
# BenchMARL: Benchmarking Multi-Agent Reinforcement Learning

**Matteo Bettini**
PhD student at University of Cambridge, ex PyTorch Intern

Amanda Prorok
University Of Cambridge

Vincent Moens
PyTorch

PyTorch

Meta

PROROKLAB

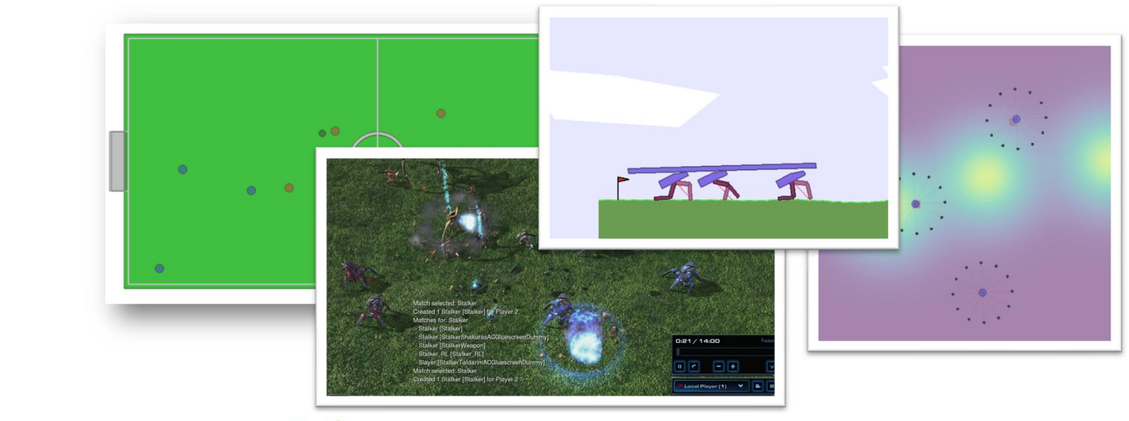
UNIVERSITY OF CAMBRIDGE

# What is BenchMARL 🧐?

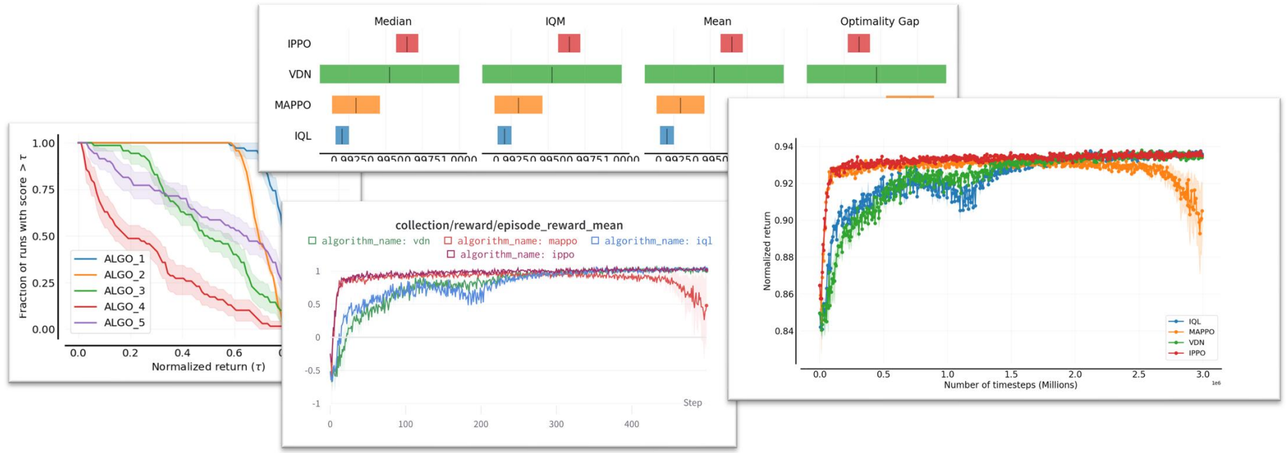
BenchMARL is a Multi-Agent Reinforcement Learning (MARL) training library created to enable reproducibility and benchmarking across different MARL algorithms and environments.

It is founded on its **core tenets**:

- **Reproducibility** through systematic grounding and standardization of configuration.
- **Statistically-Strong Evaluations**: BenchMARL provides standardized and statistically-strong plotting and reporting.
- **Experiment Independence**: our library is designed to be independent of algorithm, environment, and model choices.
- **Breadth Over the MARL Ecosystem**: compatible with different types of algorithms, tasks, and models
- **Easy Integration**: Implement new algorithms, environments, and models with ease. Focus on your research, not the setup.
- **TorchRL Backbone**: Leveraging TorchRL, we provide high performance and state-of-the-art implementations.



BenchMARL



BENCHMARL

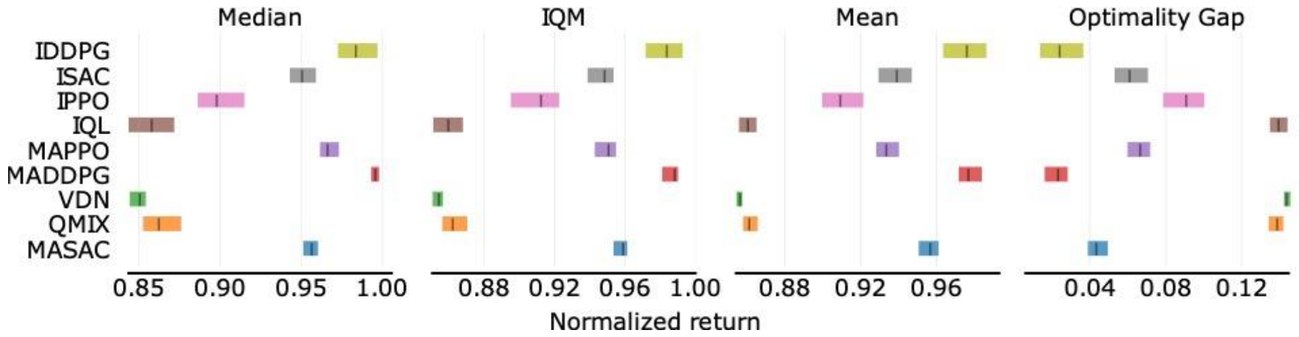
# Why does it exist?

Within machine learning domains, reinforcement learning has always been more fragmented with respect to shared community standards.

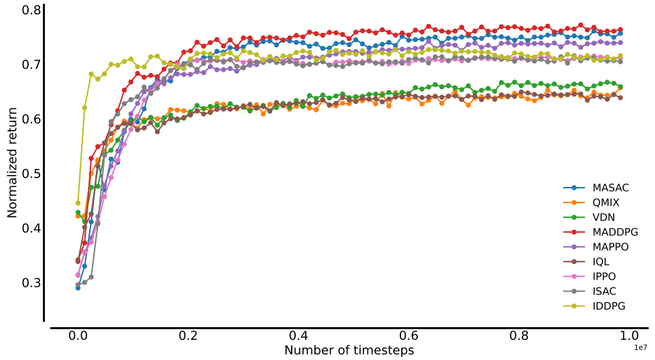
This led to a reproducibility crisis highlighted in recent NeurIPS papers [1, 2] which introduce a standardized set of tools for reporting.

BenchMARL provides a training library that uses these tools for unifying benchmarking in MARL.

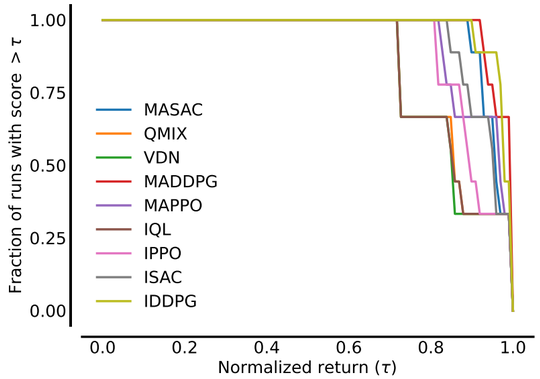
Aggregate scores




Sample efficiency curves




Performance profile





[1] Gorsane, Rihab, et al. "Towards a standardised performance evaluation protocol for cooperative marl." Advances in Neural Information Processing Systems 35 (2022): 5510-5521.

[2] Agarwal, Rishabh, et al. "Deep reinforcement learning at the edge of the statistical precipice." Advances in neural information processing systems 34 (2021): 29304-29320

BENCHMARL

# Components

## Experiment

```
python benchmarl/run.py algorithm=mappo task=vmas/balance
```

## Benchmark

```
python benchmarl/run.py -m algorithm=mappo,qmix,masac task=vmas/balance,vmas/sampling seed=0,1
```

## Algorithms

| Algorithm | On/Off policy | Actor-critic | Full-observability in critic | Action compatibility | Probabilistic actor |
|---|---|---|---|---|---|
| `Mappo` | On | Yes | Yes | Continuous + Discrete | Yes |
| `Ippo` | On | Yes | No | Continuous + Discrete | Yes |
| `Maddpg` | Off | Yes | Yes | Continuous | No |
| `Iddpg` | Off | Yes | No | Continuous | No |
| `Masac` | Off | Yes | Yes | Continuous + Discrete | Yes |
| `Isac` | Off | Yes | No | Continuous + Discrete | Yes |
| `Qmix` | Off | No | NA | Discrete | No |
| `Vdn` | Off | No | NA | Discrete | No |
| `Iql` | Off | No | NA | Discrete | No |

## Models

| Name | Decentralized | Centralized with local inputs | Centralized with global input |
|---|---|---|---|
| `Mlp` | Yes | Yes | Yes |
| `Gru` | Yes | Yes | Yes |
| `Lstm` | Yes | Yes | Yes |
| `Gnn` | Yes | Yes | No |
| `Cnn` | Yes | Yes | Yes |
| `Deepsets` | Yes | Yes | Yes |

## Tasks

| Environment | Tasks | Cooperation | Global state | Reward function | Action space | Vectorized |
|---|---|---|---|---|---|---|
| `VmasTask` | 27 | Cooperative + Competitive | No | Shared + Independent + Global | Continuous + Discrete | Yes |
| `Smacv2Task` | 15 | Cooperative | Yes | Global | Discrete | No |
| `PettingZooTask` | 10 | Cooperative + Competitive | Yes + No | Shared + Independent | Continuous + Discrete | No |
| `MeltingPotTask` | 49 | Cooperative + Competitive | Yes | Independent | Discrete | No |

# Example

BenchMARL

**INPUT**

*Run a benchmark with 3 experiments*

```
python benchmarl/run.py -m
seed=0
task=vmas/balance
model=layers/mlp
algorithm=ippo,qmix,maddpg
```

YAML ExperimentConfig

YAML Algorithm

YAML Task

YAML Model

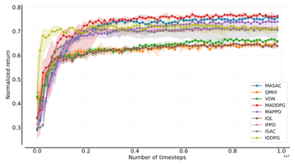
Experiment (ippo)

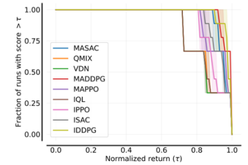
Experiment (qmix)

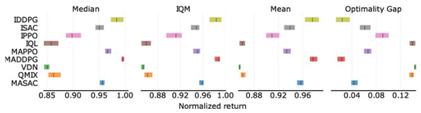
Experiment (maddpg)

Benchmark

**OUTPUT**



# Configuring with hydra

## Configuration tree

```
conf
├── algorithm
├── experiment
├── model
│   └── layers
└── task
    ├── pettingzoo
    ├── smacv2
    └── vmas
```

## Configuration loading

masac.yaml

```yaml
defaults:
  - masac_config
  - _self_

share_param_critic: True
num_qvalue_nets: 2
loss_function: "l2"
delay_qvalue: True
target_entropy: "auto"
discrete_target_entropy_weight: 0.2
alpha_init: 1.0
min_alpha: null
max_alpha: null
fixed_alpha: False
scale_mapping: "biased_softplus_1.0"
```

Load and validate using dataclass

masac.py

```python
@dataclass
class MasacConfig(AlgorithmConfig):
    share_param_critic: bool = MISSING
    num_qvalue_nets: int = MISSING
    loss_function: str = MISSING
    delay_qvalue: bool = MISSING
    target_entropy: Union[float, str] = MISSING
    discrete_target_entropy_weight: float = MISSING
    alpha_init: float = MISSING
    min_alpha: Optional[float] = MISSING
    max_alpha: Optional[float] = MISSING
    fixed_alpha: bool = MISSING
    scale_mapping: str = MISSING
```

## Overriding

### Experiment

```
python benchmarl/run.py task=vmas/balance algorithm=mappo experiment.lr=0.03 experiment.evaluation=true experiment.train_device="cpu"
```

### Algorithm

```
python benchmarl/run.py task=vmas/balance algorithm=masac algorithm.num_qvalue_nets=3 algorithm.target_entropy=auto algorithm.share_param_critic=true
```

### Model

```
python benchmarl/run.py task=vmas/balance algorithm=mappo model=layers/mlp model=layers/mlp model.layer_class="torch.nn.Linear" "model.num_cells=[32,32]" model.activation_class="torch.nn.ReLU"
```

### Task

```
python benchmarl/run.py task=vmas/balance algorithm=mappo task.n_agents=4
```

BENCHMARL

# Reporting with marl-eval

marl-eval

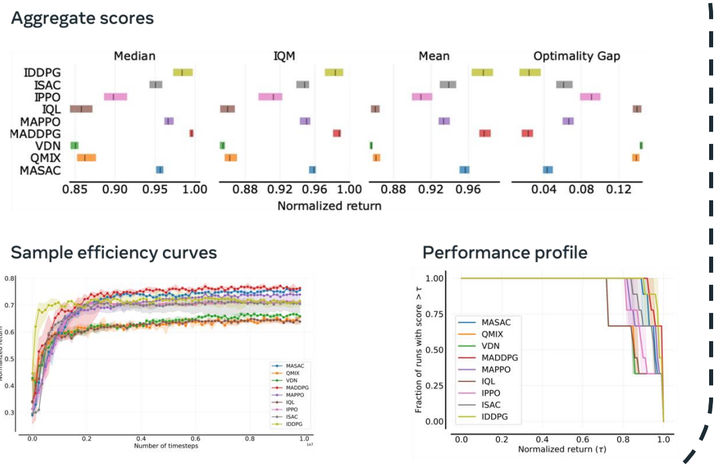
## INPUT

```
python benchmarl/run.py -m
seed=0
task=vmas/balance
algorithm=ippo,qmix,maddpg
experiment.create_json=True
```

## MARL-EVAL JSON

```
{
 "environment_name" : {
  "task_name" : {
   "algorithm_name": {
    ”seed_0": {
     "step_1" : {
      "step_count": <int>,
       "return": [<number_evaluation_episodes>],
       ”group_return": [<number_evaluation_episodes>]
}}}}}}
```

## OUTPUT

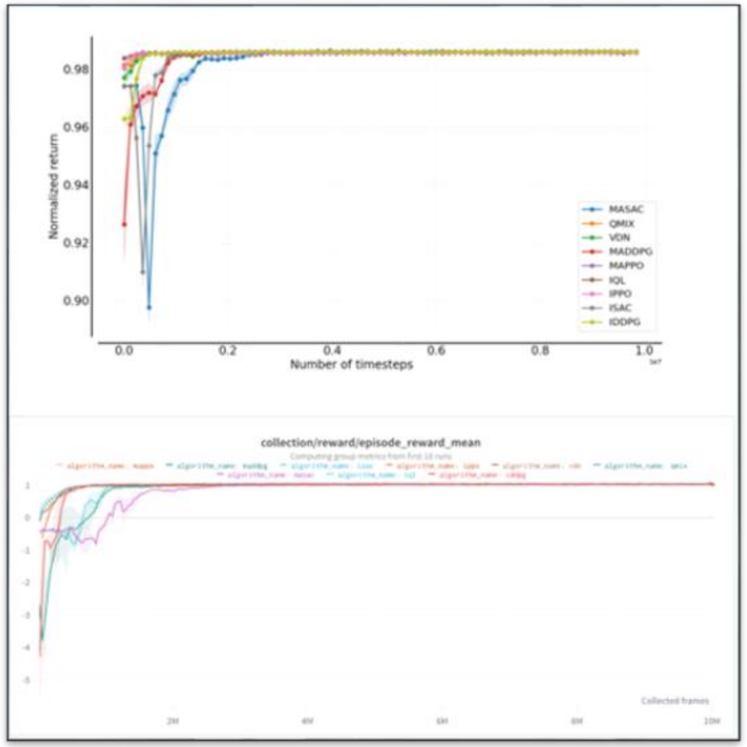



[1] Gorsane, Rihab, et al. "Towards a standardised performance evaluation protocol for cooperative marl." Advances in Neural Information Processing Systems 35 (2022): 5510-5521.

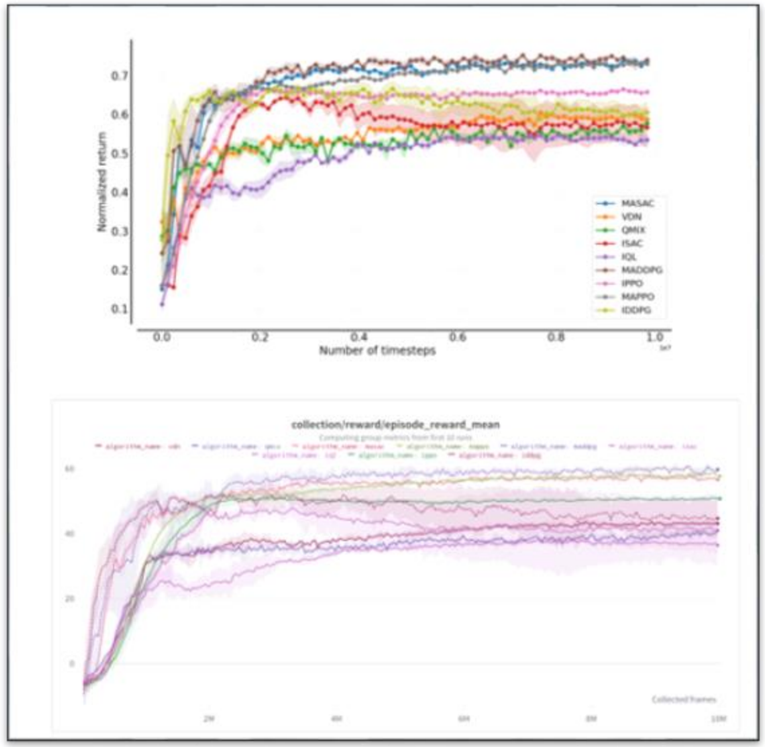
# Public benchmarks

**Tuned public benchmarks:** we tune and publish hyperparameters and plots for environments

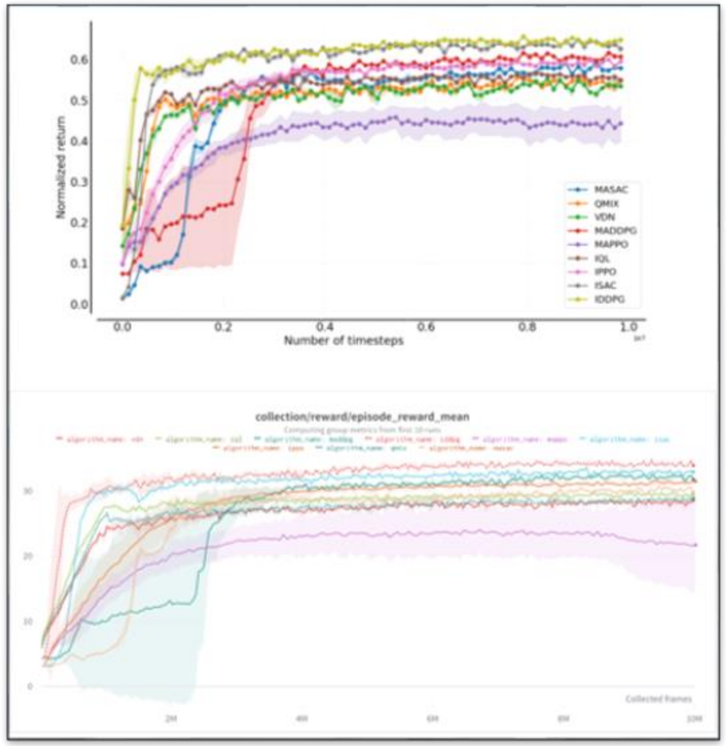
VMAS
Navigation




VMAS Balance




VMAS
Sampling



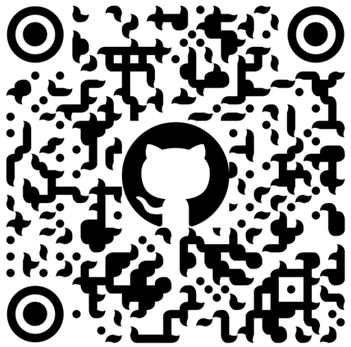
# Features

- **Logging:** wandb, csv, json, mflow, tensorboard
- **Checkpointing:** restore experiments from any state (compatible with wandb resume)
- **Callbacks:** customize your experiment with callbacks
- **Tests:** integration tests with coverage run the whole training pipeline in the CI for all tasks and all algos
- **TorchRL backend:** maintained by PyTorch org

# Thanks for your attention

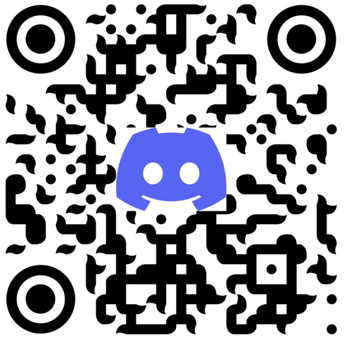
To get involved:

- Reach out
- Join discussions on GitHub
- Any contribution is always welcome!



GitHub

Documentation



Discord

Matteo Bettini
University Of Cambridge, ex PyTorch Intern

Special thanks to:

Vincent Moens
PyTorch

Amanda Prorok
University Of Cambridge