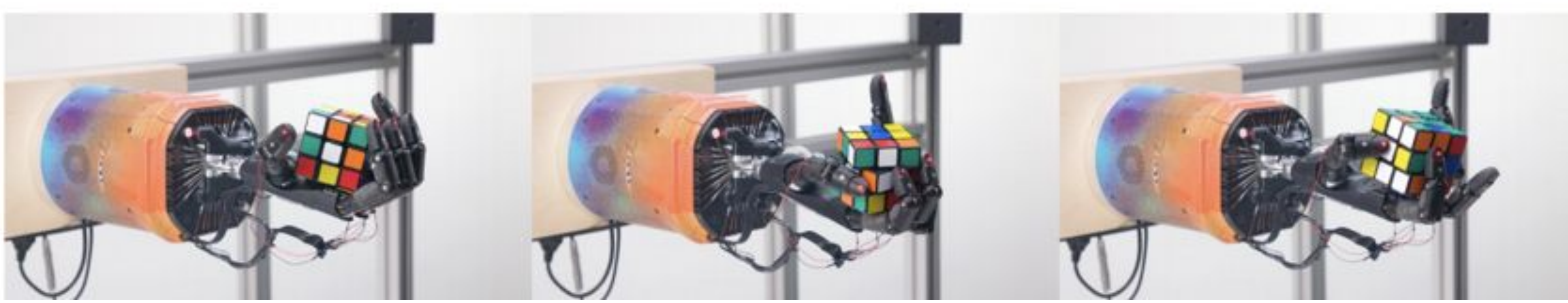
# Flexible Learning Reading Group @TU Berlin
# 6th Session: 6th of November 2019

## SOLVING RUBIK'S CUBE WITH A ROBOT HAND

A PREPRINT

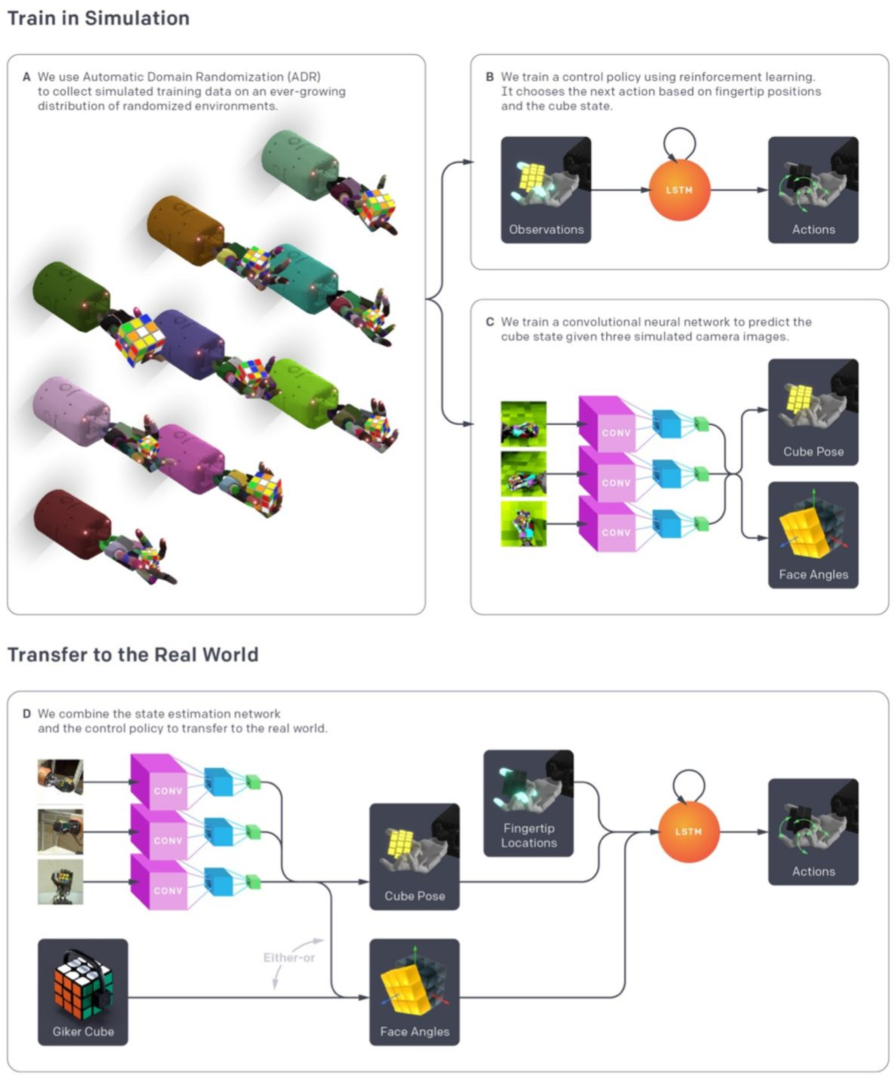
**OpenAI**
Ilge Akkaya,* Marcin Andrychowicz,* Maciek Chociej,* Mateusz Litwin,* Bob McGrew,* Arthur Petron,*
Alex Paino,* Matthias Plappert,* Glenn Powell,* Raphael Ribas,* Jonas Schneider,* Nikolas Tezak,*
Jerry Tworek,* Peter Welinder,* Lilian Weng,* Qiming Yuan,* Wojciech Zaremba,* Lei Zhang*

October 17, 2019



Picture credits - Akkaya et al. (2019: ArXiv)

# General Sim2Real Setup




A. Automatic Domain Randomization (ADR)

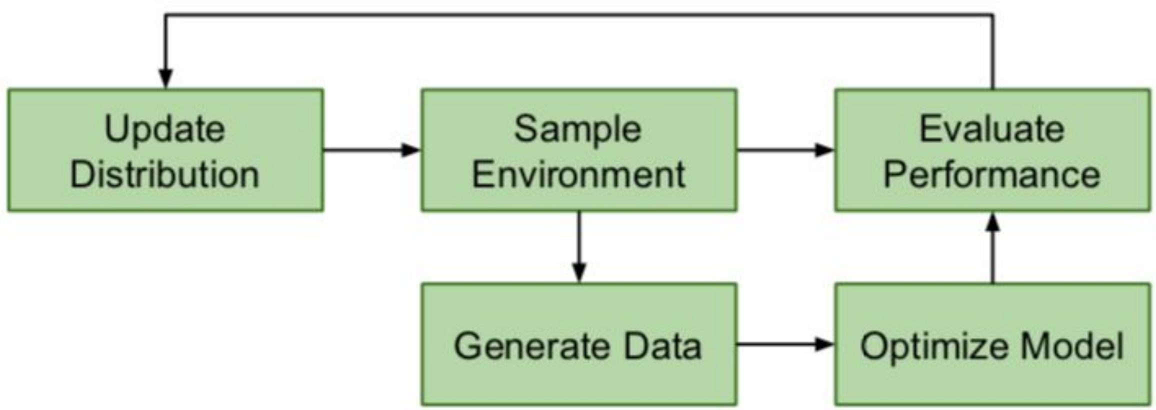
B. Distributed PPO + GAE LSTM-policy

C. ResNet-50-based State estimation

D. 0-Shot Sim2Real Transfer Results

# Automatic Domain Randomization

Hypothesis: Transfer via Meta-Learning results from training on diverse set of environments




Limited network capacity enforces learn-to-learn!

→ Natural Adaptive Curriculum!

# On-Policy RL: Policy Gradient Methods

$$\pi^\star = arg\max_\pi V^\pi(s_0) = arg\max_\pi \left[ \mathbb{E}_\pi [\sum_t \gamma^{t-1} r_t | s_0] \right]$$

Vanilla Policy Gradients (Sutton et al., 2000):

$$\nabla_\theta V^{\pi_\theta}(s_0) = \mathbb{E}_\pi [\cancel{Q^{\pi}}(s,a) \nabla_\theta \ln \pi_\theta(a|s)]$$

Generalized Advantage Estimation (GAE):

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

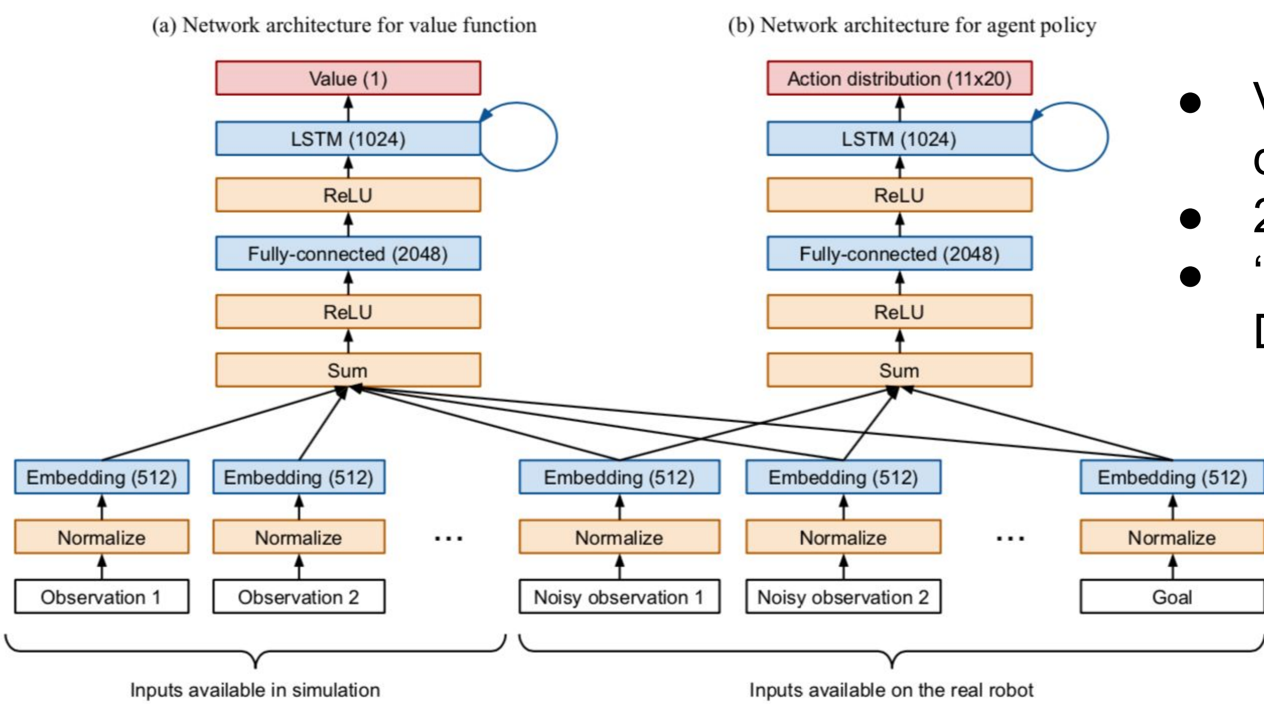
$$\hat{A}_t^{GAE}$$

$$\hat{V}_t^{GAE} = (1-\lambda) \sum_{k>0} \lambda^{k-1} \hat{V}^{(k)}, \quad 0 < \lambda < 1$$

# Training the Control Policy: PPO + GAE

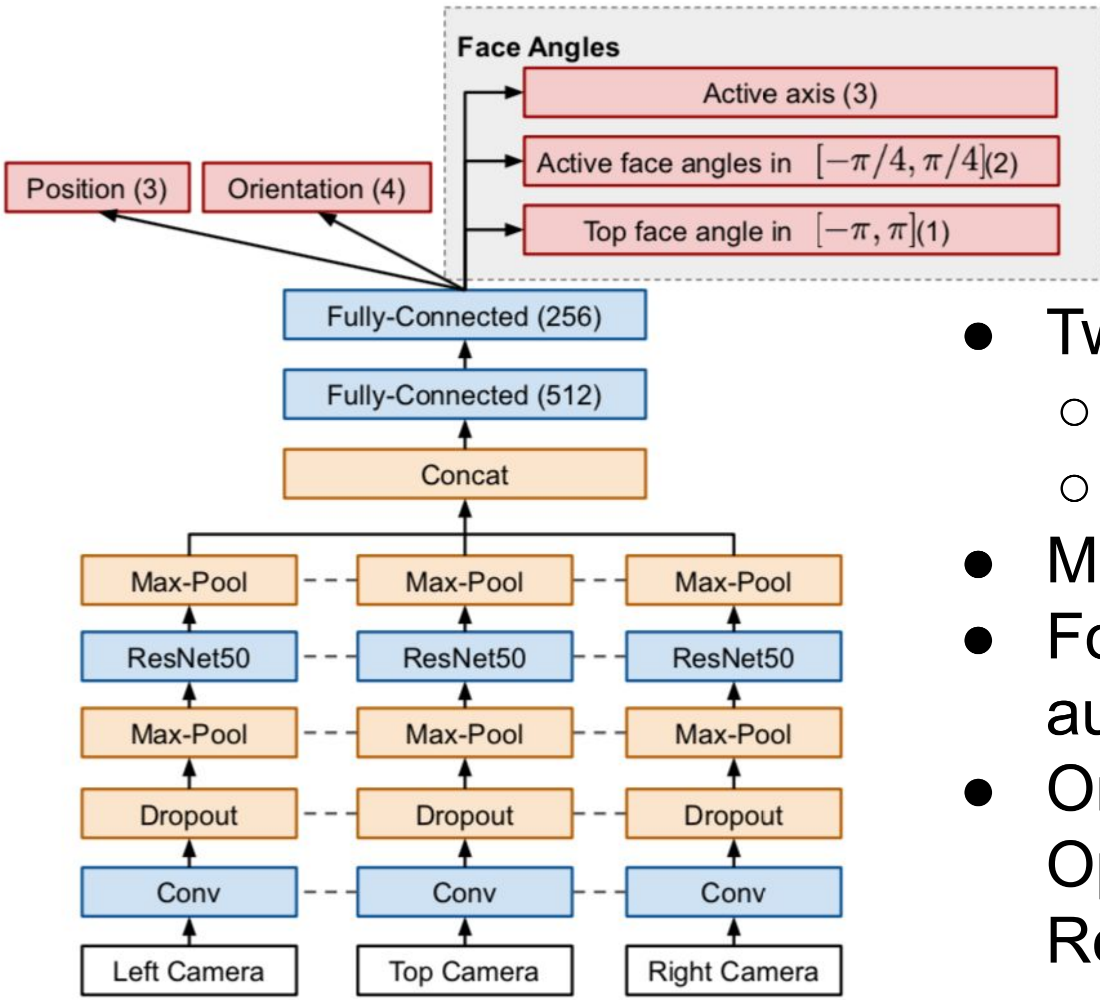
→ Proximal Policy Optimization (Schulman et al., 2017):

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




- Value Net turned off during execution!
- 20 joints - 11 bins
- ‘Multi-Categorical’ Distribution

# The State Estimation Vision System

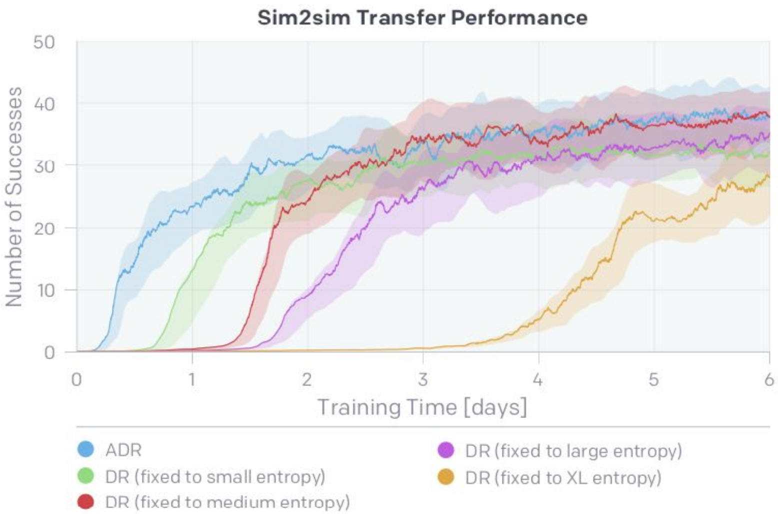


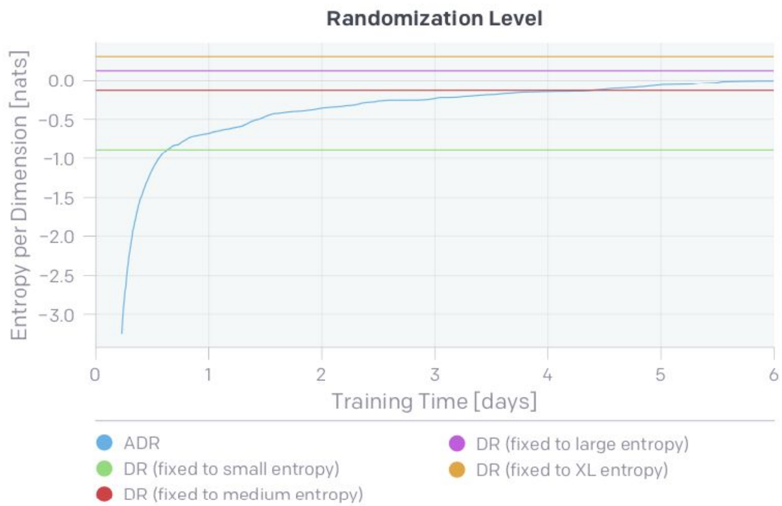

- Two Versions:
  - Sticker Support
  - Sensors for Angles
- MSE + Cross-Entropy
- Focal Loss: Dynamic + automatic loss weights
- Only synthetic data: Open AI Remote Rendering Backend

# Vision Module & Sim2Sim Results

| Experiment | Errors (Sim) | | | Errors (Real) | | |
|---|---|---|---|---|---|---|
| | Orientation | Position | Top Face | Orientation | Position | Top face |
| Full Model | 6.52° | **2.63** mm | 11.95° | **7.81°** | **6.47 mm** | **15.92°** |
| No Domain Randomization | **3.95°** | 2.97 mm | **8.56°** | 128.83° | 69.40 mm | 85.33° |
| No Focal Loss | 15.94° | 5.02 mm | 10.17° | 19.10° | 9.416 mm | 17.54° |
| Non-discrete Angles | 9.02° | 3.78 mm | 42.46° | 10.40° | 7.97 mm | 35.27° |

→ ADR + Adaptive Weighting of different tasks is crucial

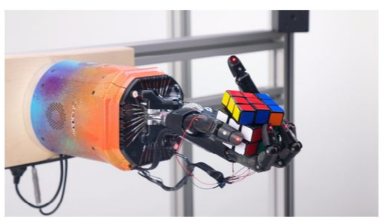


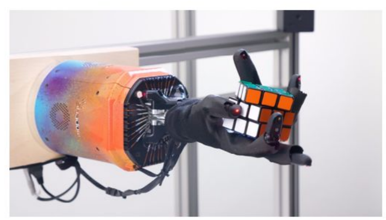

→ Curriculum outperforms fixed levels of randomness

# Sim2Real Results

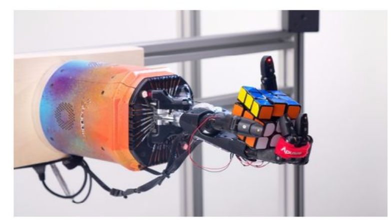
| Policy | Sensing | | ADR Entropy | Successes (Real) | | Success Rate | |
|---|---|---|---|---|---|---|---|
| | Pose | Face Angles | | Mean | Median | Half | Full |
| Manual DR | Vision | Giiker | $-0.569^*$ npd | $1.8 \pm 0.4$ | 2.0 | 0 % | 0 % |
| ADR | Vision | Giiker | $-0.084$ npd | $3.8 \pm 1.0$ | 3.0 | 0 % | 0 % |
| ADR (XL) | Vision | Giiker | 0.467 npd | $17.8 \pm 4.2$ | 12.5 | 30 % | 10 % |
| ADR (XXL) | Vision | Giiker | **0.479 npd** | $\mathbf{26.8 \pm 4.9}$ | **22.0** | **60 %** | **20 %** |
| ADR (XXL) | Vision | Vision | **0.479 npd** | $12.8 \pm 3.4$ | 10.5 | 20 % | 0 % |



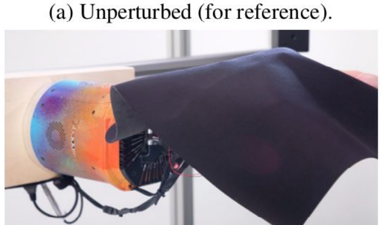
(a) Unperturbed (for reference).



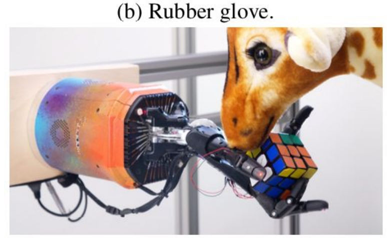
(b) Rubber glove.



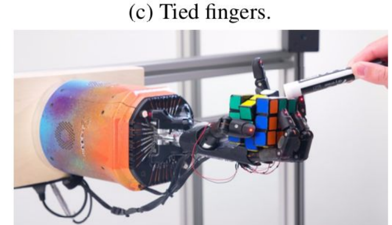
(c) Tied fingers.



(d) Blanket occlusion and perturbation.

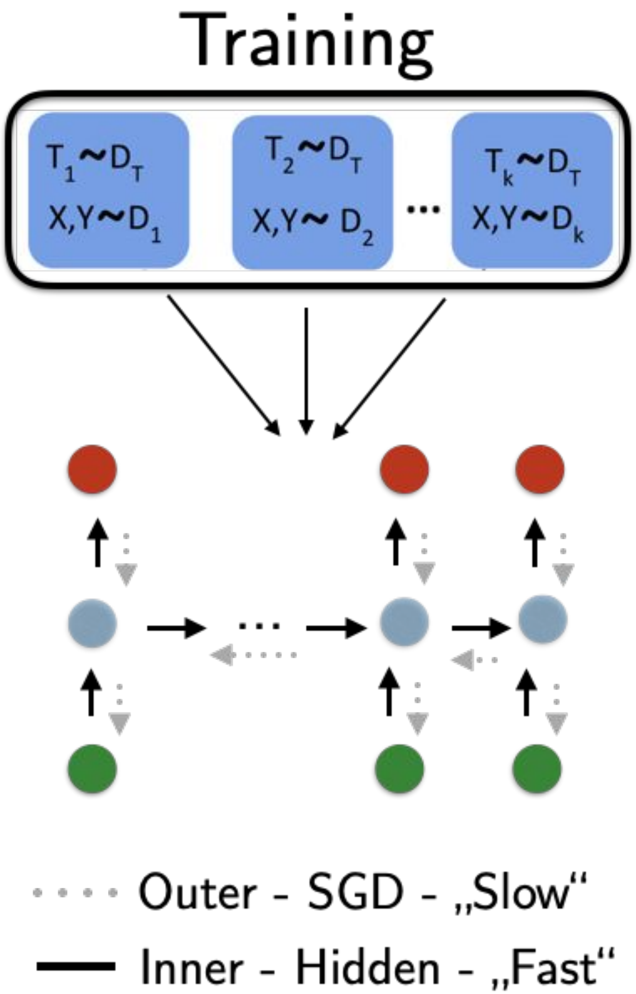


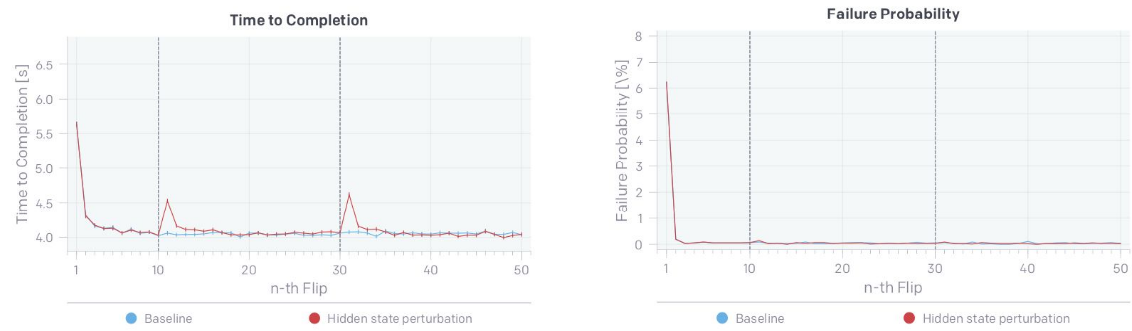
(e) Plush giraffe perturbation.[17]



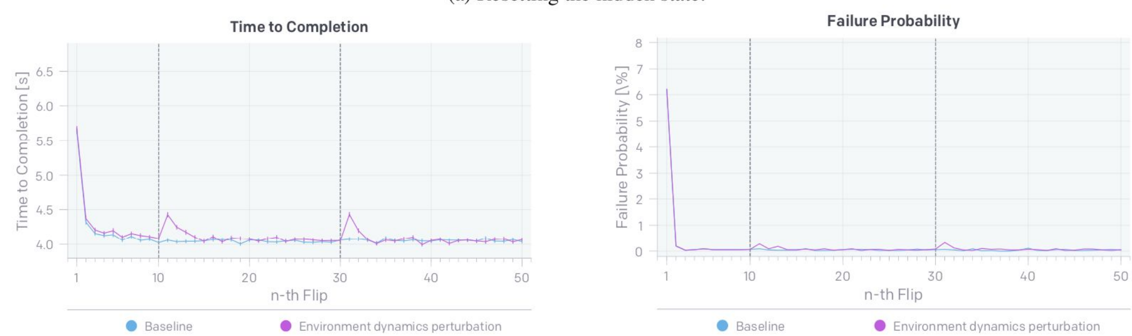
(f) Pen perturbation.

# LSTM + ADR = Meta-Learning?

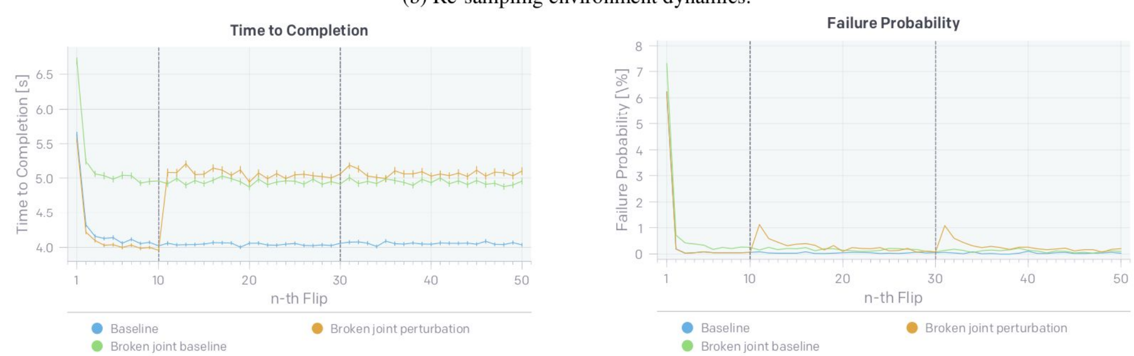



(a) Resetting the hidden state.




(b) Re-sampling environment dynamics.




(c) Breaking a random joint.

# Cool things that I learned!

1. ORRB remote rendering backend on top of Unity
2. ‘Bi-directional’ ADR: Entropy $\downarrow$ if performance < thresh
3. Redis: Centralized storage of parameters + data
4. Adversarial Random Networks - Exploration
5. Concat + Add Embedding - Flexible Inputs
6. Policy Distillation to transfer progress
7. Face Angle “Classification” - Cross-Entropy Loss
8. Multi-Task Vision with Focal Loss Weighting
9. LARS optimizer - Large Batchsize 1024

**Gary Marcus**
@GaryMarcus

Since @OpenAI still has not changed misleading blog post about "solving the Rubik's cube", I attach detailed analysis, comparing what they say and imp
they actually did. IMHO most would not be
nonexperts.

Please zoom in to read & judge for yourself

## Reality

- Neural networks didn't do the solving; a 17-year old symbolic AI algorithm did
- The solving (which face should turn where) algorithm was innate, not learned.
- Reinforcement learning played no role in the choice of which faces to turn (ie what most people call solving).
- What was learned was object manipulation, not cube solving
- Only ONE object was manipulated, and there was no test of generalizability to other objects
- That object was heavily instrumented (eg with bluetooth sensors). The hand was instrumented with LEDs, as well.
- Success rate was only 20%; hand frequently dropped cube