# Data-driven Intelligent Systems

## Lecture 17
## Reinforcement Learning II

KNOWLEDGE
TECHNOLOGY

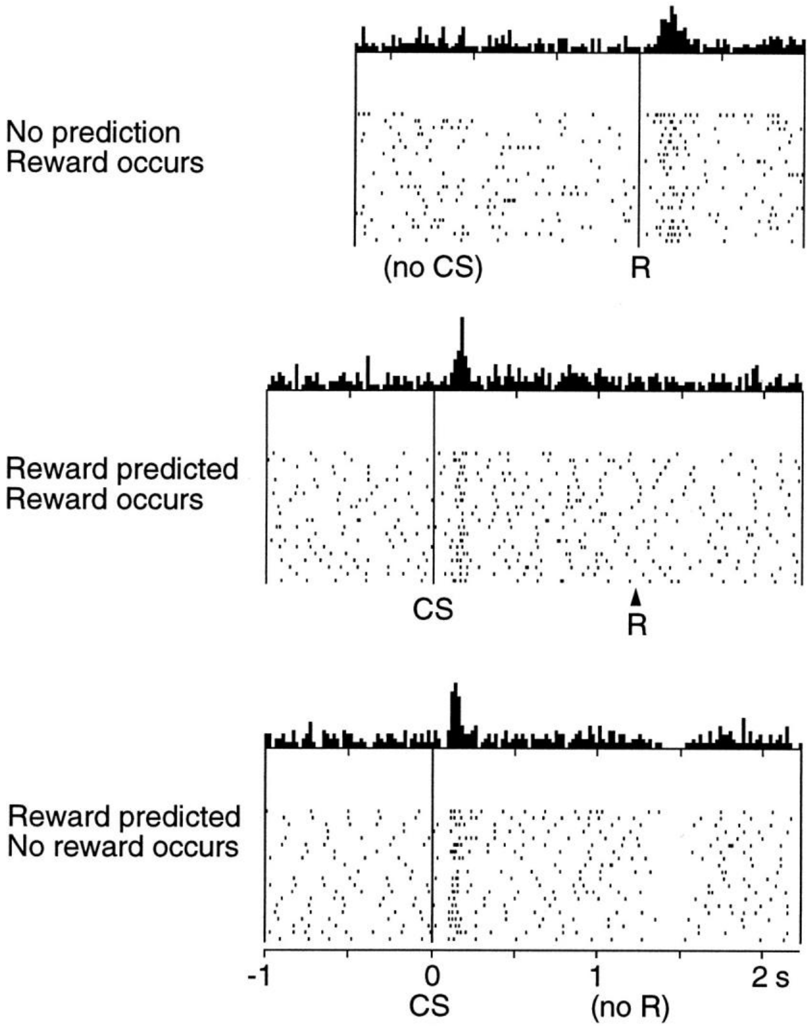
http://www.informatik.uni-hamburg.de/WTM/

# Outline

- Dopamine in the Brain Relates to TD Error
- Policy-gradient: REINFORCE
- Goal-conditioned RL and Hindsight Experience Replay
- Hierarchical RL
- Model-based RL: MuZero

# Biological Analogy: Dopamine signals TD error

Firing (spikes) of dopamine neurons




[Schultz (1998) Predictive Reward Signal of Dopamine Neurons]()

**Pavlovian conditioning:**

**Initially**: When a reward is given (e.g. juice) the animal reacts (e.g. salivates)

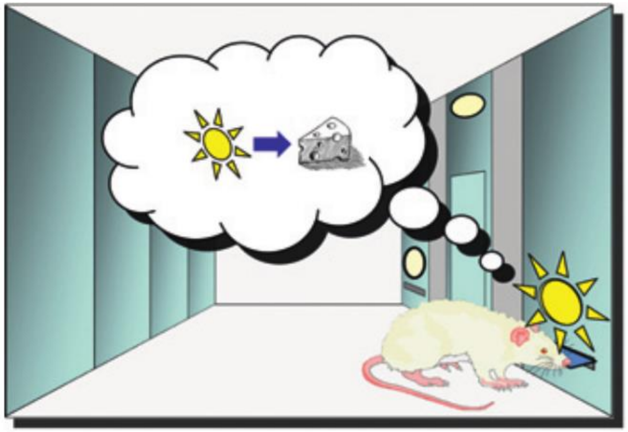
**Learning**: A neutral stimulus (e.g. bell) is consistently presented before giving the reward

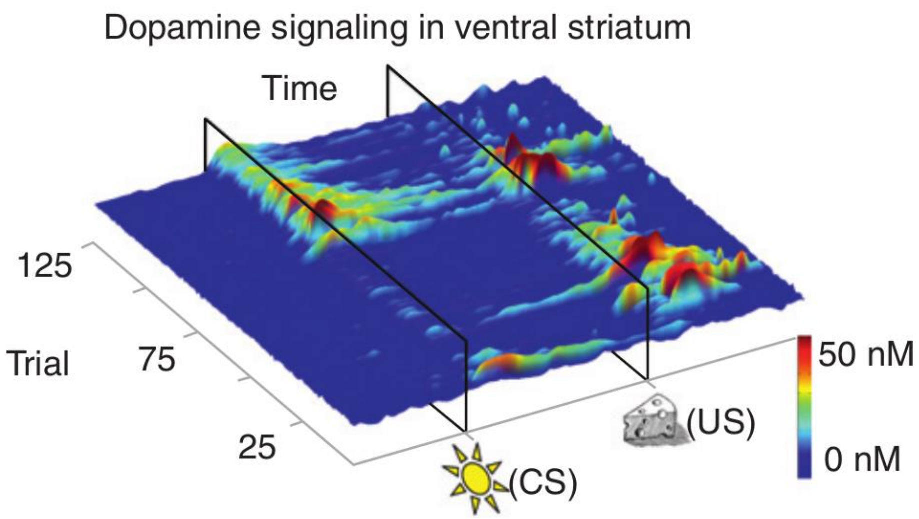
**Result**: the animal reacts at the neutral stimulus, which becomes the conditioned stimulus (CS), but not during reward.

**Interpretation**: TD error is relevant.

Dopamine neurons behave in the same way.

# Biological Analogy: Dopamine signals TD error






Clark et al. (2012) Pavlovian Valuation Systems in Learning and Decision Making

# Further Neuro-transmitters

Good evidence:

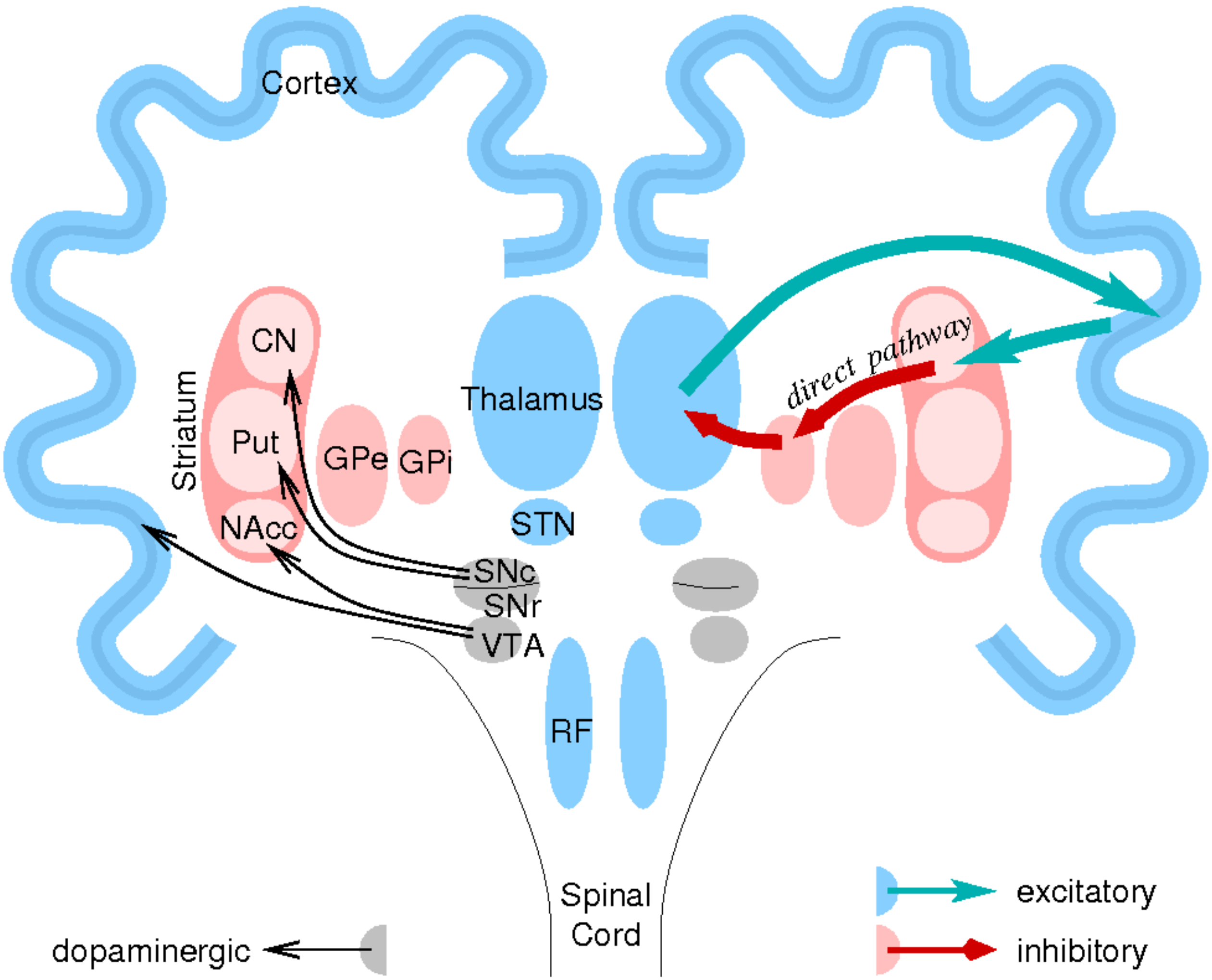
- Dopamine suggested to correspond to the TD error

Rather speculative:

- Norepinephrine / Noradrenaline signals unexpected events, i.e. surprise or uncertainty
  Dayan, Yu (2006) Phasic norepinephrine: A neural interrupt signal for unexpected events
- Acetylcholine signals unexpected uncertainty
  Yu, Dayan (2003) Acetylcholine, Norepinephrine, and Spatial Attention
- Serotonine correlates with discount factor $\gamma$

Yoshida, Uchibe, Doya (2013) Reinforcement learning with state-dependent discount factor

# Basal Ganglia as Brain Candidate for RL




SN = substantia nigra
VTA = ventral tegmental area

Model hypothesis:

- Striatum encodes the state
- Thalamus receives the action selection
- Direct pathway is via inhibition of inhibition
- Dopamine learning signals arise from SN and VTA

# Reasons for an Agent to Act

- Rewards
  - Classical rewards, e.g. food & drink
  - Social & task rewards, e.g. praise, exam marks, salary
  - Negative rewards, e.g. pain
  - Epistemic rewards
    - Reward from information gain (~curiosity)
    - Reward for efficient sensory coding, e.g. better compression
- Imitation behavior

  → Inverse RL
- Homeostasis
  - Self-protection, e.g. reflexes
  - Daily rhythm with activity and rest/sleep

# Outline

- Dopamine in the Brain Relates to TD Error
- Policy-gradient: REINFORCE
- Goal-conditioned RL and Hindsight Experience Replay
- Hierarchical RL
- Model-based RL: MuZero

# Policy Gradient

Previous lecture: Value-based RL

- Agent estimates *Q(s,a)* or *V(s)* value functions, and use these while also finding the best policy $\pi$

Alternative: Policy-gradient based RL

- Search best policy $\pi$ directly by gradient ascent
- *“But this can’t work, because we have no learning signals during steps when the reward is zero? – We need a value function to solve this temporal credit assignment problem!”*
- Solution: Use wider estimates of Return; use NN or RNN (for interpolation and backprop. through time)
  - allows learning at timesteps other than when a reward is directly present

# Policy Gradient - Equations

Objective is to maximize the expected future return $R$.

We take the gradient ∇ with respect to the parameters θ:

$$\nabla_\theta \langle R_{\pi_\theta} \rangle = \sum_s p(s) \sum_a \nabla_\theta \pi_\theta(a|s) \cdot Q(s,a)$$

$$\boxed{use: \nabla_\theta \pi_\theta(a|s) = \pi_\theta(a|s) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} = \pi_\theta(a|s) \nabla_\theta \mathrm{Log}\, \pi_\theta(a|s)}$$

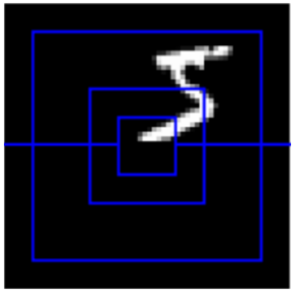
$$\frac{d}{dx} \log x = \frac{1}{x}$$

$$= \underbrace{\sum_s p(s) \sum_a \pi_\theta(a|s)}_{\text{Expectation value, compute by sampling}} \cdot \underbrace{\nabla_\theta Log\, \pi_\theta(a|s)}_{\text{Compute by gradient back-propagation from output units}} \cdot \underbrace{Q(s,a)}_{\text{Use external rewards } R}$$

Variance reduction: subtract expected baseline value *b*: *R* ← *R* - *b*

Further reading: https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html

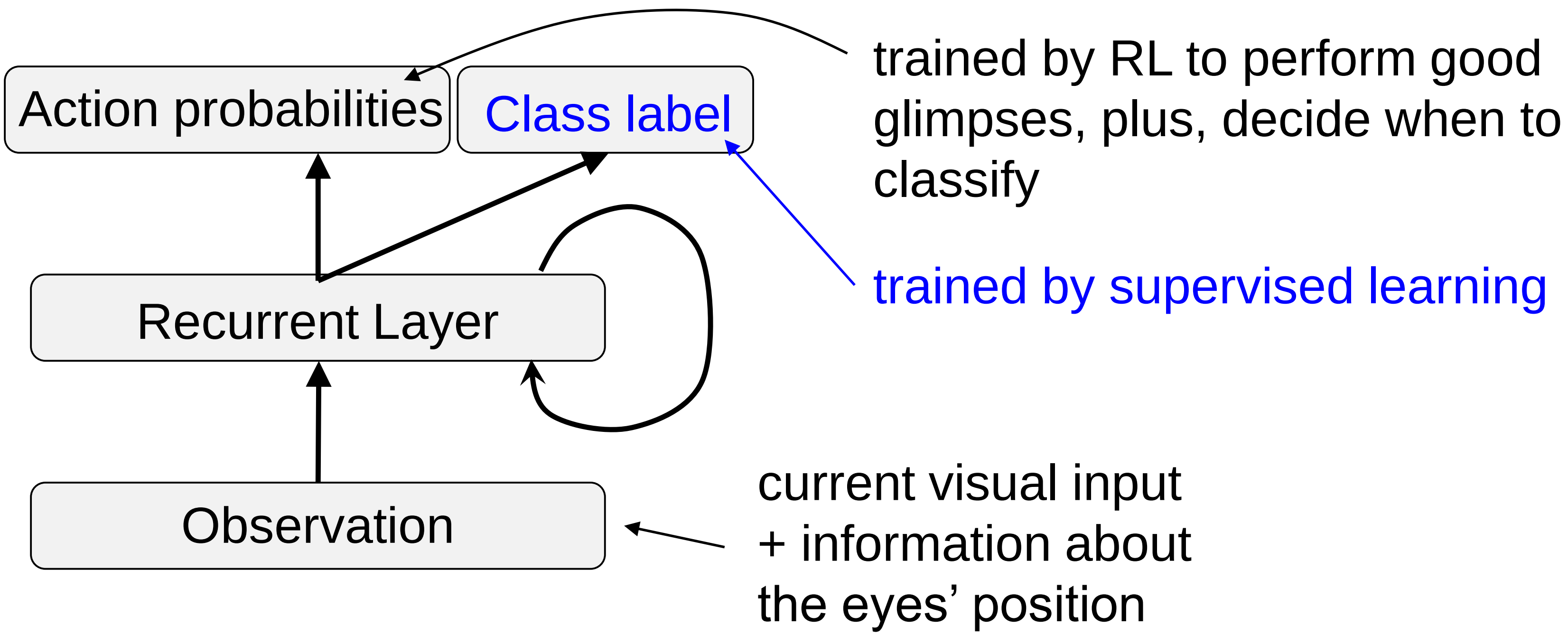
# Policy Gradient – Visual Attention



Task: visually identify a handwritten digit (MNIST)

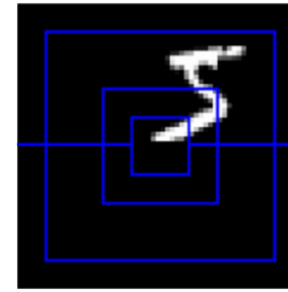
Constraint: good resolution only in small field of view

Strategy: learn a sequence to glimpse at certain positions; integrate information via RNN; then classify the digit

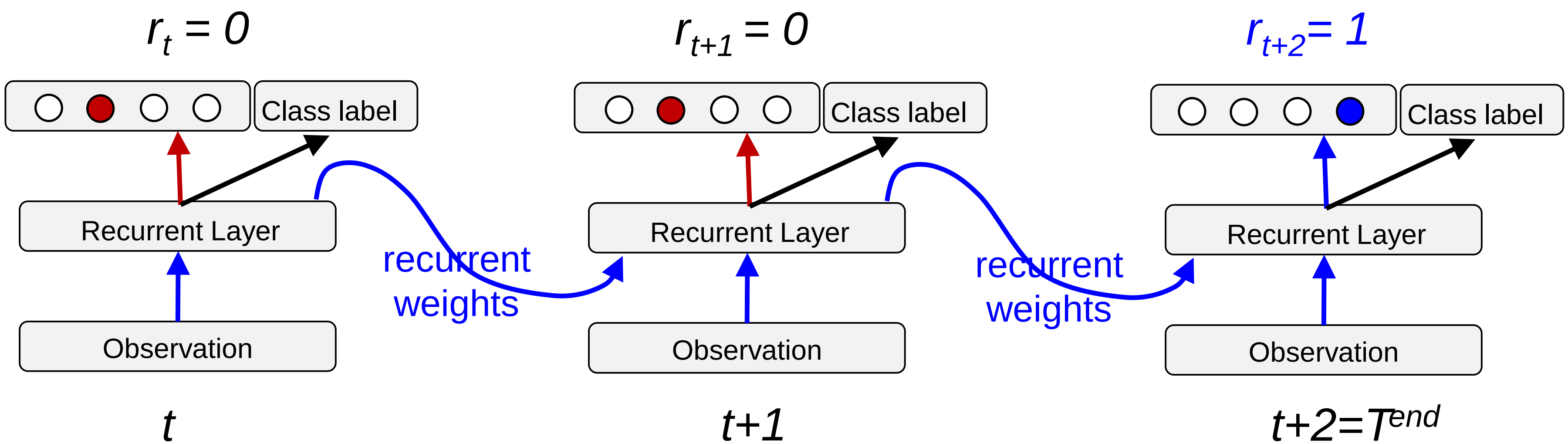



Mnih et al. (2014) Recurrent Models of Visual Attention. https://arxiv.org/pdf/1406.6247v1.pdf

# Policy Gradient – Visual Attention



Backpropagation through time pairs reward at $T^{end}$ with past activations only in input and hidden layers




Train with $R = \Sigma r_t$ for entire sequence
→ weights to all used actions can be learnt

Related technique: eligibility traces

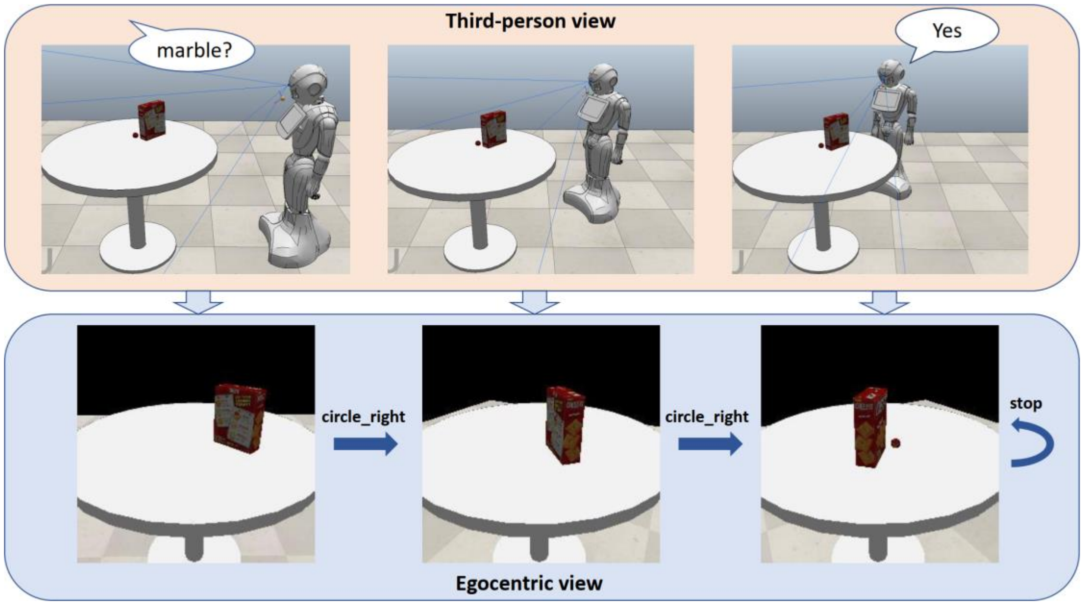
Mnih et al. (2014) Recurrent Models of Visual Attention. https://arxiv.org/pdf/1406.6247v1.pdf

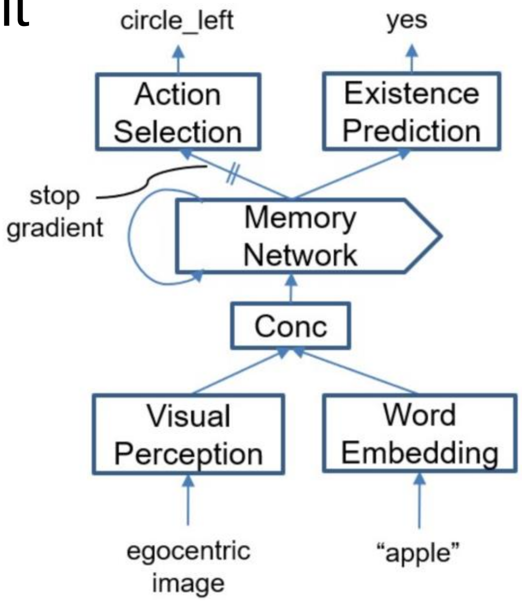
# Policy Gradient – Occlusion Reasoning

Task: answer whether a specific object is on the table

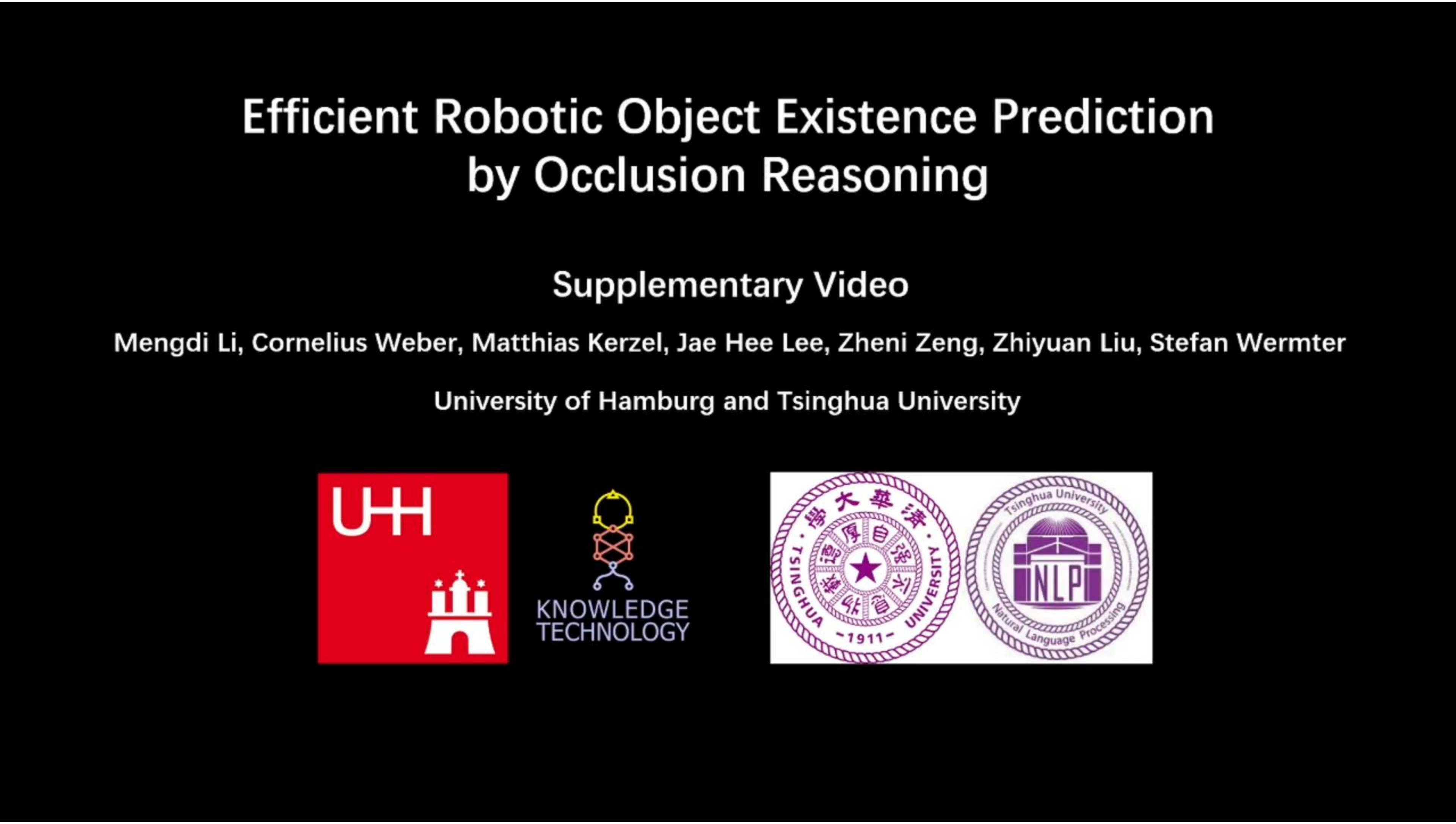
Constraint: object might be occluded by another one

Learnt strategy: walk around the table, if the object is not visible, if it is small and if a larger object might occlude it



# Policy Gradient – Occlusion Reasoning




Mengdi Li et al. IROS 2021, submitted.

# Outline

- Dopamine in the Brain Relates to TD Error
- Policy-gradient: REINFORCE
- Goal-conditioned RL and Hindsight Experience Replay
- Hierarchical RL
- Model-based RL: MuZero

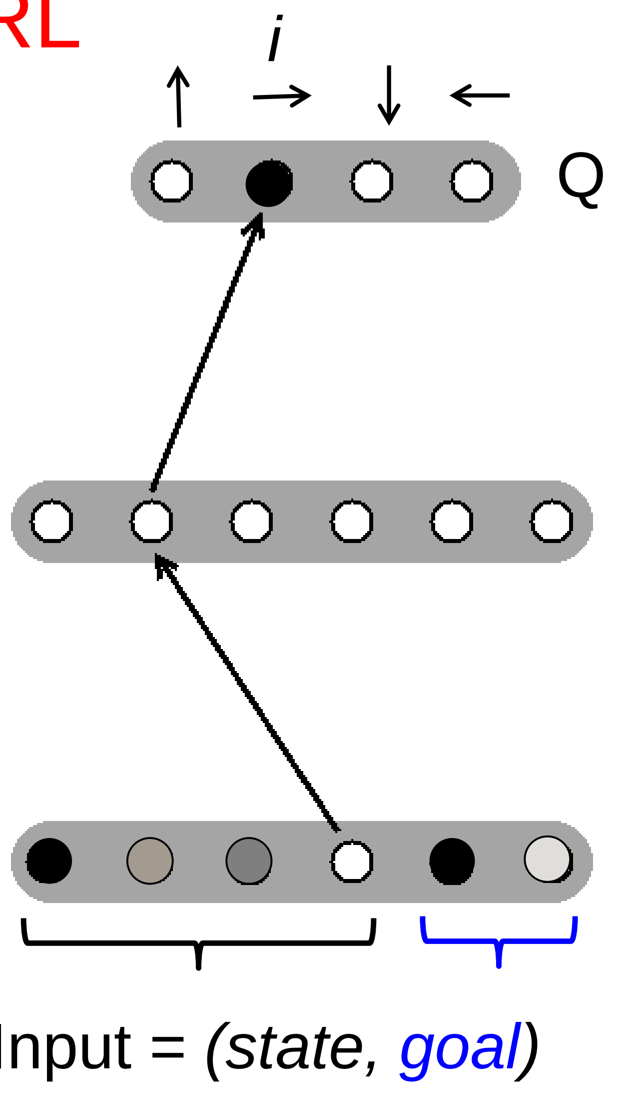
# Goal-conditioned RL

So far, problem of unflexibility:

- Typically, Q-values are assigned to states irrespective of goal
- Changing the goal requires new Q-values, hence new learning!
- Separate Q-values for every goal are costly, if a Q-table is being used

The solution is **goal-conditioned RL**:

- Enhance the input with additional units that encode the goal
  - Agent learns to reach various goals
  - Goals can be dynamically set
  - e.g. by another network → hierarchical RL



# Hindsight Experience Replay (HER)

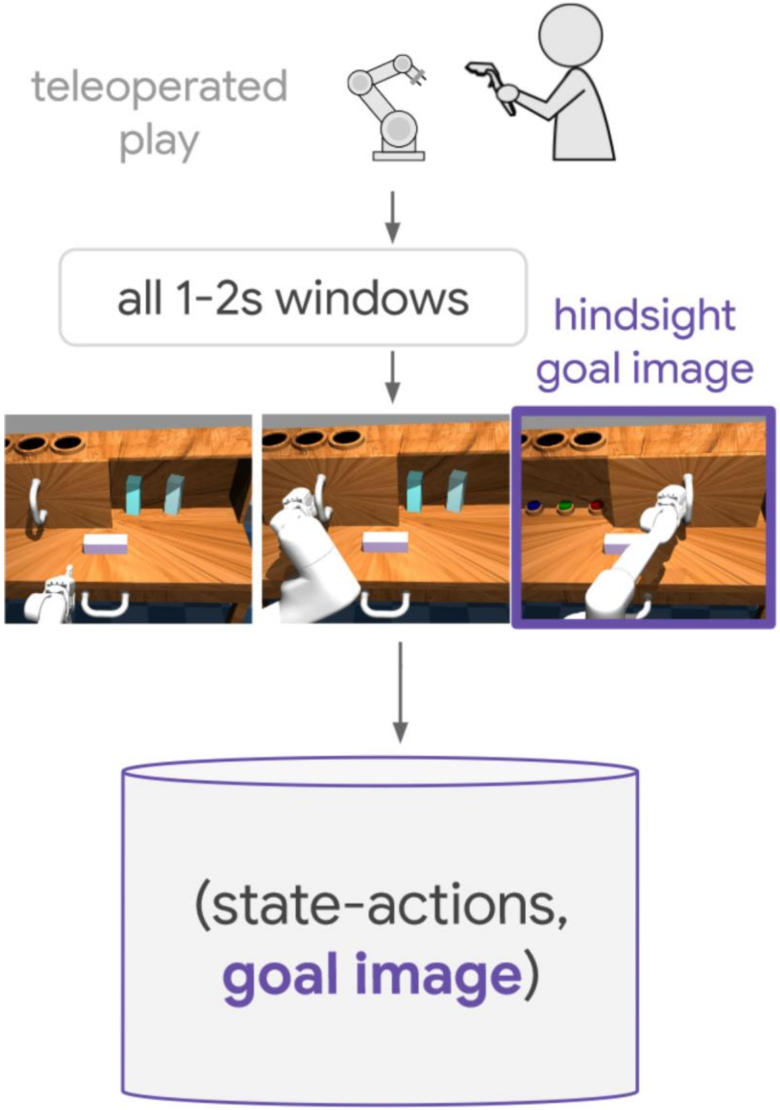
Recap: ***Experience Replay***

- During environment exploration, memorize experiences
- Learn from randomly sampled experiences from memory

***Hindsight Experience Replay***

- Use architecture for goal-conditioned RL
  - Model input is state and goal: $(s, g)$
- Take any experience (typically, an extended sequence)
  - $(s_0, s_1, \ldots, s_T)$ $\leftarrow$ state sequence from time = *0* until *T*
- In hindsight, assume $s_T$ had been the goal $g$
  - Model input: $(s_0, s_T)$
- This teaches the agent to reach $s_T$
- Will be repeated for many different $\{s_T\}$

# HER Example




Data collection

- People remote-control the robot; playful behavior without particular instructions
- Nevertheless, people do useful things, such as opening doors

HER learning

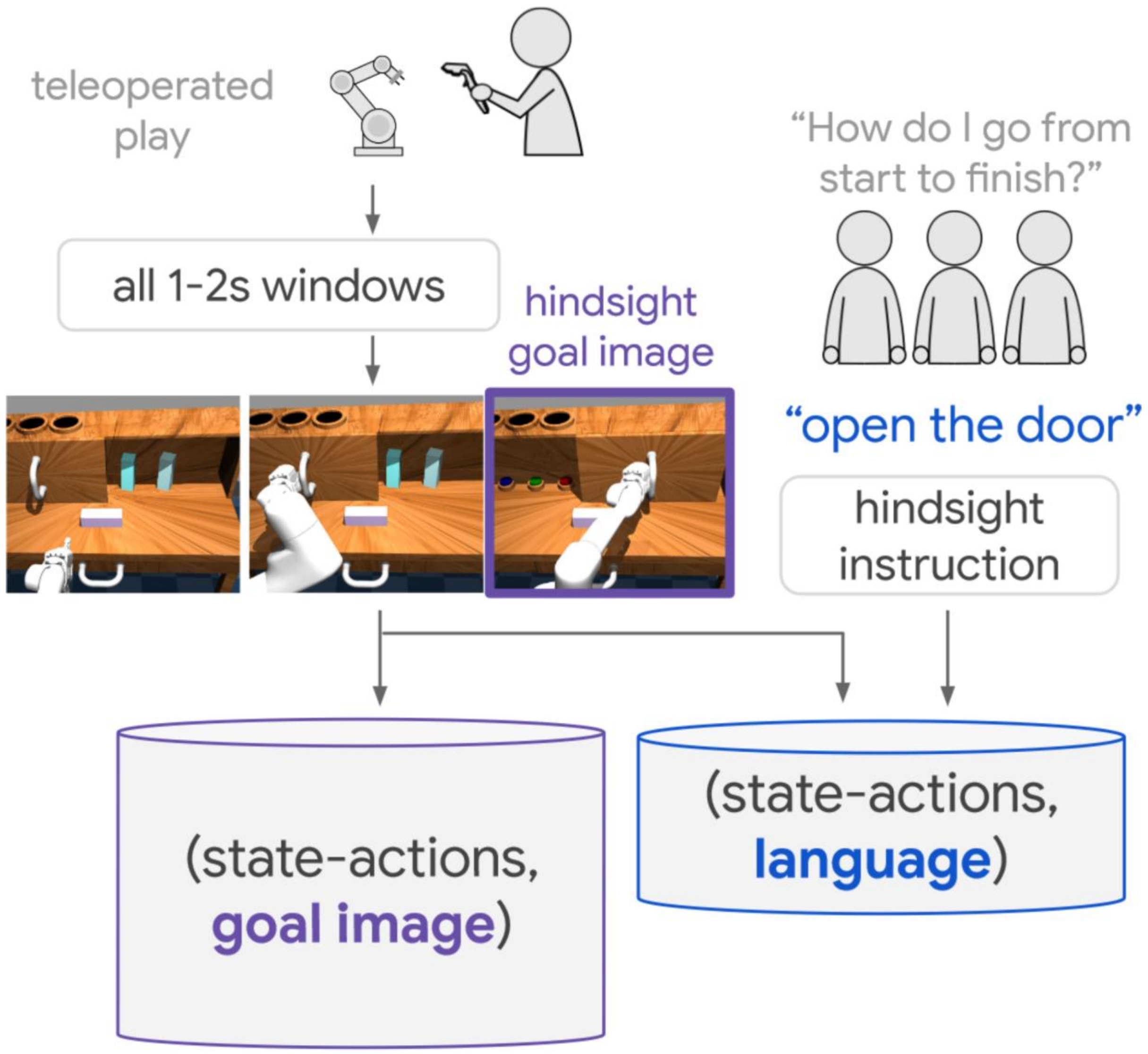
- Short sequences presented
- Last frame is the goal
- Agent learns to reach any goal

Usage

- Need to show an image of the goal state; agent can then reach it

Lynch, Sermanet (2020) Grounding Language in Play. https://arxiv.org/pdf/2005.07648.pdf

# HER Example



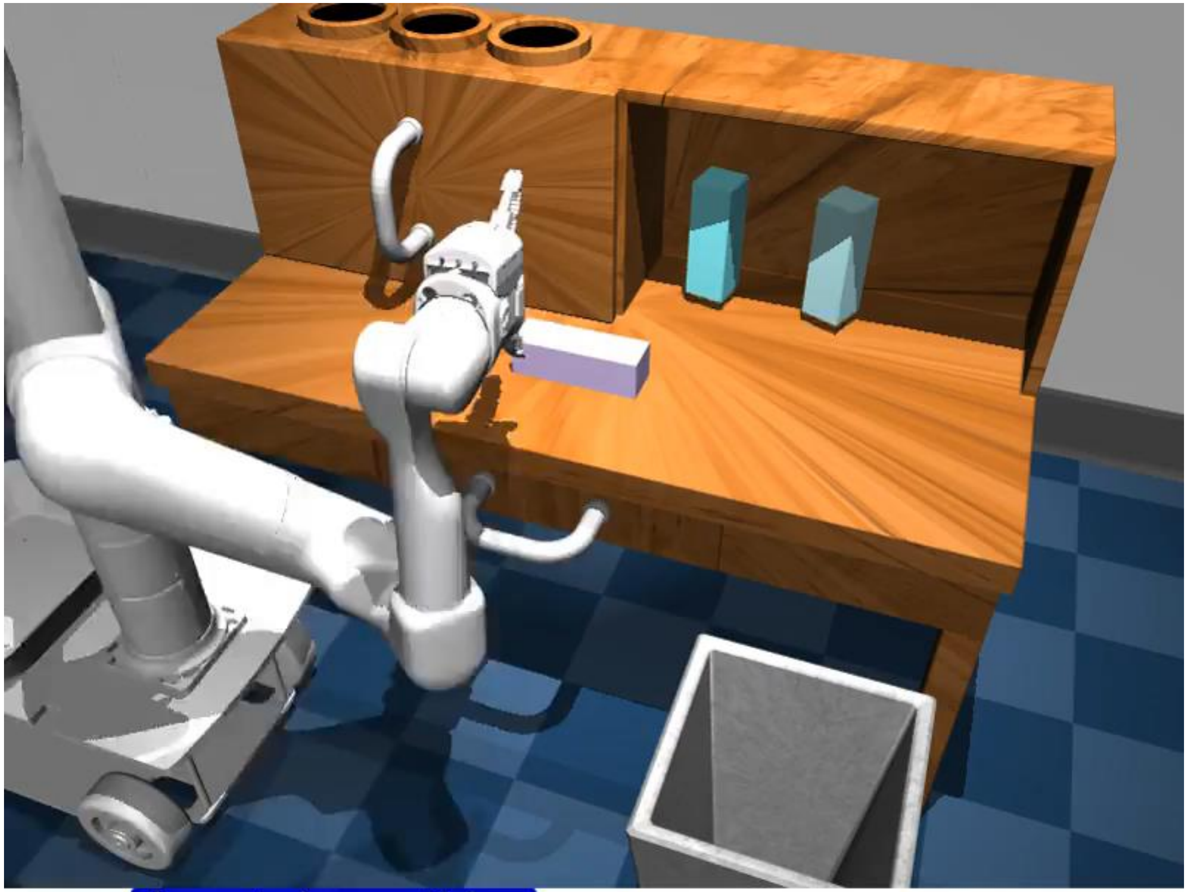

Language extension
- Image goals are supplemented by language goals

HER learning
- Deep NN learns joint representation of image and language goal

Usage
- Lang. instruction provides the goal

Lynch, Sermanet (2020) Grounding Language in Play. https://arxiv.org/pdf/2005.07648.pdf

# HER Example



now: do not do anything
next:

Lynch, Sermanet (2020) Grounding Language in Play. https://arxiv.org/pdf/2005.07648.pdf

# Outline

- Dopamine in the Brain Relates to TD Error
- Policy-gradient: REINFORCE
- Goal-conditioned RL and Hindsight Experience Replay
- Hierarchical RL
- Model-based RL: MuZero

# Hierarchical RL

"Flat" RL has limitations with hierarchically structured tasks:

- i.e. consisting of subtasks
- particularly if the subtasks are not rewarded

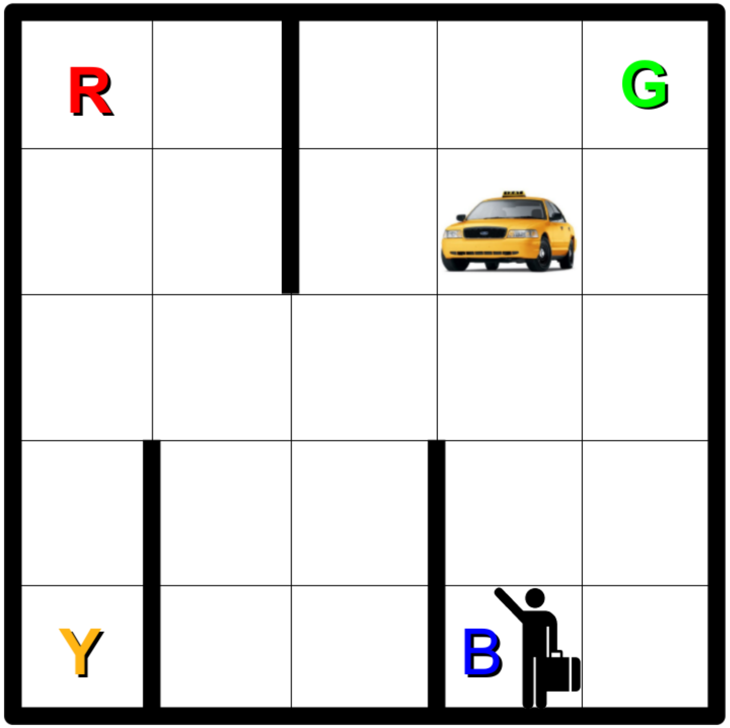
Examples in Atari games:

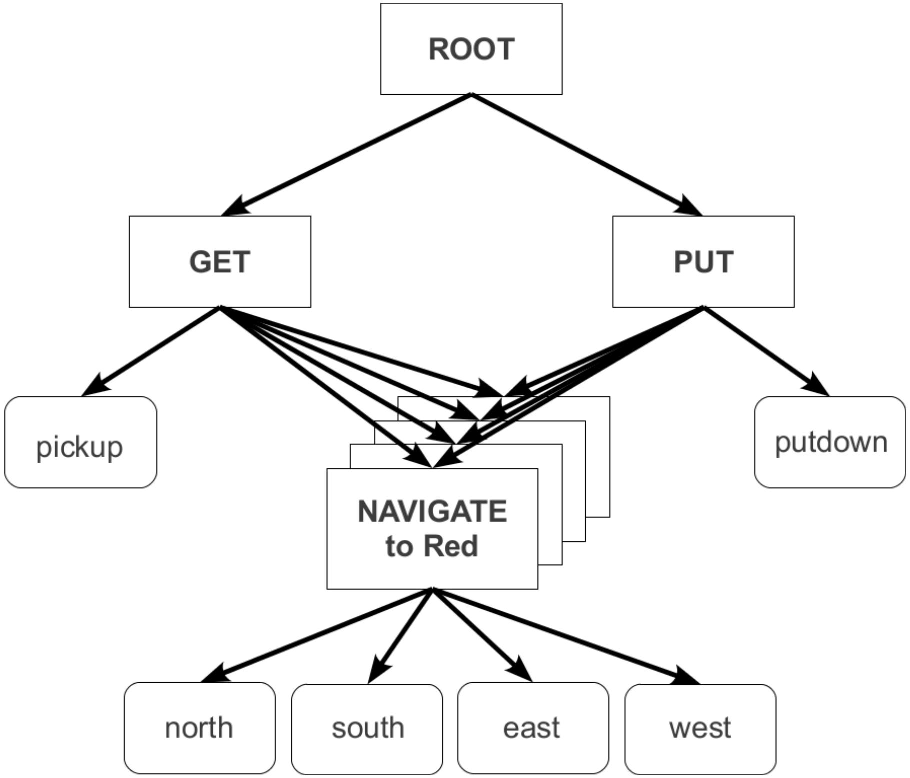
- must collect a key before a door can be opened
- multiple small missions to complete in a specific order

Hierarchical solution: decompose the task into subtasks:

- on a low hierarchical level, solve the subtasks
- on a high hierarchical level, activate the subtasks and provide their respective goals

# Hierarchical RL – taxi domain

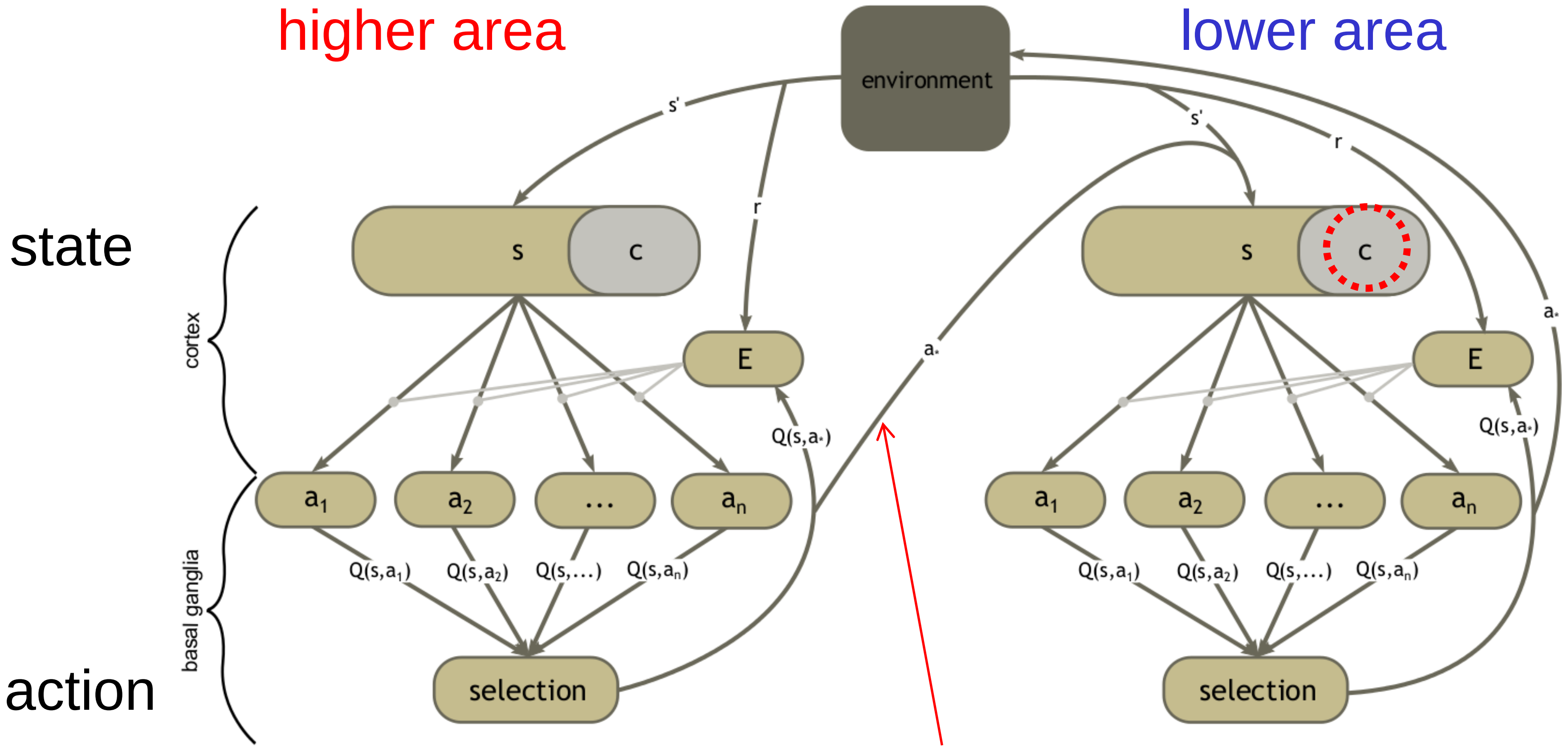



- reward given when taxi puts down passenger at destination
- hierarchical decomposition makes the problem tractable
  - low level learns “navigate to X” as a subtask
  - high level learns to arrange subtasks and set their goals

Figure: Vien Ngo, Marc Toussaint. Hierarchical Reinforcement Learning (Lecture) University of Stuttgart.

# Hierarchical RL – top-down control




top-down connections determine the current navigation goal (via context units *c*)

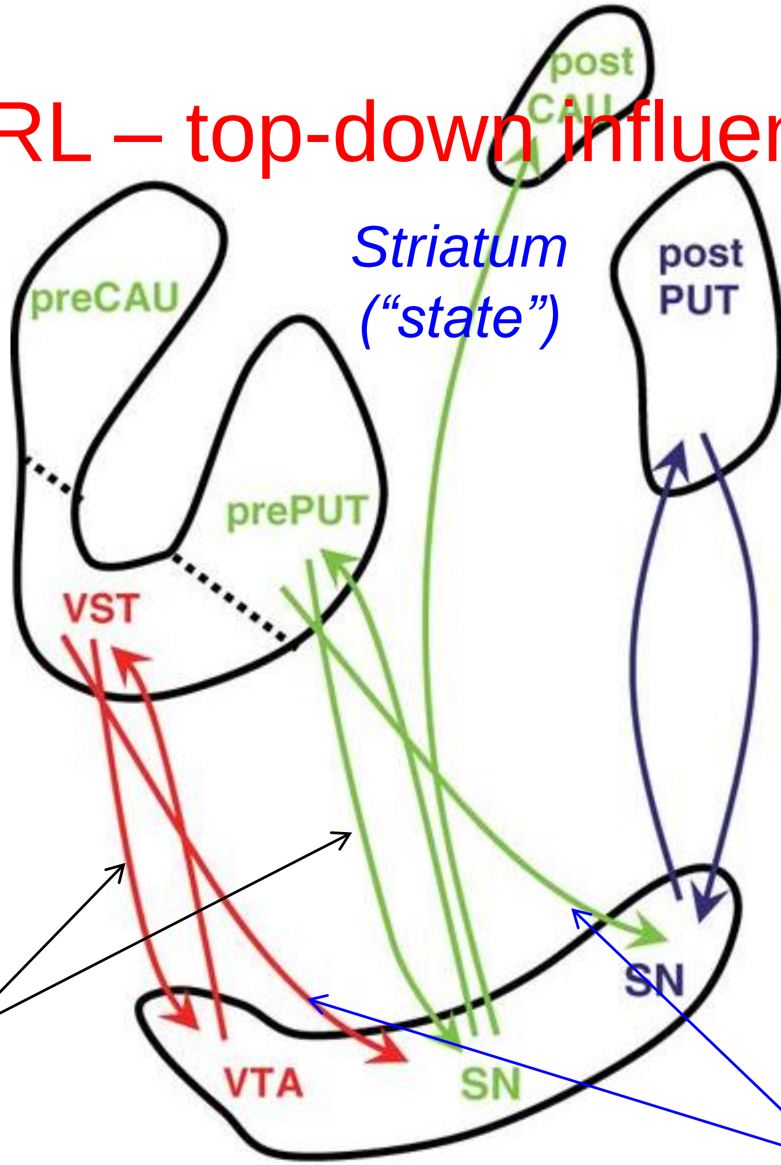
Figure: Rasmussen, Eliasmith (2014) A neural model of hierarchical reinforcement learning.

# Hierarchical RL – top-down influence in biology

*Striatum*
*(“state”)*

higher areas

lower areas




recurrent
connections
back to origin

top-down
connections

*dopamine neurons*
*(“reward” / “goal”?)*

Figure: del Campo et al. (2013) A positron emission tomography study of nigro-striatal dopaminergic mechanisms underlying attention: implications for ADHD and its treatment. Brain 136 (11), 3252-3270.

# Outline

- Dopamine in the Brain Relates to TD Error
- Policy-gradient: REINFORCE
- Goal-conditioned RL and Hindsight Experience Replay
- Hierarchical RL
- Model-based RL: MuZero

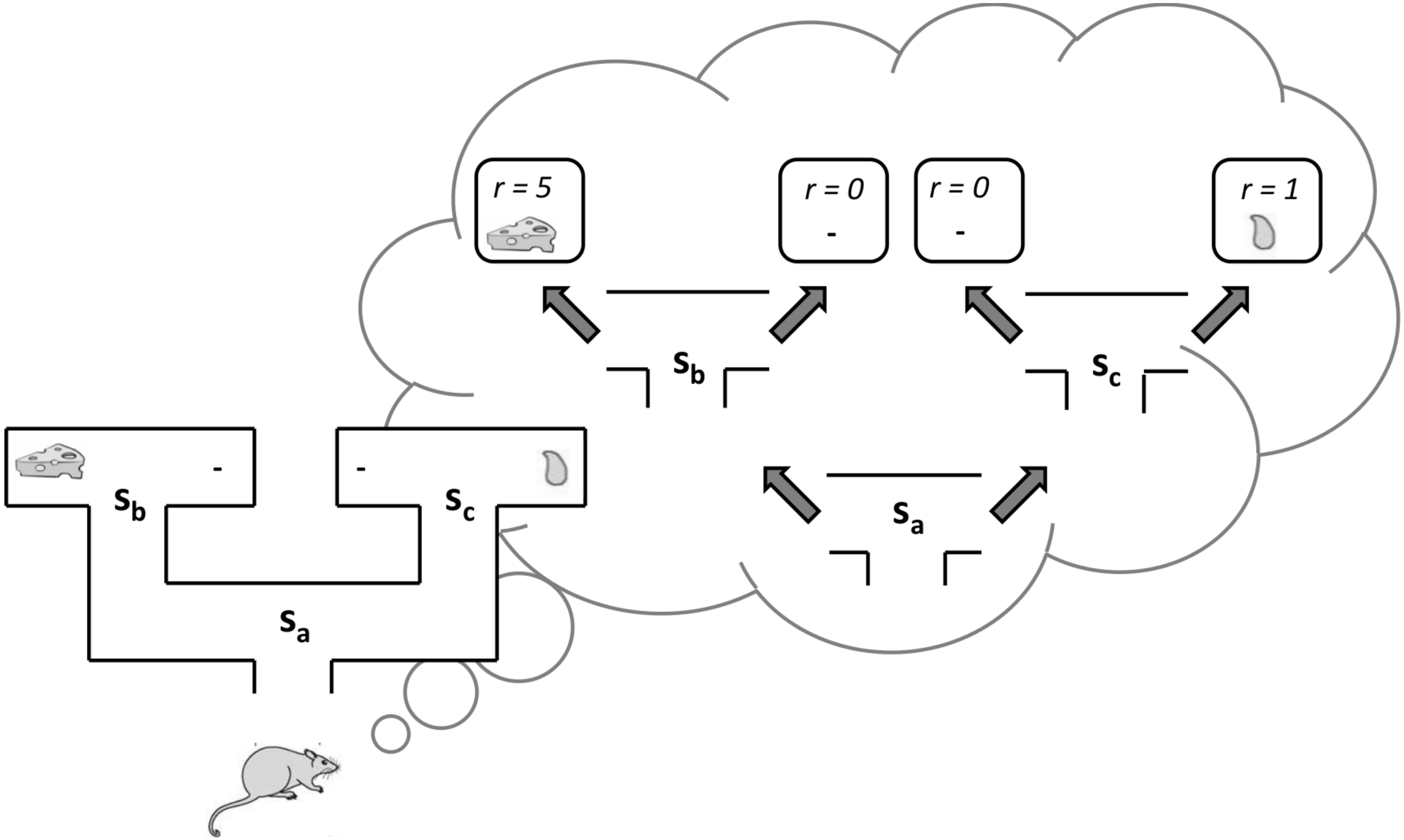
# Model-based RL

- Model of the transition structure of the world

$$P(s_{t+1}) = P(s_{t+1}|s_t,a_t)$$

  - for all states and all actions
- Can be learnt by exploration, independently of any rewards
- The model may also represent the reward structure $r(s_{t+1},s_t)$
- *Use* a model:
  - compute all Q-values based on internal simulation by model
  - compute total Return for next selection, or an entire plan
  - compare different plans, to select the best one
- Limitations:
  - exponential number of states/actions; cycles
  - breadth-first vs. depth-first

# Model-based RL

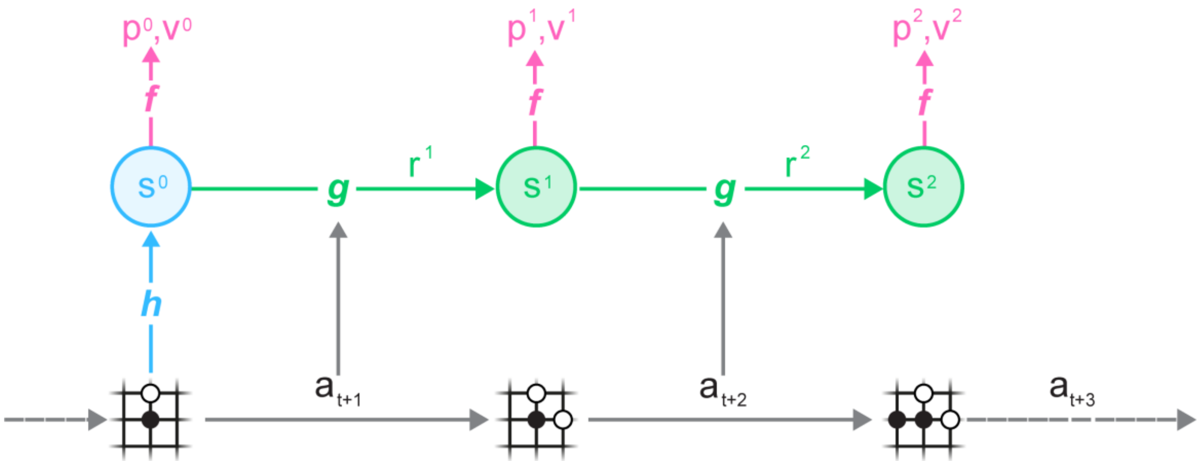



- tree search through future states and rewards

# Model-based vs model-free RL

- What if the environment changes, e.g.
  - a reward is being devalued
  - a transition becoming impossible
  - a new shortcut is introduced
- A model-free RL learner
  - will initially maintain its stimulus-response-like policy at all other states than the changed ones
  - must re-visit all states to update their values (iteratively)
- A model-based learner
  - can automatically adjust to changes in this information
- Humans used a mixture of model-free and model-based RL
  - habits vs planning

# MuZero: Architecture



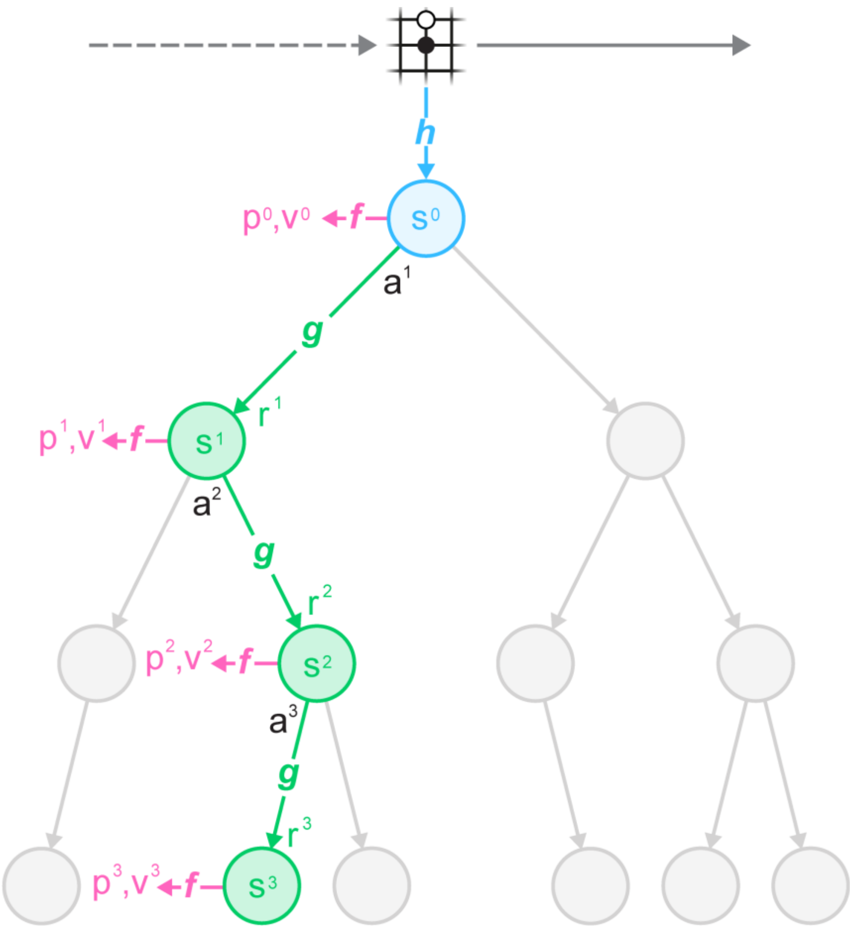

Three neural networks:

- Representation function ***h***
  - maps raw observations to latent representations $s_t$
- Dynamics function ***g***
  - maps $(s_t, a_t)$ to $s_{t+1}$ and predicts the reward $r$
- Policy and value prediction function ***f***
  - akin actor-critic network

The dynamics function is a forward model (world model) and allows recursive prediction of future outcomes, i.e. planning; but: *in latent space*

Schrittwieser et al. (2020) Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model

# MuZero: Planning by Monte-Carlo Tree Search




While exploring (in) the world, the agent plans ahead

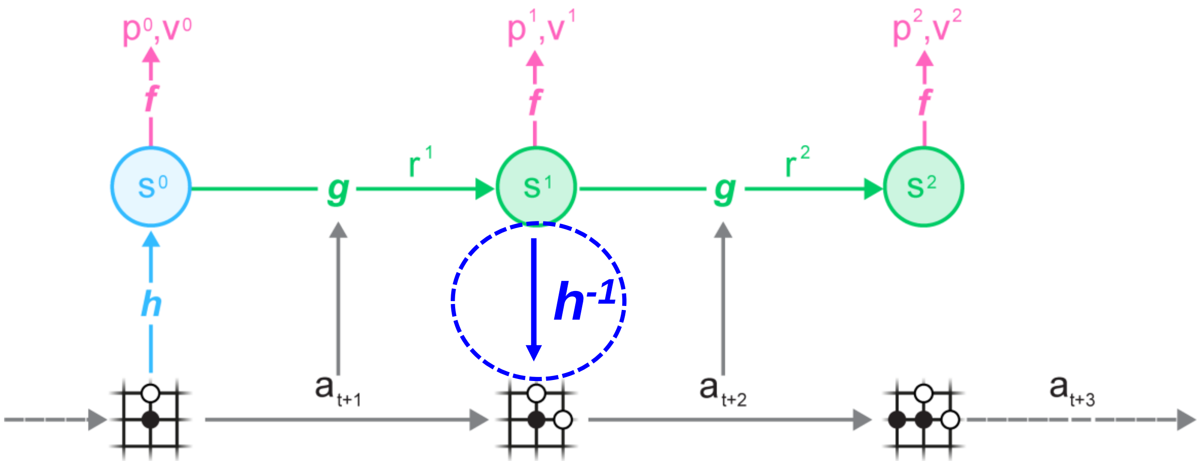
- Use the dynamics function ***g*** to simulate future states $s_{t+1}$, $s_{t+2}$, …, of which ***f*** predicts their value
- Different actions $a_t$ are explored
- MCTS extends various branches into an emerging planning tree; better branches will be more often chosen and extended
- Finally, **one** action from the root node is chosen to be executed, proportionally to the visit count of its branch

At next real time step: span another complete planning tree

# MuZero: Training via Experience Replay

- All experiences are saved in a replay buffer:
  - sequences of (observations, actions, rewards)
- Learning happens from randomly chosen experiences
  - All learning-relevant values can be computed from current network parameters
  - Using forward- and backpropagation through the network's functions ***h***, ***g***, and ***f***
- Planning happens only during "real world" exploration

# MuZero: Architecture Extension

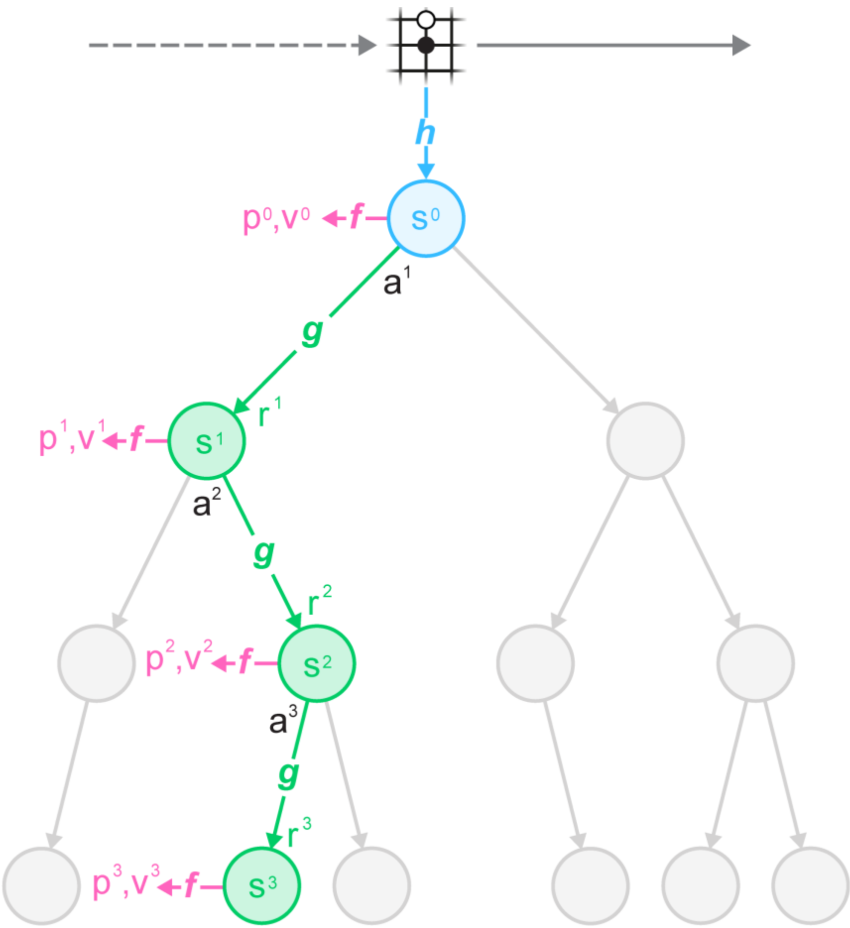



Four
~~Three~~ neural networks:

- Representation function ***h***
  - maps raw observations to latent representations $s_t$
- Dynamics function ***g***
  - maps $(s_t,a_t)$ to $s_{t+1}$ and predicts the reward $r$
- Policy and value prediction function ***f***
  - akin actor-critic network
- Reconstruction function $h^{-1}$
  - Reconstructs the input. Allows to pretrain also ***h*** and ***g*** by unsupervised random exploration of the environment

Scholz et al. (IJCNN 2021) Improving Model-Based RL …, https://arxiv.org/abs/2102.05599v1

# MuZero: Extension for Continuous Actions




So far, discrete actions:

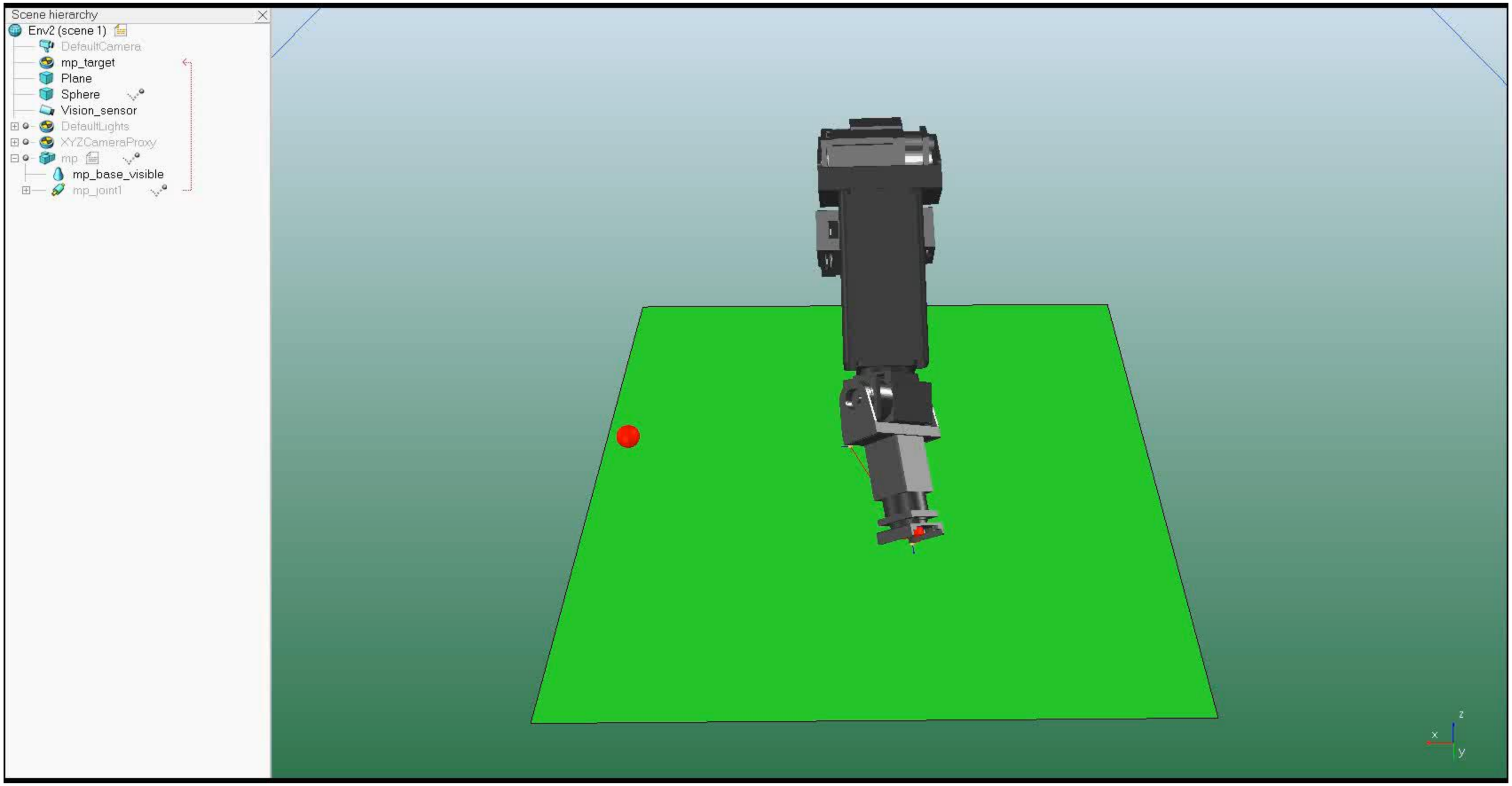
- Branching in MCTS happened along discrete action choices

Continuous action space:

- Exploration and branching happen by randomly selecting continuous action vectors
- These are chosen from a Gaussian probability density function, centered around the current most-likely action (according to the policy)
- The tree will still consist of discrete branches, as before

Yang et al. (2020) Continuous Control for Searching and Planning with a Learned Model

# Continuous MuZero: Intercepting a Rolling Ball




Video source: Lennart Clasmeier

# Summary of RL

- TD Error, derived from theory, explains biological learning
- RL complements unsupervised & supervised learning
  - adds goal-directedness
- Dynamic programming solves temporal credit assignment problem
- Techniques of supervised learning are useful helpers:
  - MLP and deep architectures approximate return
- Hierarchical RL: slow high-level actions modulate/control fast low-level actions
- Model-based RL coexists with model-free RL