Hochschule Düsseldorf
University of Applied Sciences

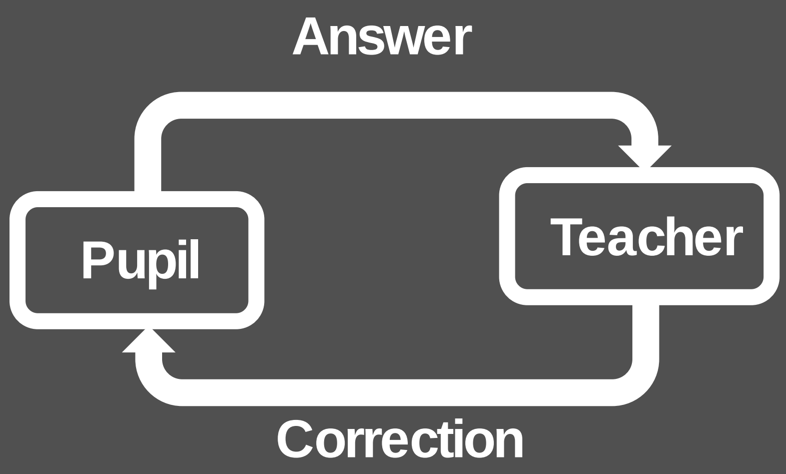
Fachbereich Medien
Faculty of Media

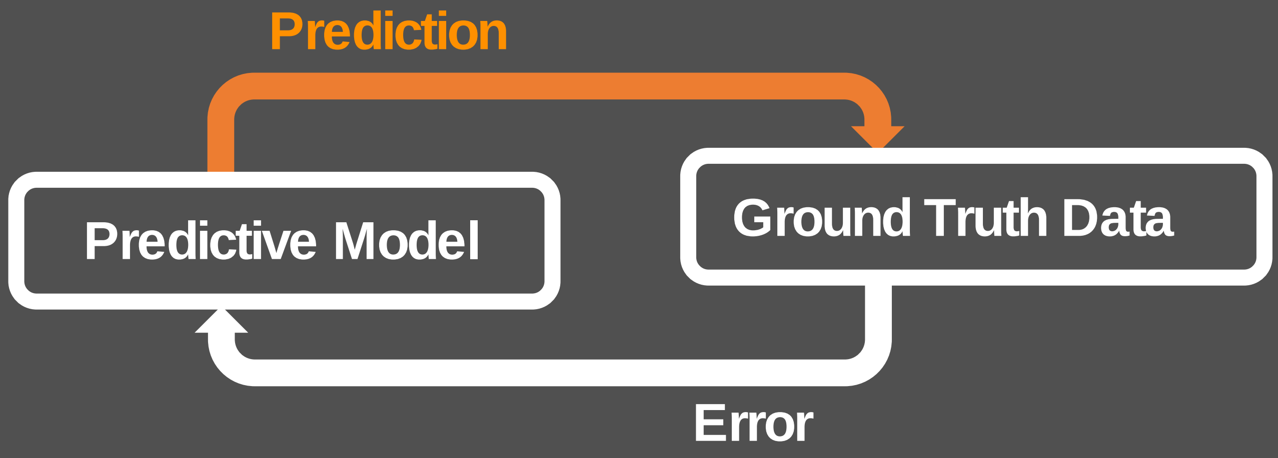
Reinforcement Learning

Introduction

Two Processes:

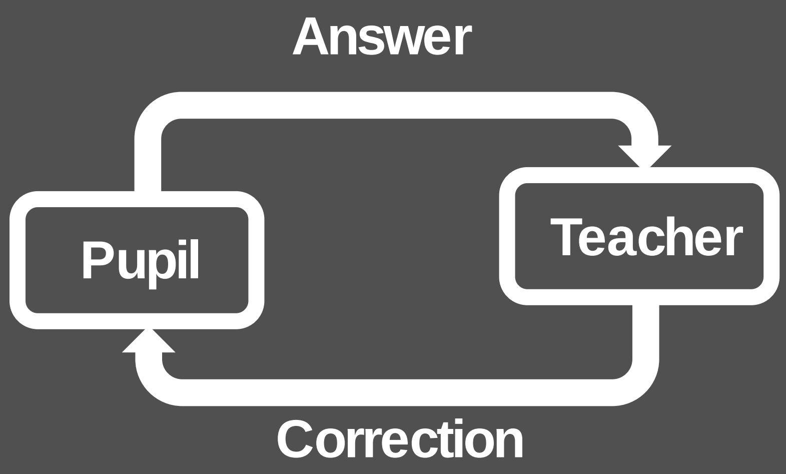
- Forward Propagation (Prediction)
- Backward Propagation (Weight Tuning)

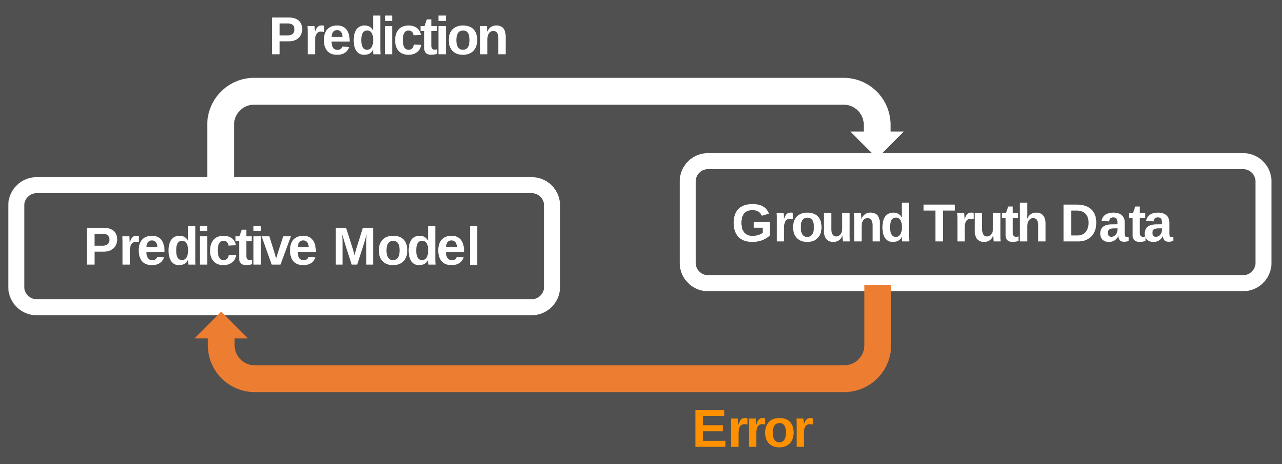




Two Processes:

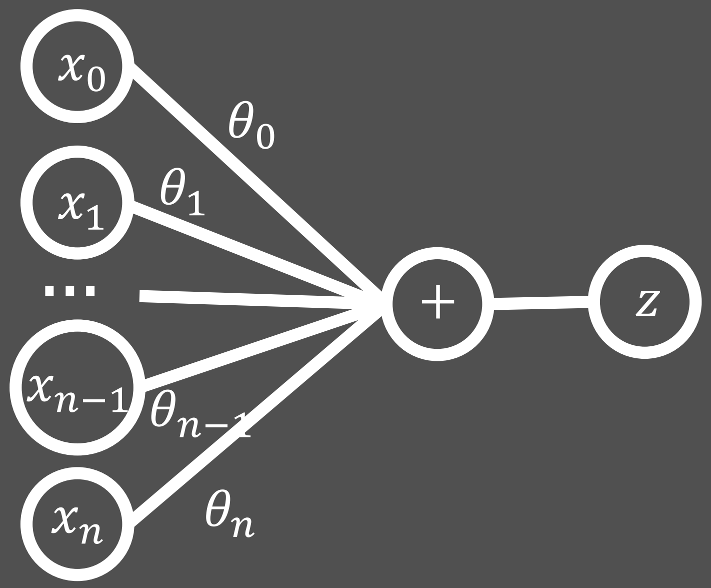
- Forward Propagation (Prediction)
- Backward Propagation (Weight Tuning)





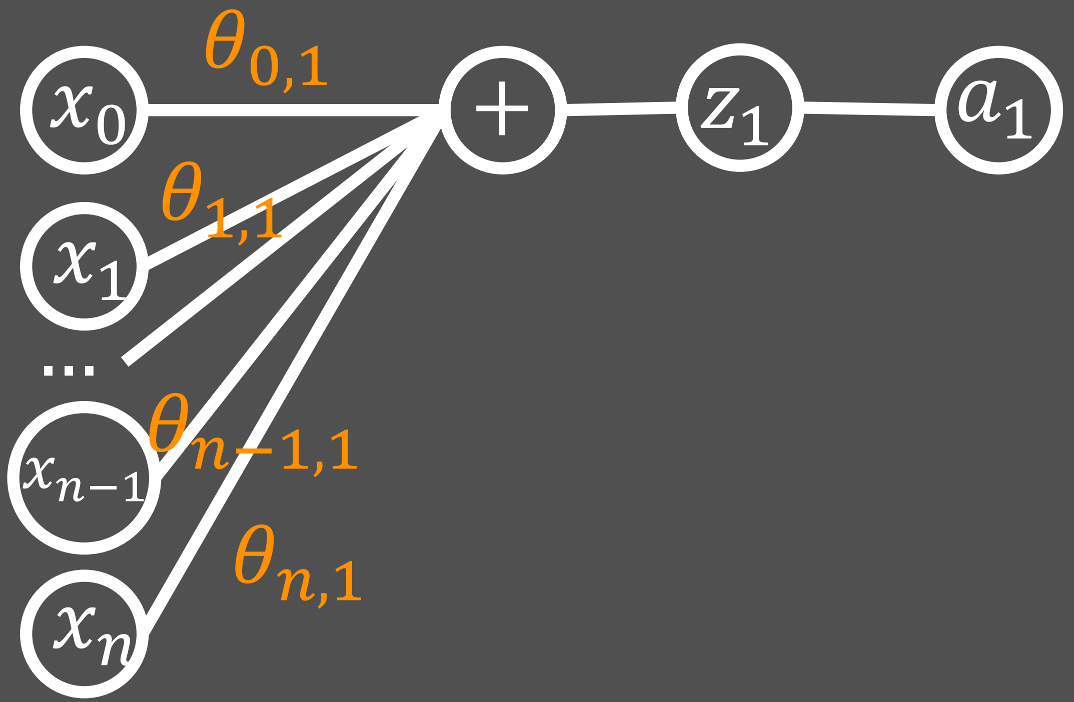
Forward Propagation

- Compute multiple weighted sums z_i
- Activate them to get $a_i(z_i)$
- Remember from Perceptron: $z = \theta x$

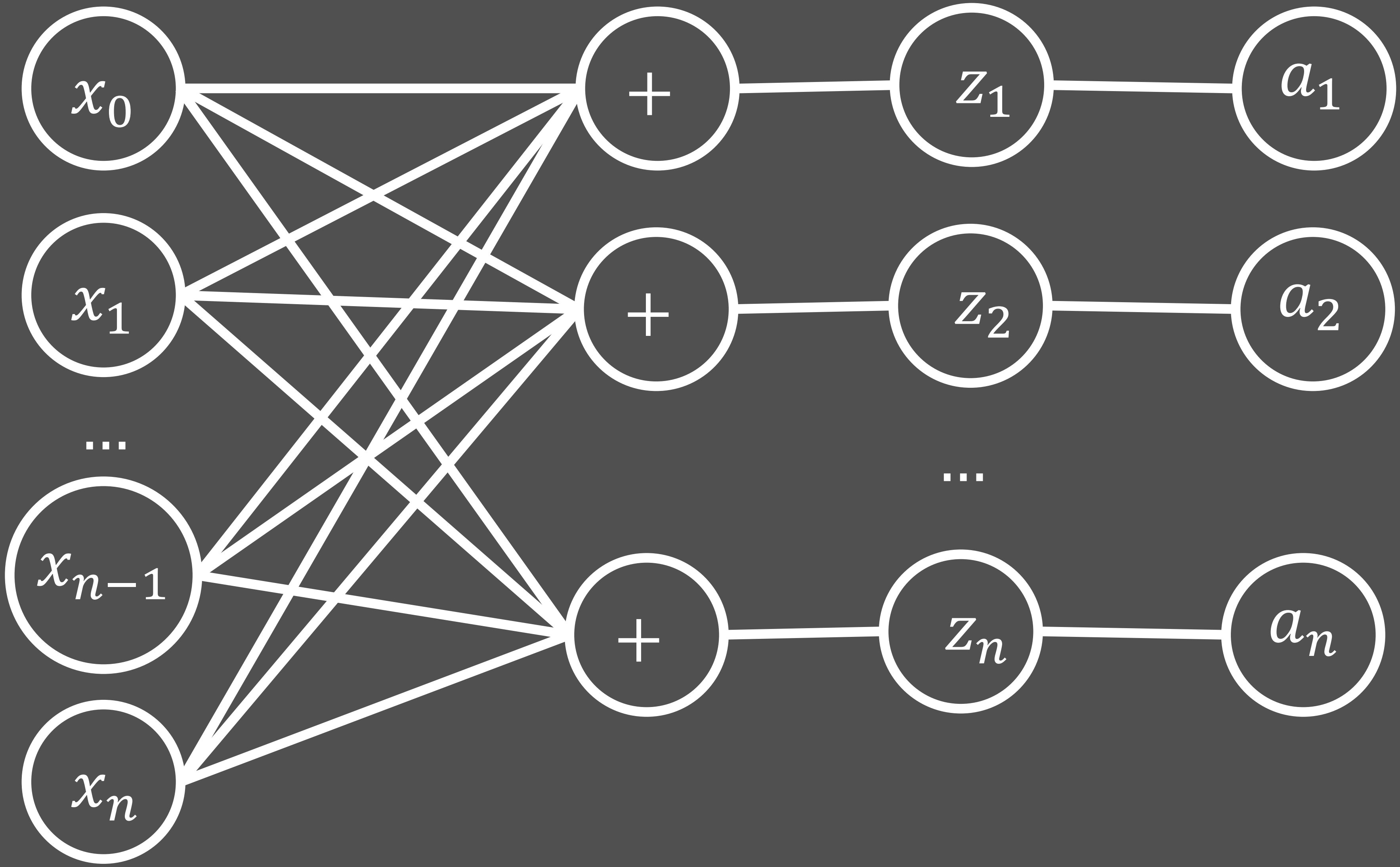


$$z_{to} = \theta_{from,to} x_{from}$$

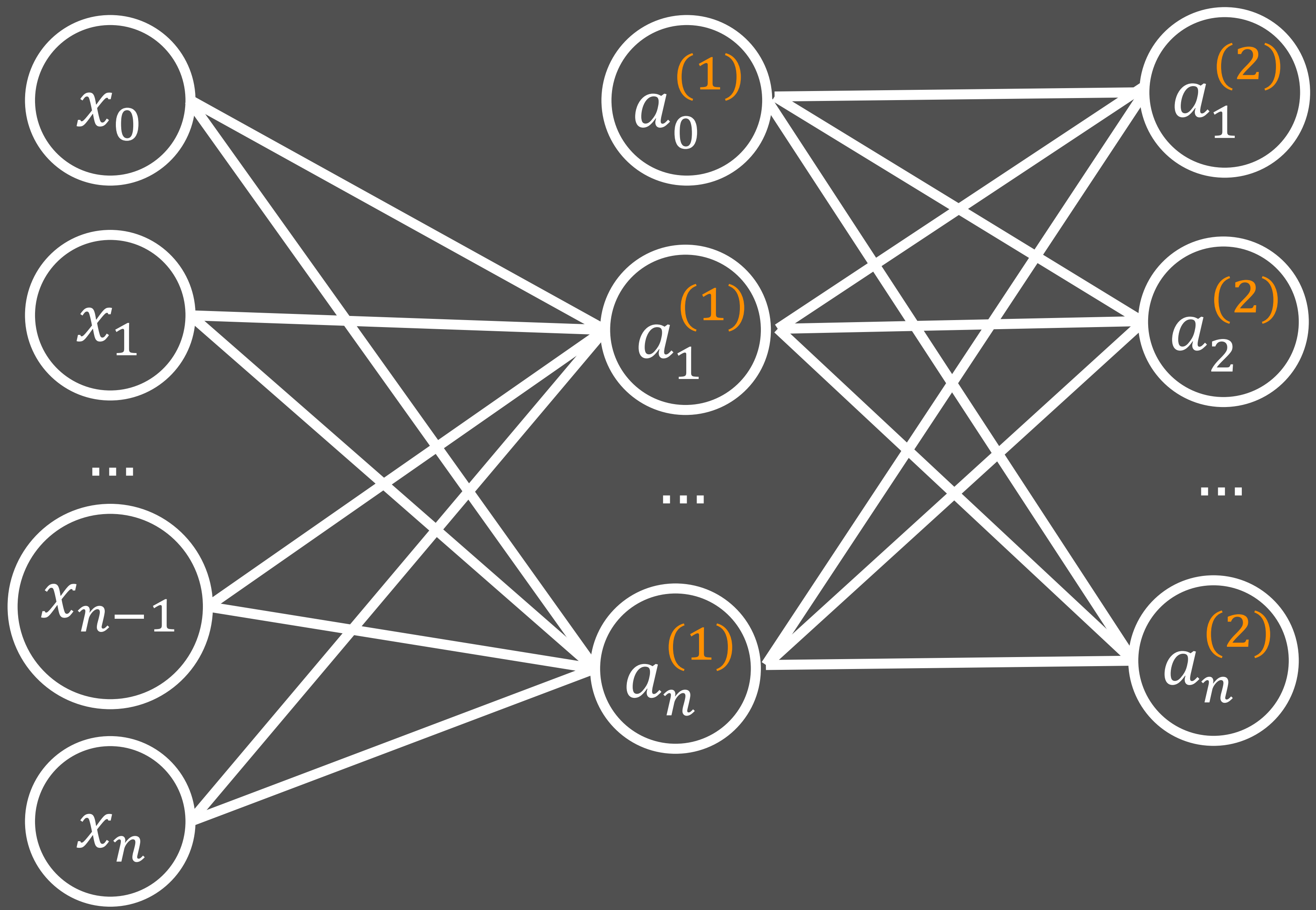
e.g. $z_1 = \theta_{0,1} x_0$



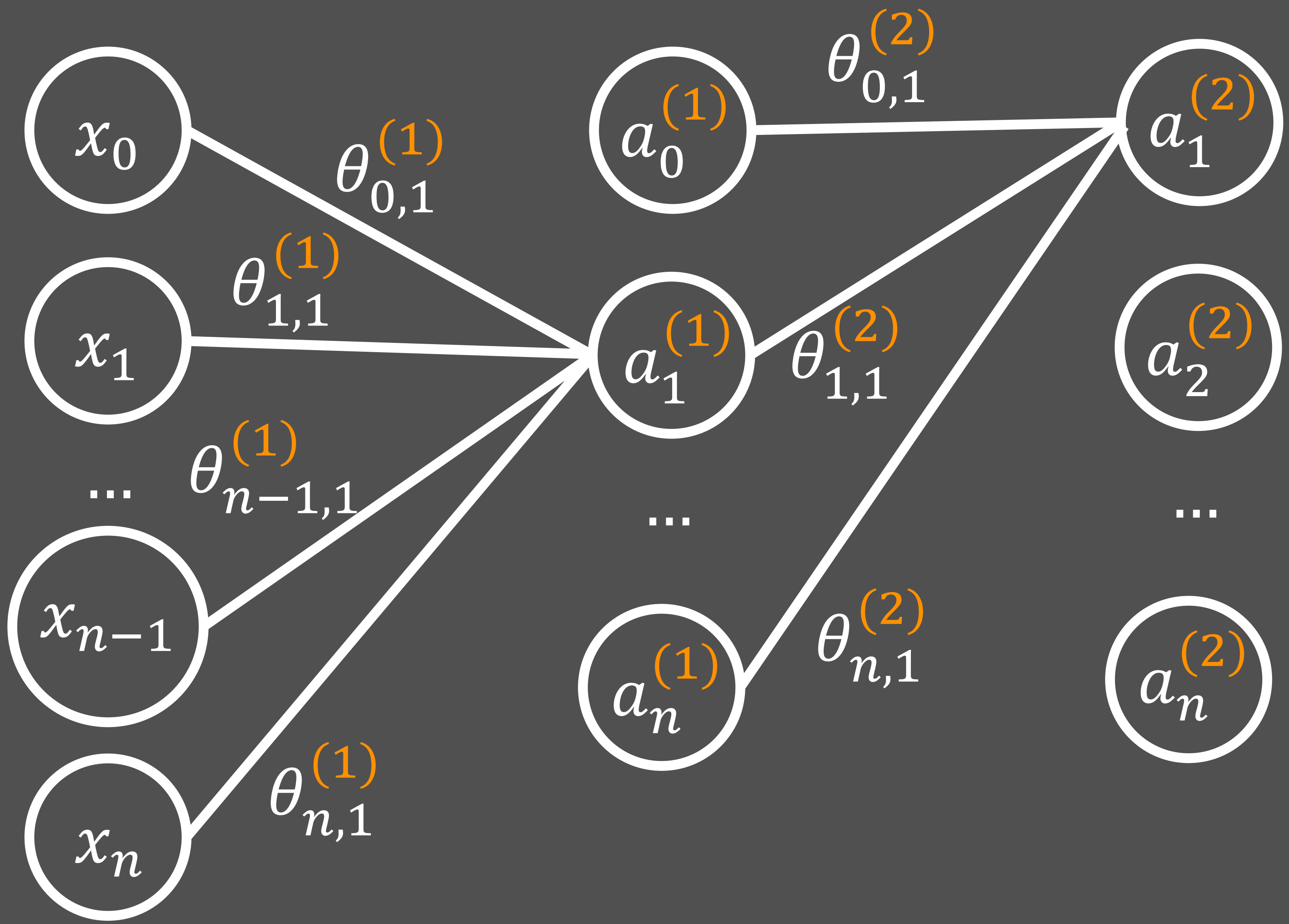
Forward Propagation



Forward Propagation

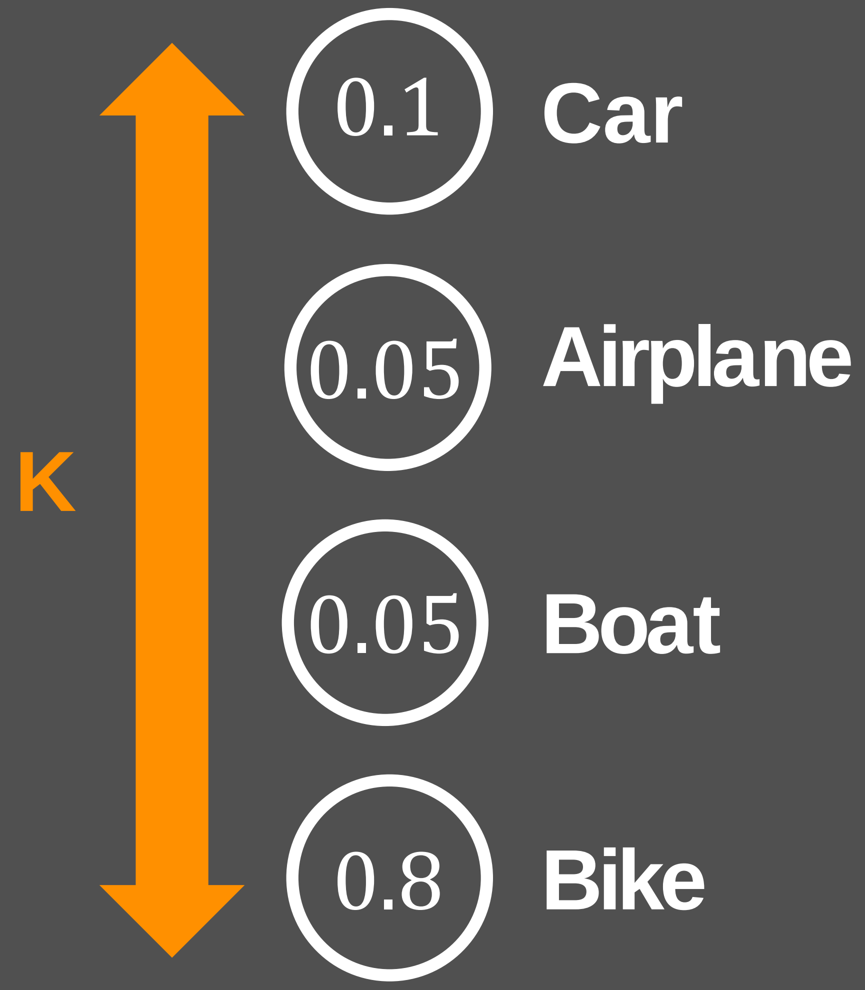


Forward Propagation



Forward Propagation

- Object Recognition
- Assume 4 classes
- L-Layer (L is the last layer)
- Softmax: $a = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$
- Use Softmax for the last Layer



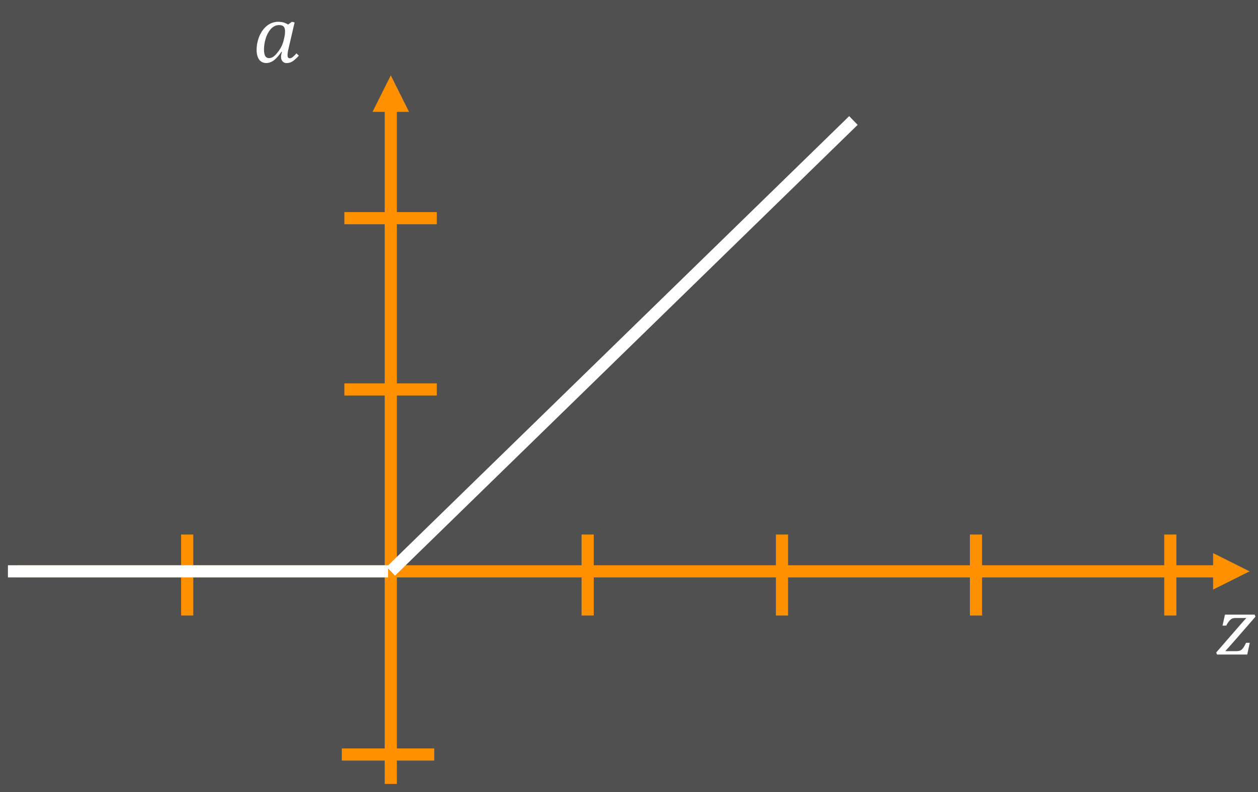
Forward Propagation

Hidden Layer Activation Functions $a(z)$:

- **Linear Neuron:** $a(z) = z$
- **Binary Threshold Unit:** $a(z) = \begin{cases} 0, z < 0 \\ 1, z \geq 0 \end{cases}$
- **Rectified Linear (ReLu):** $\boldsymbol{a(z) = \begin{cases} 0, z < 0 \\ z, z \geq 0 \end{cases}}$
- **Sigmoid:** $a(z) = \frac{1}{1+e^{-z}}$
- **Tanh:** $a(z) = \tanh(z)$

Forward Propagation

Rectified Linear (ReLu): $a(z) = \begin{cases} 0, z < 0 \\ z, z \geq 0 \end{cases}$

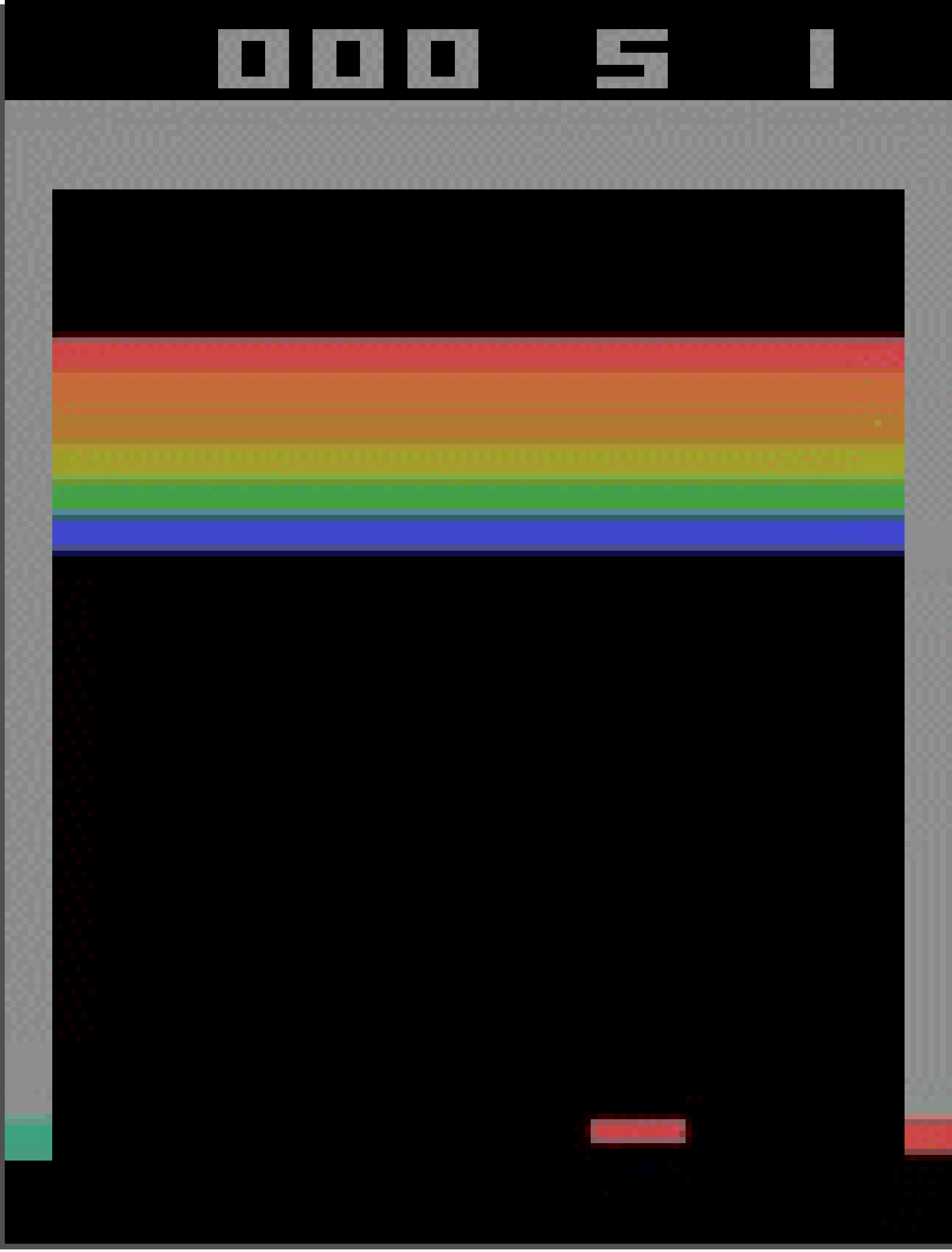


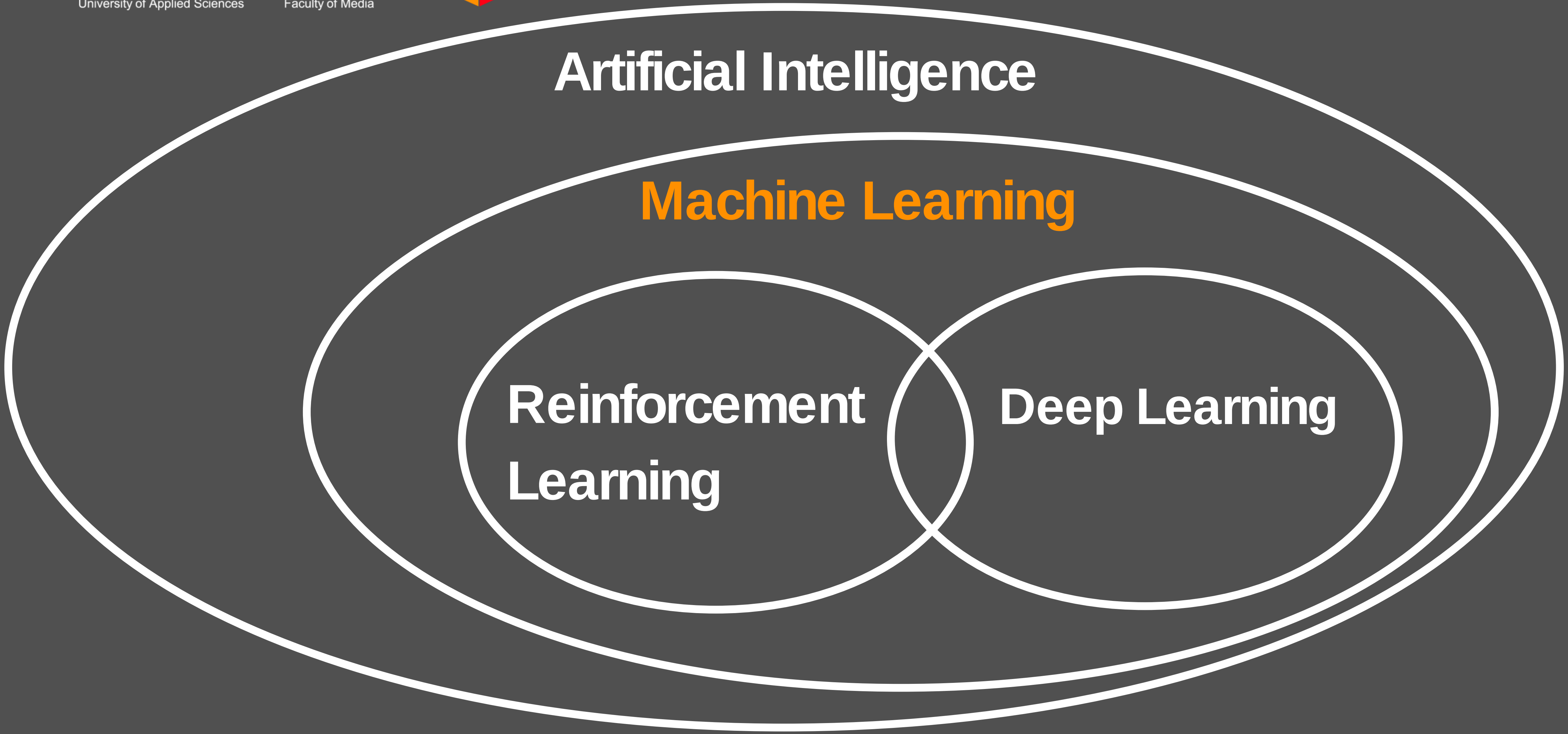
Exercise:

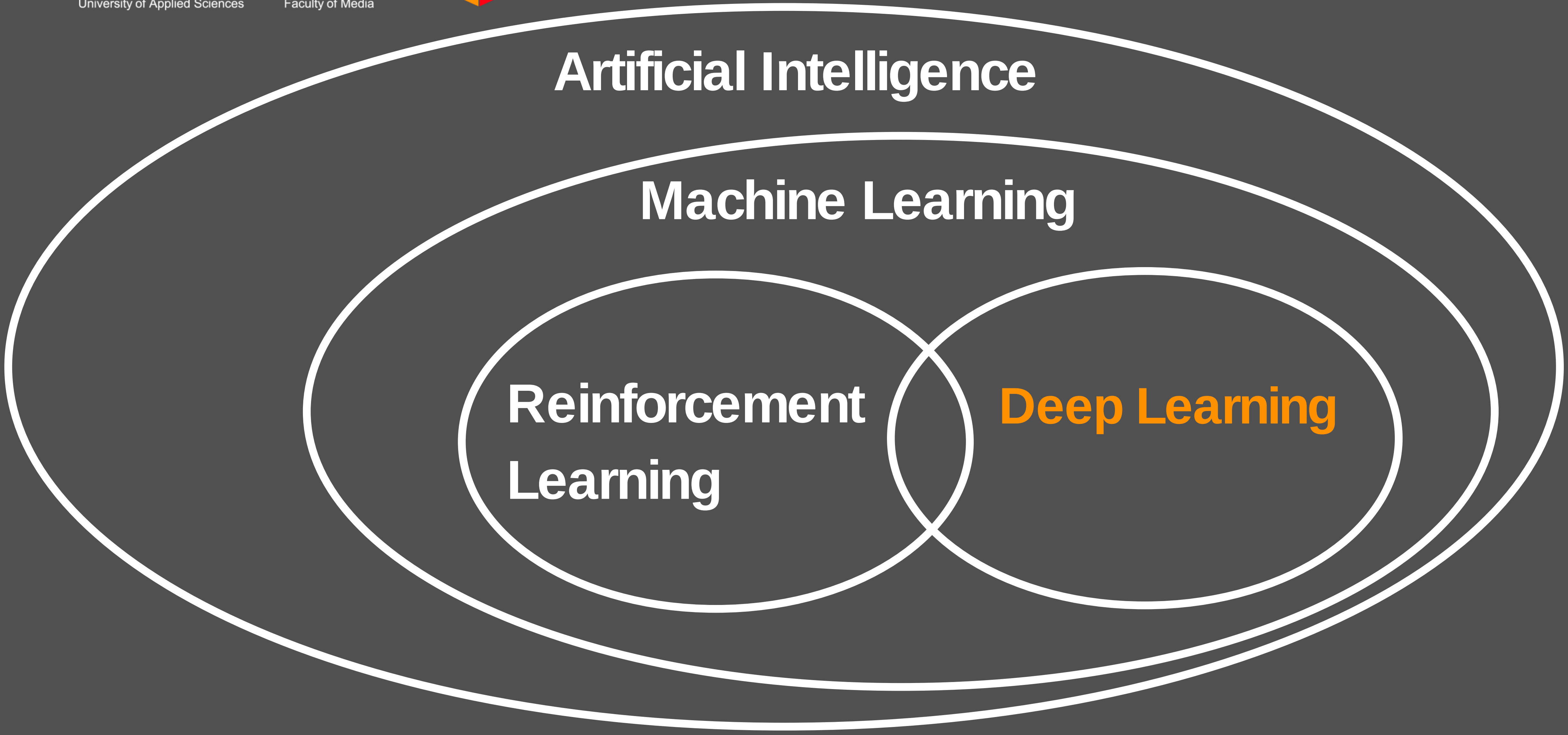
- Construct a MLP (input: 3, hidden: 4, output: 3)
- All activation functions are ReLu-Function
- The output activation is a softmax-function
- Initialize the weights randomly
  - No zero values allowed!
- The input vector is $\mathrm{x} = (1, -2, 1)^T$
- Compute one forward-propagation step!

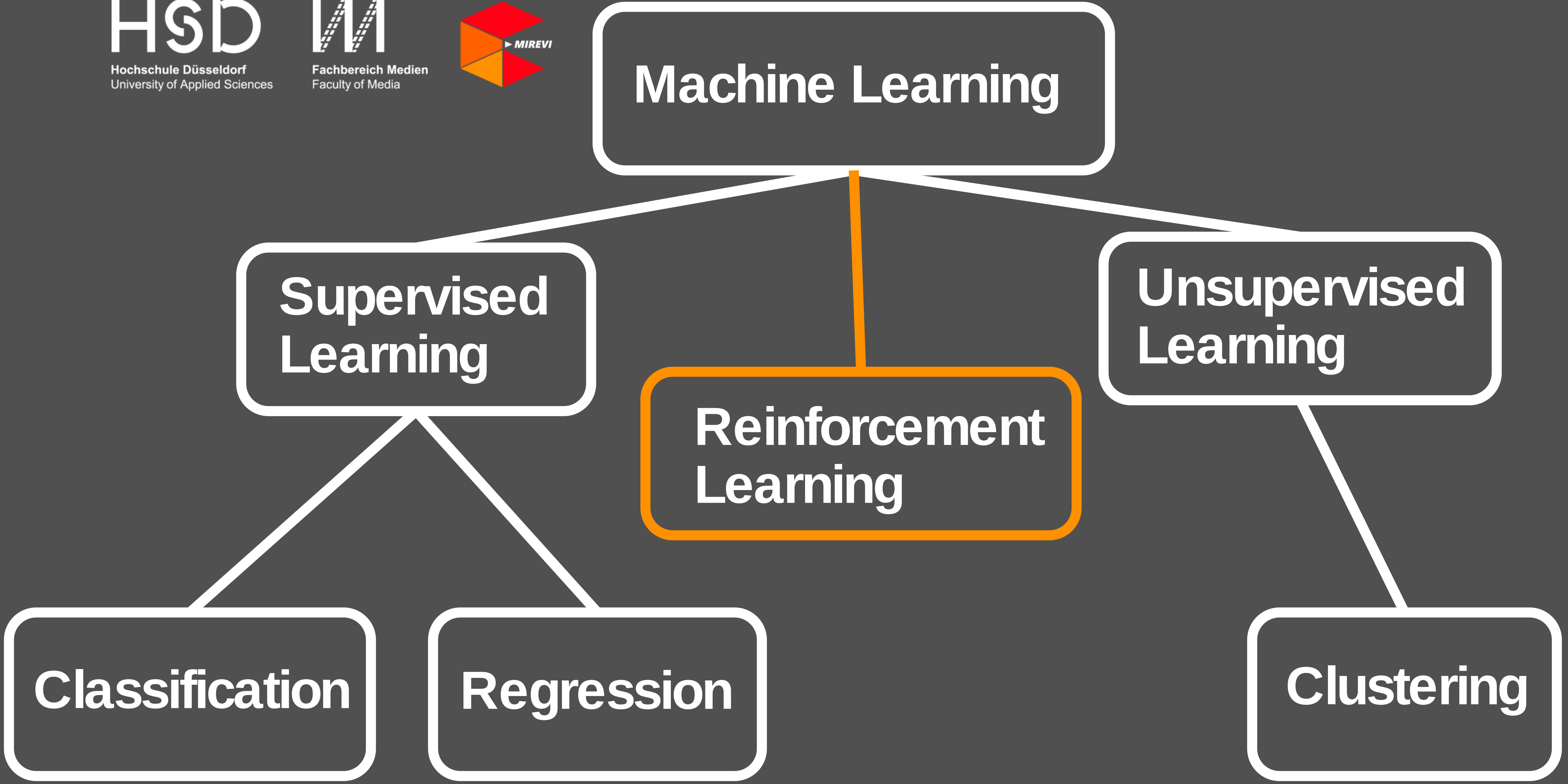
What we will do?

- Applied AI
- Reinforcement Learning
- Train Agent to play Atari-Games

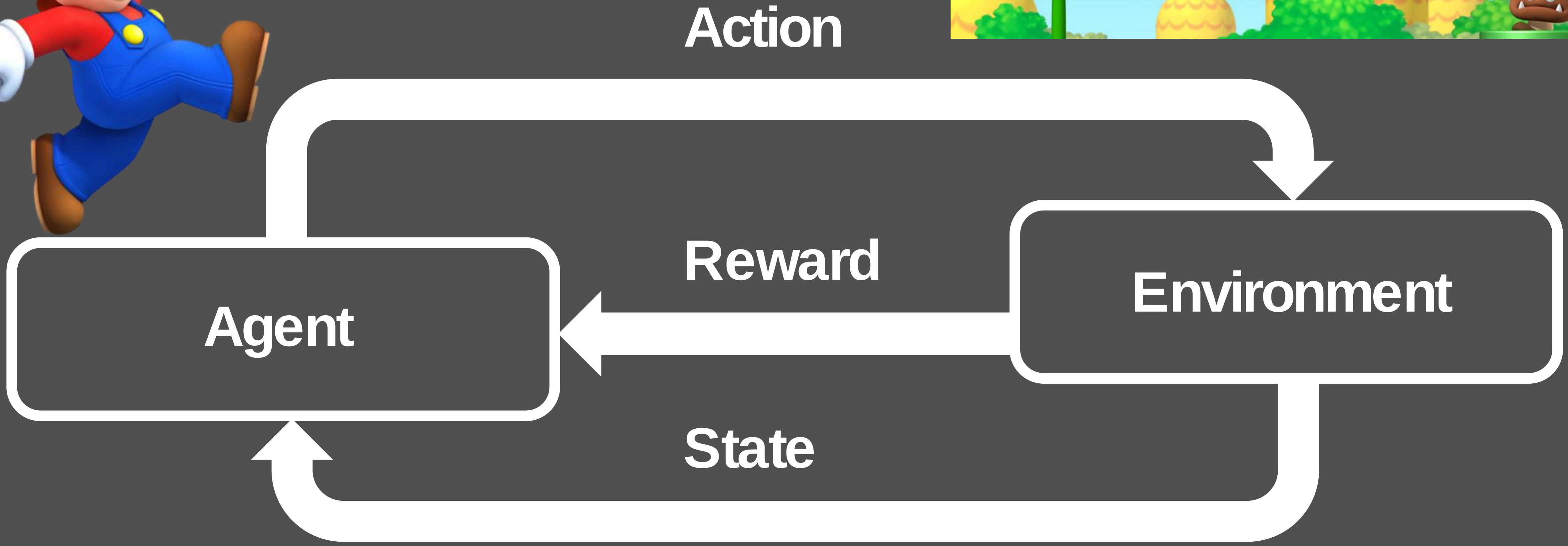



Artificial Intelligence
Machine Learning
Reinforcement Learning
Deep Learning

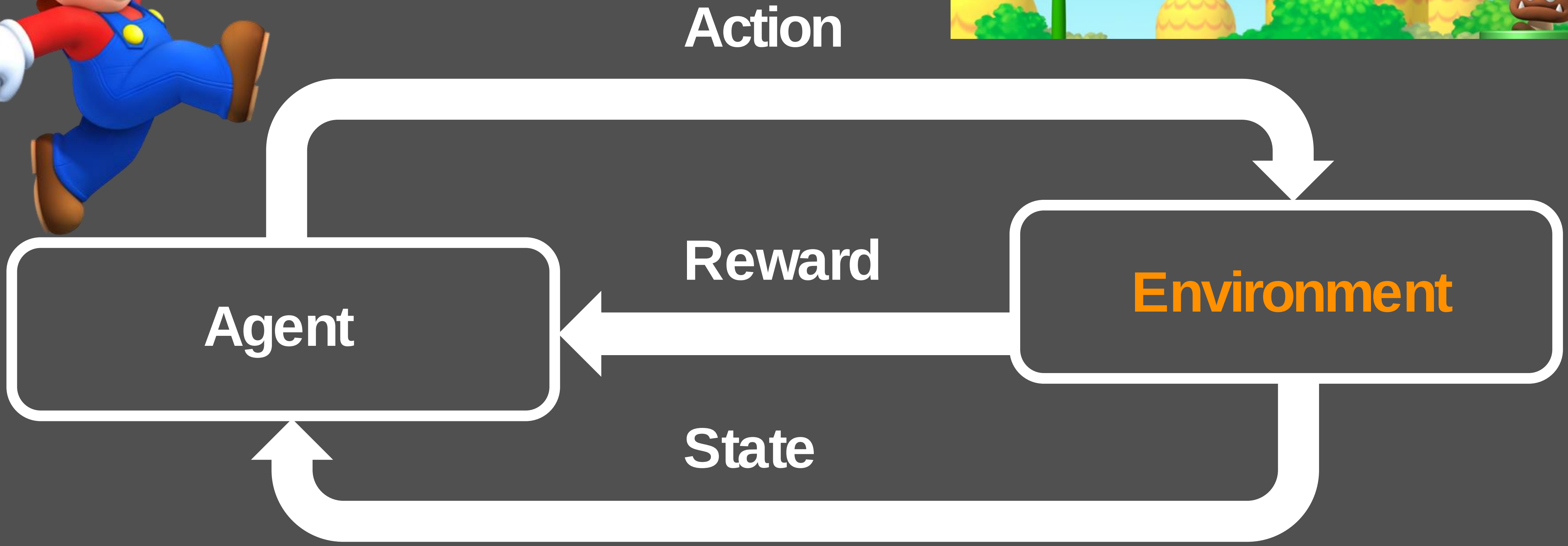

Artificial Intelligence
Machine Learning
Reinforcement Learning
Deep Learning


Machine Learning
Supervised Learning
Reinforcement Learning
Unsupervised Learning
Classification
Regression
Clustering

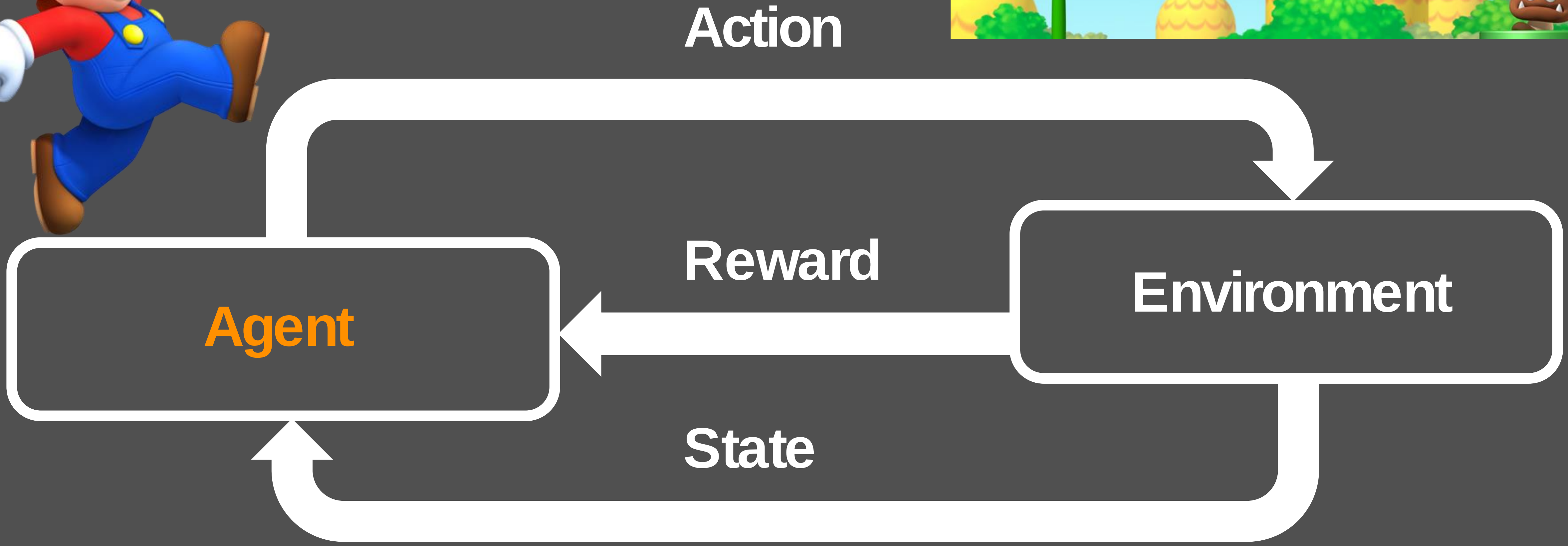
Reinforcement Learning



Reinforcement Learning



Reinforcement Learning

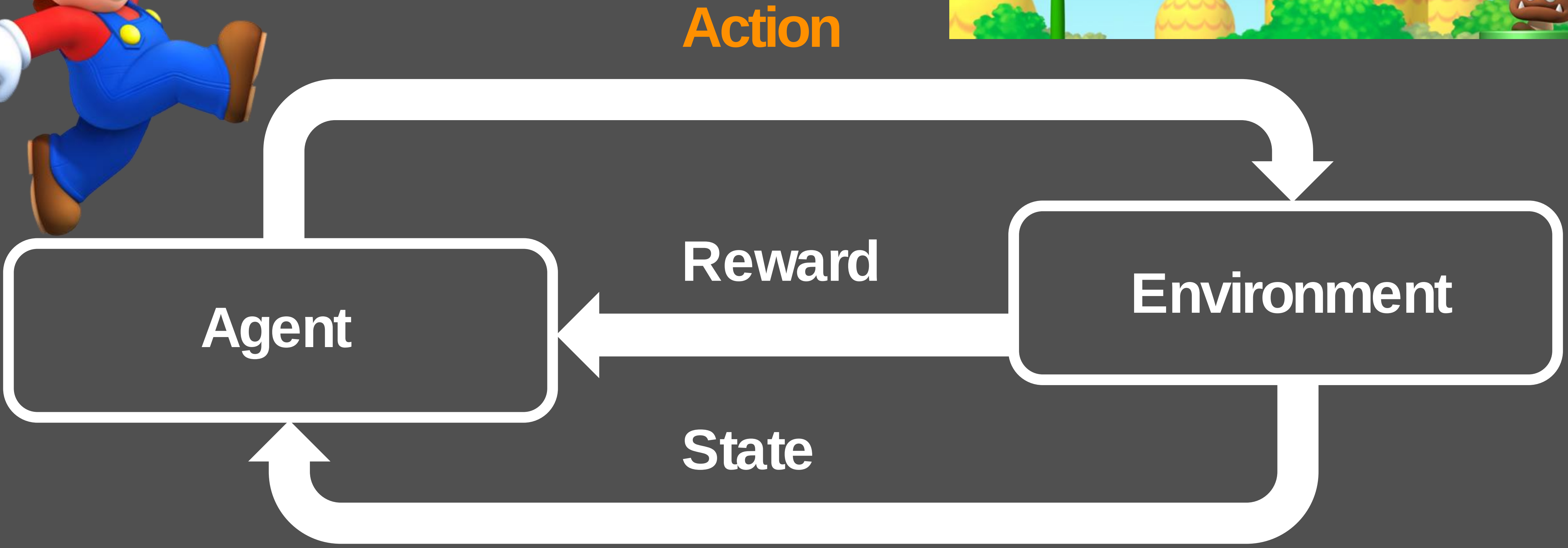


Hochschule Düsseldorf
University of Applied Sciences

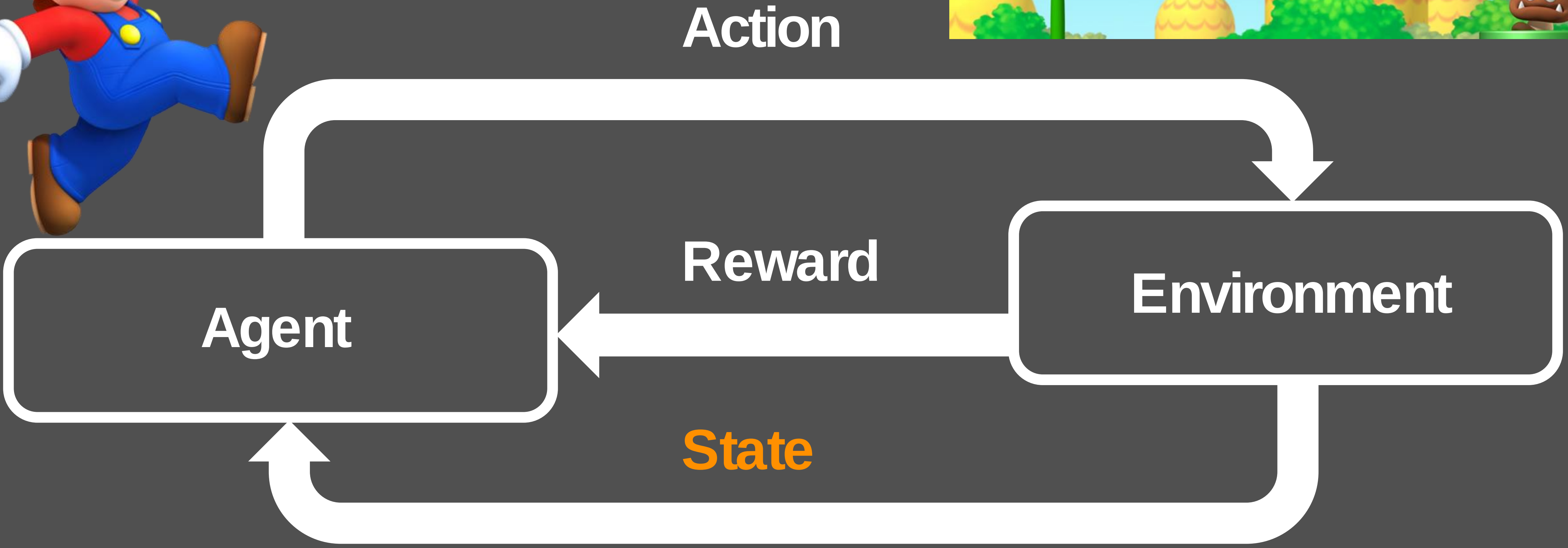
Fachbereich Medien
Faculty of Media

MIREVI

Reinforcement Learning



Reinforcement Learning

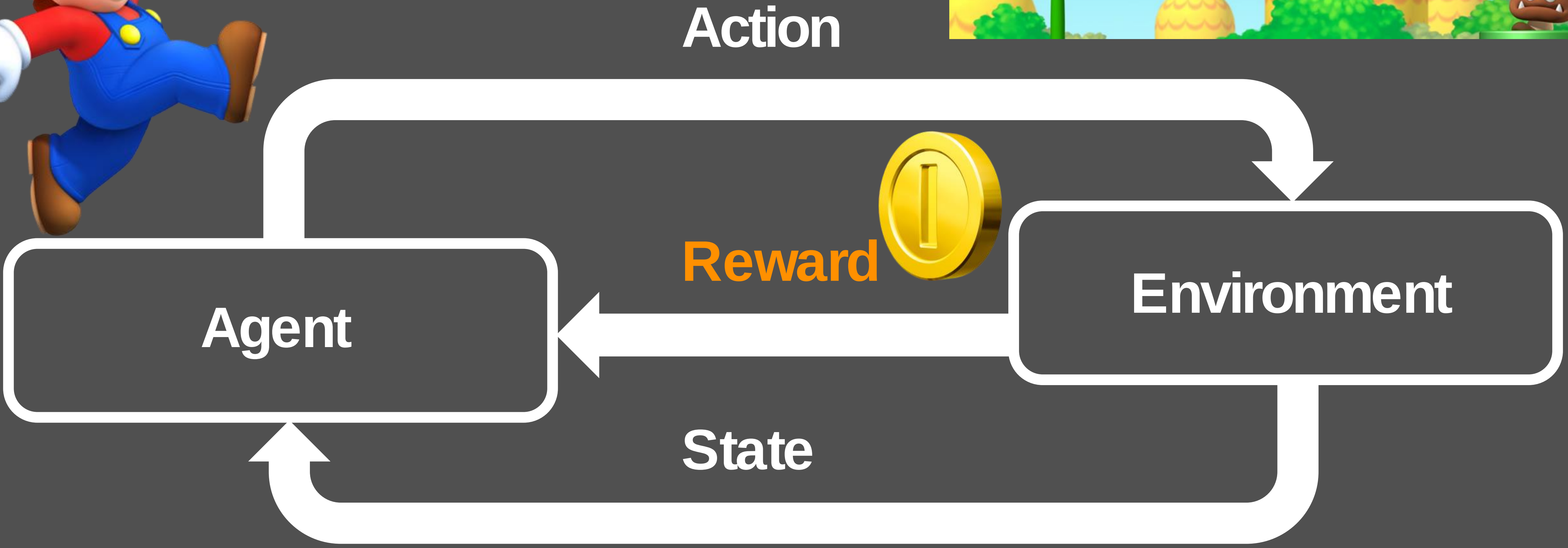


Hochschule Düsseldorf
University of Applied Sciences

Fachbereich Medien
Faculty of Media

MIREVI

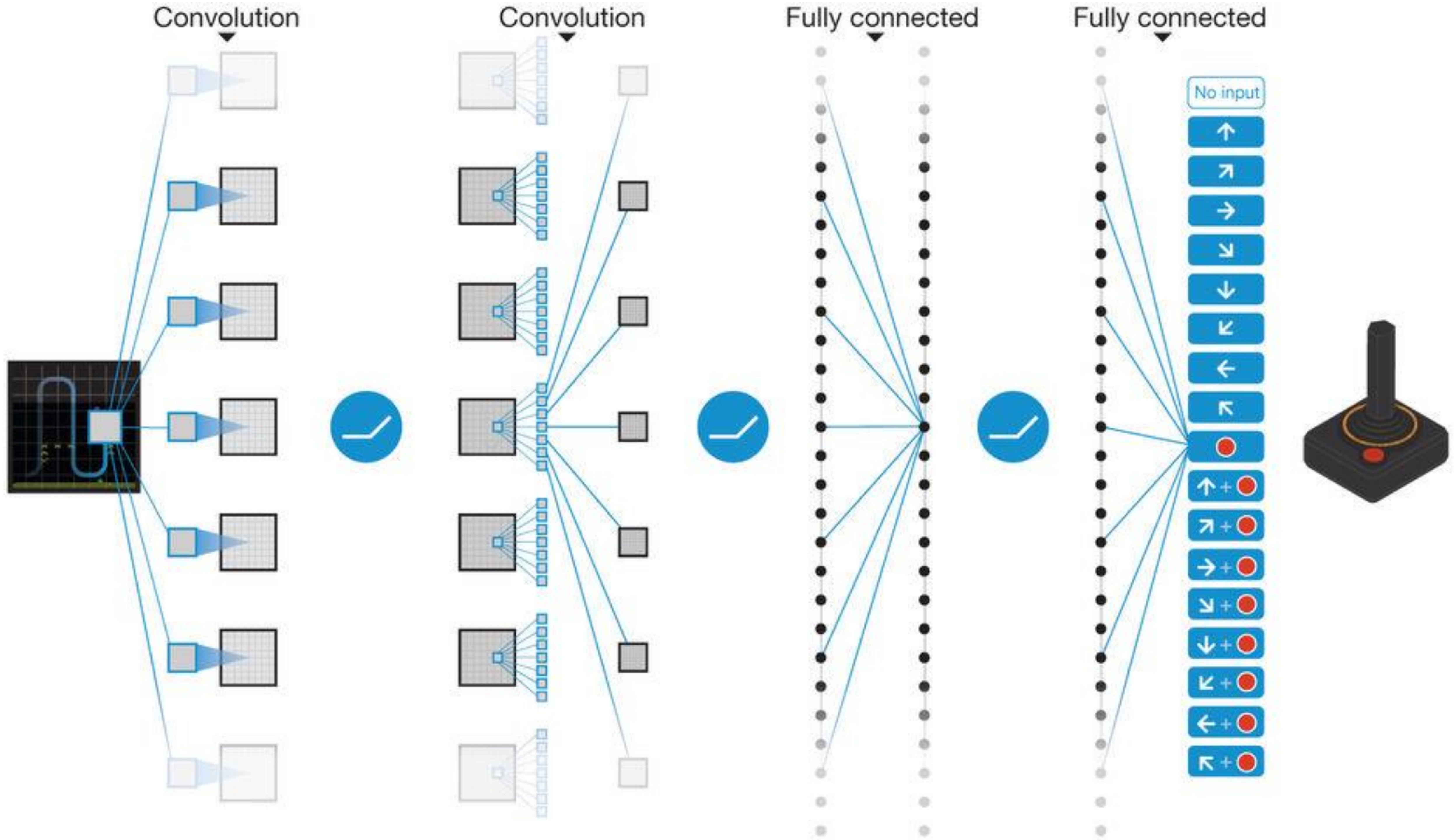
Reinforcement Learning



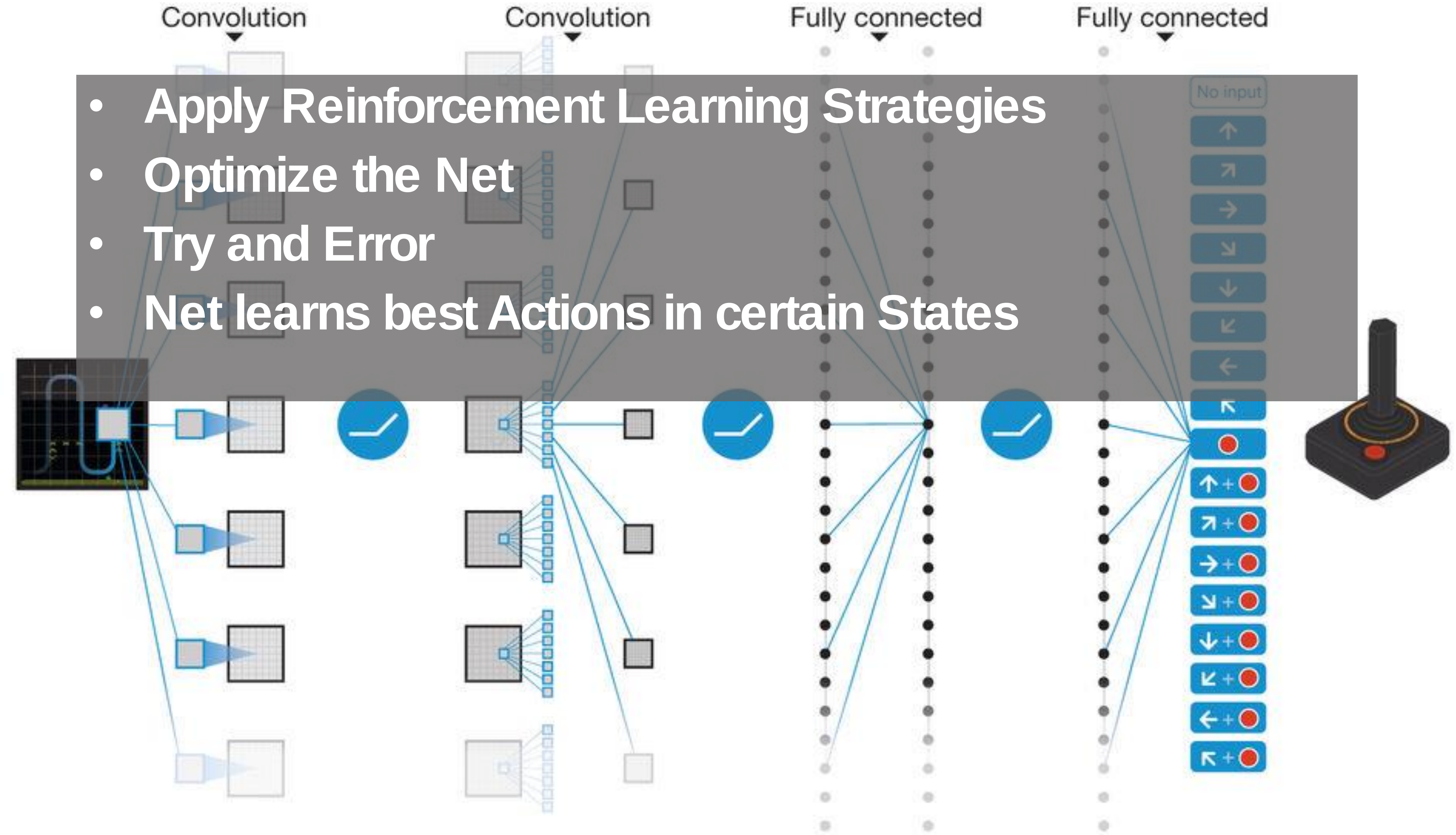
Hochschule Düsseldorf
University of Applied Sciences

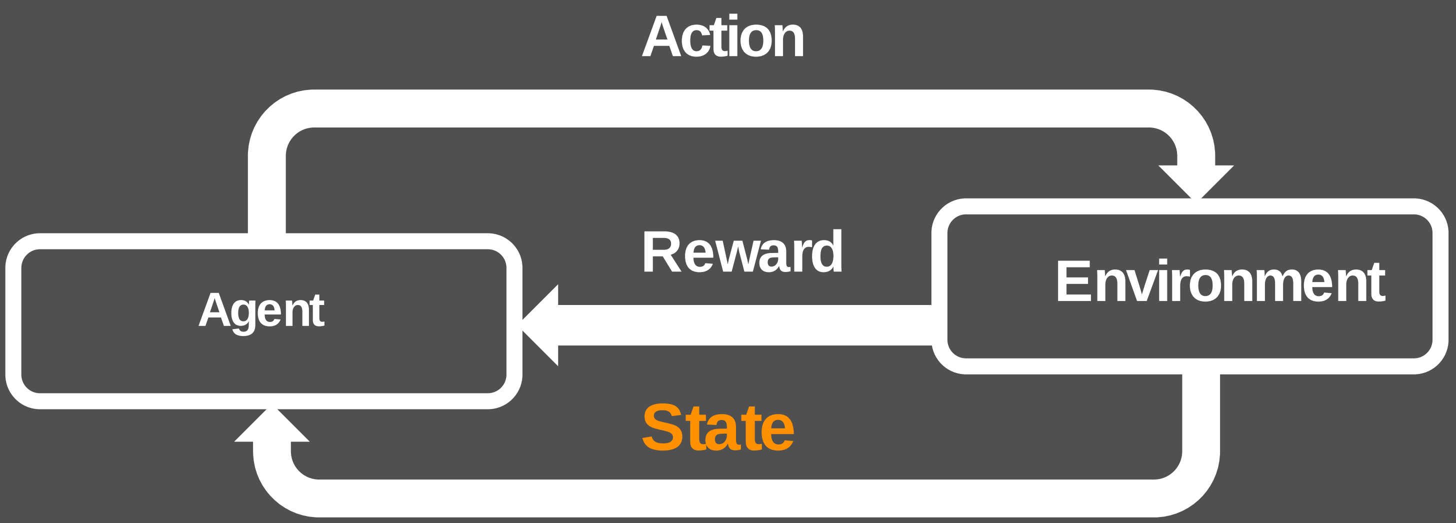
Fachbereich Medien
Faculty of Media

Rewards = Goals

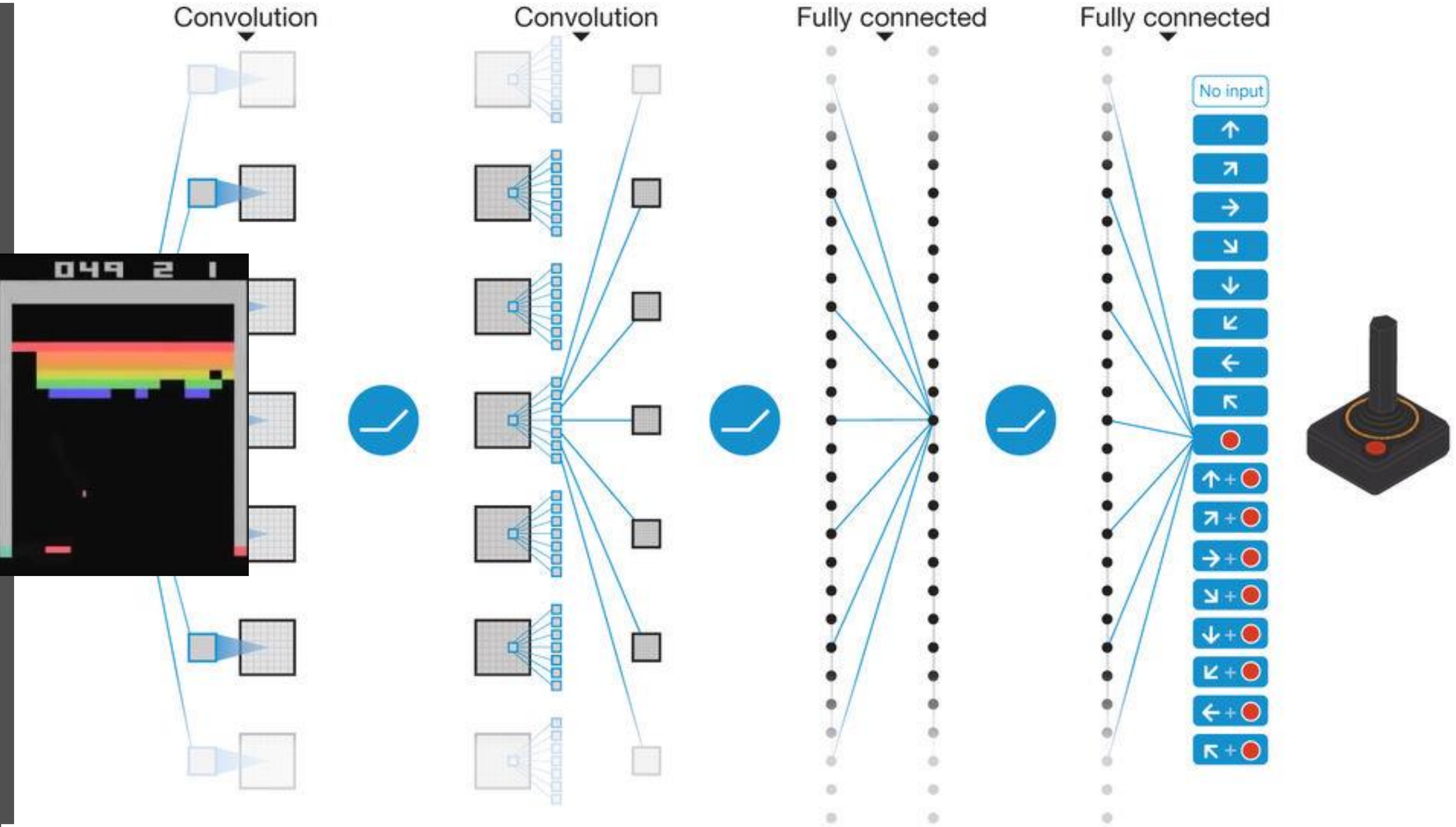

Convolution
Convolution
Fully connected
Fully connected
No input

- Apply Reinforcement Learning Strategies
- Optimize the Net
- Try and Error
- Net learns best Actions in certain States




Action
Reward
Agent
Environment
State

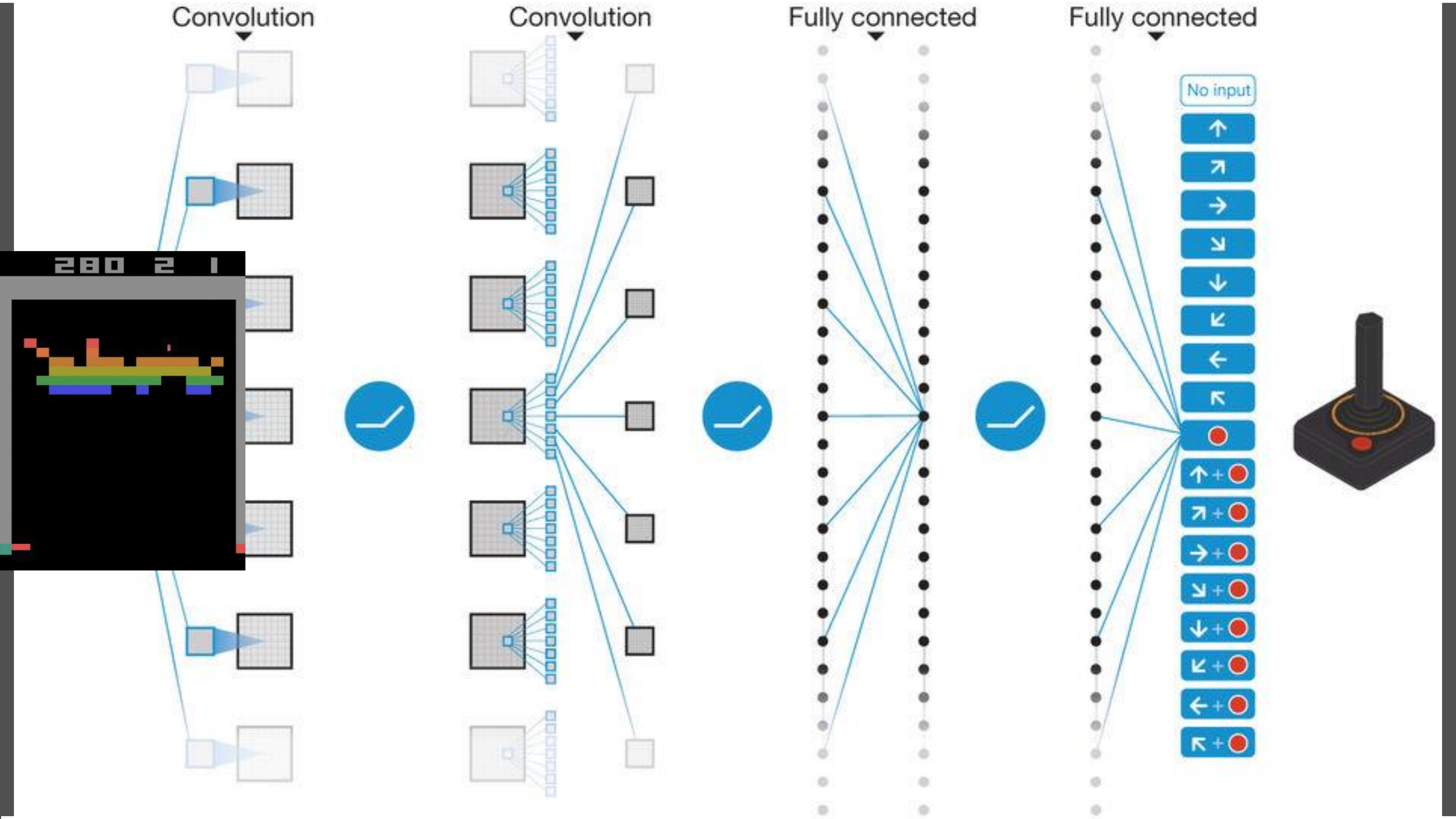

049 2 1


Convolution
Convolution
Fully connected
Fully connected
049 2 1
No input

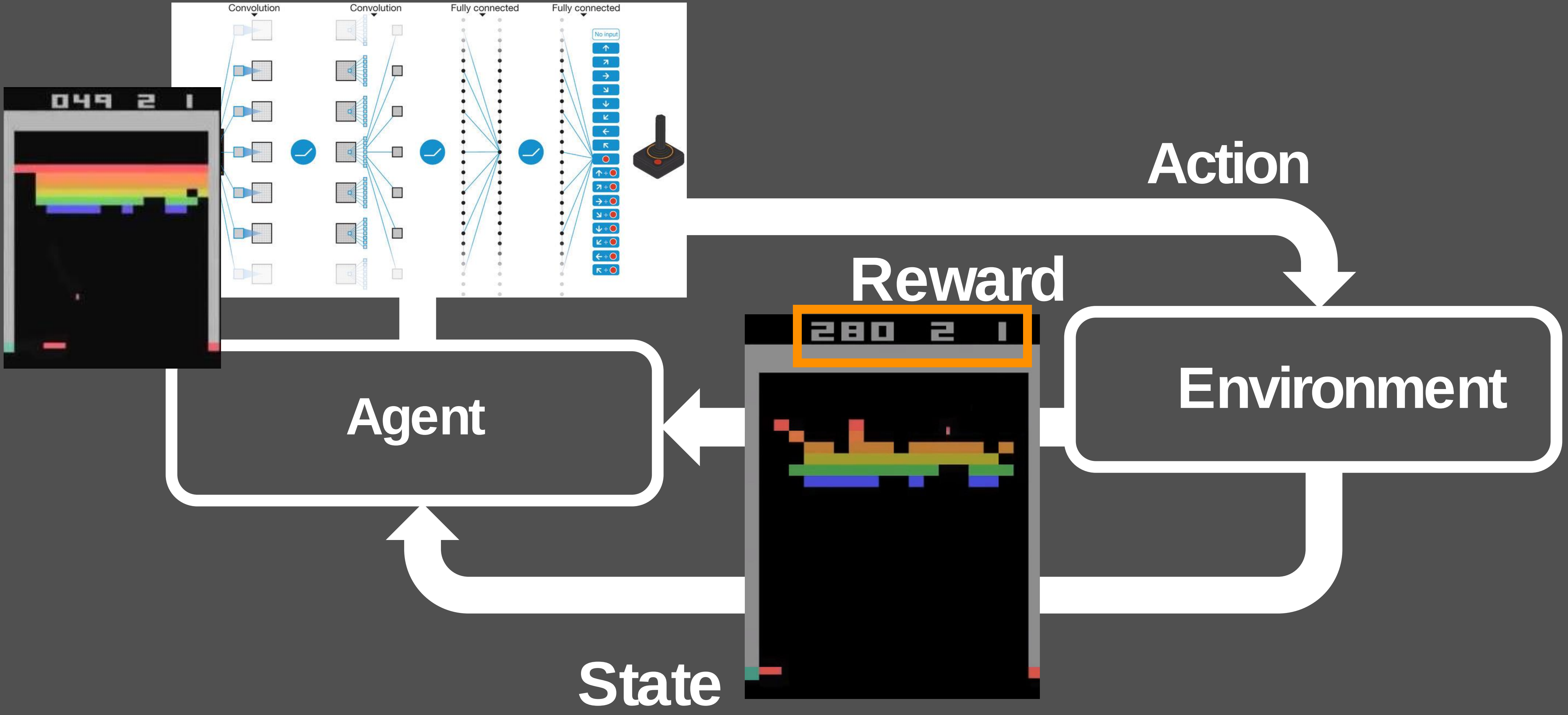

049 2 1


280 2 1

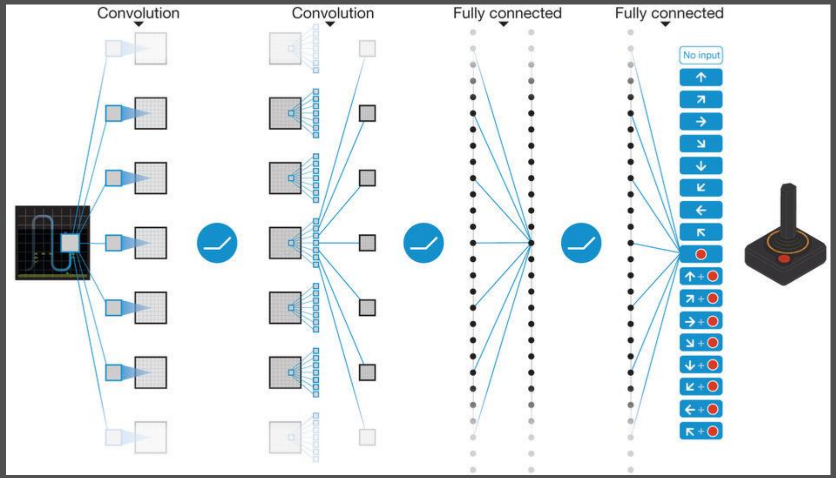
Reward



Convolution
Convolution
Fully connected
Fully connected
No input
280 2 1


Convolution
Convolution
Fully connected
Fully connected
No input
Action
Reward
Environment
Agent
State

Deep Q(uality) Network



Actions

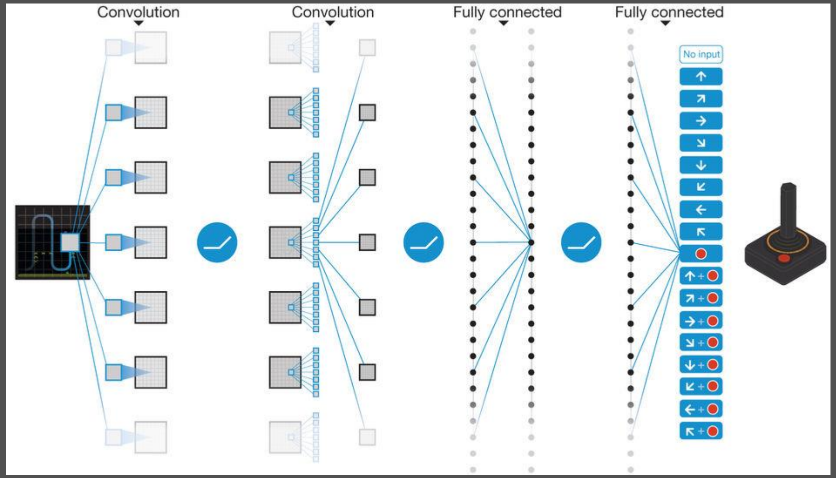
$$Q = \begin{array}{cc} \begin{array}{ccc} 0 & 1 & 2 \end{array} & \\ \begin{pmatrix} -1 & 1 & 0 \\ -1 & 0 & -1 \\ 10 & 0 & -1 \end{pmatrix} & \begin{array}{c} 0 \\ 1 \\ 2 \end{array} \end{array}$$

States

Hochschule Düsseldorf
University of Applied Sciences

Fachbereich Medien
Faculty of Media

MIREVI



Deep Q(uality) Network

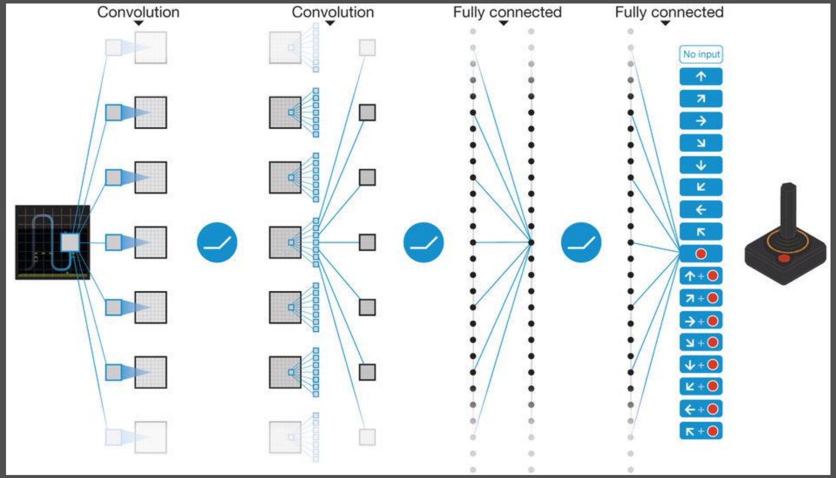
Value of take action 0 in state 0

Actions

$$Q = \begin{array}{c} \\ \\ \\ \end{array}\begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} -1 & 1 & 0 \\ -1 & 0 & -1 \\ 10 & 0 & -1 \end{pmatrix} \end{array} \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array}$$

States

Deep Q(uality) Network

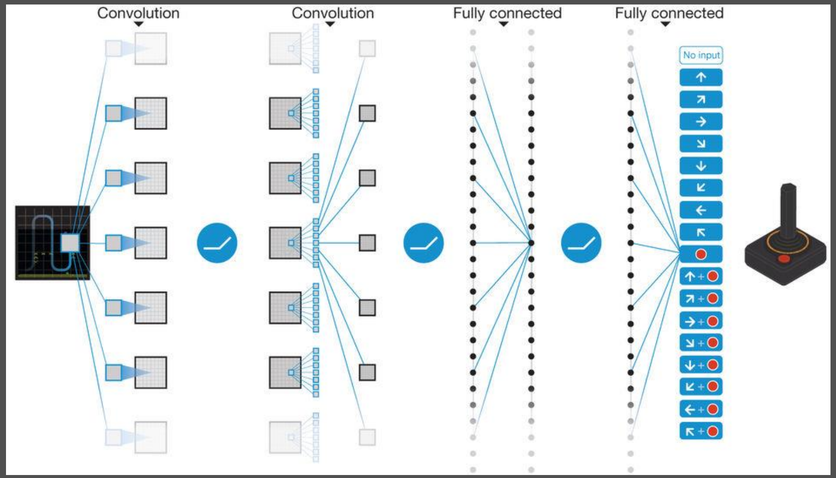


How good is it to take action 0 in state 0?

Actions

$$\begin{array}{cc} & \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ Q = & \begin{pmatrix} -1 & 1 & 0 \\ -1 & 0 & -1 \\ 10 & 0 & -1 \end{pmatrix} \begin{array}{c} 0 \\ 1 \\ 2 \end{array} \end{array}$$

States



Hochschule Düsseldorf
University of Applied Sciences

Fachbereich Medien
Faculty of Media

MIREVI

Deep Q(uality) Network

Q Table will be optimised by Neural Net

Actions

States

$$
Q = \begin{array}{cc}
\begin{array}{ccc} 0 & 1 & 2 \end{array} & \\
\begin{pmatrix} -1 & 1 & 0 \\ -1 & 0 & -1 \\ 10 & 0 & -1 \end{pmatrix} & \begin{array}{c} 0 \\ 1 \\ 2 \end{array}
\end{array}
$$

Q-Learning Algorithm:

- Estimate row in the Q table for the current state (Prediction)
- Apply action and get next state + reward
- Estimate row in the Q table for next state
- Apply Bellman Equation (Target):
  - $Q(s,a) := r + \gamma * max_{a'} Q(s',a') = Q(s,a) * max_{a'} Q(s',a')$
- Minimize Loss between Target and Prediction:
  - $L = \frac{1}{2}(r + \gamma * max_{a'} Q(s',a') - Q(s,a))^2$

- Estimate row in the Q table for the current state (Prediction)
- Apply action and get next state + reward
- Estimate row in the Q table for next state
- Apply Bellman Equation (Target):
  - $Q(s,a) := r + \gamma * max_{a'} Q(s',a') = Q(s,a) * max_{a'} Q(s',a')$
- Minimize Loss between Target and Prediction:
  - $L = \frac{1}{2}(r + \gamma * max_{a'} Q(s',a') - Q(s,a))^2$

Q-Learning:

- Estimate row in the Q table for the current state (Prediction)
- Apply action and get next state + reward
- Estimate row in the Q table for next state
- Apply Bellman Equation (Target):
  - $Q(s,a) := r + \gamma * max_{a'} Q(s',a') = Q(s,a) * max_{a'} Q(s',a')$
- Minimize Loss between Target and Prediction:
  - $L = \frac{1}{2}(r + \gamma * max_{a'} Q(s',a') - Q(s,a))^2$

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

Maximum future reward for this state and action is the immediate reward plus maximum future reward for the next state

- **Minimize Loss between Target and Prediction:**
  - $L = \frac{1}{2}(r + \gamma * max_{a'} Q(s', a') - Q(s, a))^2$
- $r + \gamma * max_{a'} Q(s', a') \in R^n$
- $Q(s, a) \in R^n$
- **Q-Learning is very similar to regression**
  - **Try to predict continous variables**

Hochschule Düsseldorf
University of Applied Sciences
Fachbereich Medien
Faculty of Media

Implementation Tricks:

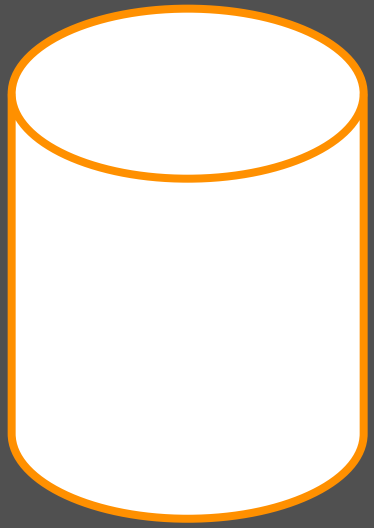
- Use ε–Greedy Policy
- Experience Replay
  - Minibatch training samples from past experience
  - Avoid to get stuck in a local minimum
- Discount Future Reward with γ
  - $Q(s,a) := r + \gamma * max_{a'} Q(s', a')$

- **4 Images form a training sample for the neural net**

Will the ball go up or down?

Your Task:

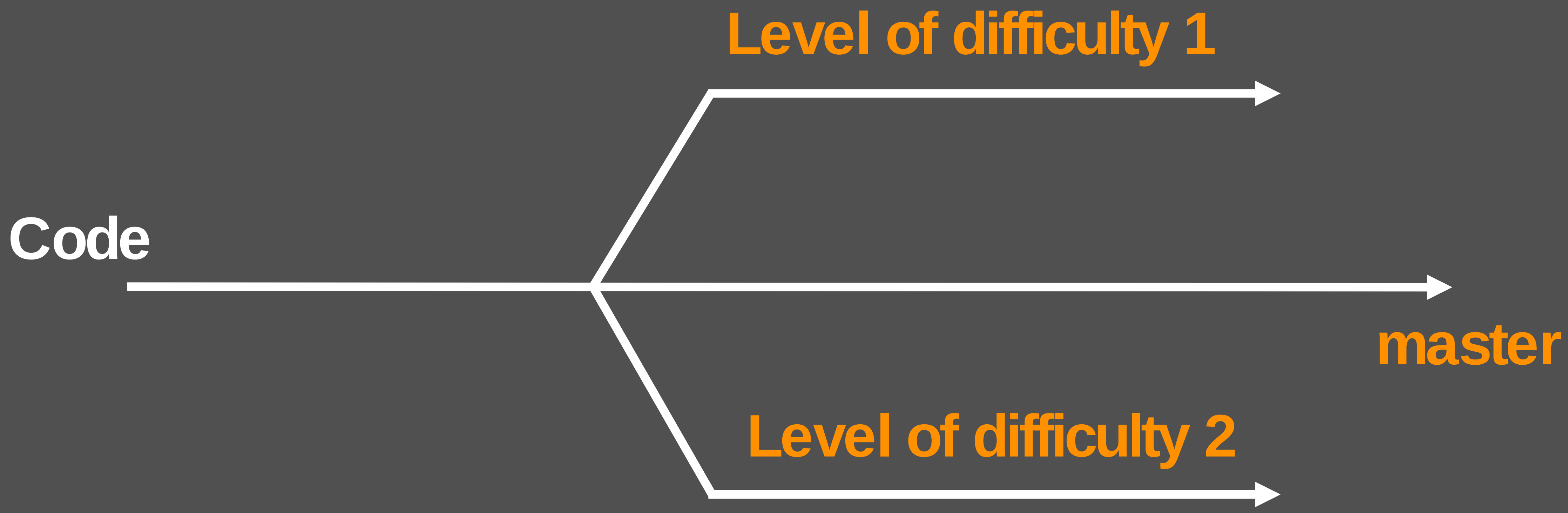
- We give you code with gaps
- You have to **fill in the gaps** with appropriate code
- The **gaps** are marked with a **comment with TODO**
- Gaps can be in all scripts
- Each script has an order of the TODOs, e.g. TODO 1, TODO 2, …



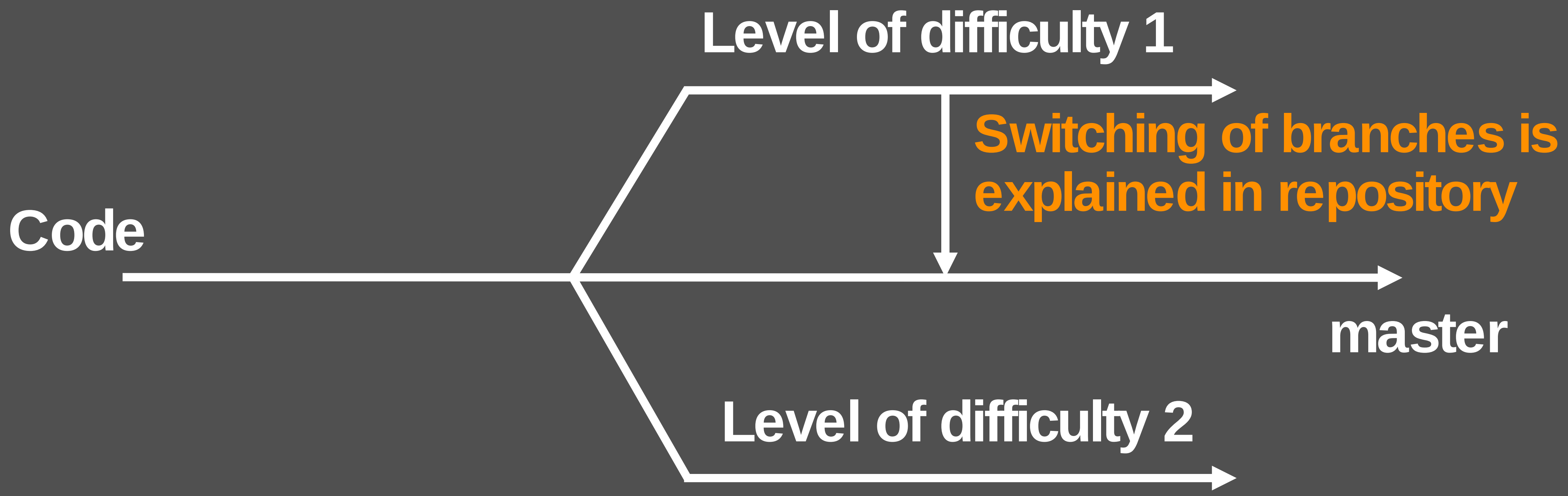
https://github.com/mati3230/MetaMarathon

- Code-Repository
- How to setup the working environment
- Tipps and recommendations
- 3 branches for different levels of difficulties
  - easy, medium, hard
- Solution will be pushed (uploaded) at the end in the master branch ;)

Branching:



Branching:


Level of difficulty 1
Switching of branches is explained in repository
Code
master
Level of difficulty 2

3 Files:

- main.py: should be executed
- estimator.py: neural net
- state_processor.py: image preprocessing

main.py: should be executed

- Mother-Script
- Management processes (training, playing)
- Execution of steps in the environment
- Optimization of Q-values

estimator.py: neural net

- Convolutional Neural Net (CNN)
- Prediction of Actions
- Training of Network

state_processor.py: image preprocessing

- Grayscale
- Cropping of relevant area
- Scaling to 84x84

First Steps:

- Go to the repository: https://github.com/mati3230/MetaMarathon
- Setup the environment (use manual)
- Clone Repository
- Choose a level of difficulty and switch branch
- Search TODOs and get familiar with code

Hochschule Düsseldorf
University of Applied Sciences

Fachbereich Medien
Faculty of Media

MIREVI

https://github.com/mati3230/MetaMarathon