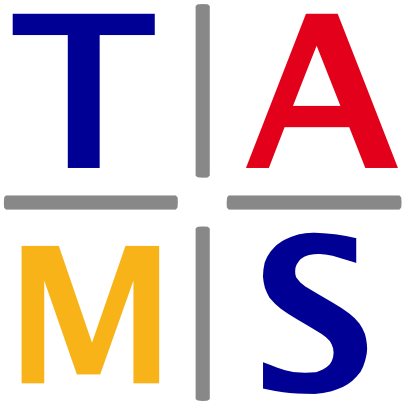
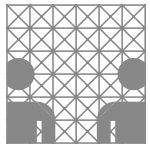
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN Faculty
Department of Informatics

# PPO Reinforcement Learning for Dexterous In-Hand Manipulation




University of Hamburg
Faculty of Mathematics, Informatics and Natural Science
Department of Informatics
**Technical Aspects of Multimodal Systems**



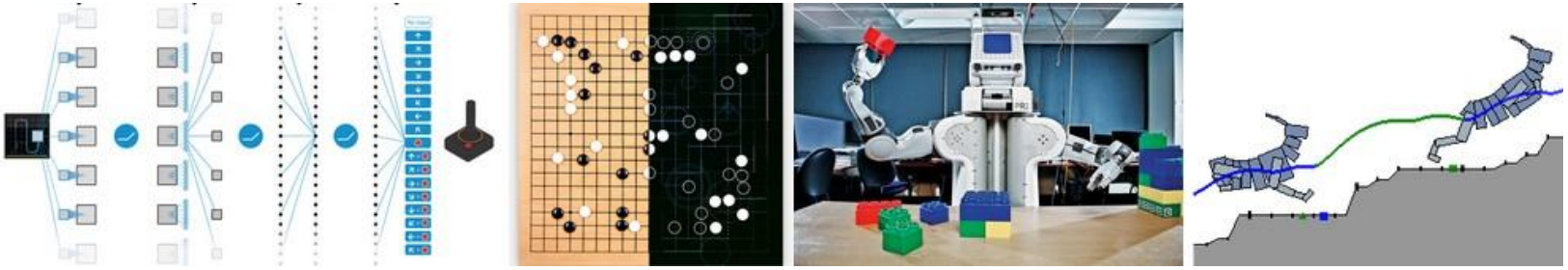
# Goals of this presentation

- ✓ Introduction to Reinforcement Learning in Robotics
- ✓ What is PPO and what are its advantages
- ✓ Using PPO to improve Robot Dexterity

# Reinforcement Learning - Introduction

Examples:

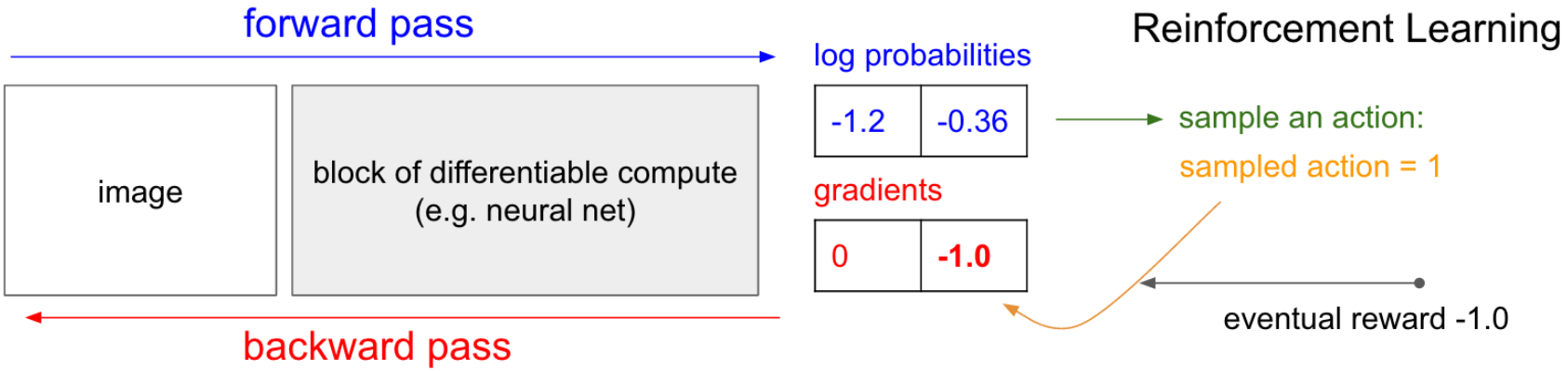
- Deep Q learning ATARI games
- AlphaGo
- Berkeley robot stacking Legos
- Physically-simulated quaruped movement



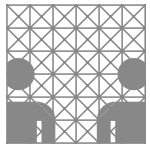
Source: [1]

# Reinforcement Learning – Policy Gradients

- On-policy vs Off-policy
- Do action, act like gradient probability is 1.0, backpropagate
- PG‘s do not require a label like in supervised learning




Source: [1]



# Reinforcement Learning – Problems

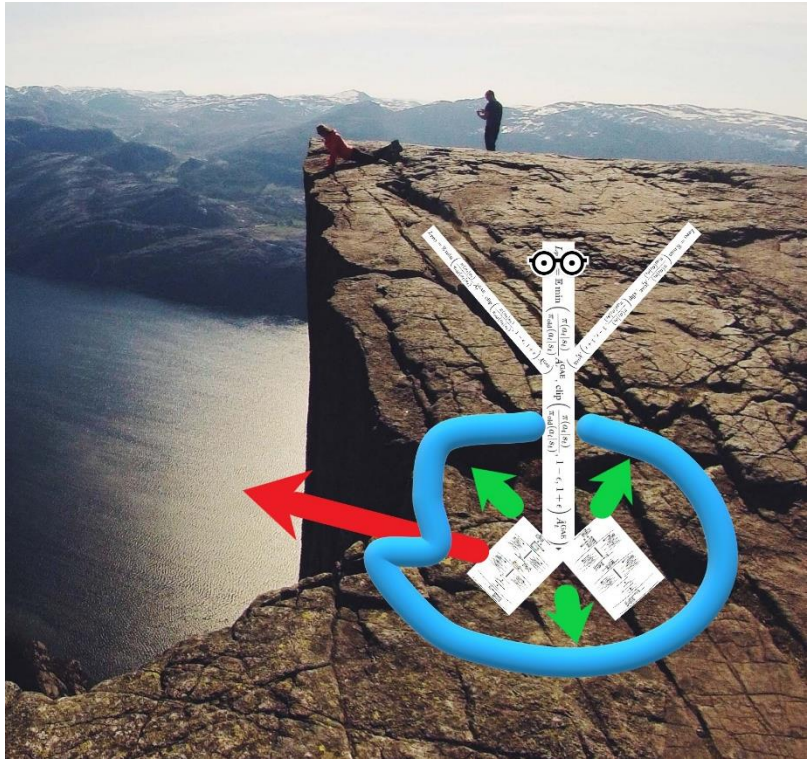
- Generated training data dependent on existing policy
- Does not scale easily to settings with difficult exploration (**Robotics**!)
- Difficult with high-dimensional actions
- Requires many samples & long training times

→ **Higher complexity problems need more sophisticated approaches**

# Reinforcement Learning – The way to improvement

- Using all data so far available and computing the best policy
- We always want the maximum discounted reward when deciding what action to take
- Optimizing the loss function without overfitting

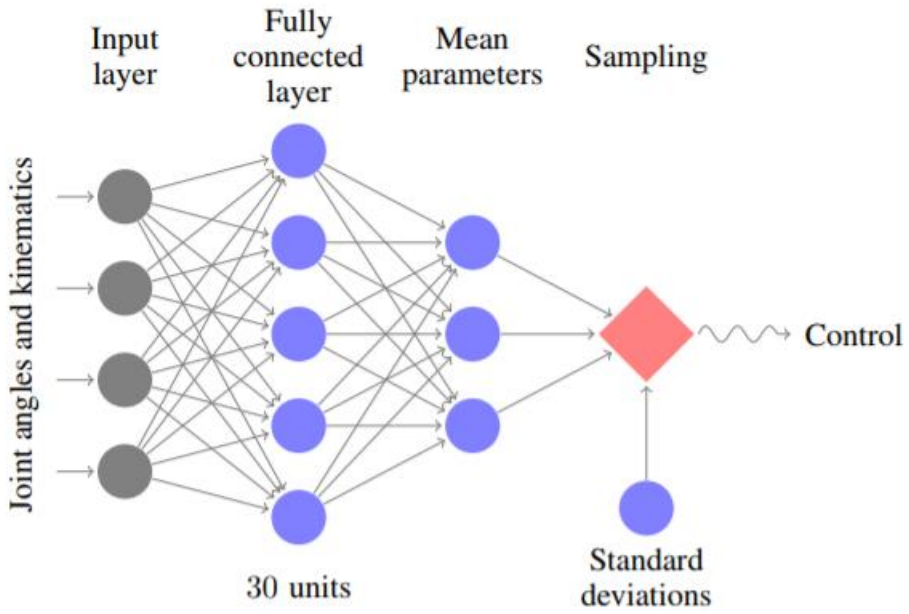
# TRPO - Visualization



Source: [2]

# Trust Region Policy Optimization (TRPO)

- Define a region in which we trust an optimized policy to not have changed catastrophically
- Measuring the change between policies using KL-divergence

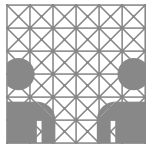



Source: [3]

# Problems with TRPO

- Performs poor on tasks with CNN‘s and RNN‘s
- Implementation is difficult
- Hard to use with architecture with multiple outputs

Source: [4]



# PPO

## Proximal Policy Optimization

# PPO – goals as set by the creators @OpenAI

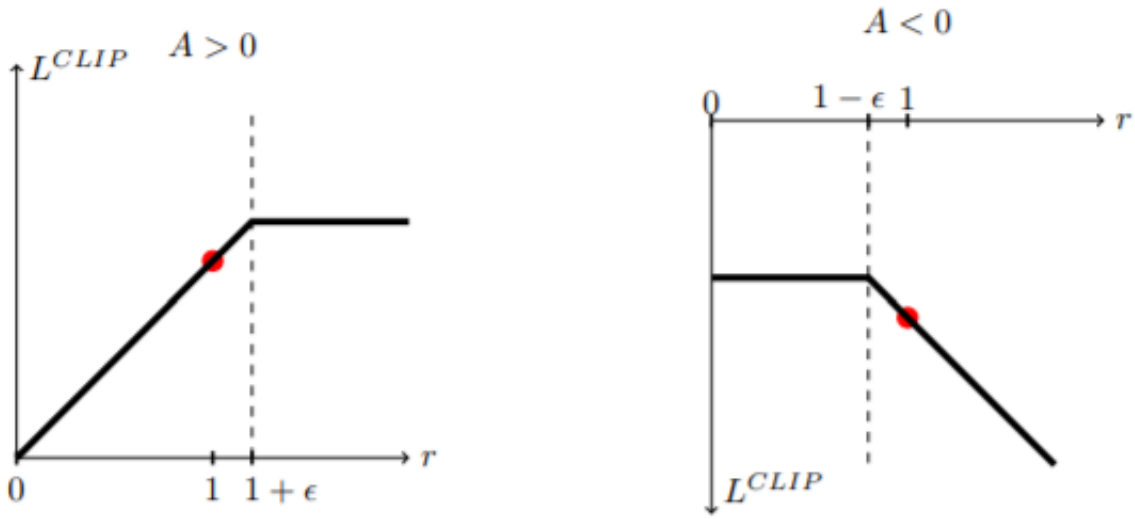
- Simple implementation
- Sample efficient
- Easily tunable

$$L_{\text{PPO}} = \mathbb{E}\min\left(\frac{\pi(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}\hat{A}_t^{\text{GAE}},\ \text{clip}\left(\frac{\pi(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)},\ 1-\epsilon,\ 1+\epsilon\right)\hat{A}_t^{\text{GAE}}\right)$$

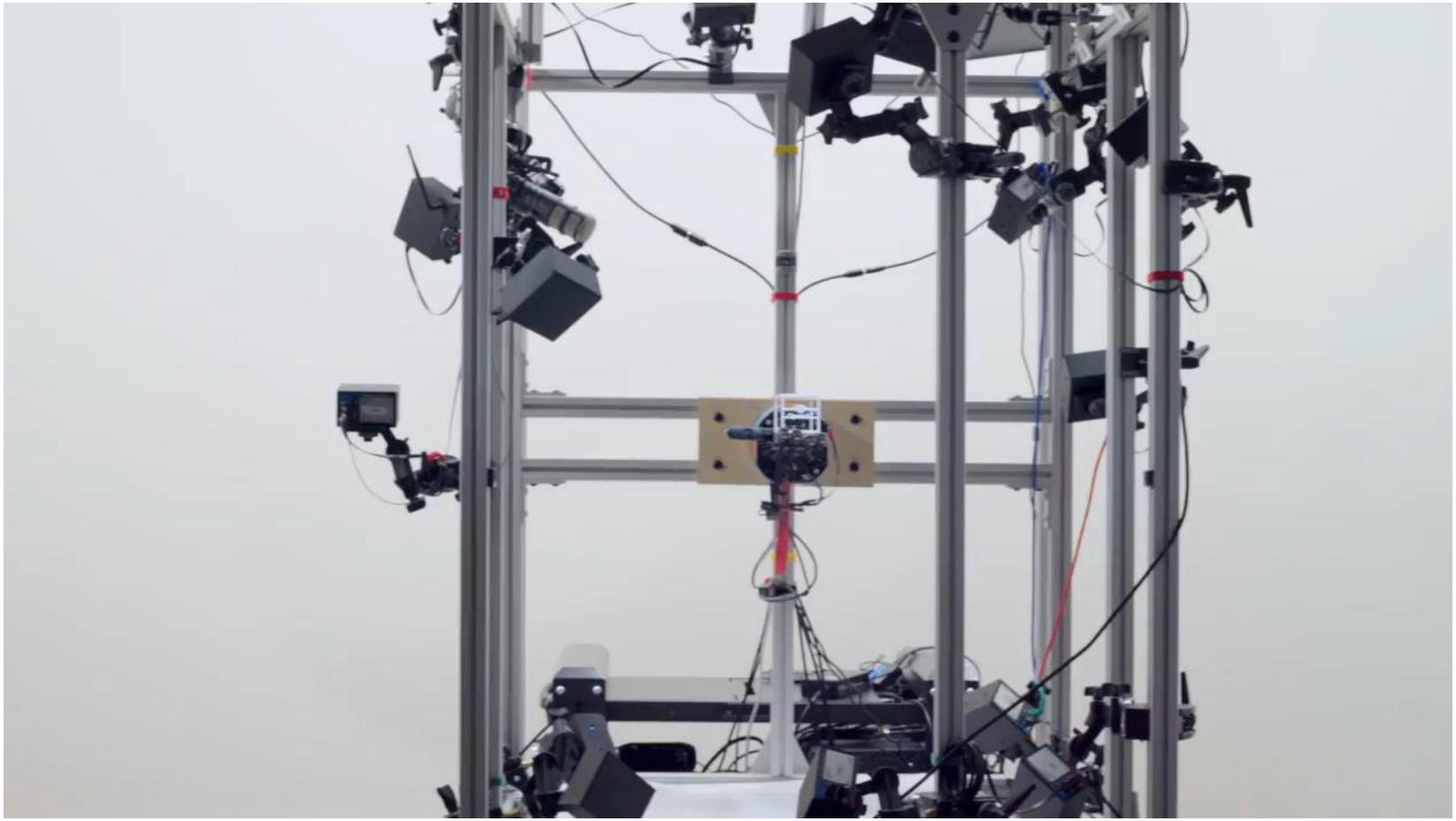
# PPO – In practice

- Training of two separate networks (with similar architecture):
  - Policy network, mapping observations to actions
  - Value network, prediction the future reward from the current state
- Value network is only used in training and has access to more information than the life version, such as:
  - Hand joint angles + velocities
  - Object velocity
  - Object + Target orientation
- Live system uses only the policy network

# PPO – Clipping aka the „Magic“

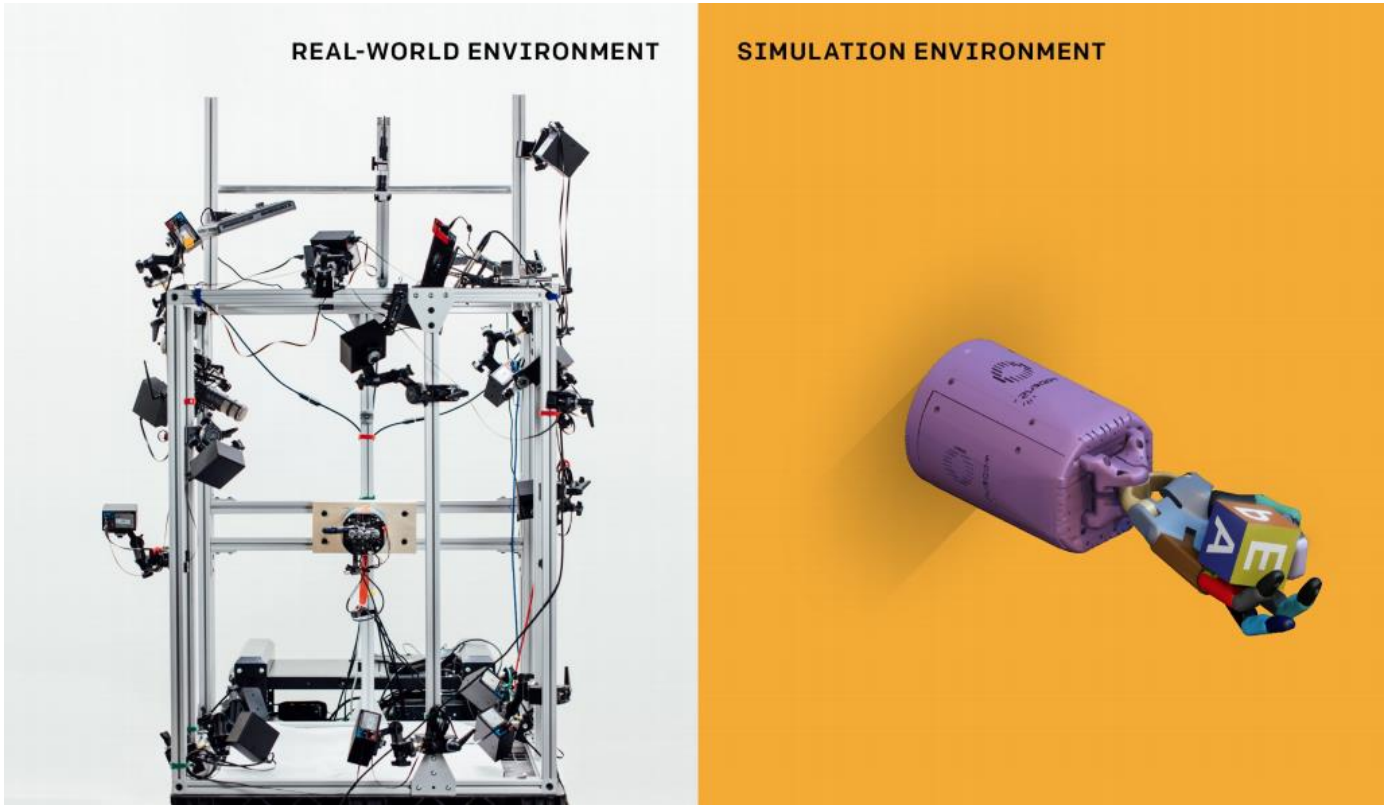



Source: [5]



Source: [6]

# Real-World vs Simulation




Source: [5]

# Real-World


- Wide range of tasks
- Diverse environments
- Fixed physics

→ **extremely hard so simulate**

Source: [5]

# How to make Simulations transferable

Limiting control policy observations:

- Simplification of pose estimator
- i.e. only fingertip pose, not pressure

Randomizations:

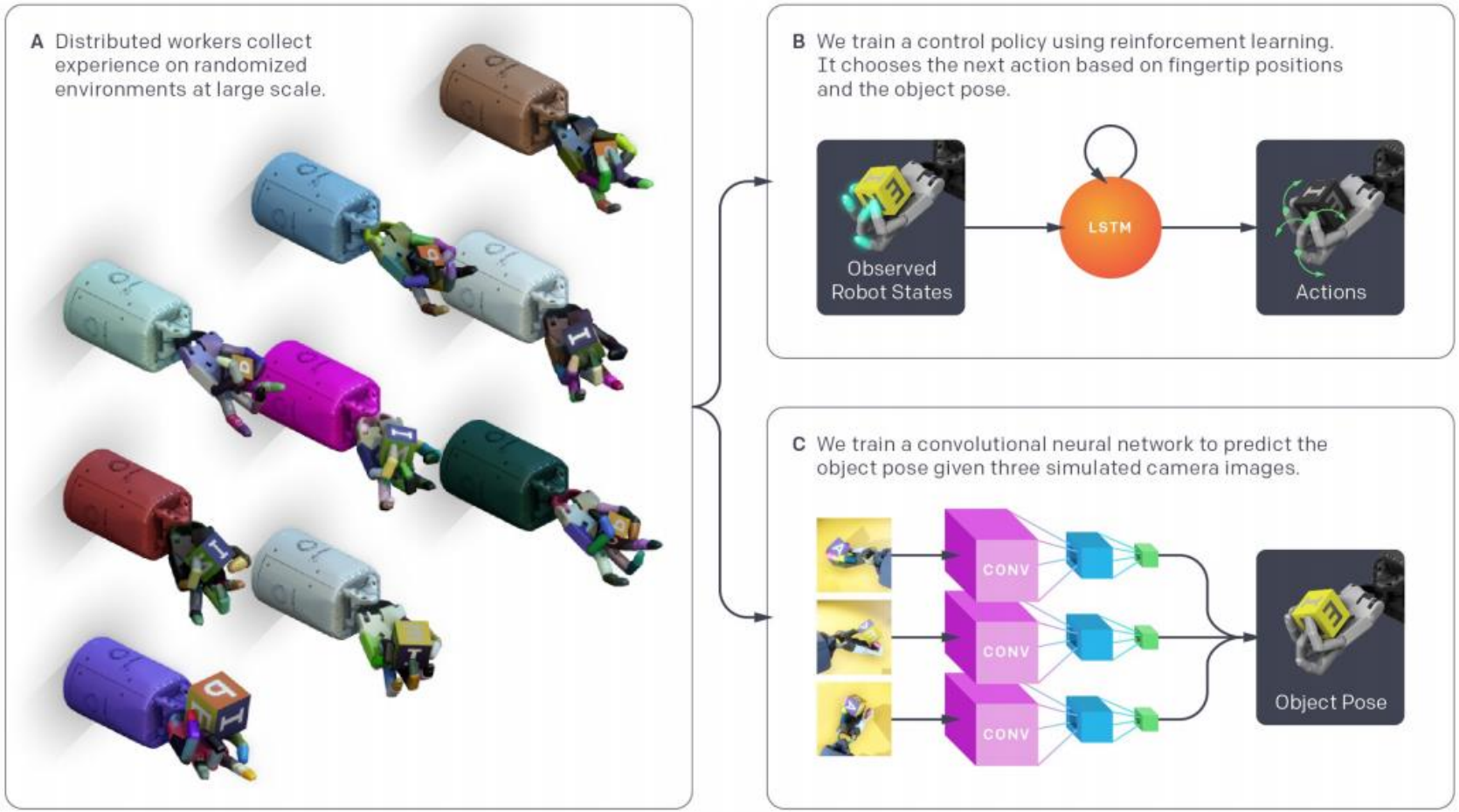
- Observation noise
- Physics randomizations
- Unmodeled effects


Source: [5]

# Examples of different simulation environments

Source: [5]

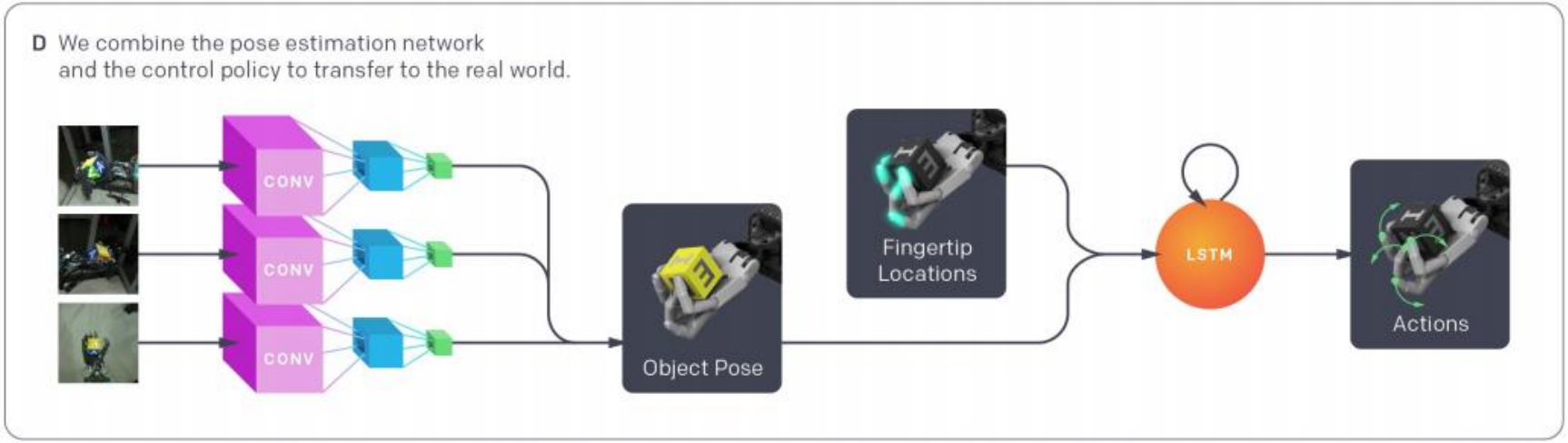
# Complete System - Overview



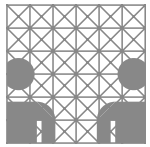

Source: [5]

# Complete System - Overview

- 3 real cameras feed CNN‘s and predict object pose
- Control policy takes that object pose & the fingertip locations
- Control policy outputs actions




Source: [5]



# Conclusion

Main **benefits** while using Reinforcement Learning in Robotics:

- No previous real life experience needed
- Robot agent does not have to collect data in the real world
- Simulations can make use of scaling and distributed infrastructure

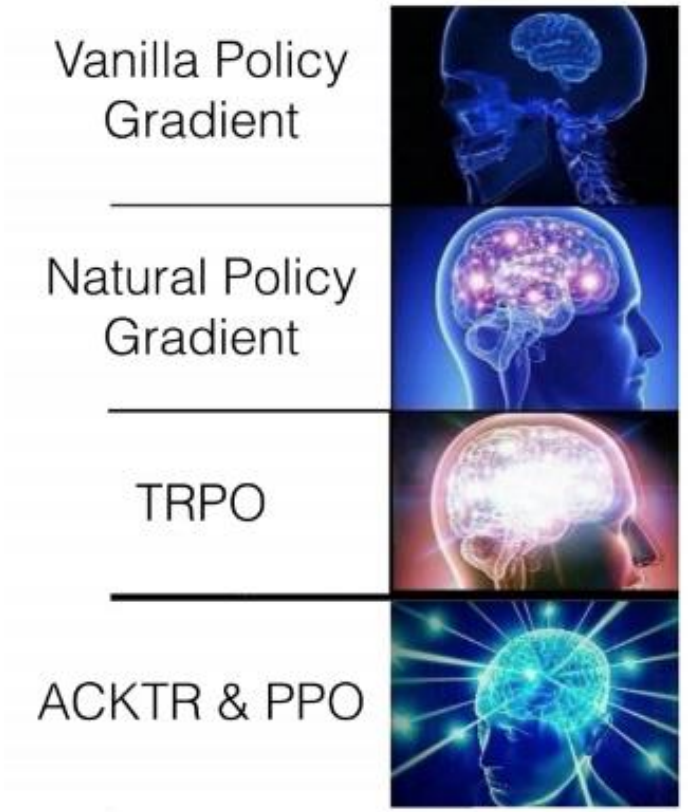
Main **challenges** for using Reinforcement Learning in Robotics

- Differences between reality and simulation need to be bridged through randomizations
- Policies need to overcome optimization obstacles, i.e. through PPO

# Outlook

Similar approaches to be tested

- KFAC
  - Blockwise approximation fo Fisher Information matrix (FIM)
  - Very fast in optimizing objectives through momentum
- ACKTR
  - Advantage actor-critic model
  - Extremely high sample efficiency




Source: [7]

# Sources

[1] Andrej Karpathy (2016). *Deep Reinforcement Learning: Pong from Pixels*, http://karpathy.github.io/2016/05/31/rl/

[2] Photo from www.pexels.com, free for personal and commercial use under CC0 Licence https://www.pexels.com/photo/adventure-cliff-lookout-people-6763/ and paint3D by the author

[3] John Schulmann, Sergey Levine, Philipp Moritz, Michael Jordan, Peter Abeel(2015). *Trust Region Policy Optimization*, Volume 37: International Conference on Machine Learning, 7-9 July 2015, Lille, France

[4] GIF from https://giphy.com/, original link not functional, free for personal and non-commercial use as per https://giphy.com/terms

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov(2017). *Proximal Policy Optimization Algorithms*. https://arxiv.org/pdf/1707.06347.pdf

# Sources cont.

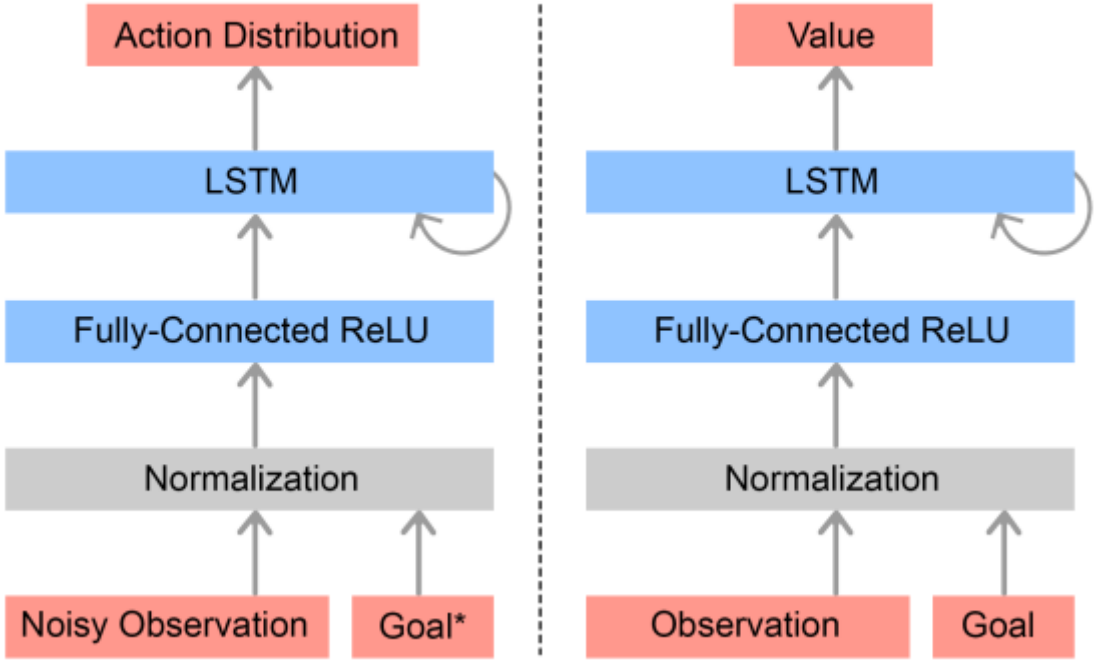
[6] *Learning Dexterity* (2018), as seen on the OpenAI Youtube channel:
https://www.youtube.com/watch?v=jwSbzNHGflM

[7] John Schulman (2017). *Advanced Policy Gradient Methods: Natural Gradient, TRPO, and More*, https://drive.google.com/file/d/0BxXI_RttTZAhMVhsNk5VSXU0U3c/

# Addendum

- Policy and Value Network
- GAE
- Distributed Training Structure
- Effect of Memory
- Physics vs Simulation Results
- Values for different Physics Parameters
- Different Training Environments Compared

# Policy and Value network




Source: [5]

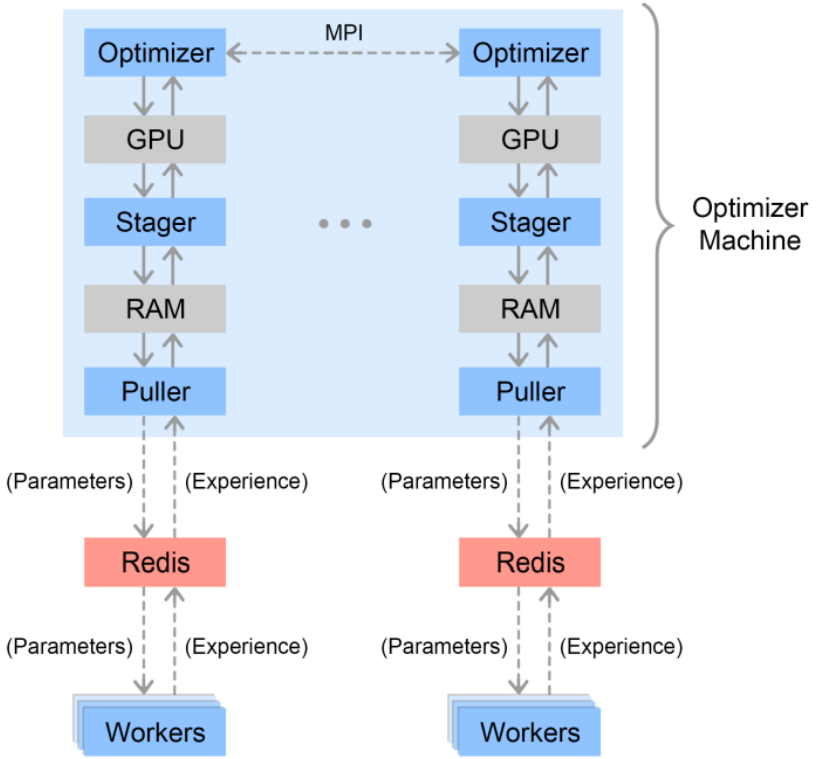
# GAE – Generalized Advantage Estimator

$$\hat{V}_t^{(k)} = \sum_{i=t}^{t+k-1} \gamma^{i-t} r_i + \gamma^k V(s_{t+k}) \approx V^\pi(s_t, a_t)$$

$$\hat{V}_t^{\text{GAE}} = (1-\lambda) \sum_{k>0} \lambda^{k-1} \hat{V}_t^{(k)} \approx V^\pi(s_t, a_t),$$

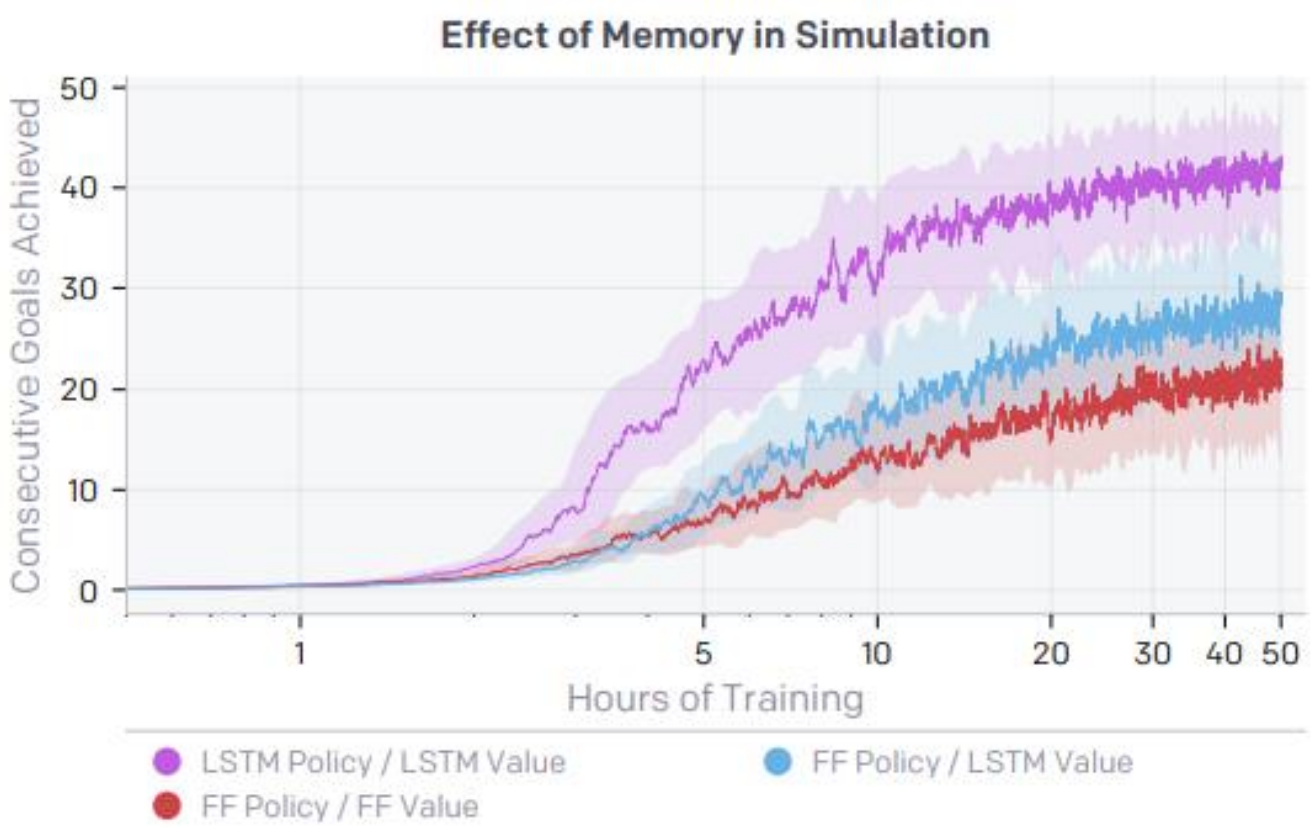
$$\hat{A}_t^{\text{GAE}} = \hat{V}_t^{\text{GAE}} - V(s_t) \approx A^\pi(s_t, a_t).$$

# Distributed Training Structure

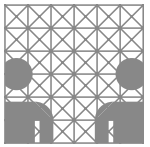



Source: [5]

# Effect of Memory in Simulation




Source: [5]



# Simulated vs Physical Results

| Simulated task | Mean | Median | Individual trials (sorted) |
|---|---|---|---|
| Block (state) | $43.4 \pm 13.8$ | 50 | - |
| Block (state, locked wrist) | $44.2 \pm 13.4$ | 50 | - |
| Block (vision) | $30.0 \pm 10.3$ | 33 | - |
| Octagonal prism (state) | $29.0 \pm 19.7$ | 30 | - |
| **Physical task** | | | |
| Block (state) | $18.8 \pm 17.1$ | 13 | 50, 41, 29, 27, 14, 12, 6, 4, 4, 1 |
| Block (state, locked wrist) | $26.4 \pm 13.4$ | 28.5 | 50, 43, 32, 29, 29, 28, 19, 13, 12, 9 |
| Block (vision) | $15.2 \pm 14.3$ | 11.5 | 46, 28, 26, 15, 13, 10, 8, 3, 2, 1 |
| Octagonal prism (state) | $7.8 \pm 7.8$ | 5 | 27, 15, 8, 8, 5, 5, 4, 3, 2, 1 |

Source: [5]

# Ranges of physics parameter randomizations

| Parameter | Scaling factor range | Additive term range |
|---|---|---|
| object dimensions | uniform($[0.95, 1.05]$) | |
| object and robot link masses | uniform($[0.5, 1.5]$) | |
| surface friction coefficients | uniform($[0.7, 1.3]$) | |
| robot joint damping coefficients | loguniform($[0.3, 3.0]$) | |
| actuator force gains (P term) | loguniform($[0.75, 1.5]$) | |
| joint limits | | $\mathcal{N}(0, 0.15)$ rad |
| gravity vector (each coordinate) | | $\mathcal{N}(0, 0.4)$ m/s$^2$ |

Source: [5]

# Different training environments compared

| Training environment | Mean | Median | Individual trials (sorted) |
|---|---|---|---|
| All randomizations (state) | $18.8 \pm 17.1$ | 13 | 50, 41, 29, 27, 14, 12, 6, 4, 4, 1 |
| No randomizations (state) | $1.1 \pm 1.9$ | 0 | 6, 2, 2, 1, 0, 0, 0, 0, 0, 0 |
| No observation noise (state) | $15.1 \pm 14.5$ | 8.5 | 45, 35, 23, 11, 9, 8, 7, 6, 6, 1 |
| No physics randomizations (state) | $3.5 \pm 2.5$ | 2 | 7, 7, 7, 3, 2, 2, 2, 2, 2, 1 |
| No unmodeled effects (state) | $3.5 \pm 4.8$ | 2 | 16, 7, 3, 3, 2, 2, 1, 1, 0, 0 |
| All randomizations (vision) | $15.2 \pm 14.3$ | 11.5 | 46, 28, 26, 15, 13, 10, 8, 3, 2, 1 |
| No observation noise (vision) | $5.9 \pm 6.6$ | 3.5 | 20, 12, 11, 6, 5, 2, 2, 1, 0, 0 |

Source: [5]