



Welcome to:

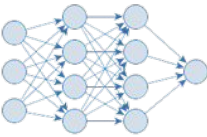
# Hands-on RL with Ray's RLlib

A beginner's tutorial for working with environments, models, and algorithms



policy (in deep-RL, this is a neural network)

multi-GPU

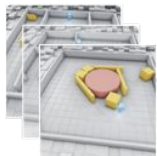


agents

tf.keras /  
torch.nn.Modules

built-in  
LSTM + attention nets

observations



arbitrary  
obs spaces

multi-agent

rewards

+0.3  
-1.0

reward shaping  
/ learning w/o  
rewards

external environments



environment

actions



15+  
available  
algorithms  
(model-free;  
model-based;  
offline RL)

arbitrary  
act. spaces

custom callbacks

parallelize and  
distribute



# Key differences: RL vs Supervised Learning

- Data collection loop is essential part of the RL algorithm, especially in the distributed setting.
- RL is forced to use unstable (bootstrapped) loss functions due to missing labels (rewards cannot cover absence of SL-style labels).

$$\mathbb{E}_{(s,a,r,s')} \left[ \frac{1}{2} \left( \underbrace{R(s, a, s') + \gamma \max_{a'} Q(s', a'; \theta)}_{\text{"labels" (bootstrapped using reward AND network output)}} - \underbrace{Q(s, a; \theta)}_{\text{predictions (actual network output)}} \right)^2 \right]$$

- Exploration vs exploitation. An (online) RL algo can try new things in the environment.
- By default, RL tries to optimize over a time axis.



# Overview of RLlib's Industry Users



**WILD  
LIFE**



Microsoft

 **pathmind**

GENERAL MOTORS



**UBISOFT**



TWO SIGMA



**ERICSSON** 





# Many Great Reasons for Using RLlib

- RLlib is based on Ray, so it benefits from all its improvements on **performance** and **scalability**.
- RLlib is backed by Anyscale, the fast-growing, well-funded company behind Ray, offering strong **OSS support**.
- RLlib is extremely **flexible**, allowing you to **customize every aspect** of the RL cycle and workflows.
- RLlib is good at solving **real-world** problems, supporting **offline RL**, **multi-agent** setups, **external simulators**, and more.



# What's a Space?

discrete

$n=3$

our action space

1

.5

multi discrete

$nvec=(3, 2, 8)$

our obs. space

1 0 5

vector

$shape=(4,)$

.1

.5

-2

.9

matrix

$shape=(4, 3)$

.1

0

.8

.1

.4

-1

.2

.2

0

8

.1

0

tensor

$shape=(4, 3, 3)$

1<sup>4</sup>

0<sup>3</sup>

8<sup>5</sup>

1<sup>0</sup>

4<sup>2</sup>

4<sup>1</sup>

2<sup>2</sup>

2<sup>4</sup>

0<sup>0</sup>

8<sup>1</sup>

1<sup>7</sup>

0<sup>9</sup>

Tuple space

$shape=((4,), (3, 3, 2), (4, 3, 2))$

Dict space

$shape=((3, 3, 2), (3,))$

"key Z":

"key A":



# A Closer Look at RLLib

```
train()
_evaluate()
save()
restore()
```

class **Trainer**(tune.Trainable)

WorkerSet

“local worker”

class **RolloutWorker**

Policy Map

Pol1

Mo  
del

Pol2

Mo  
del

@ray.remote

class **RolloutWorker**

@ray.remote

class **RolloutWorker**

@ray.remote

class **RolloutWorker**

Scalability (e.g.  
num\_workers=100)

🧠 distr. exec.  
plan (write  
your own  
custom algo)

- tf.keras.Model or torch.nn.Module
- RLLib default models
- custom models
- auto LSTM wrapping
- auto attention wrapping

Policy Map

Pol1

Model

Pol2

Model

Sampler

Vector Env

Ag1

Ag2

Offline Reader

Input File(s)

Bla blab bl abla  
bla blabla bal  
blab blab lbalb  
lb lba bal

sample()

\_\_call\_\_()

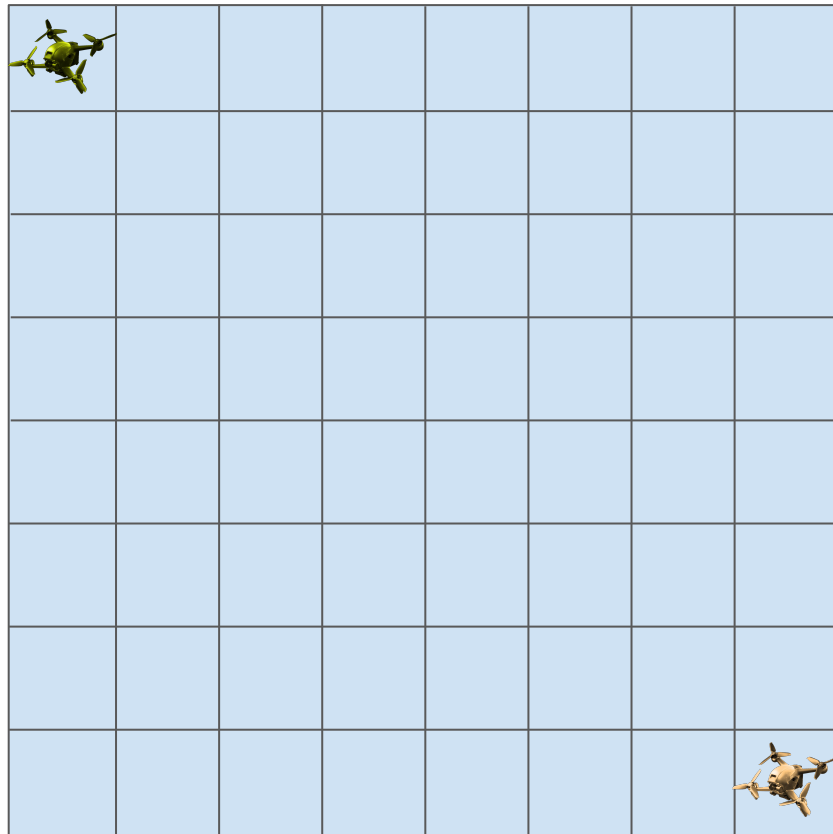
compute\_actions()



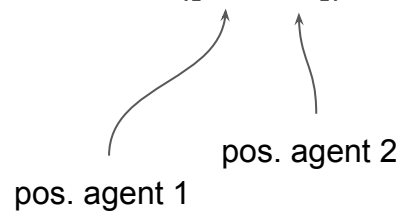
# Our Environment - The “MultiAgentArena” 🤖

action space: Discrete(4)

- 0 - up
- 1 - right
- 2 - down
- 3 - left



observation space:  
MultiDiscrete([64, 64])



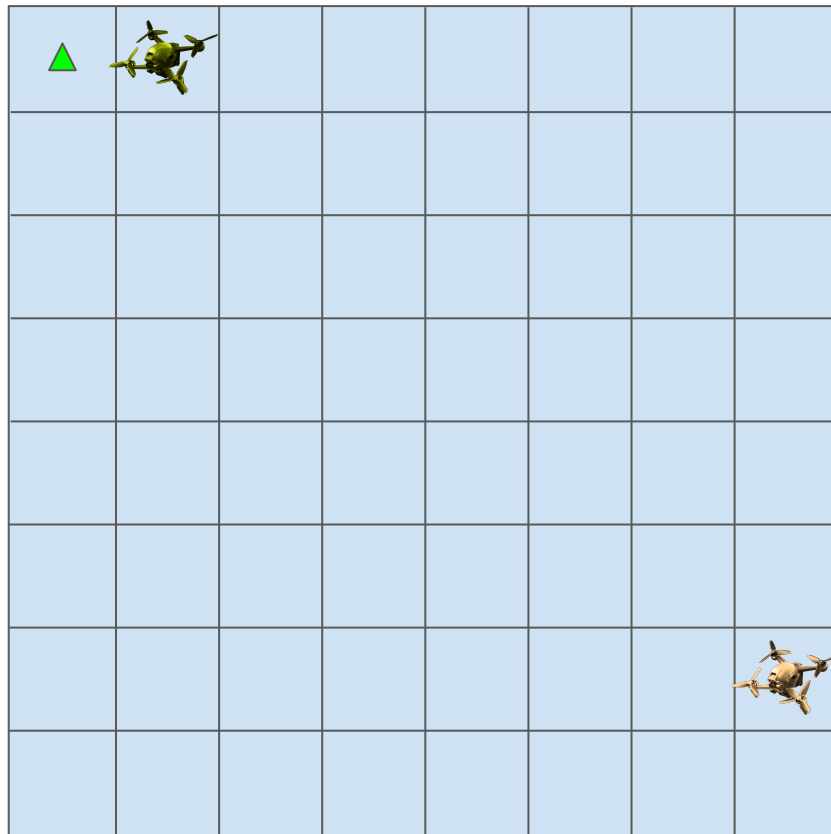




# Our Environment - The “MultiAgentArena” 🤖

action space: Discrete(4)

- 0 - up
- 1 - right
- 2 - down
- 3 - left



observation space:  
MultiDiscrete([64, 64])

pos. agent 1

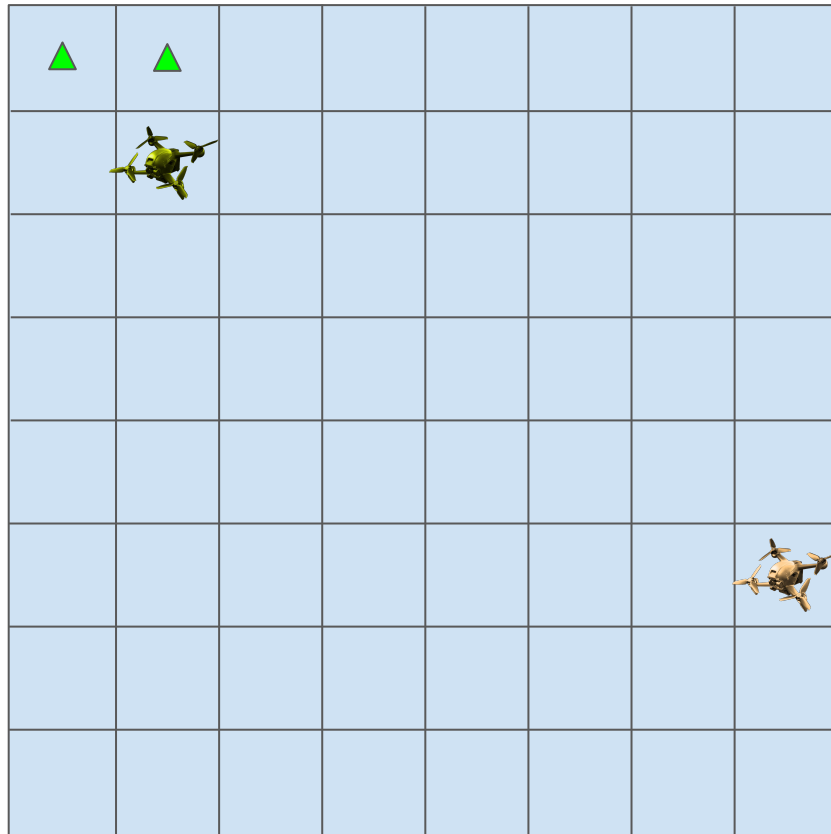
pos. agent 2



# Our Environment - The “MultiAgentArena” 🤖

action space: Discrete(4)

- 0 - up
- 1 - right
- 2 - down
- 3 - left



observation space:  
MultiDiscrete([64, 64])

pos. agent 1

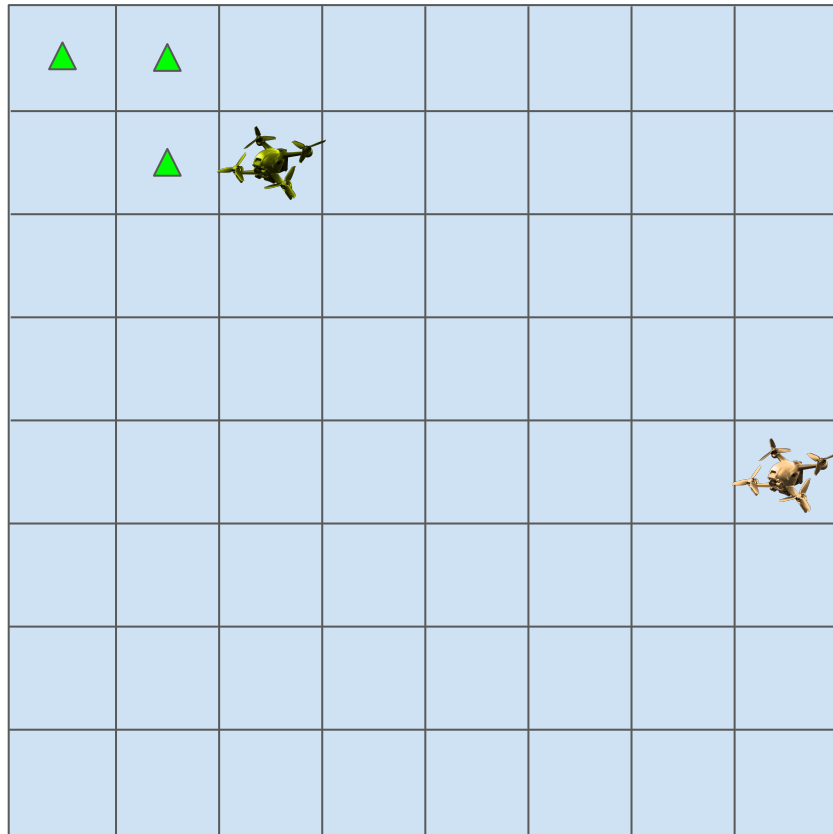
pos. agent 2



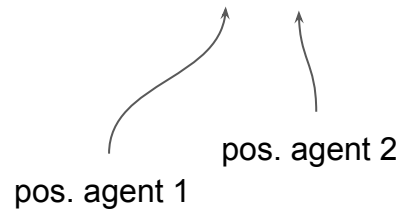
# Our Environment - The “MultiAgentArena” 🤖

action space: Discrete(4)

- 0 - up
- 1 - right
- 2 - down
- 3 - left



observation space:  
MultiDiscrete([64, 64])

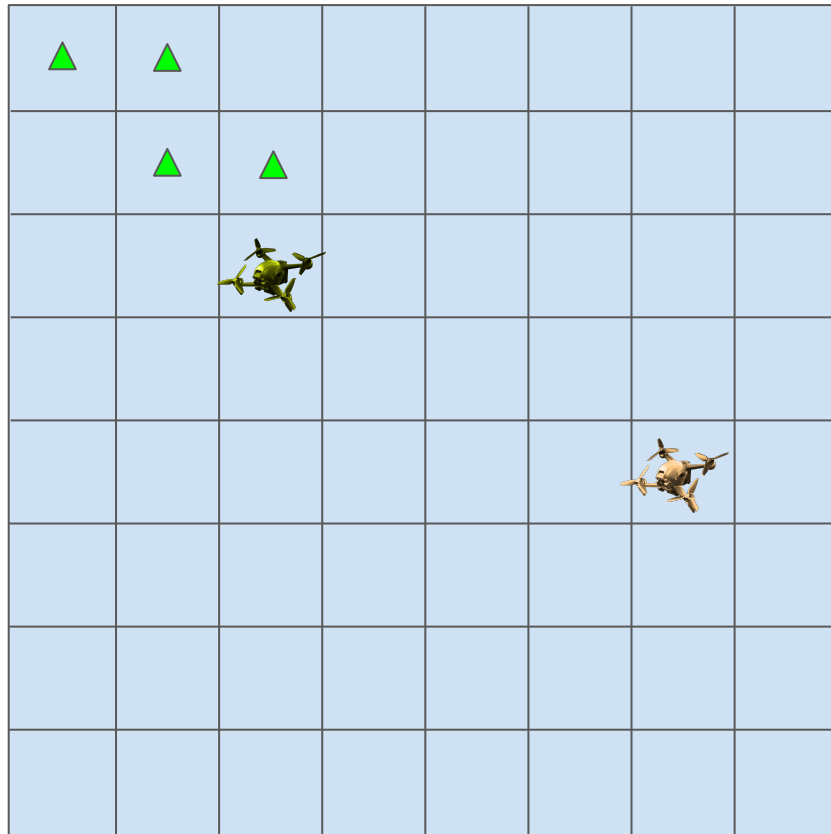




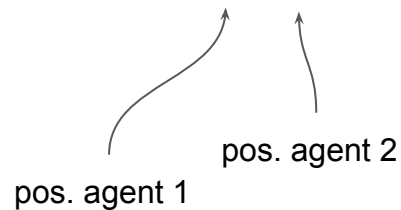
# Our Environment - The “MultiAgentArena” 🤖

action space: Discrete(4)

- 0 - up
- 1 - right
- 2 - down
- 3 - left



observation space:  
MultiDiscrete([64, 64])

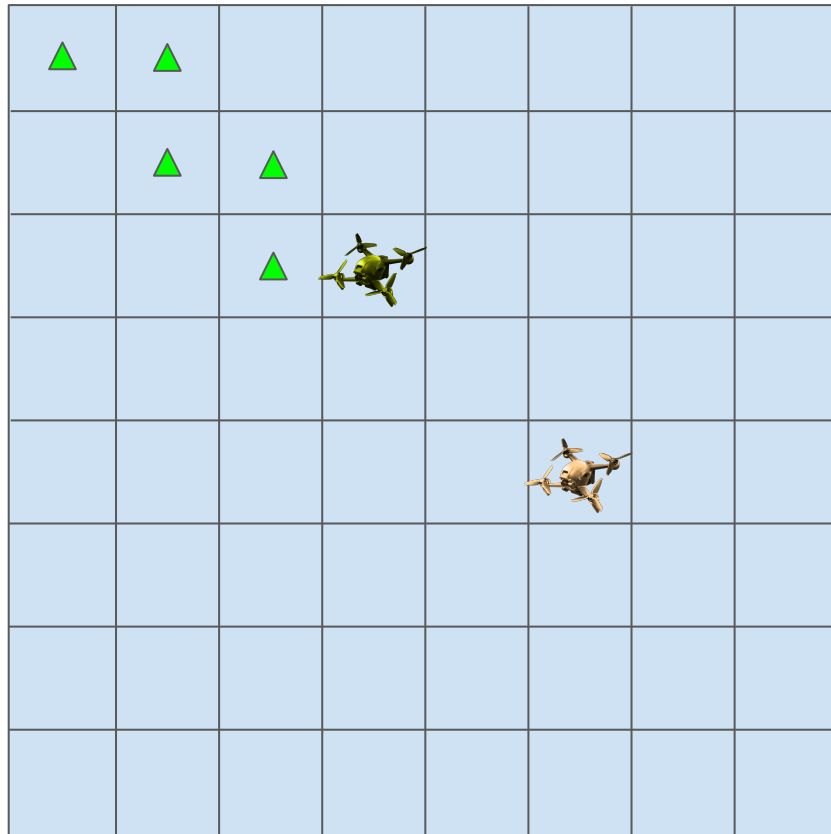




# Our Environment - The “MultiAgentArena” 🤖

action space: Discrete(4)

- 0 - up
- 1 - right
- 2 - down
- 3 - left



observation space:  
MultiDiscrete([64, 64])

pos. agent 1

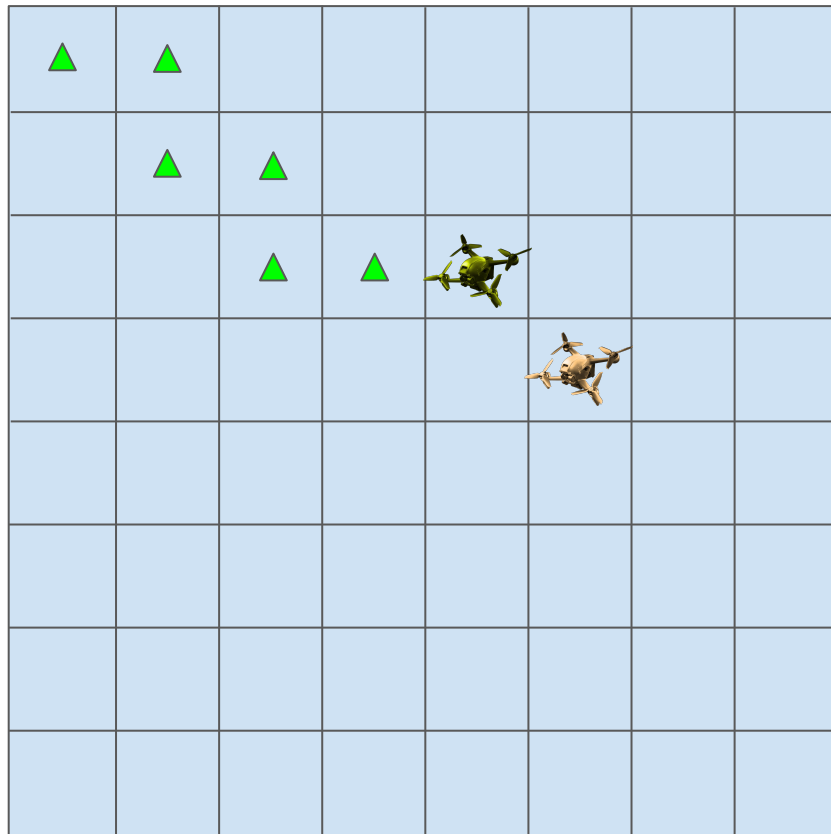
pos. agent 2



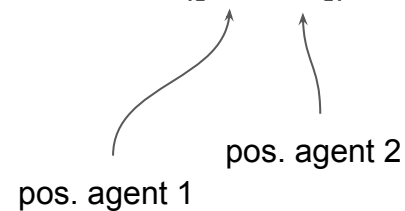
# Our Environment - The “MultiAgentArena” 🤖

action space: Discrete(4)

- 0 - up
- 1 - right
- 2 - down
- 3 - left



observation space:  
MultiDiscrete([64, 64])

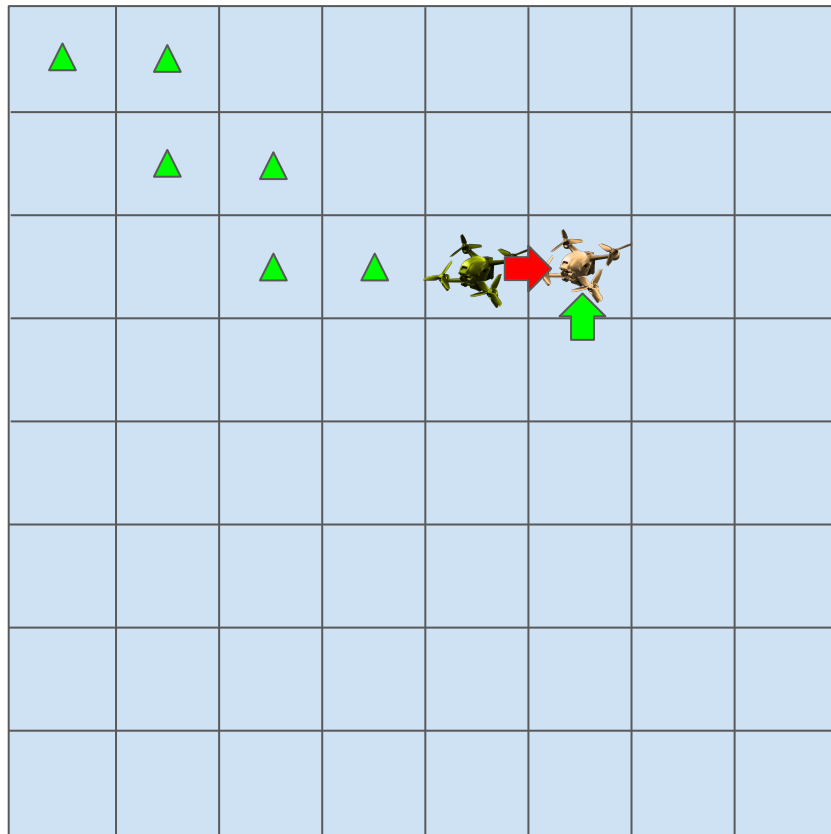




# Our Environment - The “MultiAgentArena” 🤖

action space: Discrete(4)

- 0 - up
- 1 - right
- 2 - down
- 3 - left



observation space:  
MultiDiscrete([64, 64])

