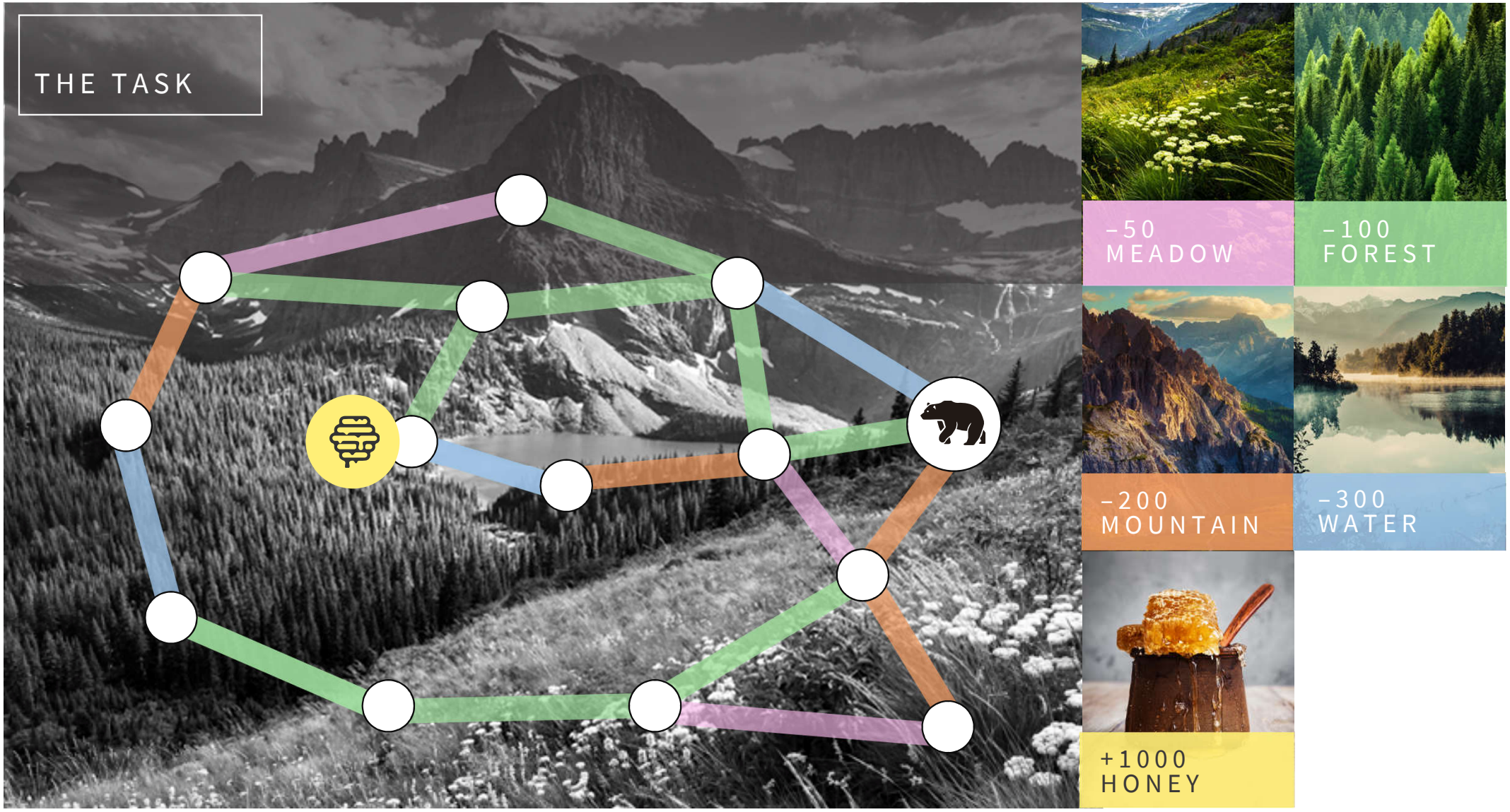
INTRODUCTION

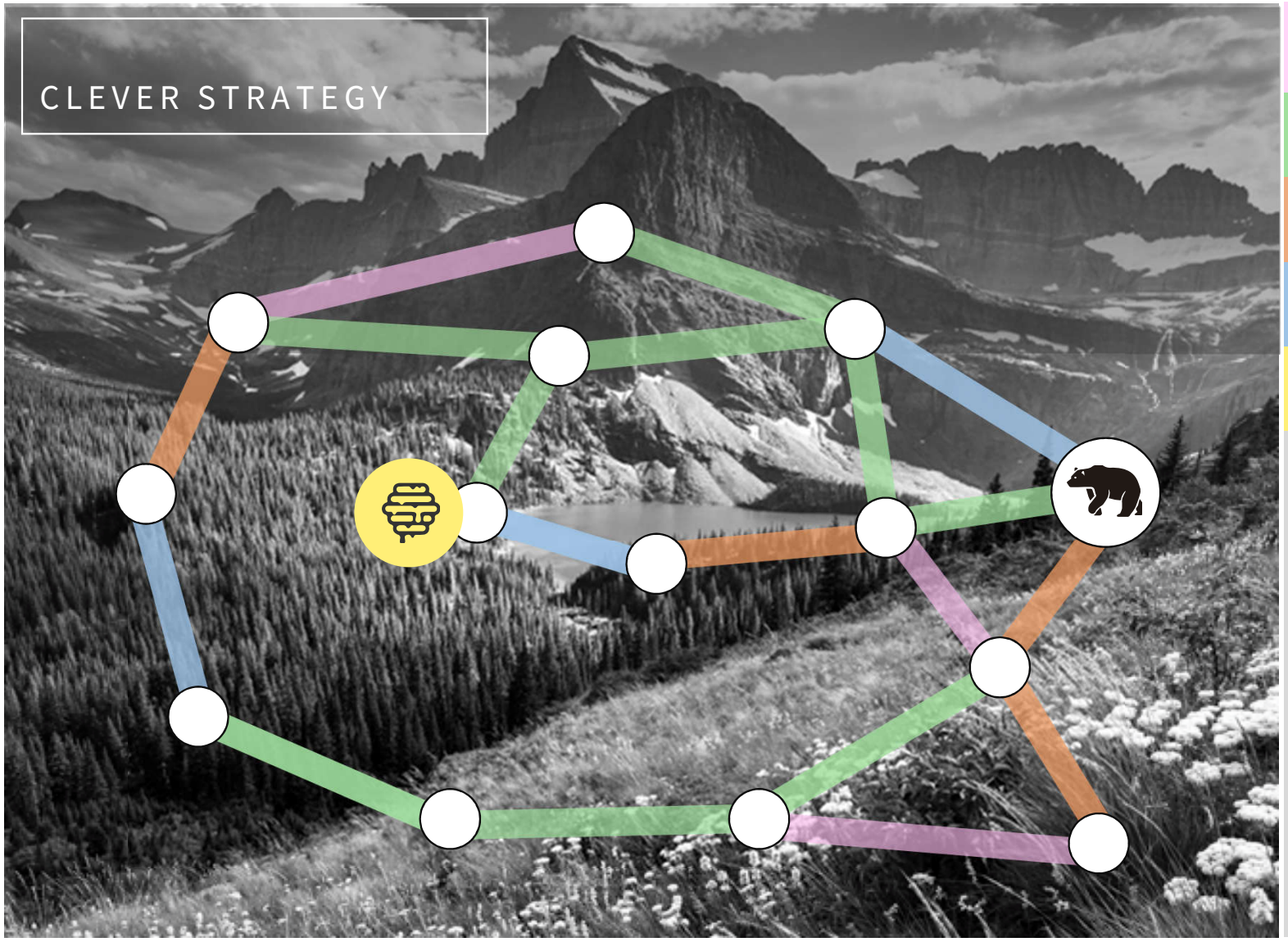
TO REINFORCEMENT LEARNING WITH TF–AGENTS & TENSOR FLOW 2.0

OLIVER ZEIGERMANN
EMBARC

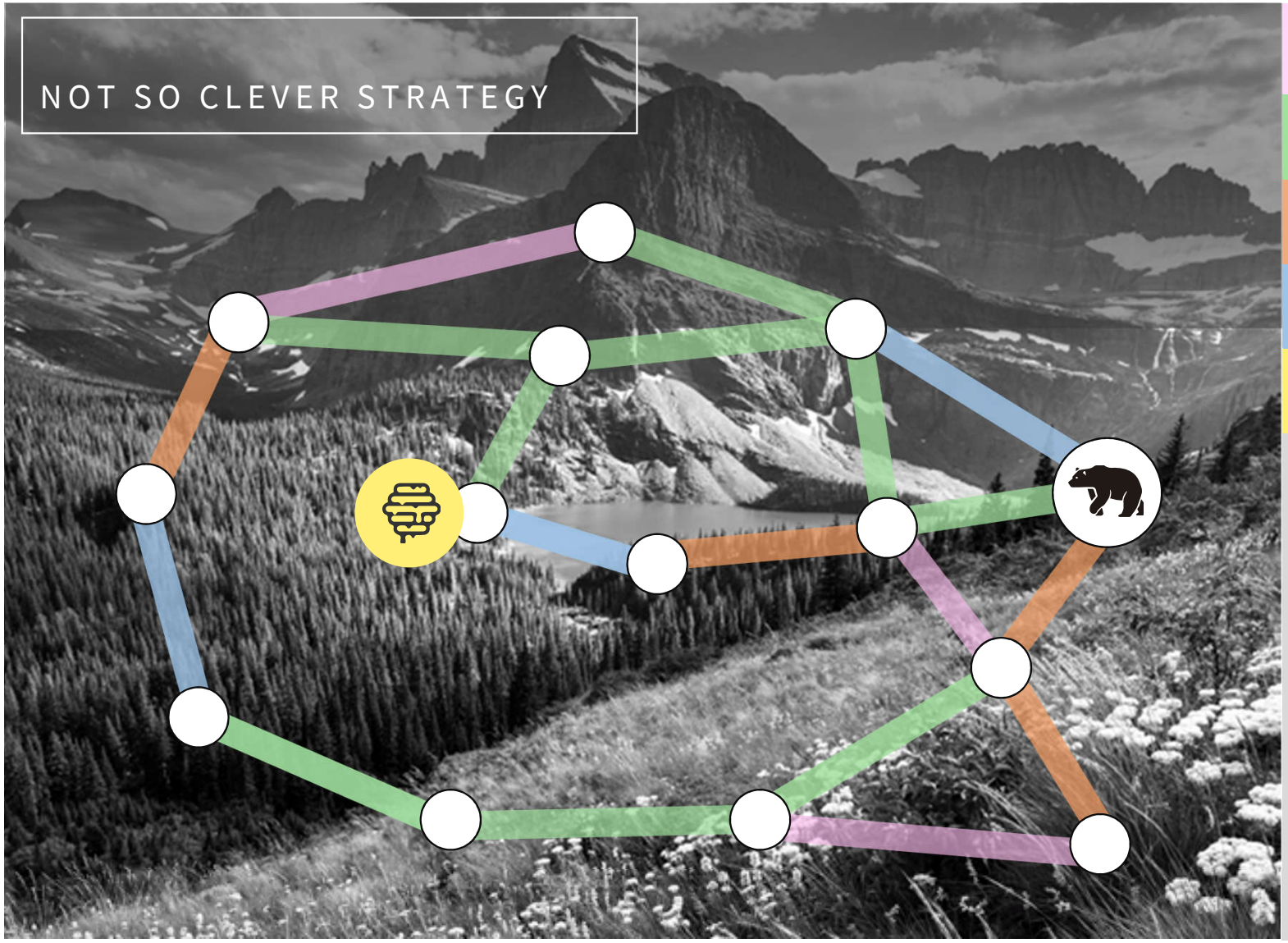
CHRISTIAN HIDBER
BSQUARE


Orso‘s World


THE TASK
−50
MEADOW
−100
FOREST
−200
MOUNTAIN
−300
WATER
+1000
HONEY



–50 MEADOW	
–100 FOREST	
–200 MOUNTAIN	
–300 WATER	
+1000 HONEY	
	-100
	-100
	-100
	-100
	+1000
	600



−50
MEADOW

−100
FOREST

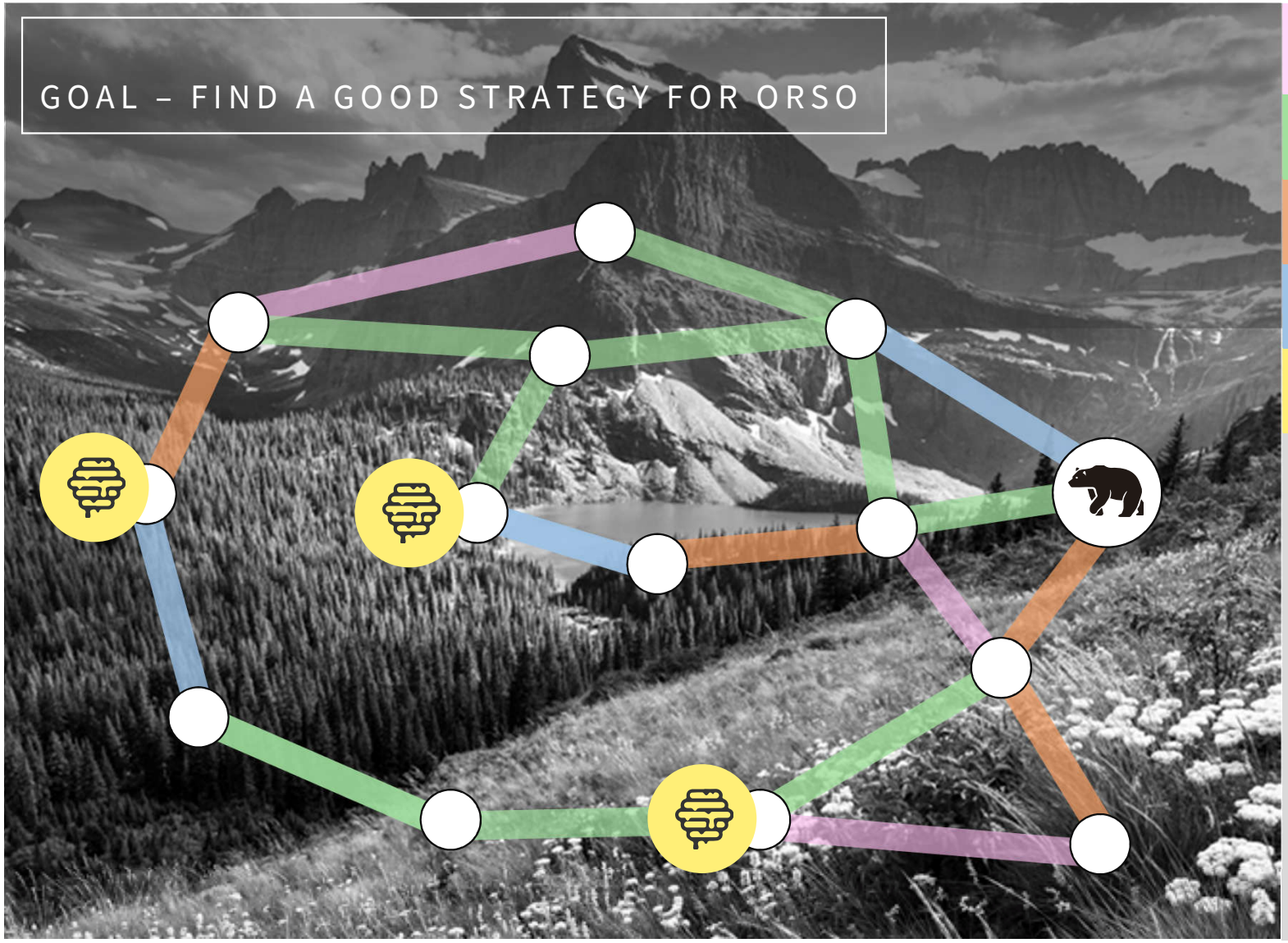
−200
MOUNTAIN

−300
WATER

+1000
HONEY

-100
-200
-300
+1000

400



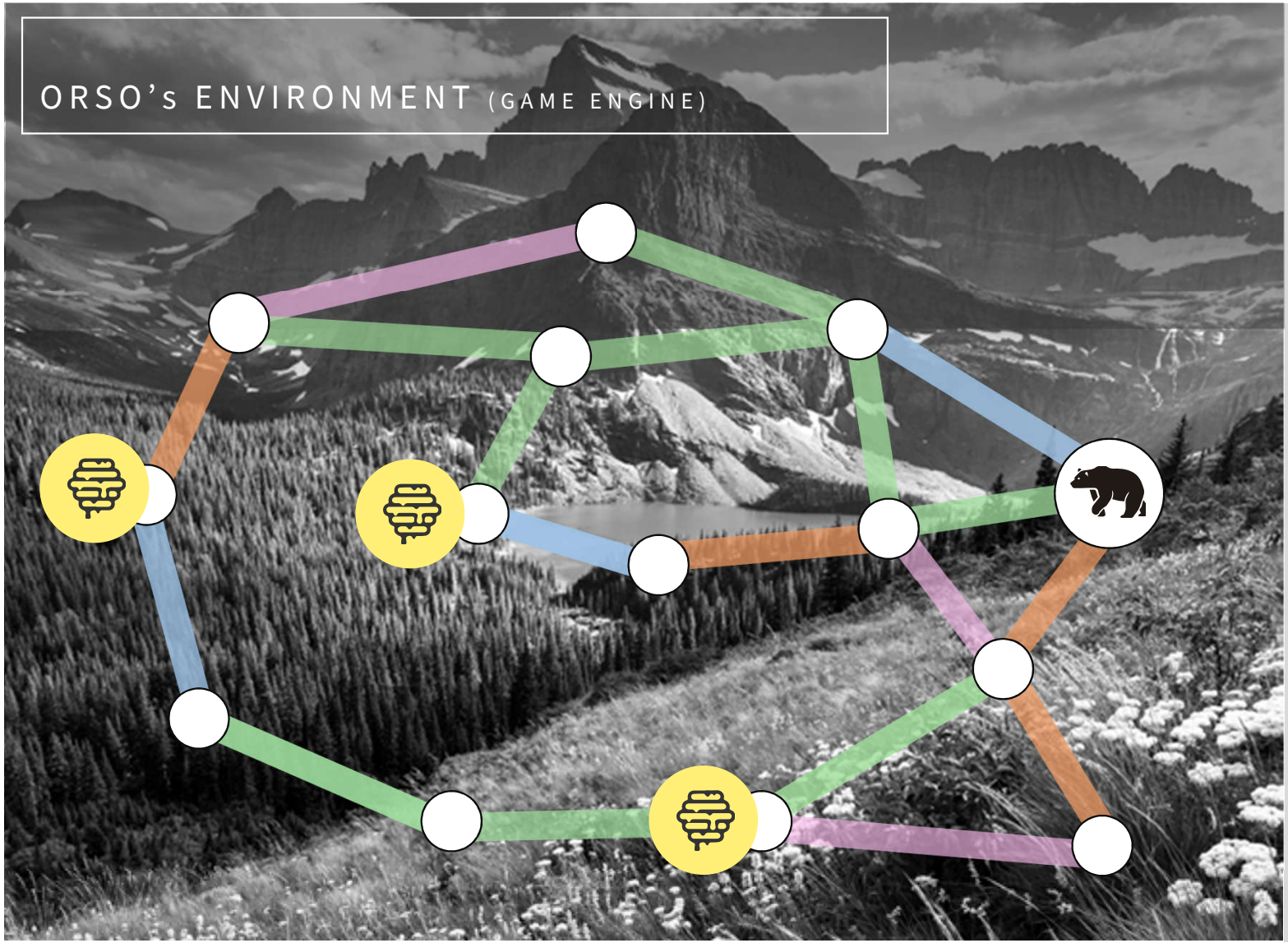
−50
MEADOW

−100
FOREST

−200
MOUNTAIN

−300
WATER

+1000
HONEY



GRAPH

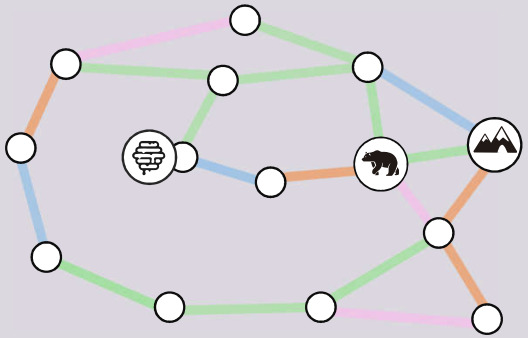
POSITIONS

(CAVE,HONEY,ORSO)

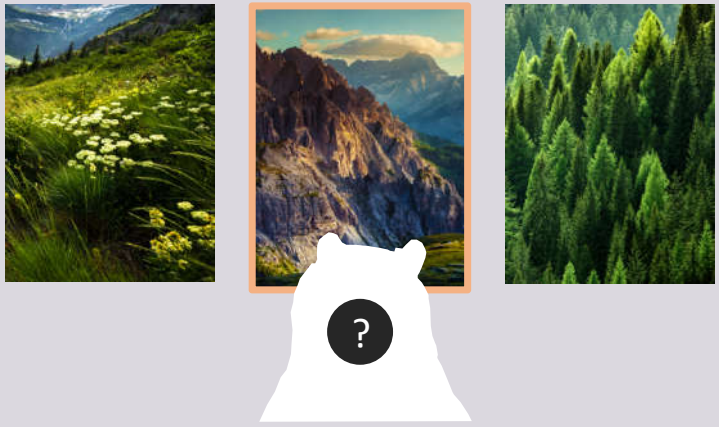
WORLD-LOGIC

...

CHOOSING AN ACTION



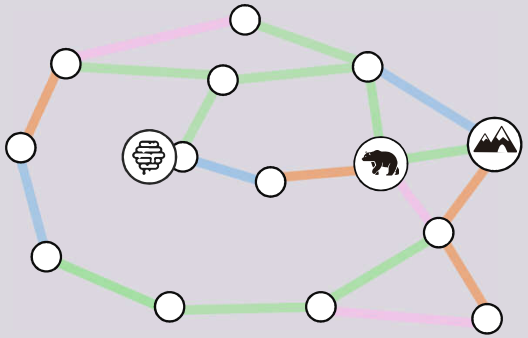
OBSERVATION
(GAME STATE)



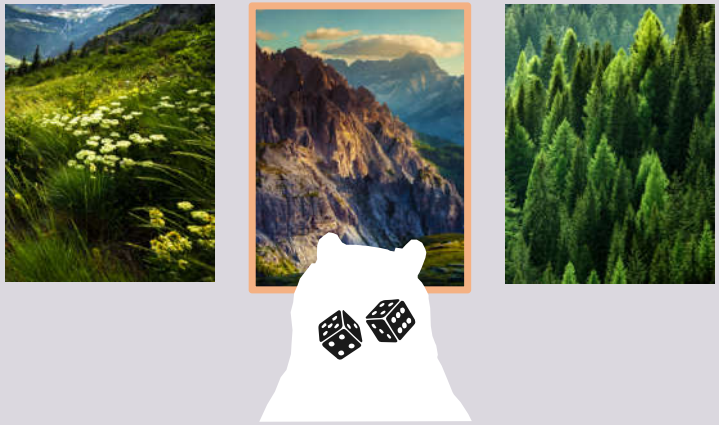
POLICY
(GAMING STRATEGY)

ACTION

CHOOSING AN ACTION



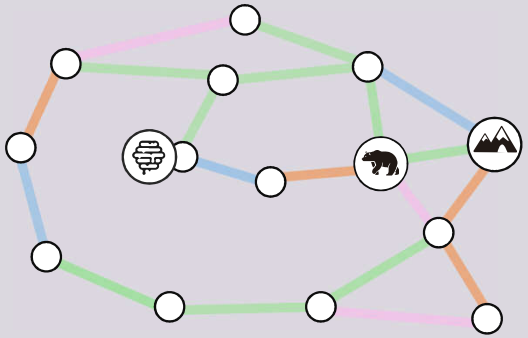
OBSERVATION
(GAME STATE)



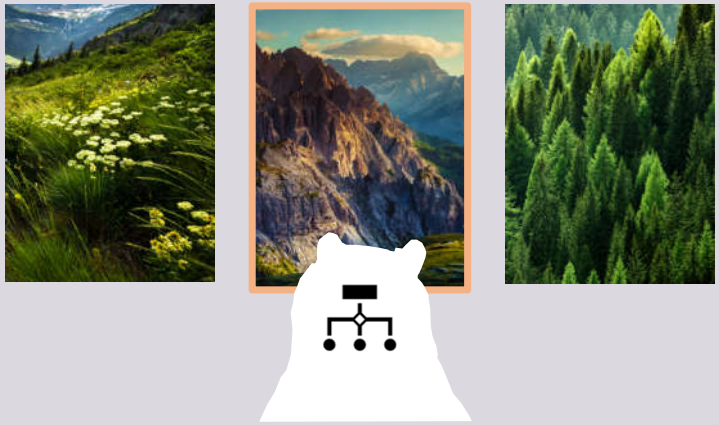
POLICY
(GAMING STRATEGY)

ACTION

CHOOSING AN ACTION



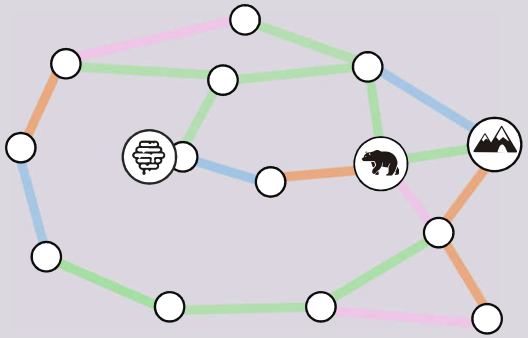
OBSERVATION
(GAME STATE)



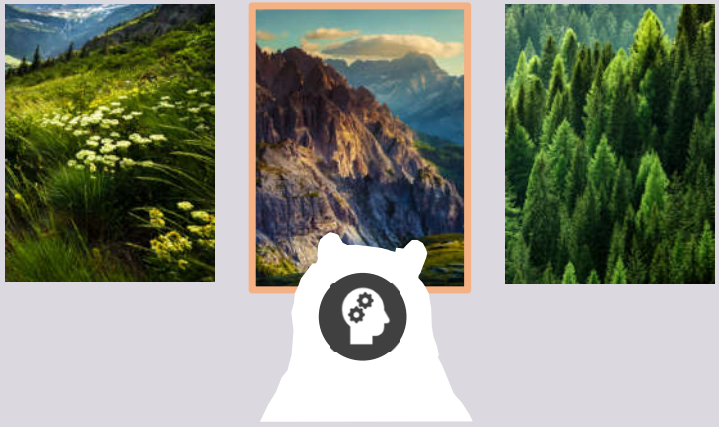
POLICY
(GAMING STRATEGY)

ACTION

CHOOSING AN ACTION



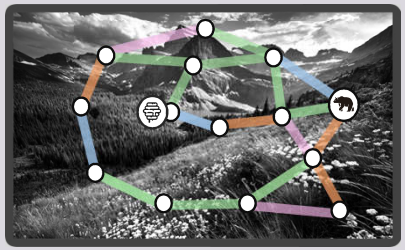
OBSERVATION
(GAME STATE)



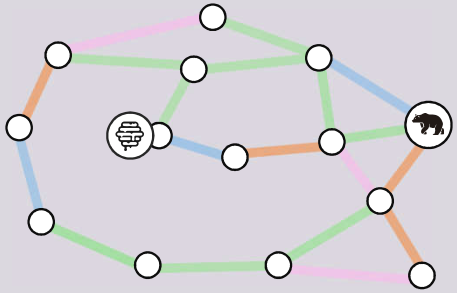
POLICY
(GAMING STRATEGY)

ACTION

REINFORCEMENT LEARNING APPROACH



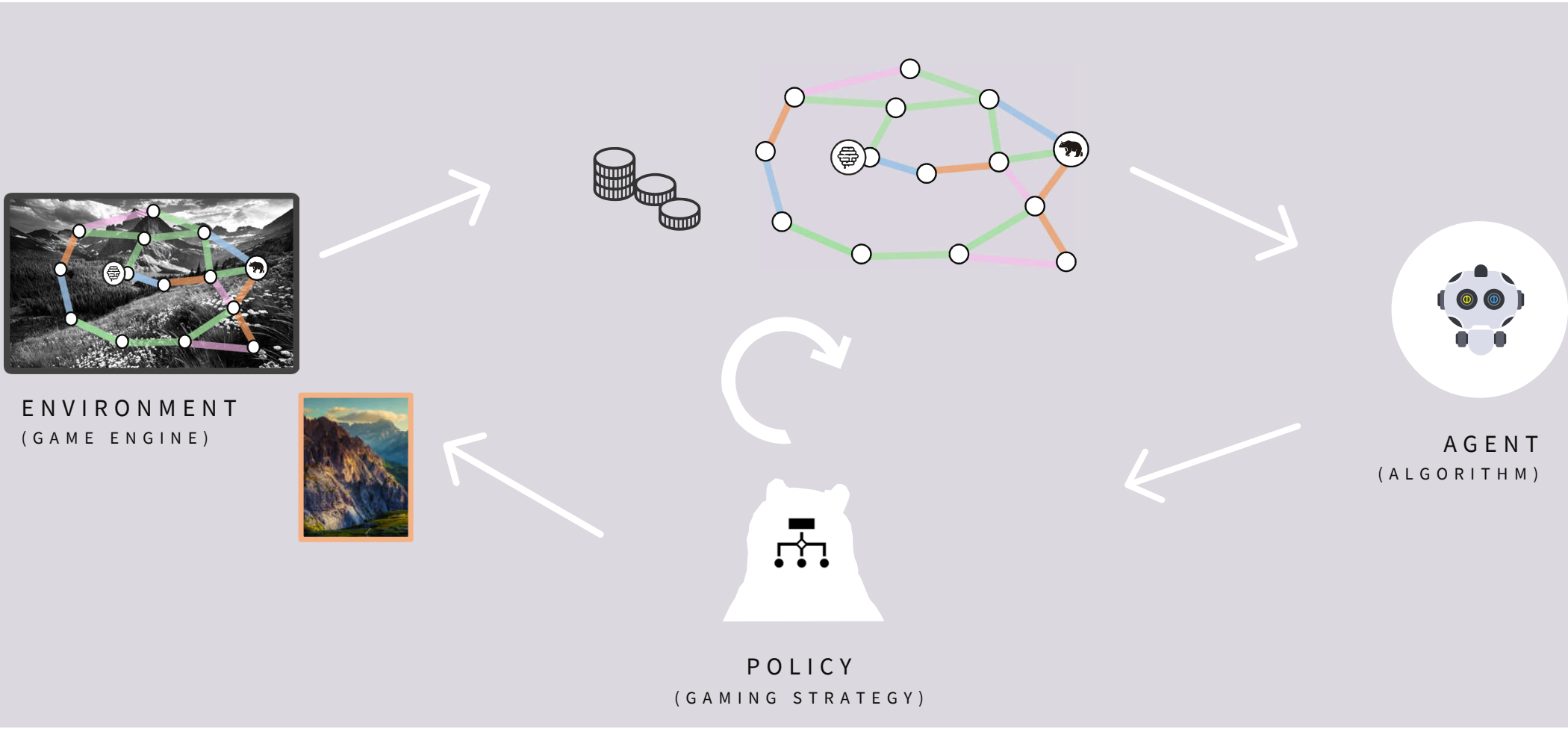
ENVIRONMENT
(GAME ENGINE)



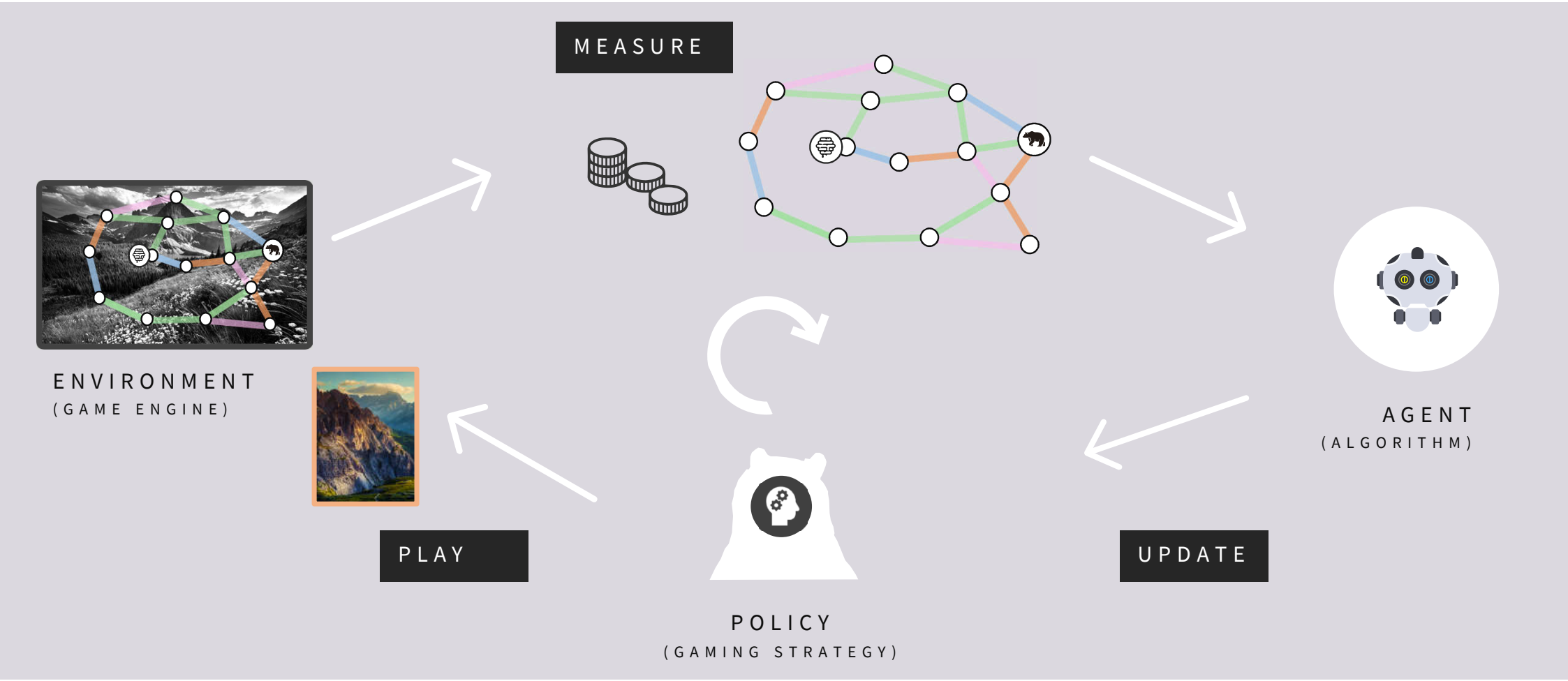
AGENT
(ALGORITHM)

POLICY
(GAMING STRATEGY)

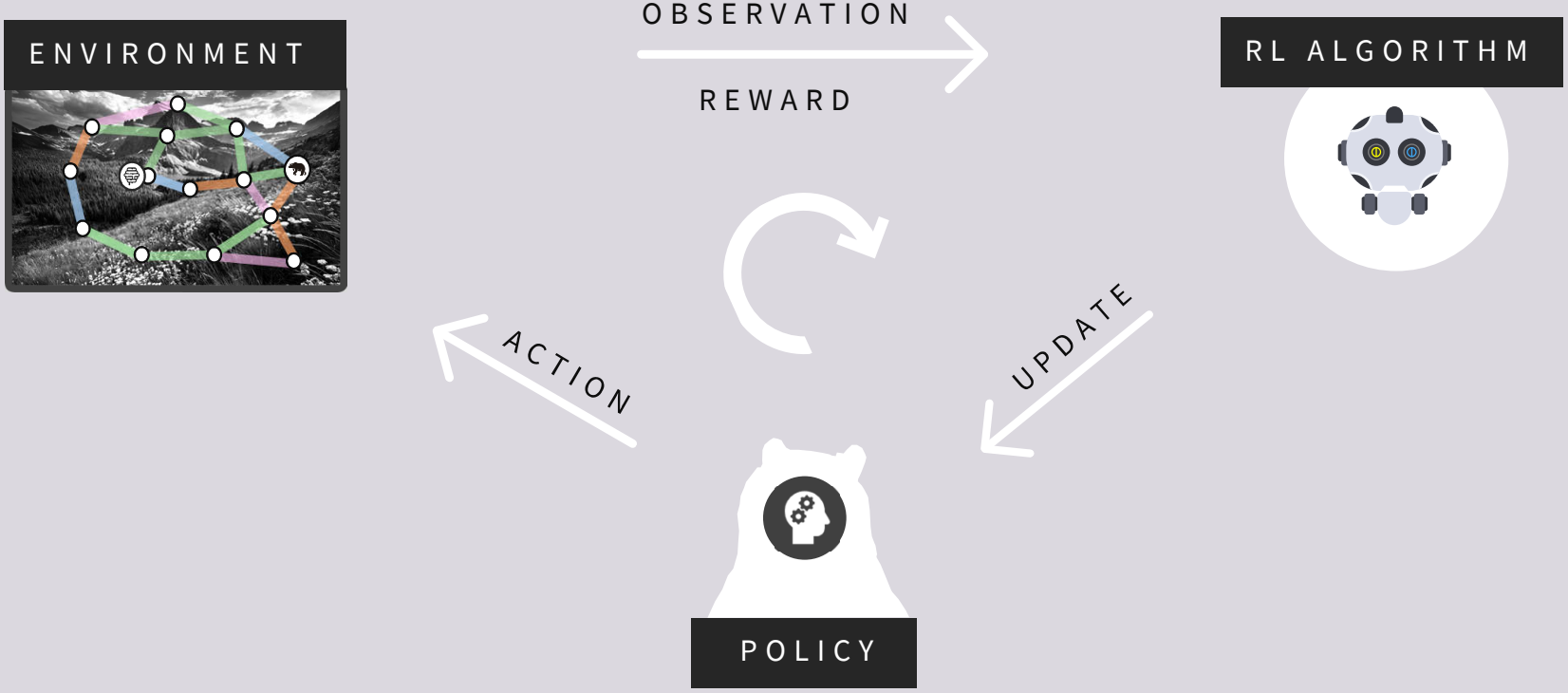
REINFORCEMENT LEARNING APPROACH



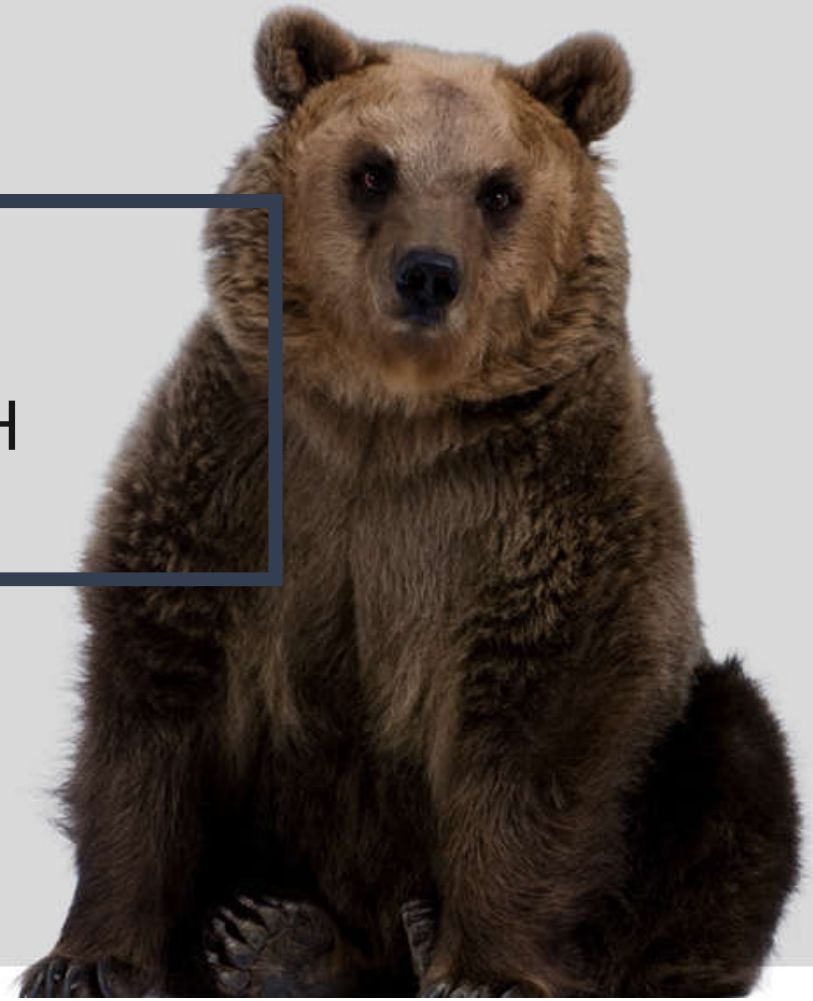
REINFORCEMENT LEARNING APPROACH



REINFORCEMENT LEARNING APPROACH



IMPLEMENTATION WITH TF–AGENTS



TF-AGENTS & TF 2.0

ENVIRONMENT

OpenAI Gym
- Step
- Reset

OBSERVATION

REWARD

RL ALGORITHM

TF Agents
- Reinforce
- DQN
- PPO
- SAC
- ...

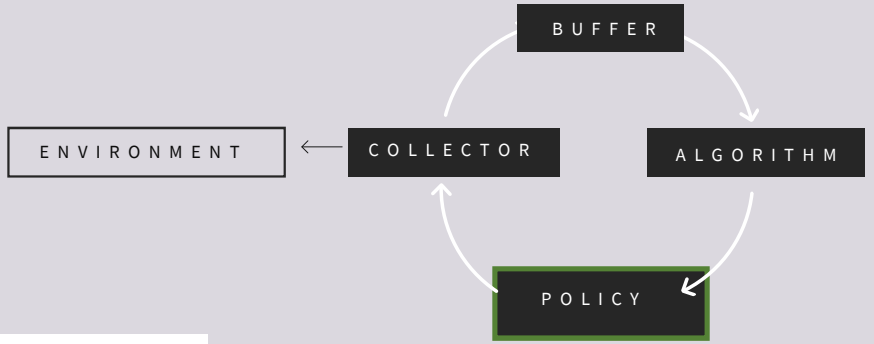
ACTION

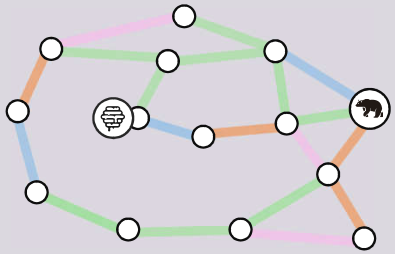
UPDATE

POLICY

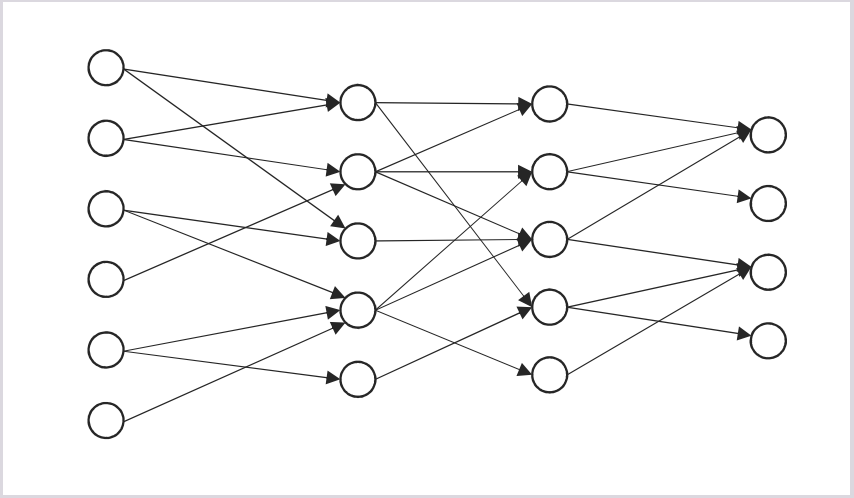
TF 2.0/KERAS

POLICY: FROM OBSERVATION TO ACTION



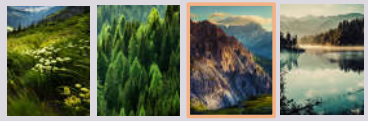


x_1
x_2
.
.
.
x_n



SOFTMAX

P_1
P_2
P_3
P_4



INPUT LAYER

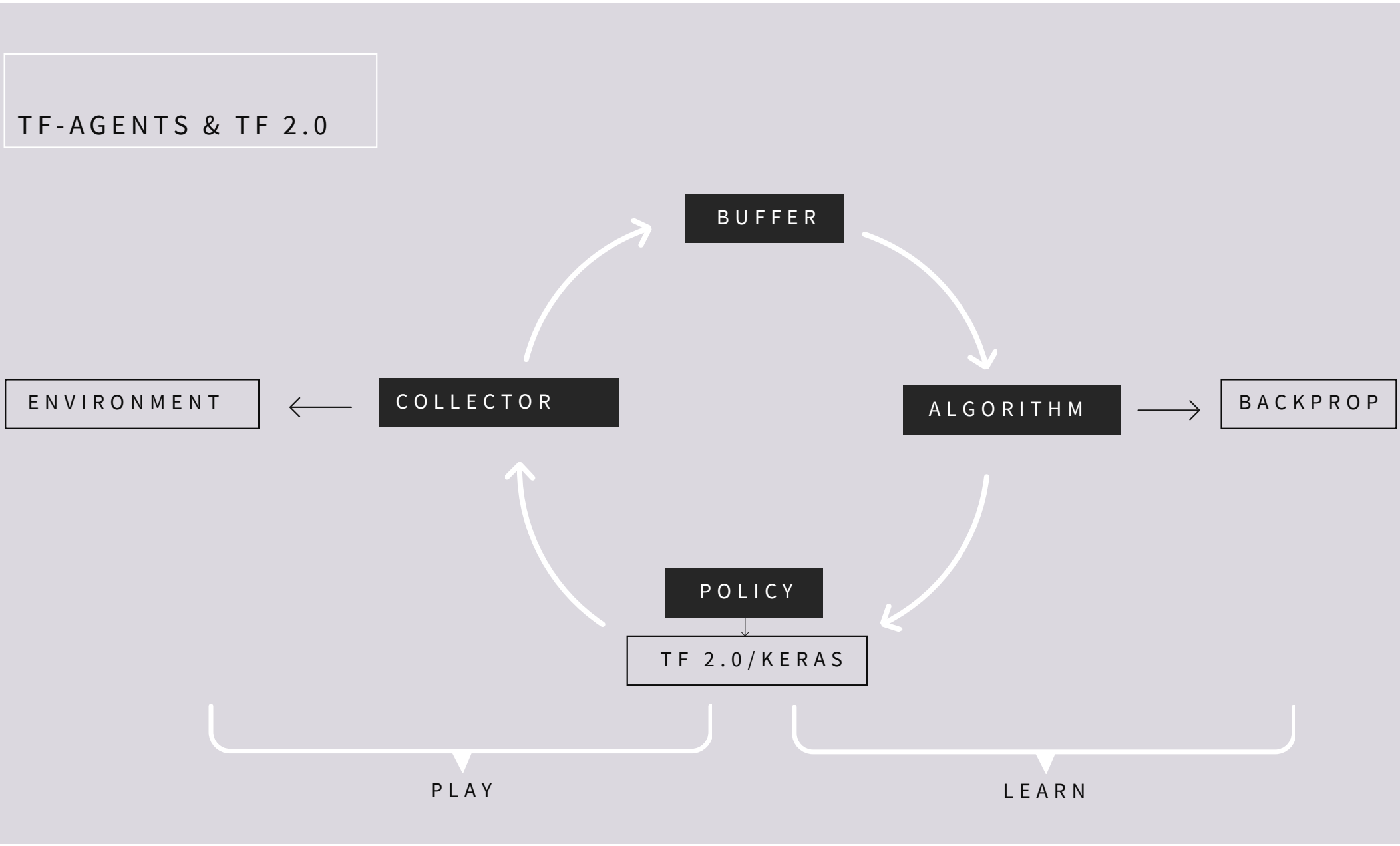
HIDDEN LAYER

OUTPUT LAYER

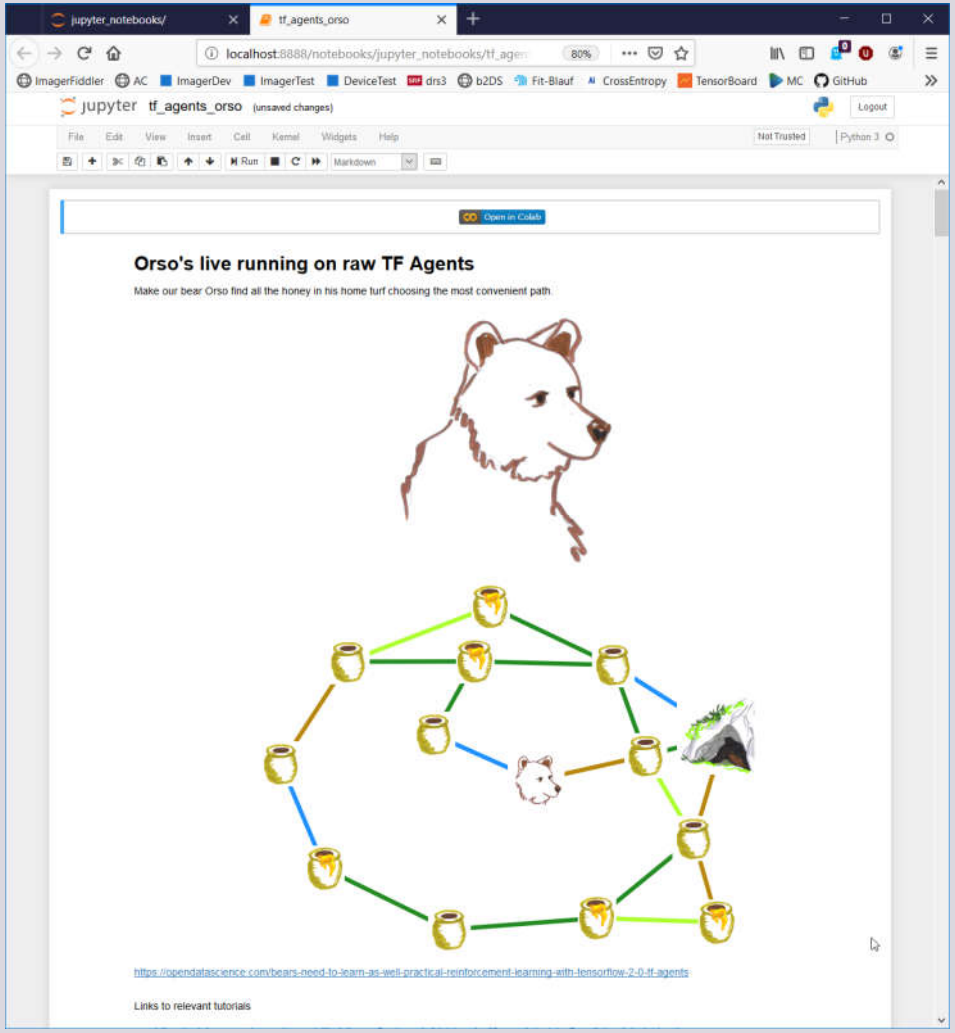
```
observation_spec = train_env.observation_spec()
```

```
action_spec = train_env.action_spec()
```

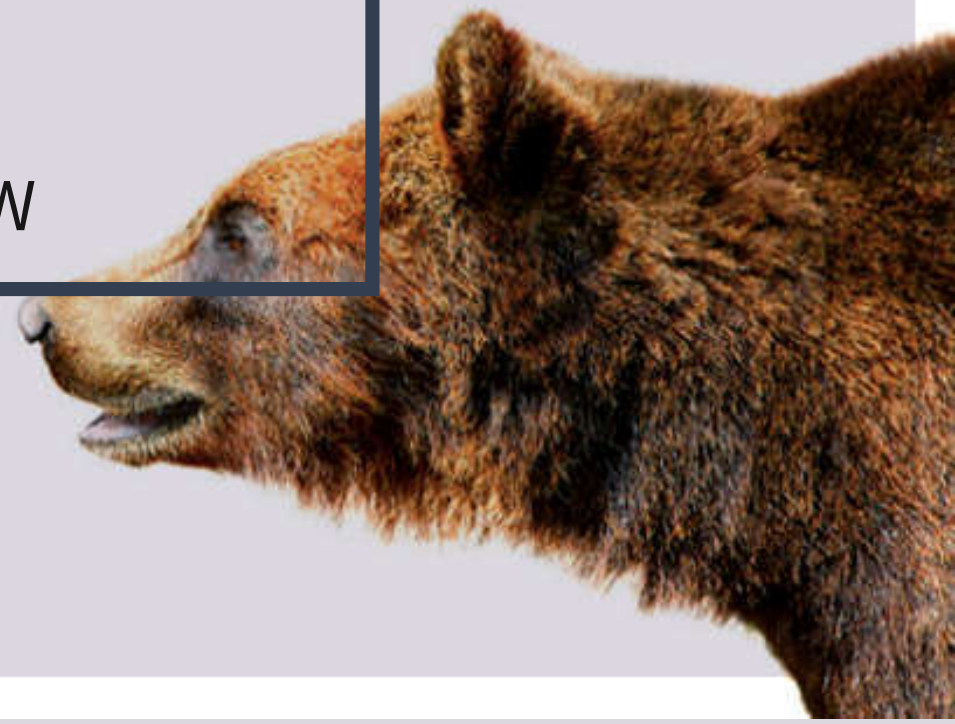
```
fc_layer_params=(500,500)
```


TF-AGENTS & TF 2.0
BUFFER
ENVIRONMENT
COLLECTOR
ALGORITHM
BACKPROP
POLICY
TF 2.0/KERAS
PLAY
LEARN

DEMO
ORSO ON TF–AGENTS



PRACTITIONERS VIEW



PRACTITIONERS VIEW

PPO (outline)

```
py_env = suite_gym.load("Orso-v1")
train_env = tf_py_environment.TFPyEnvironment(py_env)

observation_spec = train_env.observation_spec()
action_spec = train_env.action_spec()
actor_net = ActorDistributionNetwork(observation_spec, action_spec, fc_layer_params=(500,500))
value_net = ValueNetwork(observation_spec, fc_layer_params=(500,500))

optimizer = tf.compat.v1.train.AdamOptimizer(learning_rate=0.001)
timestep_spec = train_env.time_step_spec()
tf_agent = PPOAgent(timestep_spec, action_spec, optimizer, actor_net=actor_net, value_net=value_net, num_epochs=5)
tf_agent.initialize()

replay_buffer = TFUniformReplayBuffer( tf_agent.collect_data_spec, batch_size=1, max_length=10000 )

collect_driver = DynamicEpisodeDriver(train_env, tf_agent.collect_policy, observers=[replay_buffer.add_batch],
                                      num_episodes=10)

collect_driver.run = common.function(collect_driver.run, autograph=False)
tf_agent.train = common.function(tf_agent.train, autograph=False)
for i in range(250):
    if i % 10 == 0:
        eval_policy(tf_agent.policy)
    collect_driver.run()
    trajectories = replay_buffer.gather_all()
    loss, _ = tf_agent.train(experience=trajectories)
    print(f'iteration={i} loss={loss}')
    replay_buffer.clear()
```

DQN (outline)

```
py_env = suite_gym.load("Orso-v1")
train_env = tf_py_environment.TFPyEnvironment(py_env)

observation_spec = train_env.observation_spec()
action_spec = train_env.action_spec()
q_net = q_network.QNetwork(observation_spec, action_spec(), fc_layer_params=(500, 500))


optimizer = tf.compat.v1.train.AdamOptimizer(learning_rate=self._learning_rate)
time_spec = train_env.time_step_spec()
tf_agent = dqn_agent.DqnAgent(time_spec, action_spec, q_network=q_net, optimizer=optimizer)
tf_agent.initialize()

replay_buffer = TFUniformReplayBuffer(data_spec=tf_agent.collect_data_spec, batch_size=64, max_length=100000)
random_policy = random_tf_policy.RandomTFPolicy(time_spec, action_spec)
preload_driver = DynamicEpisodeDriver(env=train_env, policy=random_policy, observers=[replay_buffer.add_batch],
                                      num_episodes=1000)
preload_driver.run()

driver = DynamicEpisodeDriver(env=train_env, policy=self._trained_policy, observers=[replay_buffer.add_batch])
dataset = replay_buffer.as_dataset(num_parallel_calls=3, sample_batch_size=64, num_steps=2).prefetch(3)
iter_dataset = iter(dataset)
for iteration in range(1, 20000):
    driver.run()

    for t in range(1, self._training.num_epochs_per_iteration + 1):
        trajectories, _ = next(iter_dataset)
        tf_agent.train(experience=trajectories)
```

PRACTITIONERS VIEW

PPO (outline)

```
py_env = suite_gym.load("Orso-v1")
train_env = tf_py_environment.TFPyEnvironment(py_env)

observation_spec = train_env.observation_spec()
action_spec = train_env.action_spec()
actor_net = ActorDistributionNetwork(observation_spec, action_spec, fc_layer_params=(500,500))
value_net = ValueNetwork(observation_spec, fc_layer_params=(500,500))

optimizer = tf.compat.v1.train.AdamOptimizer(learning_rate=0.001)
timestep_spec = train_env.time_step_spec()
tf_agent = PPOAgent(timestep_spec, action_spec, optimizer, actor_net=actor_net, value_net=value_net, num_epochs=5)
tf_agent.initialize()

replay_buffer = TFUniformReplayBuffer( tf_agent.collect_data_spec, batch_size=1, max_length=10000 )

collect_driver = DynamicEpisodeDriver(train_env, tf_agent.collect_policy, observers=[replay_buffer.add_batch],
                                      num_episodes=10)

collect_driver.run = common.function(collect_driver.run, autograph=False)
tf_agent.train = common.function(tf_agent.train, autograph=False)
for i in range(250):
    if i % 10 == 0:
        eval_policy(tf_agent.policy)
    collect_driver.run()
    trajectories = replay_buffer.gather_all()
    loss, _ = tf_agent.train(experience=trajectories)
    print(f'iteration={i} loss={loss}')
    replay_buffer.clear()
```

DQN (outline)

```
py_env = suite_gym.load("Orso-v1")
train_env = tf_py_environment.TFPyEnvironment(py_env)

observation_spec = train_env.observation_spec()
action_spec = train_env.action_spec()
q_net = q_network.QNetwork(observation_spec, action_spec(), fc_layer_params=(500, 500))

optimizer = tf.compat.v1.train.AdamOptimizer(learning_rate=self._learning_rate)
time_spec = train_env.time_step_spec()
tf_agent = dqn_agent.DqnAgent(time_spec, action_spec, q_network=q_net, optimizer=optimizer)
tf_agent.initialize()

replay_buffer = TFUniformReplayBuffer(data_spec=tf_agent.collect_data_spec, batch_size=64, max_length=100000)
random_policy = random_tf_policy.RandomTFPolicy(time_spec, action_spec)
preload_driver = DynamicEpisodeDriver(env=train_env, policy=random_policy, observers=[replay_buffer.add_batch],
                                      num_episodes=1000)
preload_driver.run()

driver = DynamicEpisodeDriver(env=train_env, policy=self._trained_policy, observers=[replay_buffer.add_batch])
dataset = replay_buffer.as_dataset(num_parallel_calls=3, sample_batch_size=64, num_steps=2).prefetch(3)
iter_dataset = iter(dataset)
for iteration in range(1, 20000):
    driver.run()

    for t in range(1, self._training.num_epochs_per_iteration + 1):
        trajectories, _ = next(iter_dataset)
        tf_agent.train(experience=trajectories)
```

PRACTITIONERS VIEW

IMPLEMENTING REINFORCEMENT LEARNING


```
ppoAgent = PpoAgent( gym_env_name = 'CartPole-v0', fc_layers=(100, 50, 25) )
ppoAgent.train()
```

Tensorforce: a TensorFlow library for applied reinforcement learning

docs passing | chat on gitter | build failing | license Apache 2.0

Introduction

Tensorforce is an open-source deep reinforcement learning framework, with an emphasis on modularized flexible library design and straightforward usability for applications in research and practice. Tensorforce is built on top of Google's TensorFlow framework and compatible with Python 3 (Python 2 support was dropped with version 0.5).

Tensorforce follows a set of high-level design choices which differentiate it from other similar libraries:

- **Modular component-based design**: feature implementations, provided components are as configurable as possible, potentiall
- **Separation of RL algorithm and ap**: agent classes specify how (states/observations) and outputs (
- **Full-on TensorFlow models**: The er

cur)

cement learning

OpenAI

Husk

Huskarl is

Dopamine

rototyping of reinforcement learning algorithms. It aims to fill the need for a
s can freely experiment with wild ideas (speculative research).

Coach

build failing | License Apache 2.0 | docs passing | DOI 10.5281/zenodo.1134898

Coach is a python reinforcement learning framework containing implementation of many state-of-the-art algorithms.

It exposes a set of easy-to-use APIs for experimenting with new RL algorithms, and allows simple integration of new environments to solve. Basic RL components (algorithms, environments, neural network architectures, exploration policies, ...) are well decoupled, so that extending and reusing existing components is fairly painless.

DEMO
ORSO ON EASYAGENTS

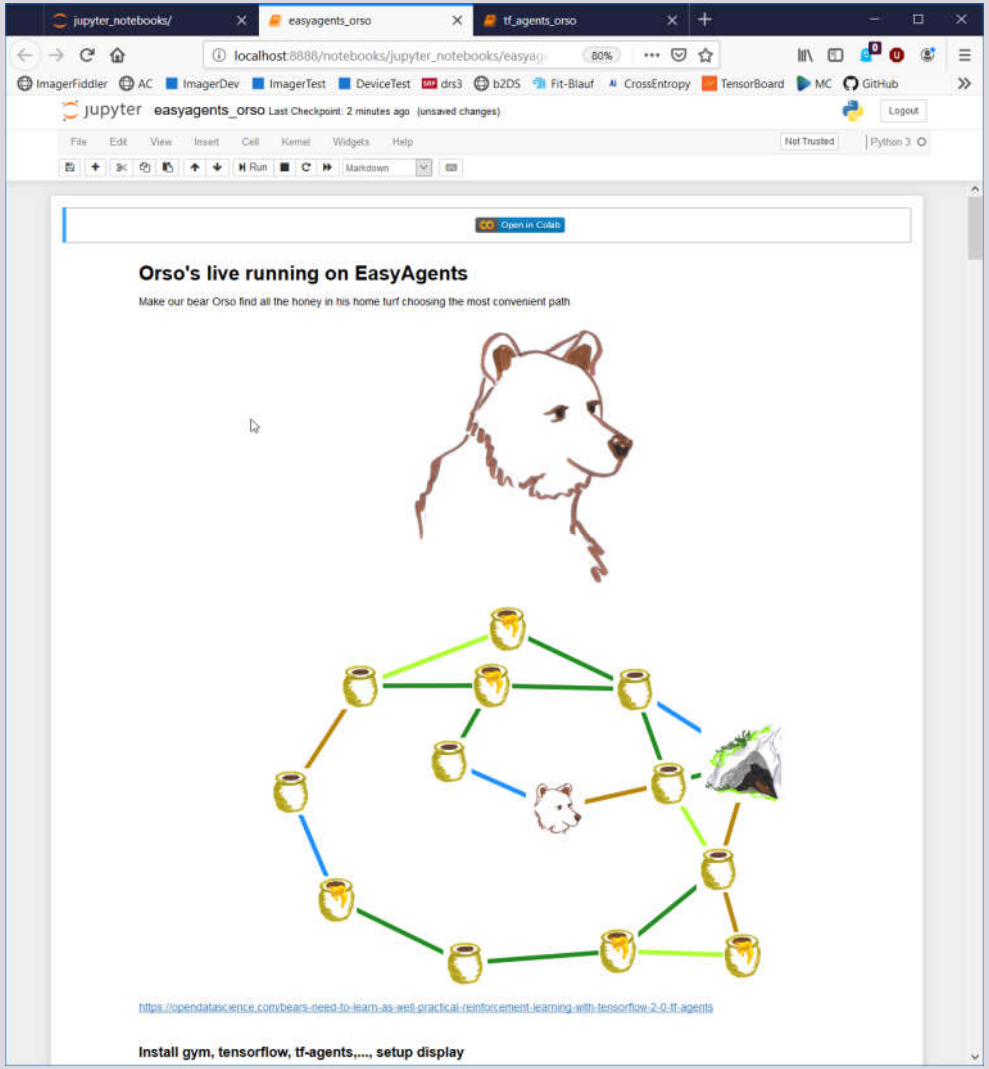
Reinforcement Learning for Practitioners (v1.2, 19Q4)

build passing | coverage 94% | license MIT | downloads/month 3k | api docs

Status: under active development, breaking changes may occur. Release notes.

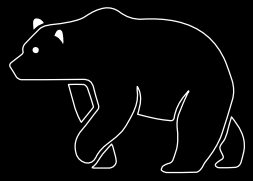
EasyAgents

EasyAgents is a high level reinforcement learning api focusing on ease of use and simplicity. Written in Python and running on top of established reinforcement learning libraries like tf-Agents, tensorforce or keras-rl. Environments are implemented in OpenAI gym. For an example of an industrial application of reinforcement learning see here.



DISCLAIMER: EASYAGENTS IS DEVELOPED BY THE SPEAKERS.

EasyAgents –



Feedback / Suggestions

Ideas –

YOUR SPEAKERS

OLIVER ZEIGERMANN

OliverZeigermann@gmail.com
@DJCordhose
https://www.linkedin.com/in/oliver-zeigermann-34989773
https://github.com/DJCordhose

CHRISTIAN HIDBER

Christian.Hidber@bsquare.ch
+41 44 260 54 00
https://www.linkedin.com/in/christian-hidber/
https://github.com/christianhidber


THANK YOU