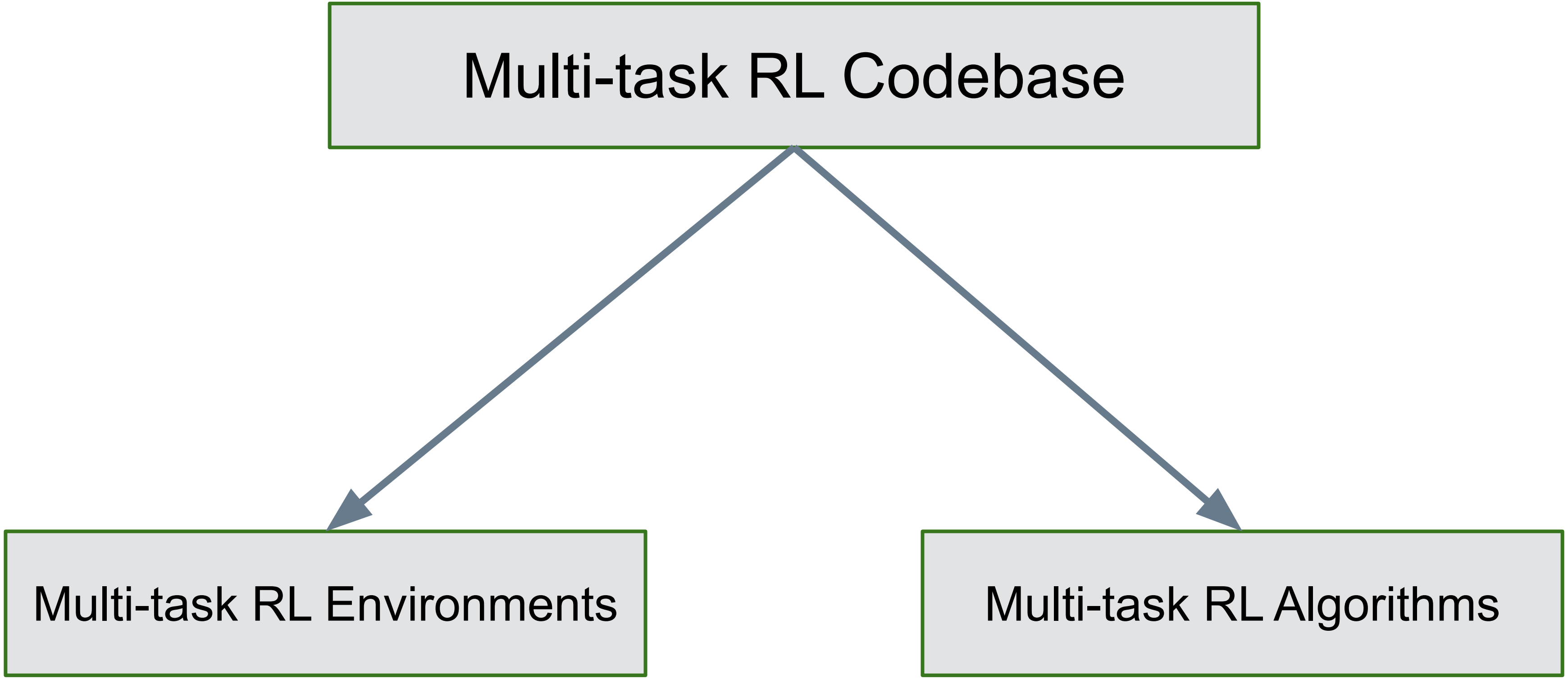
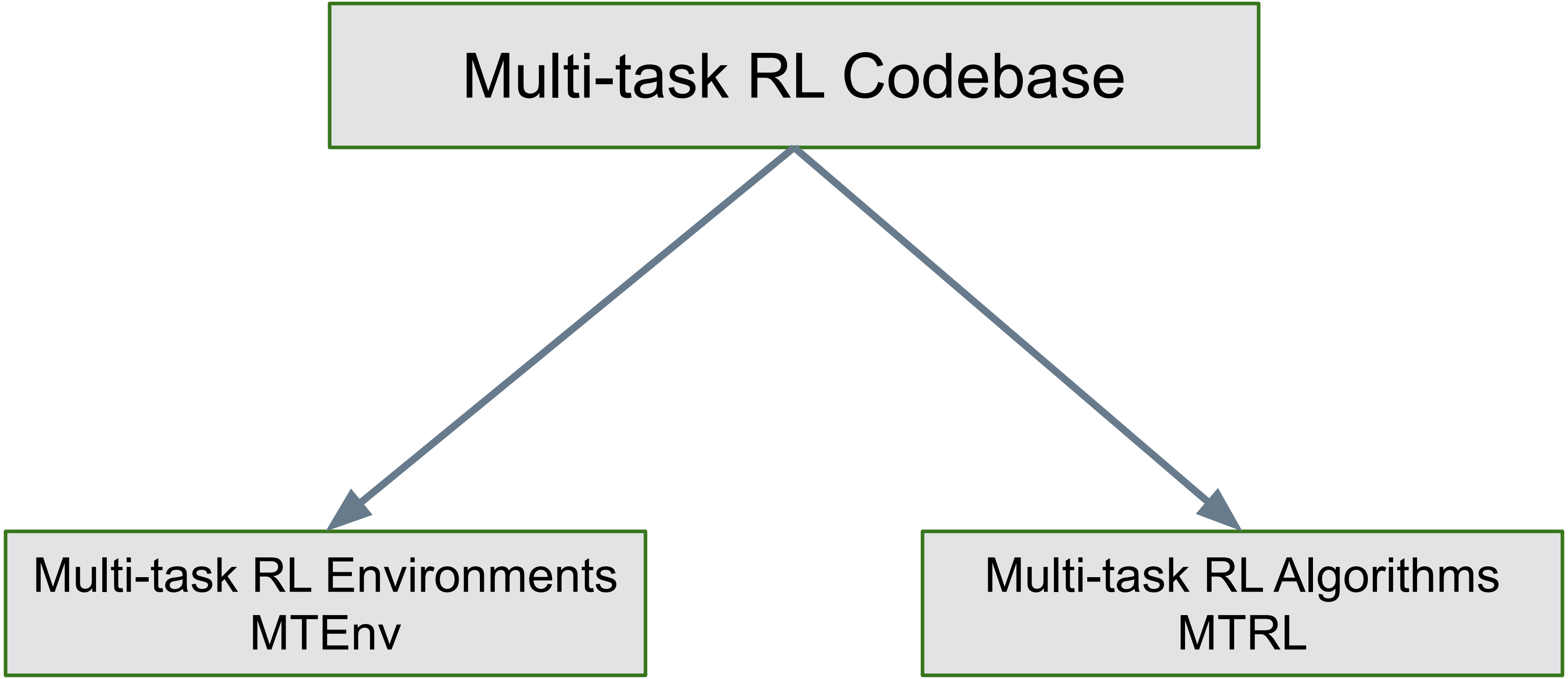
Environments and Baselines for Multi-task RL

Shagun Sodhani

Facebook AI Research

FACEBOOK AI


Multi-task RL Codebase
Multi-task RL Environments
Multi-task RL Algorithms


Multi-task RL Codebase
Multi-task RL Environments
MTEnv
Multi-task RL Algorithms
MTRL

MTEnv

MTEnv: Goals

1. Standardize Multitask RL Environments

2. Provide Better Benchmarks

MTEnv: Standardize Multitask RL Environments

1. OpenAI Gym [1] provides the most popular interface for RL environments.

[1]: https://gym.openai.com/

MTEnv: Standardize Multitask RL Environments

1. OpenAI Gym [1] provides the most popular interface for RL environments.

```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000):
  action = policy.get_action(observation)
  observation, reward, done, info = env.step(action)
  if done:
    observation = env.reset()
env.close()
```


[1]: https://gym.openai.com/

MTEnv: Standardize Multitask RL Environments

1. OpenAI Gym [1] provides the most popular interface for RL environments.

```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000):
  action = policy.get_action(observation)
  observation, reward, done, info = env.step(action)
  if done:
    observation = env.reset()
env.close()
```


[1]: https://gym.openai.com/

MTEnv: Standardize Multitask RL Environments

1. OpenAI Gym [1] provides the most popular interface for RL environments.

```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000):
  action = policy.get_action(observation)
  observation, reward, done, info = env.step(action)
  if done:
    observation = env.reset()
env.close()
```


[1]: https://gym.openai.com/

MTEnv: Standardize Multitask RL Environments

1. OpenAI Gym [1] provides the most popular interface for RL environments.

```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000):
  action = policy.get_action(observation)
  observation, reward, done, info = env.step(action)
  if done:
    observation = env.reset()
env.close()
```


[1]: https://gym.openai.com/

MTEnv: Standardize Multitask RL Environments

1. OpenAI Gym [1] provides the most popular interface for RL environments.

```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000):
  action = policy.get_action(observation)
  observation, reward, done, info = env.step(action)
  if done:
    observation = env.reset()
env.close()
```


[1]: https://gym.openai.com/

MTEnv: Standardize Multitask RL Environments

1. OpenAI Gym [1] provides the most popular interface for RL environments.

```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000):
  action = policy.get_action(observation)
  observation, reward, done, info = env.step(action)
  if done:
    observation = env.reset()
env.close()
```


[1]: https://gym.openai.com/

MTEnv: Standardize Multitask RL Environments

1. OpenAI Gym [1] provides the most popular interface for RL environments.

```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000):
  action = policy.get_action(observation)
  observation, reward, done, info = env.step(action)
  if done:
    observation = env.reset()
env.close()
```


[1]: https://gym.openai.com/

MTEnv: Standardize Multitask RL Environments

1. OpenAI Gym [1] provides the most popular interface for RL environments.

```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000):
  action = policy.get_action(observation)
  observation, reward, done, info = env.step(action)
  if done:
    observation = env.reset()
env.close()
```


[1]: https://gym.openai.com/

MTEnv: Standardize Multitask RL Environments

1. OpenAI Gym [1] provides the most popular interface for RL environments.

2. The biggest benefit of the interface is the stanadarization.

[1]: https://gym.openai.com/

MTEnv: Standardize Multitask RL Environments

1. OpenAI Gym [1] provides the most popular interface for RL environments.

2. The biggest benefit of the interface is the stanadarization.

3. It is very easy to swap the agent, without changing the environment.

[1]: https://gym.openai.com/

MTEnv: Standardize Multitask RL Environments

1. OpenAI Gym [1] provides the most popular interface for RL environments.

2. However, the standardization is lost when used for multi-task RL.

[1]: https://gym.openai.com/

MTEnv: Standardize Multitask RL Environments

1. OpenAI Gym [1] provides the most popular interface for RL environments.

2. However, the standardization is lost when used for multi-task RL.

3. Gym is not designed to control the task state; unlike environment observation, task information is not a first-class citizen.

[1]: https://gym.openai.com/

MTEnv: Standardize Multitask RL Environments

1. OpenAI Gym [1] provides the most popular interface for RL environments.

2. However, the standardization is lost when used for multi-task RL.

3. Gym is not designed to control the task state; unlike environment observation, task information is not a first-class citizen.

4. Multiple ways of supporting the task information.

[1]: https://gym.openai.com/

MTEnv: Standardize Multitask RL Environments

1. Multiple ways of supporting the task information.

2. This precisely is the problem - we lost standardization.

MTEnv: Standardize Multitask RL Environments

1. Multiple ways of supporting the task information.

2. This precisely is the problem - we lost standardization.

3. This adds overhead for using other people environments.

MTEnv: Standardize Multitask RL Environments

We extend the Gym API to support multiple tasks, with two guiding principles:

MTEnv: Standardize Multitask RL Environments

We extend the Gym API to support multiple tasks, with two guiding principles:

1. Make minimal changes to the Gym Interface (which the community is very familiar with).

MTEnv: Standardize Multitask RL Environments

We extend the Gym API to support multiple tasks, with two guiding principles:

1. Make minimal changes to the Gym Interface (which the community is very familiar with).

2. Make it easy to port existing environments to MTEnv.

MTEnv: Standardize Multitask RL Environments

1. Gym returns an unstructured observation.

MTEnv: Standardize Multitask RL Environments

1. Gym returns an unstructured observation.

2. MTEnv returns a structured observation.
   a. A dictionary with two keys - env_obs and task_obs

MTEnv: Standardize Multitask RL Environments

1. Gym returns an unstructured observation.

2. MTEnv returns a structured observation.
   a. A dictionary with two keys - env_obs and task_obs

3. Minimal change to the environment interface.

MTEnv: Standardize Multitask RL Environments

1. Task Information is a first class citizen.
    a. Separate seed, observation space, etc
    b. Easy to control task

MTEnv: Standardize Multitask RL Environments

1. Task Information is a first class citizen.
    a. Separate seed, observation space, etc
    b. Easy to control task
2. Easy to port existing environments to MTEnv
    a. Provide several wrappers to help with that.

MTEnv: Supported Environments

1. Acrobot
2. Cartpole
3. DeepMind Control Suite
4. Meta-World
5. Multi-armed Bandit
6. Tabular MDP
7. Two-Goal Maze

MTEnv: Future Steps

1. Add new envs
2. Release new benchmark
3. Extending the library
    a. Dealing with variable action and observation spaces changes

MTEnv: How to play with it

1. PyPI: mtenv

2. GitHub: https://github.com/facebookresearch/mtenv

MTRL

MTRL: Components

MTRL has two components

1. Base Policy

2. **Plug and play components** to make the policy work on multi-task setup

MTRL: Base Policy

1. Actor
2. Critic
3. Encoder
4. Decoder
5. Transition model
6. Reward model

MTRL: Base Policy

1. SAC
2. SAC-AE
3. DeepMDP

MTRL: Components for Multi-task RL

1. Task/Context Encoder

2. Gradient Manipulation Algorithms

3. Task specific components and selection mechanism - eg Multi-headed policies

4. Centralized policy - Distral

MTRL: Plug and play

Base Components

Actor
Critic
Encoder
Decoder

Multitask RL Components

Multi-head policy

MTRL: Plug and play

Base Components

Actor
Critic
Encoder
Decoder

Multitask RL Components

Multi-head policy

PYTHONPATH=. python3 -u main.py

*setup=metaworld env=metaworld-mt10 *

*agent=state_sac *

agent.multitask.should_use_multi_head_policy=True

MTRL: Plug and play

Base Components

Actor
Critic
Encoder
Decoder

Multitask RL Components

Multi-head policy

PYTHONPATH=. python3 -u main.py

*setup=metaworld env=metaworld-mt10 *

*agent=state_sac *

agent.multitask.should_use_multi_head_policy=True

MTRL: Plug and play

Base Components

Actor
Critic
Encoder
Decoder

Multitask RL Components

Multi-head policy

PYTHONPATH=. python3 -u main.py

*setup=metaworld env=metaworld-mt10 *

*agent=state_sac *

agent.multitask.should_use_multi_head_policy=True

MTRL: Plug and play

Base Components

Actor
Critic
Encoder
Decoder

Multitask RL Components

Multi-head policy

```
PYTHONPATH=. python3 -u main.py
setup=metaworld env=metaworld-mt10 \
agent=state_sac \
agent.multitask.should_use_multi_head_policy=True
```

MTRL: Plug and play

Base Components

Actor
Critic
Encoder
Decoder

Multitask RL Components

Use Mixture-of-encoders

MTRL: Plug and play

Base Components

Actor
Critic
Encoder
Decoder

Multitask RL Components

Use Mixture-of-encoders

PYTHONPATH=. python3 -u main.py

*setup=metaworld env=metaworld-mt10 *

*agent=state_sac *

agent.encoder.type_to_select=moe

MTRL: Plug and play

Base Components

Actor
Critic
Encoder
Decoder

Multitask RL Components

Use Mixture-of-encoders

PYTHONPATH=. python3 -u main.py

*setup=metaworld env=metaworld-mt10 *

*agent=state_sac *

agent.encoder.type_to_select=moe

MTRL: Plug and play

Base Components

Actor
Critic
Encoder
Decoder

Multitask RL Components

Use Mixture-of-encoders (4 encoders)

MTRL: Plug and play

Base Components

Actor
Critic
Encoder
Decoder

PYTHONPATH=. python3 -u main.py

*setup=metaworld env=metaworld-mt10 *

*agent=state_sac *

*agent.encoder.type_to_select=moe *

agent.encoder.moe.num_experts=4

Multitask RL Components

Use Mixture-of-encoders (4 encoders)

MTRL: Plug and play

Base Components

Actor
Critic
Encoder
Decoder

Multitask RL Components

Use Mixture-of-encoders

PYTHONPATH=. python3 -u main.py

*setup=metaworld env=metaworld-mt50 *

*agent=state_sac *

agent.encoder.type_to_select=moe

MTRL: How to play with it

1. GitHub: https://github.com/facebookresearch/mtrl

[1]: https://gym.openai.com/

MTRL: Future Steps

1. Add more base policies and components
    a. PPO, IMPALA etc
    b. Context Aware Dynamics Model
    c. HyperNetworks
    d. Trajectory based context encoders

[1]: https://gym.openai.com/

MTRL: Future Steps

1. Scaling and Ease of Use
    a. Memory-efficient replay buffers
    b. Scaling policy components
    c. Add examples of complex training pipelines
    d. Pre-trained models and weights

[1]: https://gym.openai.com/

References

[1]: https://gym.openai.com/

[1]: https://gym.openai.com/

Thank You

FACEBOOK AI