



meetup

STATWORX

ZÜRICH R-USER GROUP REINFORCEMENT LEARNING USING R

STATWORX GmbH

Sebastian Heinz, CEO

Oliver Guggenbühl, Consultant

Zürich, 18th June 2019

AGENDA

R-Users Zurich meetup

STATWORX



1

COMPANY PROFILE

2

INTRODUCTION TO REINFORCEMENT LEARNING

3

THEORETICAL OVERVIEW

4

IMPLEMENTATION IN R

5

SUPER MARIO AI USE CASE

6

QUESTIONS

STATWORX COMPANY PROFILE

Information
Services
Project approach

STATWORX

Facts and figures

STATWORX

CLIENT EXCERPT



COMPANY PROFILE

STATWORX is a consulting company for data science, machine learning, and AI located in Frankfurt, Vienna and Zurich. We support our customers in the development and implementation of data science and machine learning projects as well as data driven products.

2011

FOUNDED

3

OFFICES

40

EMPLOYEES

200+

DATA SCIENCE
PROJECTS

50+

INDUSTRY
CUSTOMERS

1000+

DATA ACADEMY
PARTICIPANTS

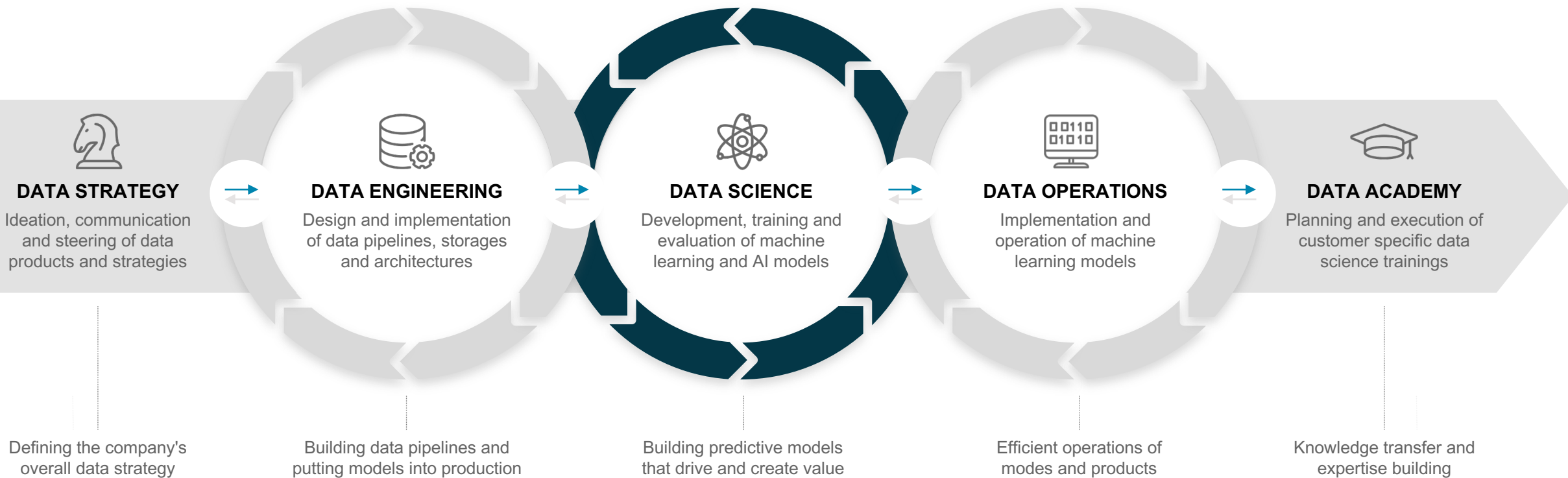
TOOL STACK & PARTNERS



END-2-END DATA CONSULTING

STATWORX

We support our customers along the whole process of data driven decision making



INTRODUCTION

REINFORCEMENT LEARNING

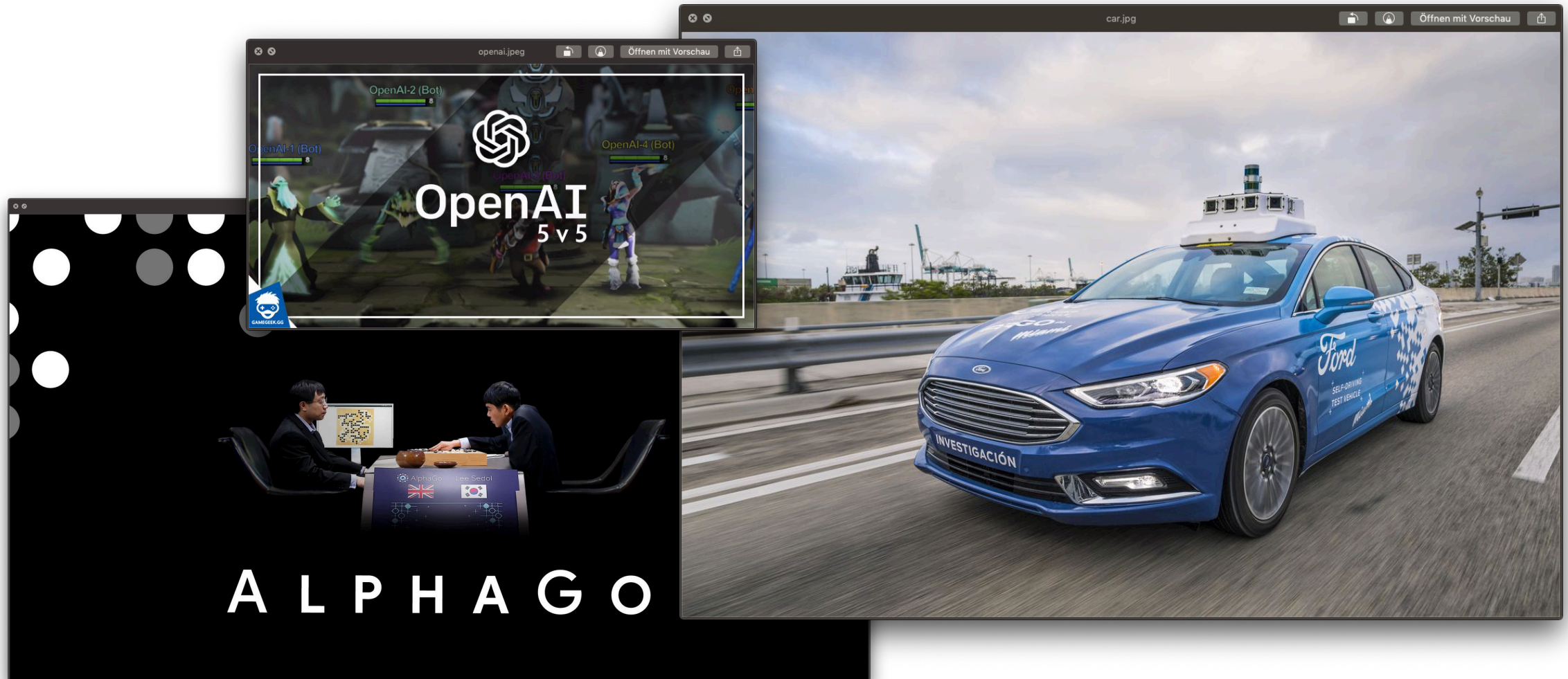
Where is Reinforcement Learning being used?

A brief history of Data Science

What distinguishes Reinforcement Learning from Supervised & Unsupervised Learning?

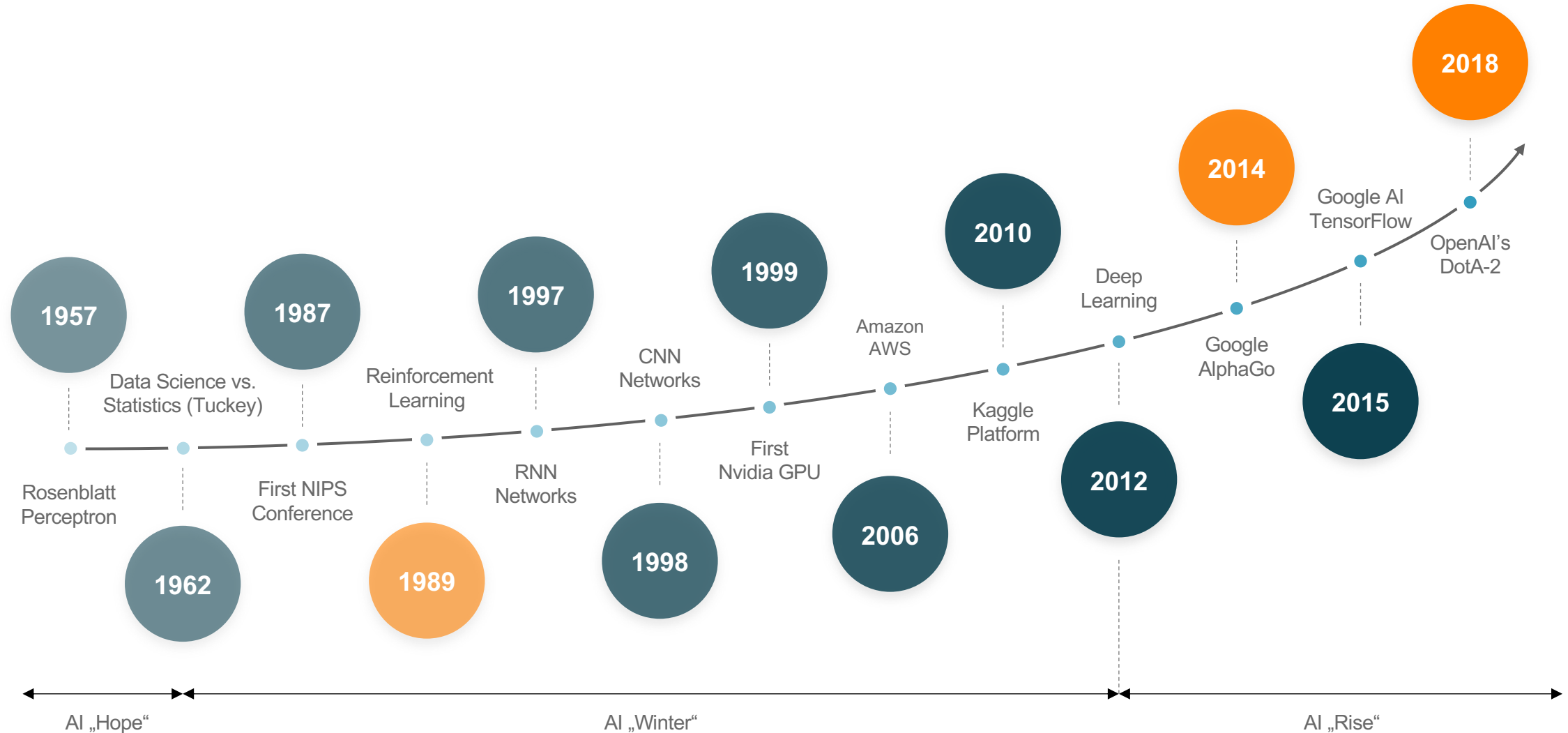
INTRODUCTION

Reinforcement Learning is currently one of the hottest ML topics



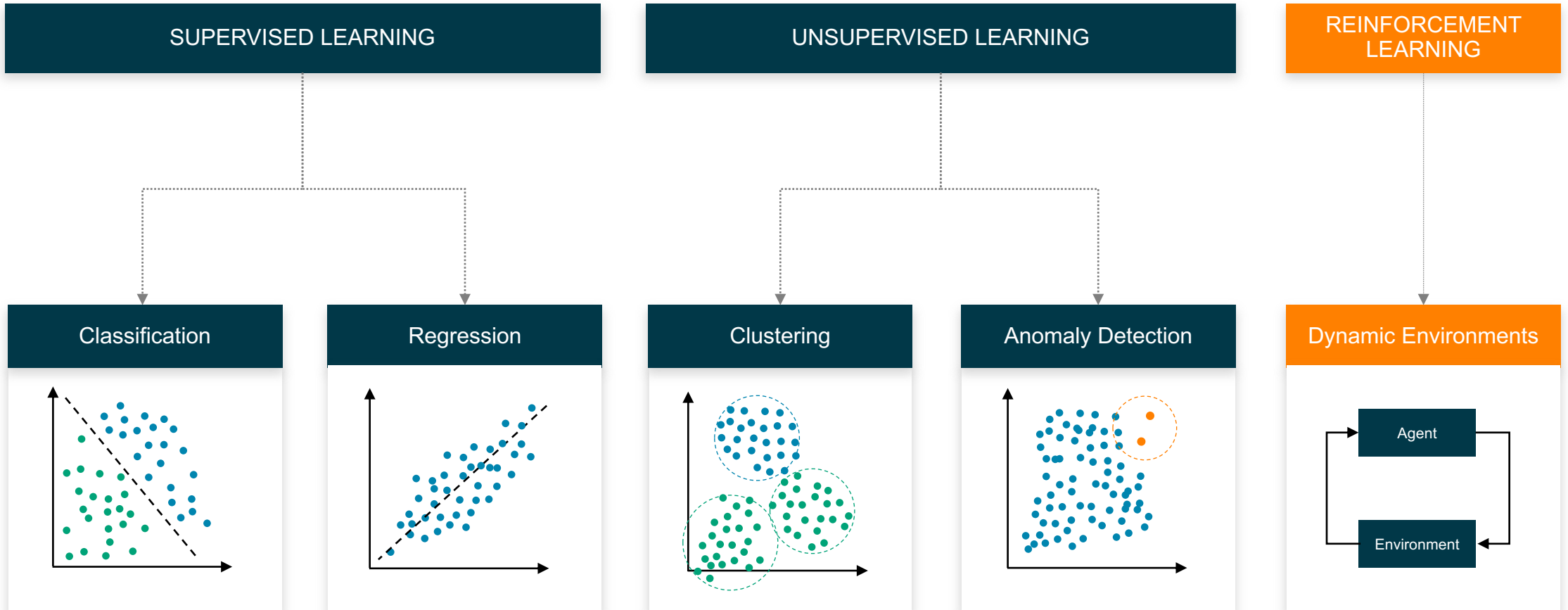
A BRIEF HISTORY OF DATA SCIENCE

The history of Data Science and AI



MACHINE LEARNING OVERVIEW

Machine Learning Applications



INTRODUCTION

What is Reinforcement Learning?

STATWORX

„Instead of relying on a set of (labelled or unlabelled) training data, Reinforcement Learning relies on being able to monitor the response of the actions taken by the agent.“

THEORY

REINFORCEMENT LEARNING

How does Reinforcement Learning work?
The Gridworld problem as an example

THEORY

How does Reinforcement Learning work?



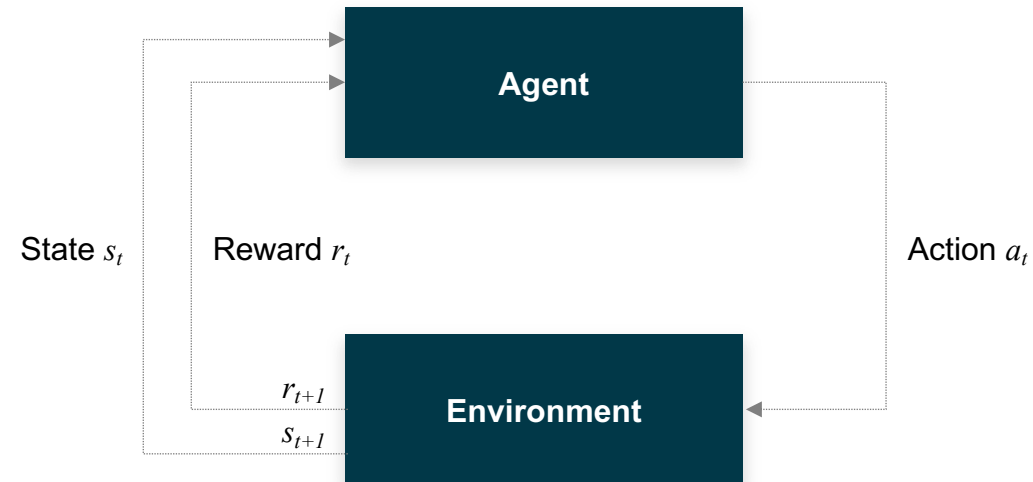
THEORY

How does Reinforcement Learning work?



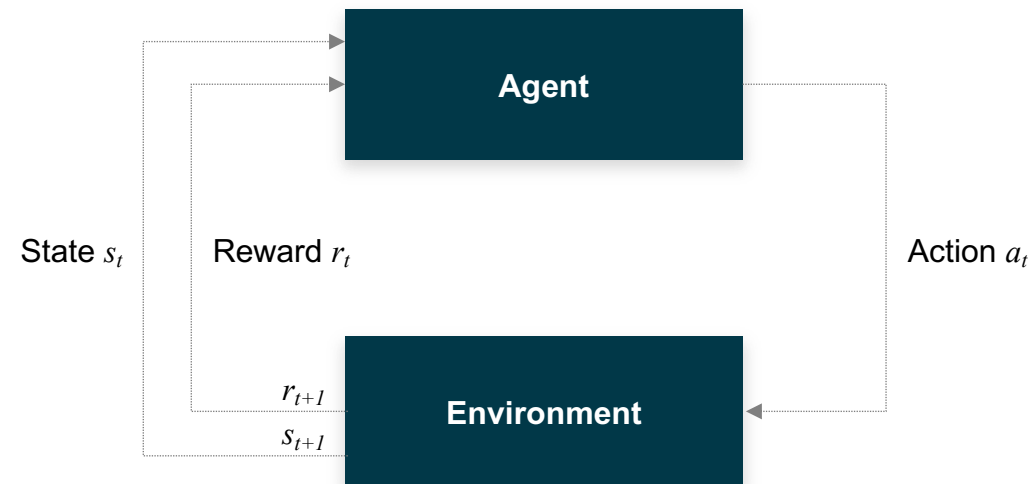
THEORY

How does Reinforcement Learning work?



THEORY

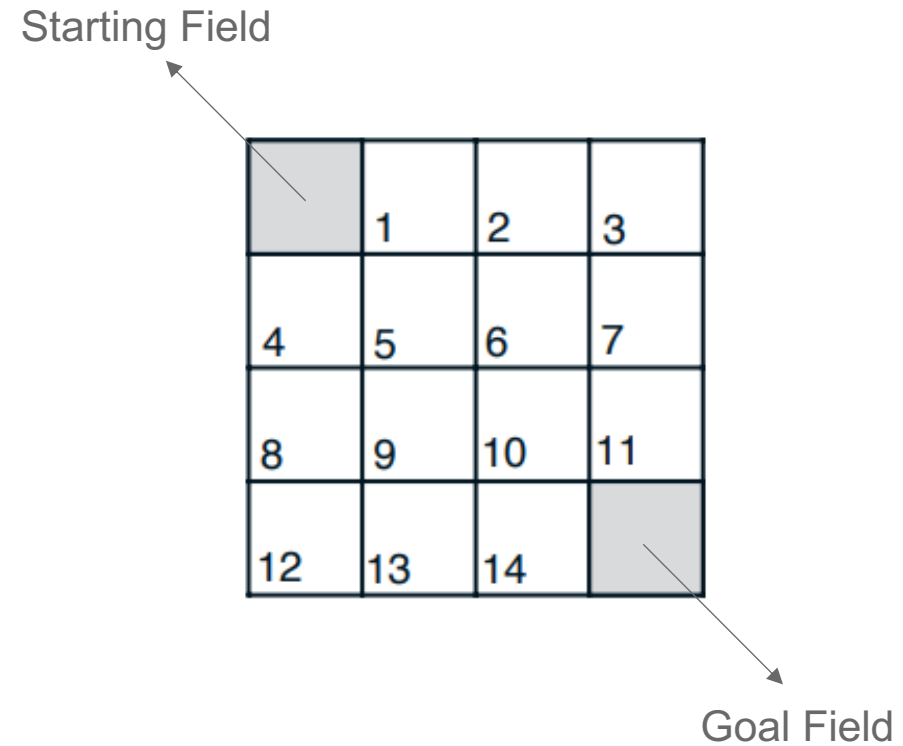
How does Reinforcement Learning work?



Agent tries to maximize his reward by choosing appropriate actions at a given state of the environment.

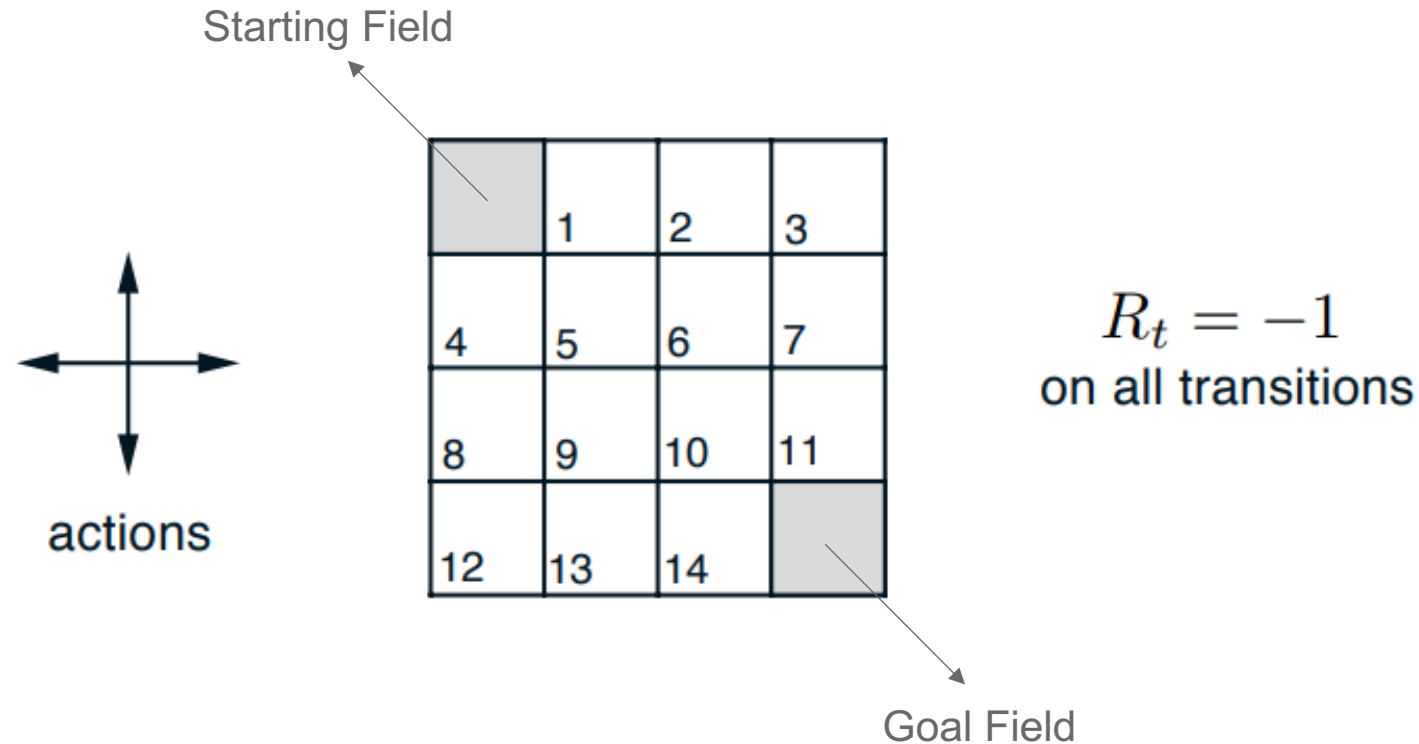
EXAMPLE: GRIDWORLD

Reinforcement Learning Use Case



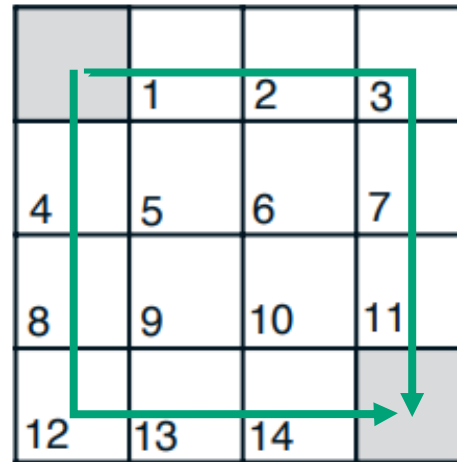
EXAMPLE: GRIDWORLD

Reinforcement Learning Use Case



EXAMPLE: GRIDWORLD

Reinforcement Learning Use Case



*Ideal return: -6
(6 steps to complete the episode)*

Q-LEARNING

Determining the optimal policy for an environment

$$Q(s, a) = r + \gamma \max_{a'} Q(s', a')$$

Q-Value

*immediate
reward*

*discount
factor*

*future
reward*

Q-Value

- represents the maximum possible reward at the end of the game for action a in state S
- *does so for each possible action a for the current state S*

Immediate reward

- the immediate reward of a possible action taken, as defined by the environment
- there might be no immediate rewards, but only delayed rewards

Discount factor

- Steers the rate of considering future rewards
- Small values promote decisions that generate immediate reward
- Larger values favor decisions that generate future reward

Future reward

- the maximum possible future reward after transitioning to the next state S' by choosing action a

Q-LEARNING

Determining the optimal policy for an environment

$Q = .59$	$Q = .656$	$Q = .73$	$Q = .81$
$Q = .656$	$Q = .73$	$Q = .81$	$Q = .9$
$Q = .73$	$Q = .81$	$Q = .9$	$Q = 1$
$Q = .81$	$Q = .9$	$Q = 1$	$r = 1$

Assuming that the discount factor $\gamma = 0.9$ and the final reward $r = 1$: $Q(14, right) = 1 + 0.9 \cdot 0$



IMPLEMENTATION IN R

THE reinforcelearn PACKAGE

The reinforcelearn Package
Live Demo
Advanced Functionalities


THE reinforcelearn PACKAGE

Reinforcement Learning implementation in R

STATWORX

reinforcelearn package:

The reinforcelearn package offers easy tools to create environments, agents and let them interact.



```
library(reinforcelearn)

# Create an environment
env <- makeEnvironment()

# Create an agent
agent <- makeAgent()

# Let the agent interact with the environment
interact(env, agent)
```

THE reinforcelearn PACKAGE

Reinforcement Learning implementation in R

STATWORX

Gridworld in reinforcelearn:

The Gridworld environment can be easily created with only a few lines of code:

```
library(reinforcelearn)

# Create an environment
env <- makeEnvironment("gridworld",
                       shape = c(4, 4),
                       goal.states = 15,
                       initial.state = 0)
```



THE reinforcelearn PACKAGE

Reinforcement Learning implementation in R

STATWORX

Gridworld in reinforcelearn:

The agent consists of several parts:

- The policy defines the type of decision rules.
- The value function determines how the current state of the agent is to be evaluated.
- The algorithm determines how the optimal policy is to be found and learnt.

```
library(reinforcelearn)

# Set up the agent
policy <- makePolicy("epsilon.greedy",
                    epsilon = 0.1)
val.fun <- makeValueFunction("table")
algorithm <- makeAlgorithm("qlearning")

# Create the agent
agent <- makeAgent(policy = policy,
                  val.fun = val.fun,
                  algorithm = algorithm)
```



THE reinforcelearn PACKAGE

Reinforcement Learning implementation in R

STATWORX

Interaction:

Once the environment and agent have been created we can let them interact with the `reinforcelearn::interact()` function.

```
library(reinforcelearn)

# Let the agent interact with the environment
interact(env, agent,
         n.episodes = 500,
         visualize = TRUE,
         learn = TRUE)
```



LIVE DEMO

Reinforcement Learning in action

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for setting up a gridworld environment and training an agent using the `reinforcelearn` package. The code includes environment creation, policy and value function setup, and the `interact` function to run the training.
- Environment Panel:** Shows the current environment variables: `agent` (Environment), `algorithm` (List of 2), `env` (Environment), `policy` (List of 2), and `val.fun` (List of 2).
- Console:** Displays the output of the training process, including the message "Episode 100 finished after 31 steps with a return of -31" and a matrix of returns for each step.
- Package Manager:** Shows the system library of installed packages, including `archdata`, `argonDash`, `argonR`, `askpass`, `assertthat`, `AzureAuth`, `AzureRMR`, `AzureStor`, `backports`, `base`, `base64enc`, and `bayesplot`.

```
1 library(reinforcelearn)
2
3 # set up the environment
4 env = makeEnvironment("gridworld", shape = c(4, 4), goal.states = 15, initial.state = 0)
5
6 # set up the agent
7 policy = makePolicy("softmax")
8 val.fun = makeValueFunction("table", n.states = env$n.states, n.actions = env$n.actions)
9 algorithm = makeAlgorithm("qlearning")
10
11 agent = makeAgent(policy = policy, val.fun = val.fun, algorithm = algorithm)
12
13 # let agent interact with environment
14 interact(env, agent, n.episodes = 100, visualize = TRUE)
```

```
14:57 (Top Level) R Script
~/Intern/reticulate/
- - - -
- - - -
- - - -
- - - o

Episode 100 finished after 31 steps with a return of -31
$return
[1] -40 -9 -56 -66 -46 -35 -17 -75 -21 -89 -36 -83 -34 -51 -67 -41 -29 -16 -10 -179 -101 -30 -53 -15 -16 -
72 -43
[28] -33 -28 -45 -48 -51 -19 -31 -10 -30 -21 -14 -18 -10 -6 -43 -32 -37 -22 -10 -30 -16 -42 -56 -18 -19 -
23 -8
[55] -15 -22 -12 -20 -17 -12 -53 -27 -16 -32 -10 -8 -10 -13 -14 -19 -26 -11 -6 -12 -9 -6 -17 -10 -15 -
11 -12
[82] -60 -10 -29 -19 -28 -15 -11 -17 -9 -37 -10 -26 -32 -13 -17 -17 -6 -9 -31

$steps
[1] 40 9 56 66 46 35 17 75 21 89 36 83 34 51 67 41 29 16 10 179 101 30 53 15 16 72 43 33 28 45 48 51
```

Name	Description	Version
archdata	Example Datasets from Archaeological Research	1.2
argonDash	Argon Shiny Dashboard Template	0.1.0
argonR	R Interface to Argon HTML Design	0.1.0
askpass	Safe Password Entry for R, Git, and SSH	1.1
assertthat	Easy Pre and Post Assertions	0.2.1
AzureAuth	Authentication Services for Azure Active Directory	1.1.0
AzureRMR	Interface to 'Azure Resource Manager'	2.1.1
AzureStor	Storage Management in 'Azure'	2.0.1
backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.4
base	The R Base Package	?
base64enc	Tools for base64 encoding	?
bayesplot	Plotting for Bayesian Models	?

Live Demo

OPENAI GYMS

Advanced functionalities with OpenAI gyms

STATWORX

Using OpenAI gyms in `reinforcelearn`:

`reinforcelearn` allows for easy access to gym environments created by OpenAI.



```
library(reinforcelearn)
library(reticulate)

# Create an environment
env <- makeEnvironment("gym",
                       gym.name = "SpaceInvaders-v0")
```



NEURAL NETWORKS

Advanced functionalities with Keras

STATWORX



Using neural networks in reinforcelearn:

reinforcelearn allows for easy integration of neural networks made in keras into your value function.

```
library(reinforcelearn)
library(keras)

env <- makeEnvironment("gridworld",
                      shape = c(4, 4),
                      goal.states = 15)

model <- keras_model_sequential() %>%
  layer_dense(units = 4, input_shape = 1,
             activation = "linear") %>%
  compile(optimizer = optimizer_sgd(lr = 0.1),
         loss = "mae")

policy <- makePolicy("epsilon.greedy", epsilon = 0.2)
algorithm <- makeAlgorithm("qlearning")
val.fun <- makeValueFunction("neural.network",
                           model = model)
agent <- makeAgent(policy, val.fun, algorithm)

interact(env, agent, n.episodes = 100)
```



USE CASE DEMONSTRATION SUPER MARIO BROS. AI

Overview
States, Actions & Rewards
Training and Results

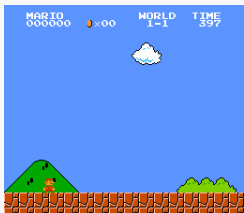
GYM-SUPER-MARIO-BROS

There is a great gym for Super Mario Bros (NES)

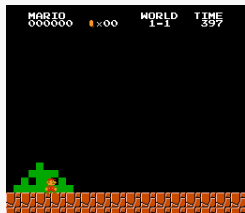
GYM-SUPER-MARIO-BROS

An OpenAI Gym environment for Super Mario Bros. & Super Mario Bros. 2 (Lost Levels) on The Nintendo Entertainment System (NES) using the nes-py emulator.

GAME MODES



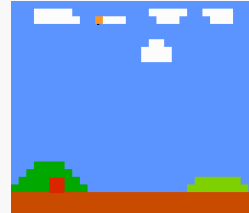
Standard



Downsample



Pixel



Rectangle

Kautenja / gym-super-mario-bros

Watch 8 Star 175 Fork 33

Code Issues 3 Pull requests 0 Projects 0 Wiki Security Insights

An OpenAI Gym interface to Super Mario Bros. & Super Mario Bros. 2 (Lost Levels) on The NES

openai-gym super-mario-bros super-mario-bros-2-lost-levels nes-py

539 commits 1 branch 114 releases 2 contributors View license

Branch: master New pull request Create new file Upload files Find File Clone or download

Kautenja bump version	Latest commit 87ac38c 14 days ago
.github	Update bug_report.md 4 months ago
gym_super_mario_bros	Update smb_env.py 14 days ago
tools	move scripts to dedicated folder last year
.gitignore	cleanup last year
.travis.yml	Update .travis.yml 5 months ago
LICENSE	Update LICENSE 10 months ago
README.md	Update README.md 14 days ago
__main__.py	update main script 11 months ago
makefile	test before deploy 5 months ago
requirements.txt	fix requirements 5 months ago
setup.py	bump version 14 days ago

gym-super-mario-bros

build passing pypi package 7.2.3 python 2.7 | 3.5 | 3.6 status stable format wheel license Proprietary

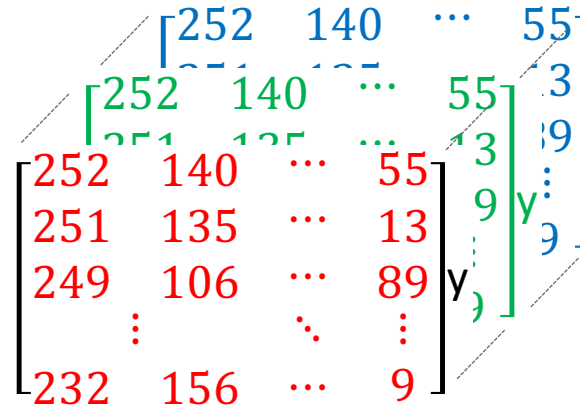
An OpenAI Gym environment for Super Mario Bros. & Super Mario Bros. 2 (Lost Levels) on The Nintendo Entertainment System (NES) using the nes-py emulator.

GAME STATES

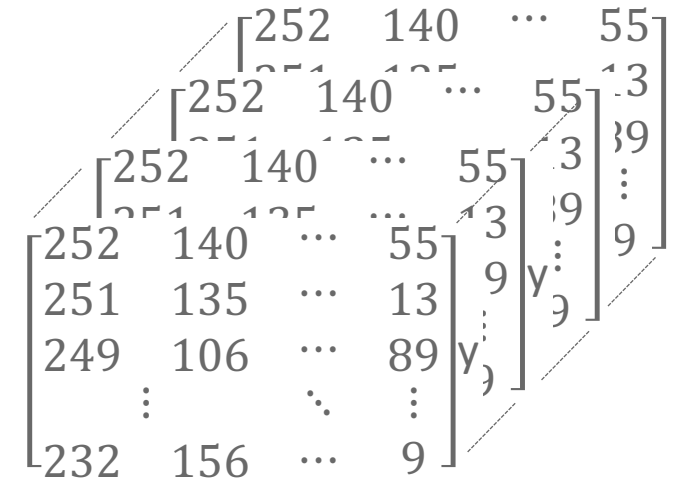
Game states are defined as 4-D tensors containing the last four game states as grayscale matrices



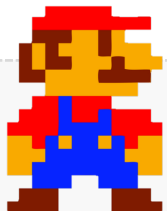
Single Game Screen



$\{n \times p \times c\}$ RGB Pixel Tensor



$\{n \times p \times 4\}$ Grayscale Pixel Tensor



"For video game environments, it is important to provide stacked subsequent game frames to the agent. Otherwise, it would not be possible to detect any "movement" on the screen. Furthermore, we use only every n^{th} frame."

Mario Pro-Tip

ACTIONS

Mario's action space is mapped to integers

CONTROL CROSS

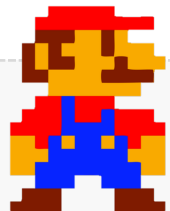
Used to control Mario into 8 different directions: right, right-down, down, down-left, left, left-top, top, top-right.

START / SELECT

Usually used to pause / start / exit the game.

BUTTONS

Used for jumping (A) and running (B hold).



"During training, controller actions (and their combinations) are mapped to integers, usually limited to game-relevant actions. This is done by using wrapper functions around the environment."

Mario Pro-Tip

REWARDS

The game's reward is composed of three different components

$$r = v + c + d$$

reward *velocity* *clock* *death*

VELOCITY

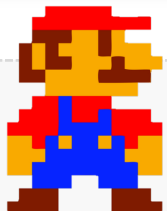
- v : the difference in agent x values between states
- $v = x_1 - x_0$
 - x_0 is the x position before the step
 - x_1 is the x position after the step
- moving right $\Leftrightarrow v > 0$
- moving left $\Leftrightarrow v < 0$
- not moving $\Leftrightarrow v = 0$

CLOCK

- c : the difference in the game clock between frames
- The clock encourages to be fast
- $c = c_0 - c_1$
 - c_0 is the clock before the step
 - c_1 is the clock after the step
- no clock tick $\Leftrightarrow c = 0$
- clock tick $\Leftrightarrow c < 0$

DEATH

- d : a death penalty that penalizes the agent for dying in a state
- this penalty encourages the agent to avoid death
- alive $\Leftrightarrow d = 0$
- dead $\Leftrightarrow d = -15$

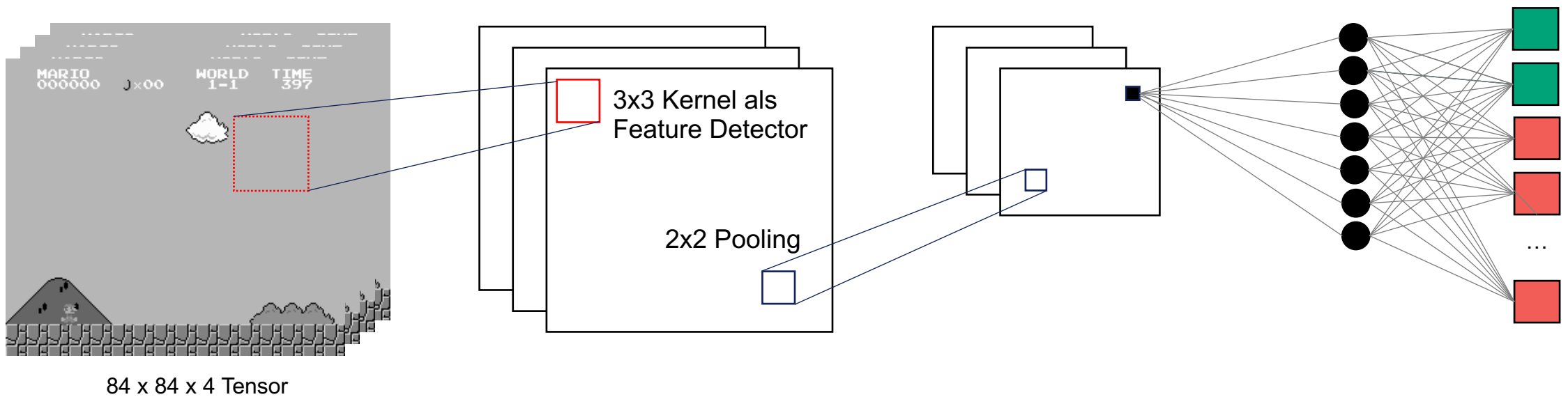


"The reward function assumes the objective of the game is to move as far right as possible (increase the agent's x value), as fast as possible (decrease time), without dying."

Mario Pro-Tip

MODEL

We are using a deep CNN to approximate the Q-value function of our agent



INPUT

We are using the last 4 game frames as input tensor to the neural network.

2D CONVOLUTIONS

Convolutional layers extract relevant information from the game screens.

POOLED FEATURE MAPS

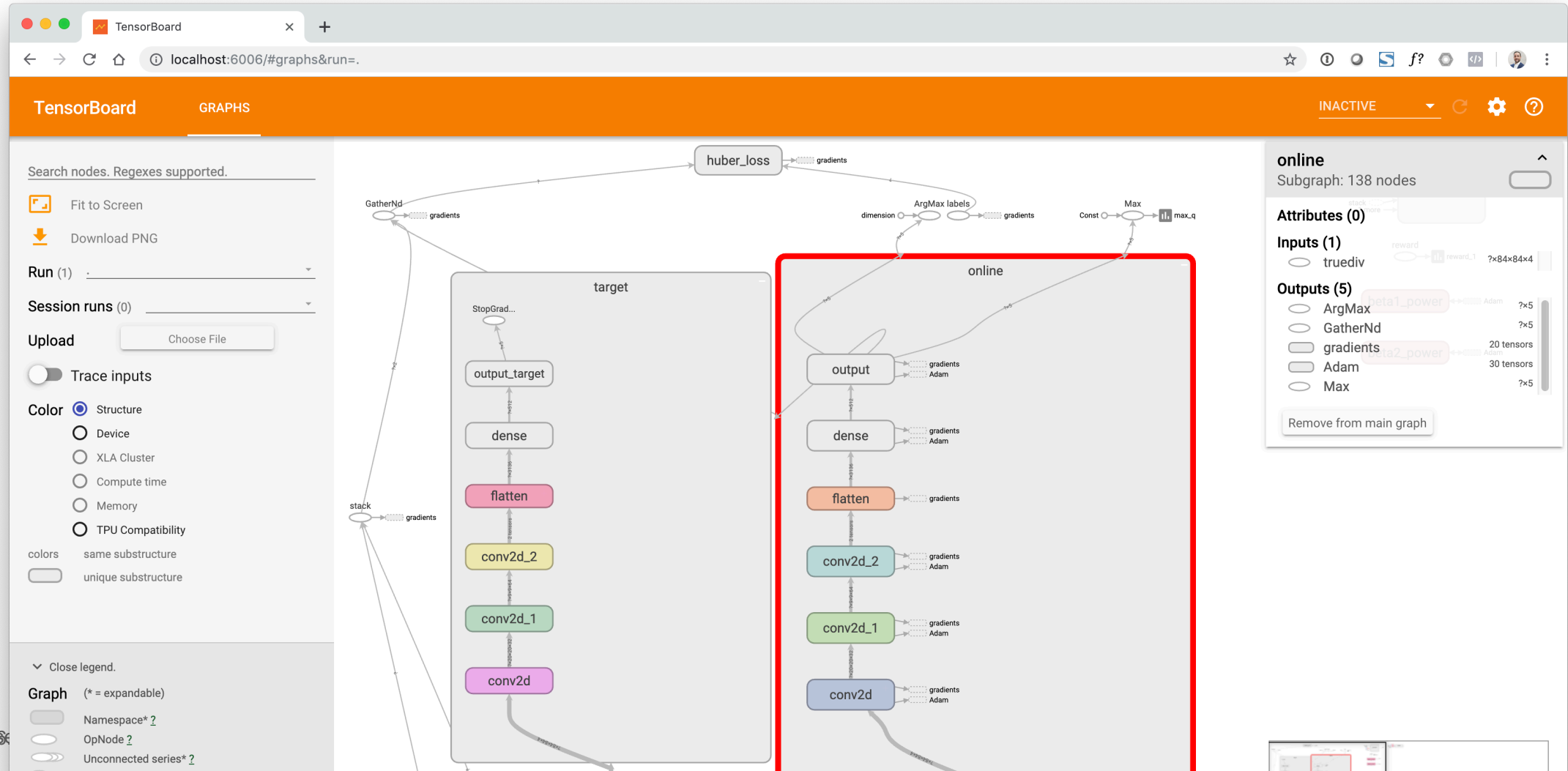
Pooling compresses information and shrinks the dimensionality of the problem.

DENSE & OUTPUT

The network approximates the Q-values by comparing the

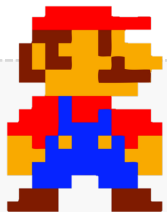
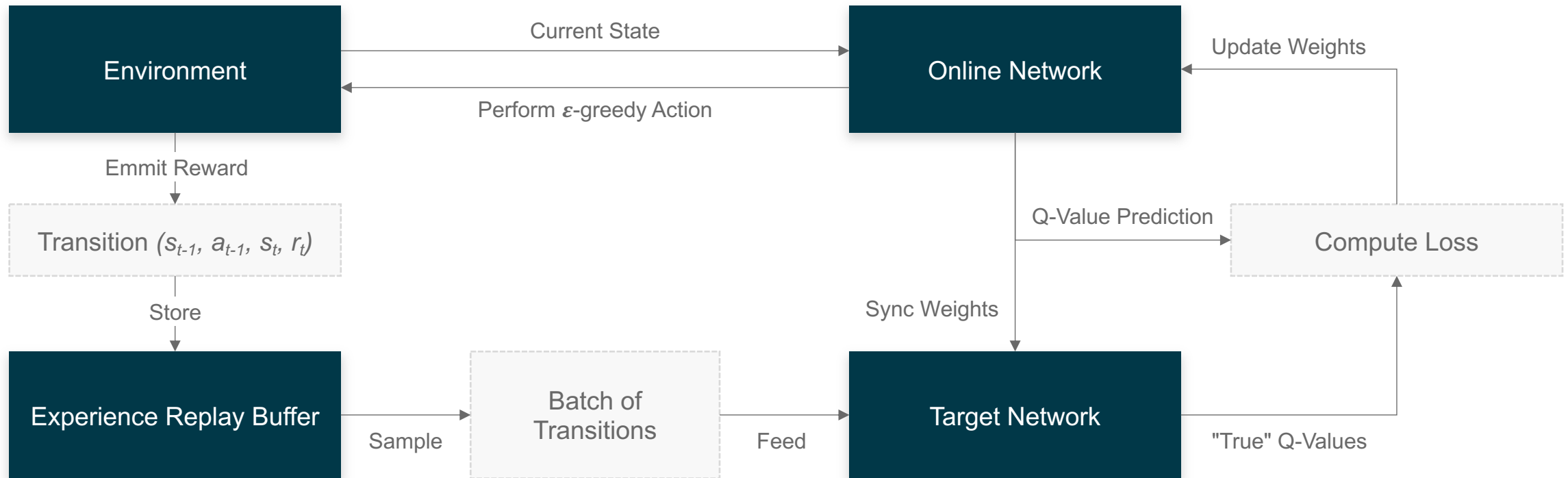
MODEL

A cool screenshot of our model architecture in TensorBoard 😊



TRAINING PIPELINE

Overview of the actual training process of the agent



"Experience replay is needed because subsequent game states are highly correlated which violates the i.i.d. assumption of stochastic gradient descent, which is used for updating the online network's weights."

Mario Pro-Tip

RESULTS

Watch the Super Mario Agent in action as it masters the first level of the game



LIVE DEMO

Learning to play Super Mario using R and reinforcelearn (well, kind of) 😊

The screenshot displays the RStudio interface. The main editor window shows the R script 'mario.R' with the following code:

```
1 library(reinforcelearn)
2 library(reticulate)
3 library(keras)
4 library(imager)
5 import("gym_super_mario_bros")
6
7
8 # Define CNN
9 model <- keras_model_sequential() %>%
10   layer_reshape(input_shape = 4096L, target_shape = c(64, 64, 1)) %>%
11   layer_conv_2d(filters = 32, kernel_size = c(3, 3)) %>%
12   layer_conv_2d(filters = 64, kernel_size = c(3, 3)) %>%
13   layer_conv_2d(filters = 64, kernel_size = c(3, 3)) %>%
14   layer_flatten() %>%
15   layer_dense(units = 512L, activation = "relu") %>%
16   layer_dense(units = 6L, activation = "linear") %>%
17   compile(optimizer = "adam", loss = "mae")
18
19 # An ugly hack due to bug in package ;)
20 class(model)[7] <- "keras.models.Sequential"
21
22 # Value function
23 value <- makeValueFunction("neural.network", model = model)
24
25 # Make environment
26 env <- makeEnvironment("gym", gym.name = "SuperMarioBros-1-1-v0")
27
28 # Epsilon greedy policy
29 policy <- makePolicy("epsilon.greedy", epsilon = 0.1)
30
31 # Algorithm
32 algorithm <- makeAlgorithm("qlearning")
33
34 # Experience replay memory
35 memory <- makeReplayMemory(size = 50000, batch.size = 32L)
36
37 # Preprocess
38 # For neural network training the outcome of preprocess must be a one-row matrix in order to be able to learn.
39 preprocess <- function(state) {
40   img <- image_array_resize(state, height = 64, width = 64, data_format = "channels_last")
41   img <- (img[, ,1] + img[, ,2] + img[, ,3]) / 3
42 }
```

The Environment pane on the right shows the following objects:

Object	Type
agent	Environment
algorithm	List of 2
env	Environment
memory	List of 2
policy	List of 2
result	Large list (3 elements, 720.8 Kb)
value	List of 2
action	3L
i	100L
model	Model
reward	0
state	Large array (184320 elements, 720.2 Kb)
preprocess	function (state)

The bottom pane shows the R documentation for the `interact` function from the `reinforcelearn` package:

Interaction between agent and environment.

Description

Run interaction between agent and environment for specified number of steps or episodes.

Usage

```
interact(env, agent, n.steps = Inf, n.episodes = Inf,
         max.steps.per.episode = Inf, learn = TRUE, visualize = FALSE)
```

Live Demo

KEY FINDINGS

What we have learned

1. Reinforcement learning is a very promising area of ML / AI research
2. R is still quite limited in the area of modern reinforcement learning methods
3. The `reinforce` learn package is not up-2-date when it comes to network policies
4. `reticulate` is a way to play around with RL for the moment but not the final solution
5. If you are considering diving deeper into RL, take a look at Python ;)

WANT TO LEARN MORE?

Join one of our next training events!

STATWORX

STATWORX

Data Science Bootcamp

A 5-day introduction to the field of data science and machine learning.

July 1st – July 5th
STATWORX Office
Zurich, Switzerland
German Language

www.statworx.com/ch/academy

JOIN TRAINING

STATWORX

Data University

A 2-day immersive learning experience with 24 sessions from can be selected.

October 9th – October 10th
Goethe University
Frankfurt, Germany
German Language

www.data-university.de

JOIN TRAINING

STATWORX

Deep Learning Bootcamp

A 5-day introduction to the field of deep learning with Keras and TensorFlow.

November 4th – November 8th
STATWORX Office
Frankfurt, Germany
German Language

www.statworx.com/ch/academy

JOIN TRAINING

Q&A and Discussion

THANK YOU
FOR THE ATTENTION.

CONTACT

Sebastian Heinz, CEO
sebastian.heinz@statworx.com
www.statworx.com

STATWORX

Wöhlerstr. 8-10
60323 Frankfurt am Main