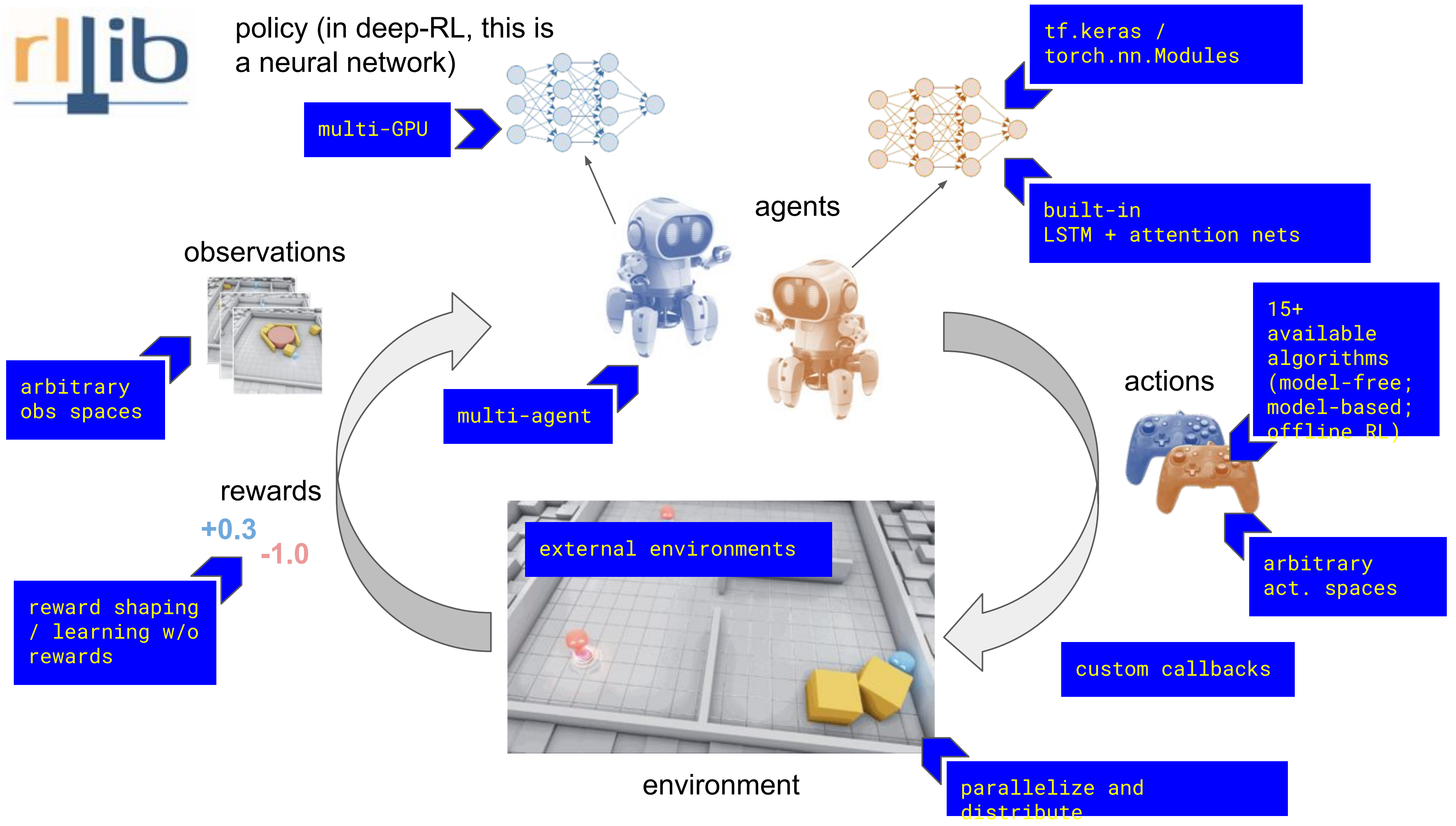
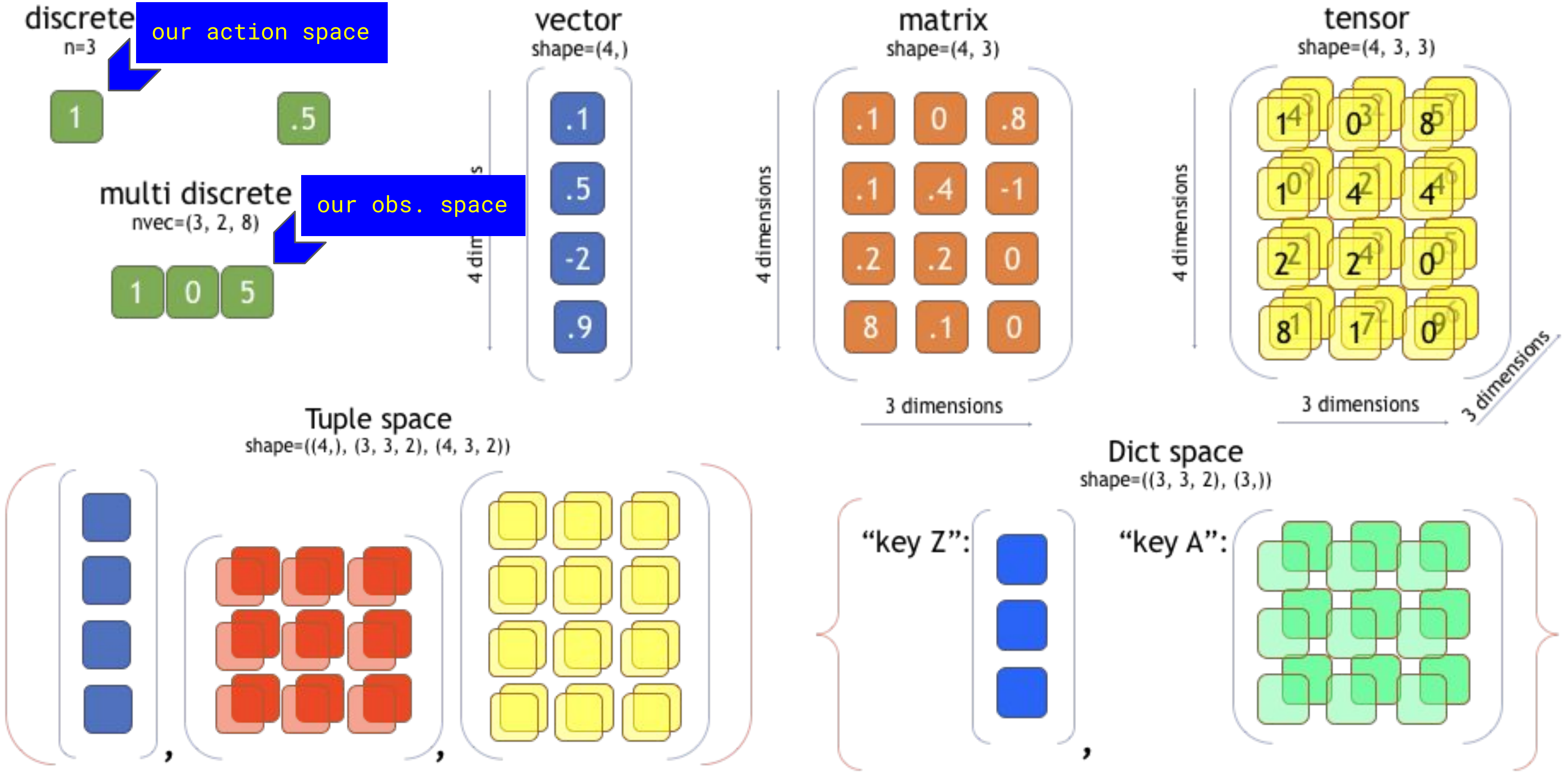
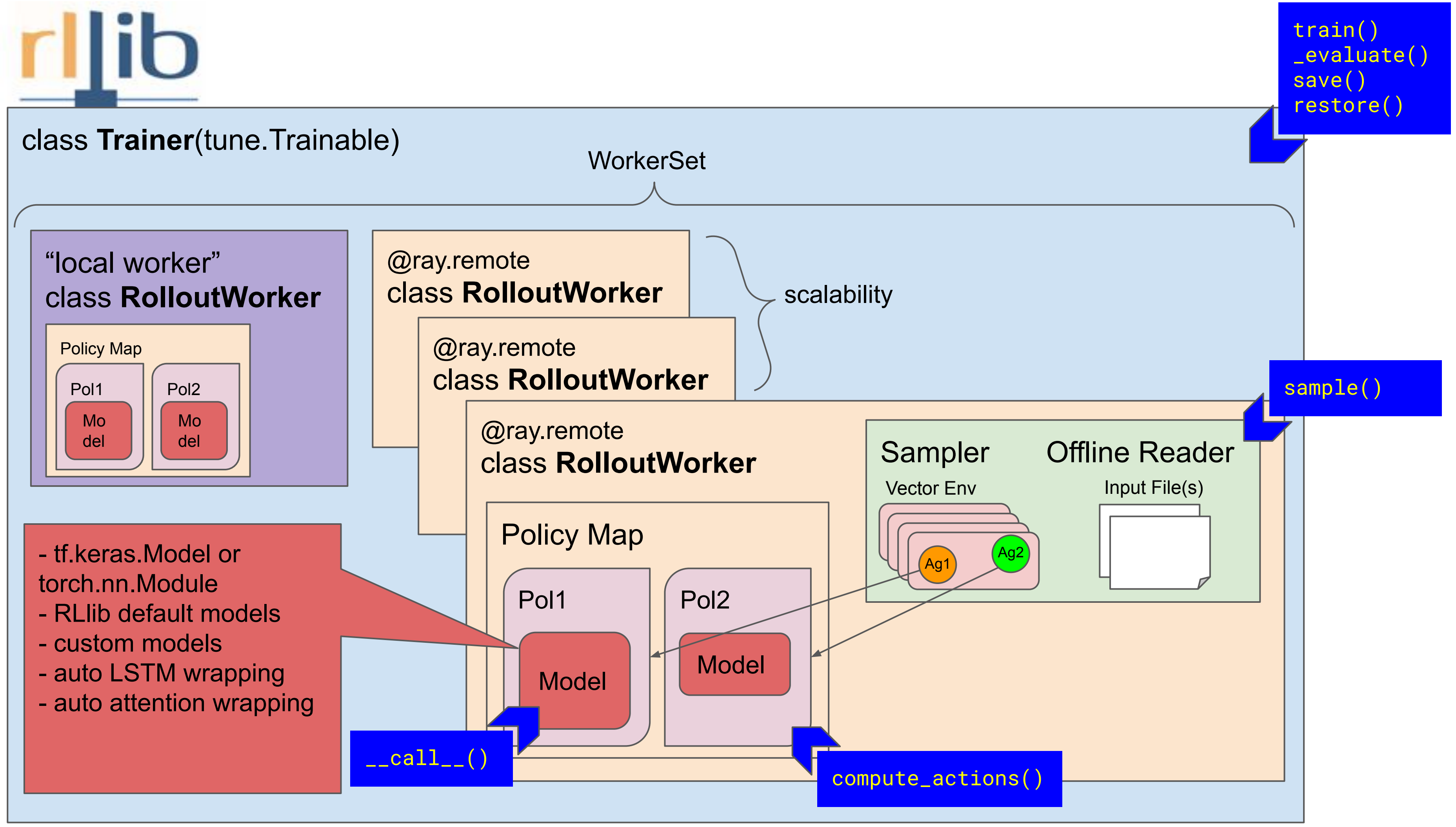

rllib
policy (in deep-RL, this is a neural network)
multi-GPU
tf.keras / torch.nn.Modules
built-in LSTM + attention nets
agents
observations
arbitrary obs spaces
multi-agent
15+ available algorithms (model-free; model-based; offline RL)
actions
rewards
+0.3
-1.0
reward shaping / learning w/o rewards
external environments
arbitrary act. spaces
custom callbacks
environment
parallelize and distribute


discrete
n=3
our action space
1
.5
multi discrete
nvec=(3, 2, 8)
our obs. space
1 0 5
vector
shape=(4,)
4 dimensions
.1
.5
-2
.9
matrix
shape=(4, 3)
4 dimensions
.1 0 .8
.1 .4 -1
.2 .2 0
8 .1 0
3 dimensions
tensor
shape=(4, 3, 3)
4 dimensions
1 0 8
1 4 4
2 2 0
8 1 0
3 dimensions
3 dimensions
Tuple space
shape=((4,), (3, 3, 2), (4, 3, 2))
Dict space
shape=((3, 3, 2), (3,))
"key Z":
"key A":


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
scalability
@ray.remote
class RolloutWorker
@ray.remote
class RolloutWorker
sample()
Sampler
Vector Env
Ag1
Ag2
Offline Reader
Input File(s)
Policy Map
Pol1
Model
Pol2
Model
- tf.keras.Model or torch.nn.Module
- RLlib default models
- custom models
- auto LSTM wrapping
- auto attention wrapping
__call__()
compute_actions()

Multi Agent Arena