*module*), 72
`mushroom_rl_benchmark.builders.network.td3_network` (*module*), 72
`mushroom_rl_benchmark.builders.network.trpo_network` (*module*), 73
`mushroom_rl_benchmark.builders.policy_search.black_box_optimization` (*module*), 57

[mushroom_rl_benchmark.builders.policy_search.policy_gradient \(module\)](#), 56
[mushroom_rl_benchmark.builders.value.dqn.averaged_dqn \(module\)](#), 63
[mushroom_rl_benchmark.builders.value.dqn.categorical_dqn \(module\)](#), 65
[mushroom_rl_benchmark.builders.value.dqn.double_dqn \(module\)](#), 63
[mushroom_rl_benchmark.builders.value.dqn.dqn \(module\)](#), 62
[mushroom_rl_benchmark.builders.value.dqn.dueling_dqn \(module\)](#), 64
[mushroom_rl_benchmark.builders.value.dqn.maxmin_dqn \(module\)](#), 64
[mushroom_rl_benchmark.builders.value.dqn.noisy_dqn \(module\)](#), 64
[mushroom_rl_benchmark.builders.value.dqn.prioritized_dqn \(module\)](#), 64
[mushroom_rl_benchmark.builders.value.td.sarsa_lambda_continuous \(module\)](#), 62
[mushroom_rl_benchmark.builders.value.td.td_finite \(module\)](#), 59
[mushroom_rl_benchmark.builders.value.td.td_trace \(module\)](#), 61
[mushroom_rl_benchmark.builders.value.td.true_online_sarsa_lambda \(module\)](#), 62
[mushroom_rl_benchmark.core.experiment \(module\)](#), 48
[mushroom_rl_benchmark.core.logger \(module\)](#), 50
[mushroom_rl_benchmark.core.suite \(module\)](#), 46
[mushroom_rl_benchmark.core.suite_visualizer \(module\)](#), 54
[mushroom_rl_benchmark.core.visualizer \(module\)](#), 53
[mushroom_rl_benchmark.experiment.run \(module\)](#), 73
[mushroom_rl_benchmark.experiment.slurm.aggregate_results \(module\)](#), 74
[mushroom_rl_benchmark.experiment.slurm.arguments \(module\)](#), 74
[mushroom_rl_benchmark.experiment.slurm.run_script \(module\)](#), 74
[mushroom_rl_benchmark.experiment.slurm.slurm_script \(module\)](#), 74
[mushroom_rl_benchmark.utils.plot \(module\)](#), 76
[mushroom_rl_benchmark.utils.primitive \(module\)](#), 75
[mushroom_rl_benchmark.utils.sweep \(module\)](#), 76
[mushroom_rl_benchmark.utils.utils \(module\)](#), 75

N

[NoisyDQNBuilder \(class in mushroom_rl_benchmark.builders.value.dqn.noisy_dqn\)](#), 64

O

[object_to_primitive \(\) \(in module mushroom_rl_benchmark.utils.primitive\)](#), 75

P

[PGPEBuilder \(class in mushroom_rl_benchmark.builders.policy_search.black_box_optimization\)](#), 58
[plot_counter \(BenchmarkSuiteVisualizer attribute\)](#), 54
[plot_counter \(BenchmarkVisualizer attribute\)](#), 53
[plot_mean_conf \(\) \(in module mushroom_rl_benchmark.utils.plot\)](#), 76
[PolicyGradientBuilder \(class in mushroom_rl_benchmark.builders.policy_search.policy_gradient\)](#), 56
[PPOBuilder \(class in mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.ppo\)](#), 68
[print_experiments \(\) \(BenchmarkSuite method\)](#), 48
[print_metrics \(\) \(in module mushroom_rl_benchmark.experiment.run\)](#), 74
[PrioritizedDQNBuilder \(class in mushroom_rl_benchmark.builders.value.dqn.prioritized_dqn\)](#), 64

Q

[QLambdaBuilder \(class in mushroom_rl_benchmark.builders.value.td.td_trace\)](#), 62
[QLearningBuilder \(class in mushroom_rl_benchmark.builders.value.td.td_finite\)](#), 60

R

[read_arguments_aggregate \(\) \(in module mushroom_rl_benchmark.experiment.slurm.arguments\)](#), 74
[read_arguments_run \(\) \(in module mushroom_rl_benchmark.experiment.slurm.arguments\)](#), 74
[REINFORCEBuilder \(class in mushroom_rl_benchmark.builders.policy_search.policy_gradient\)](#), 57
[REPSBuilder \(class in mushroom_rl_benchmark.builders.policy_search.black_box_optimization\)](#), 58
[reset \(\) \(BenchmarkExperiment method\)](#), 49
[resume \(\) \(BenchmarkExperiment method\)](#), 49
[run \(\) \(BenchmarkExperiment method\)](#), 48
[run \(\) \(BenchmarkSuite method\)](#), 48
[run_parallel \(\) \(BenchmarkExperiment method\)](#), 49
[run_sequential \(\) \(BenchmarkExperiment method\)](#), 49
[run_slurm \(\) \(BenchmarkExperiment method\)](#), 49
[RWRBuilder \(class in mushroom_rl_benchmark.builders.policy_search.black_box_optimization\)](#), 58

S

[SACActorNetwork \(class in mushroom_rl_benchmark.builders.network.sac_network\)](#), 72

SACBuilder (class in mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.sac), 69
SACriticNetwork (class in mushroom_rl_benchmark.builders.network.sac_network), 72
SARSABuilder (class in mushroom_rl_benchmark.builders.value.td.td_finite), 60
SARSALambdaBuilder (class in mushroom_rl_benchmark.builders.value.td.td_trace), 62
SarsaLambdaContinuousBuilder (class in mushroom_rl_benchmark.builders.value.td.sarsa_lambda_continuous), 62
save_agent_builder() (BenchmarkLogger method), 52
save_best_agent() (BenchmarkLogger method), 52
save_boxplots() (BenchmarkSuiteVisualizer method), 54
save_builders() (BenchmarkExperiment method), 50
save_config() (BenchmarkLogger method), 52
save_entropy() (BenchmarkLogger method), 51
save_environment_builder() (BenchmarkLogger method), 52
save_figure() (BenchmarkLogger method), 53
save_J() (BenchmarkLogger method), 51
save_last_agent() (BenchmarkLogger method), 52
save_parameters() (BenchmarkSuite method), 48
save_params() (BenchmarkLogger method), 53
save_plot() (BenchmarkExperiment method), 50
save_plots() (BenchmarkSuite method), 48
save_R() (BenchmarkLogger method), 51
save_report() (BenchmarkVisualizer method), 54
save_reports() (BenchmarkSuiteVisualizer method), 54
save_stats() (BenchmarkLogger method), 52
save_V() (BenchmarkLogger method), 51
set_and_save_config() (BenchmarkExperiment method), 50
set_and_save_stats() (BenchmarkExperiment method), 50
set_eval_mode() (AgentBuilder method), 56
set_eval_mode() (COPDAC_QBuilder method), 66
set_eval_mode() (DQNBuilder method), 63
set_eval_mode() (EnvironmentBuilder static method), 55
set_eval_mode() (TDFiniteBuilder method), 60
set_log_dir() (BenchmarkLogger method), 50
set_log_id() (BenchmarkLogger method), 51
set_preprocessors() (AgentBuilder method), 56
show_agent() (BenchmarkVisualizer method), 54
show_plot() (BenchmarkExperiment method), 50
show_plots() (BenchmarkSuite method), 48
show_report() (BenchmarkVisualizer method), 54
show_reports() (BenchmarkSuiteVisualizer method), 55
SpeedyQLearningBuilder (class in mushroom_rl_benchmark.builders.value.td.td_finite), 60
start_timer() (BenchmarkExperiment method), 50
StochasticACBuilder (class in mushroom_rl_benchmark.builders.actor_critic.classic_actor_critic.stochastic_ac), 65
stop_timer() (BenchmarkExperiment method), 50

T

TD3ActorNetwork (class in mushroom_rl_benchmark.builders.network.td3_network), 73
TD3Builder (class in mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.td3), 69
TD3CriticNetwork (class in mushroom_rl_benchmark.builders.network.td3_network), 72
TDFiniteBuilder (class in mushroom_rl_benchmark.builders.value.td.td_finite), 59
TDTraceBuilder (class in mushroom_rl_benchmark.builders.value.td.td_trace), 61
to_duration() (in module mushroom_rl_benchmark.experiment.slurm.slurm_script), 75
TRPOBuilder (class in mushroom_rl_benchmark.builders.actor_critic.deep_actor_critic.trpo), 70
TRPONetwork (class in mushroom_rl_benchmark.builders.network.trpo_network), 73
TrueOnlineSarsaLambdaBuilder (class in mushroom_rl_benchmark.builders.value.td.true_online_sarsa_lambda), 62

W

WeightedQLearningBuilder (class in mushroom_rl_benchmark.builders.value.td.td_finite), 61