



Introduction to Embodied AI

Lecture 8 – Policy I

Prof. Li Yi



Logistics: Midterm

- Midterm (40% of the total score)
 - 4/21, 3:10 – 5:00, in class
 - First two sections, in total 110 min
 - The third class will be used for lecture
- One-page double-sided A4 cheat sheet
- Scope: from Lecture 1 - Lecture 9 (4/14)

Logistics: Midterm

- Questions:
 - Multi-choice questions
 - True or False
 - Short answer questions
 - simply state why and how
 - some may require mathematical derivations

Logistics: Assignment 2

- Assignment 2 (15%): will be released on 04/11, due 04/26 (Saturday) 11:59PM. (Still has 2 grace days with -10% and -20% score)

☐ Assignment 2 答疑	题目将在4.11发布	0	0	0	0
--	------------	---	---	---	---

- In this assignment you will grasp a known object in the simulator. You need to estimate the object's pose using the point cloud in order to transform grasping poses from the object frame to the world frame.

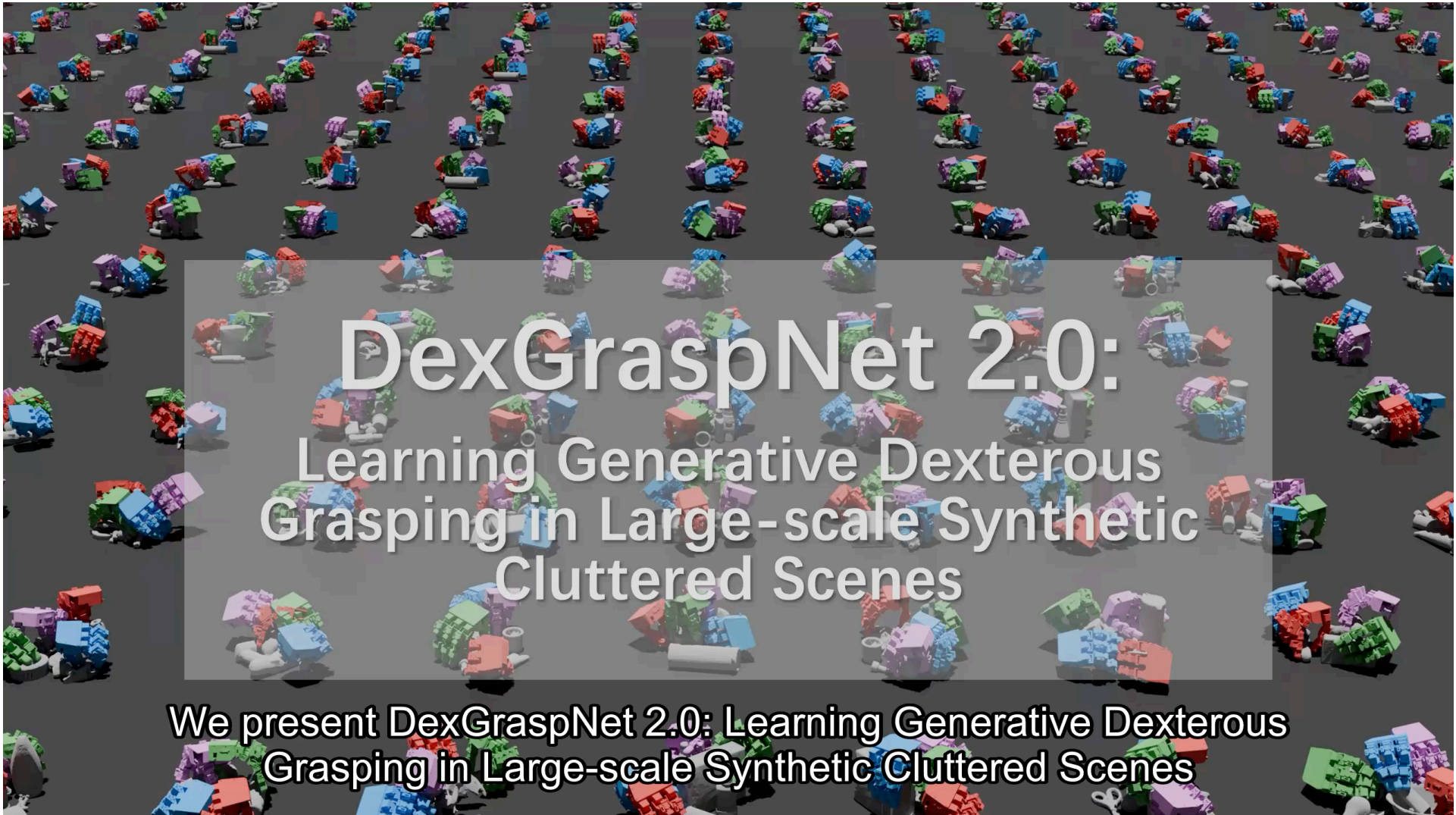
Logistics: Assignment 2

- Assignment 2 and 3 will involve real-world experiments (10% for each). You need to form teams of no more than 4 members to complete the real-world experiments together. The specific schedule will be announced later. (It will be latter than the ddl of assignment 2)



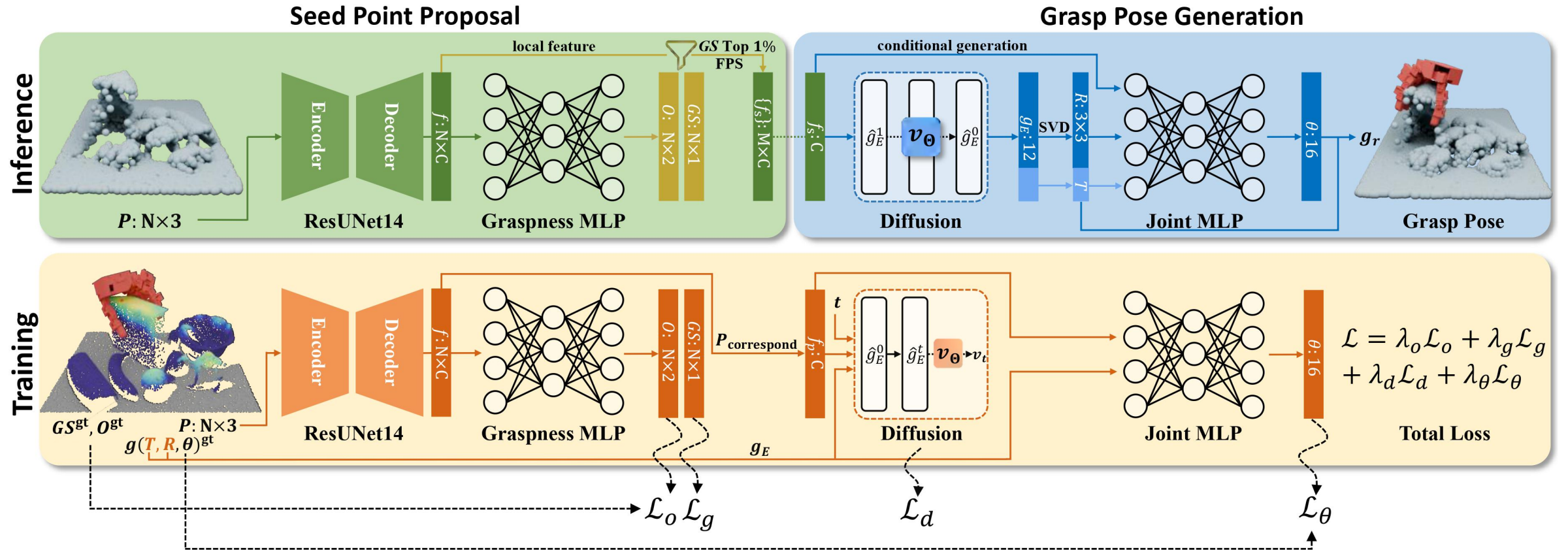
Conditional Grasp Generative Model

Data: DexGraspNet 2.0



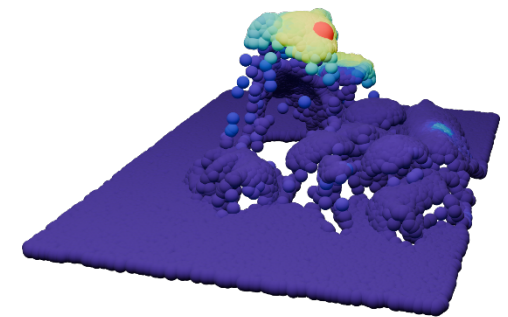
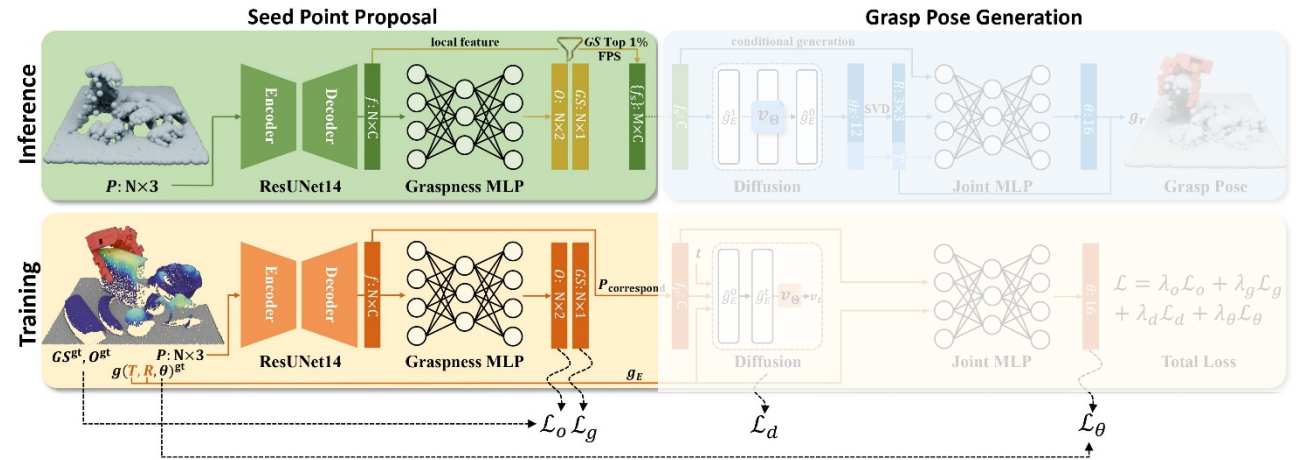
In submission.

Diffusion-based Dexterous Hand Pose Generation



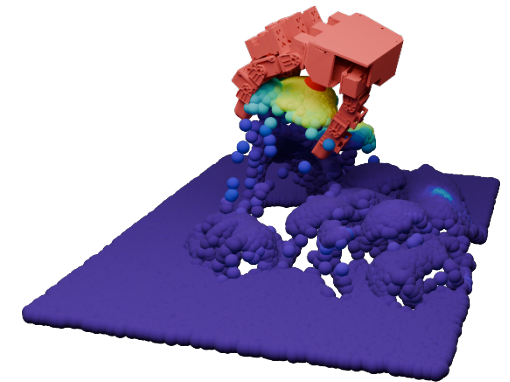
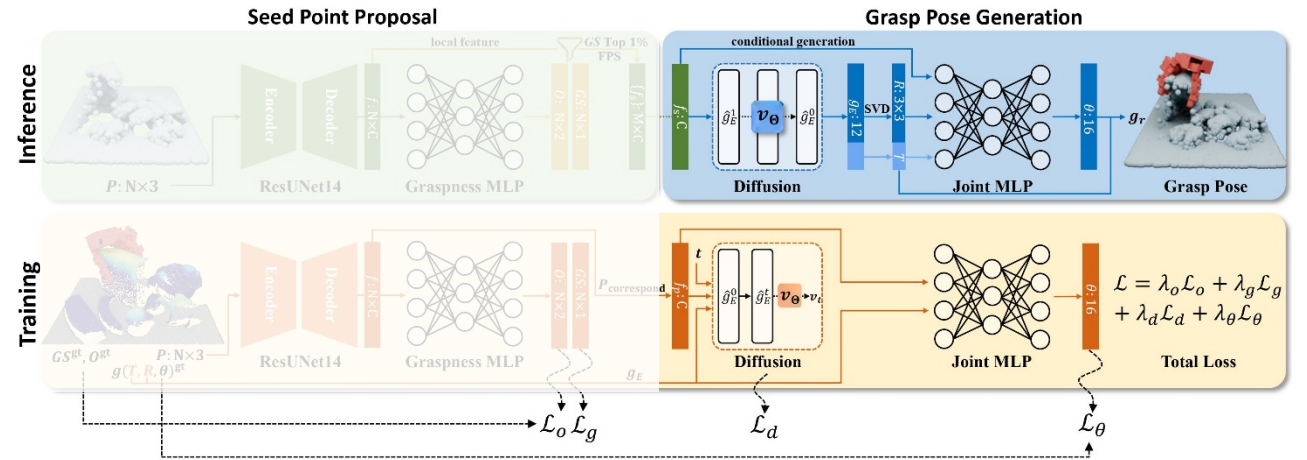
Method

- Stage 1: Where to grasp
 - Use sparse convolution to get the **local feature**
 - Predict graspness and objectness scores
 - Accumulated with nearby gt seed points

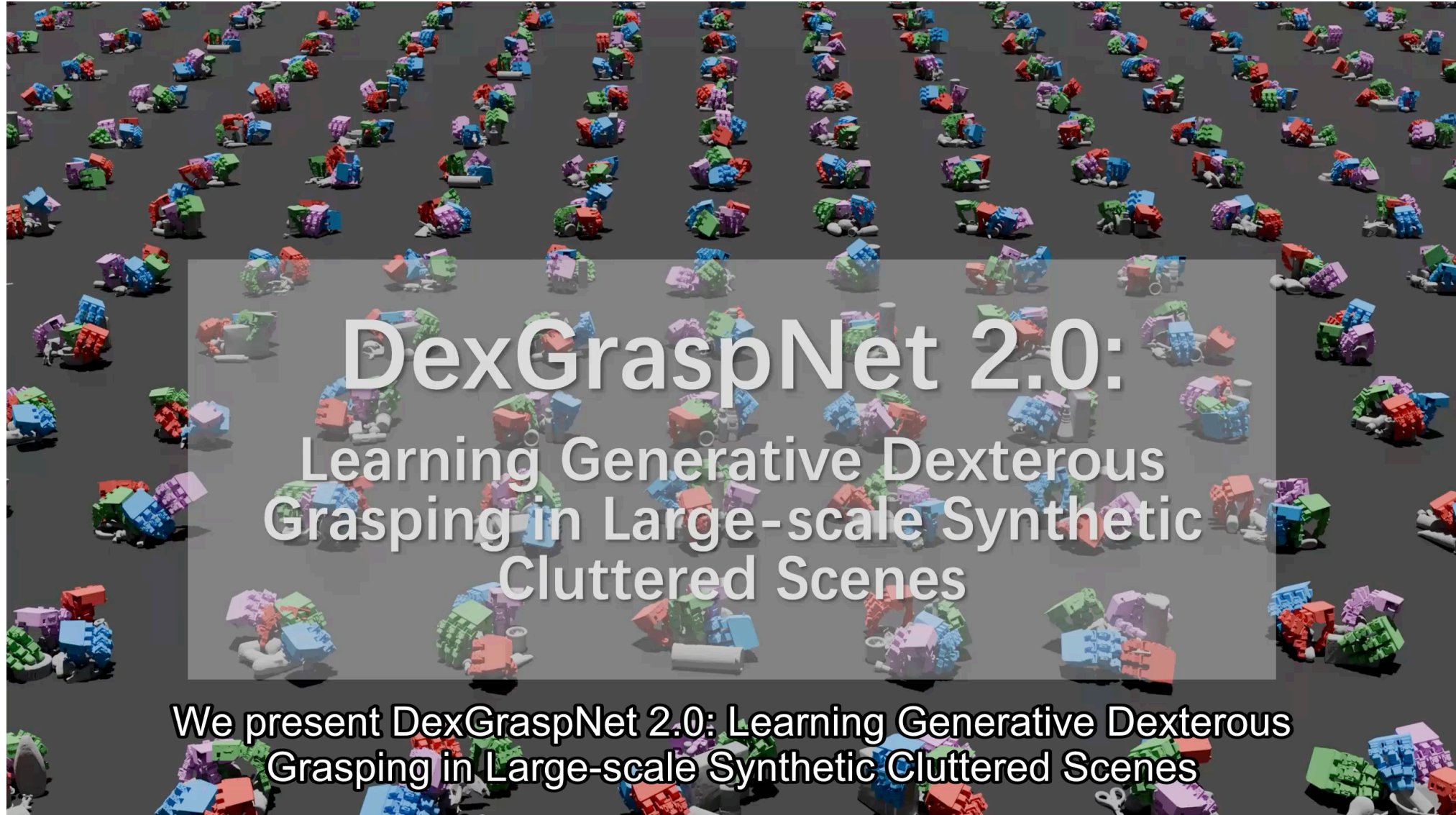


Method

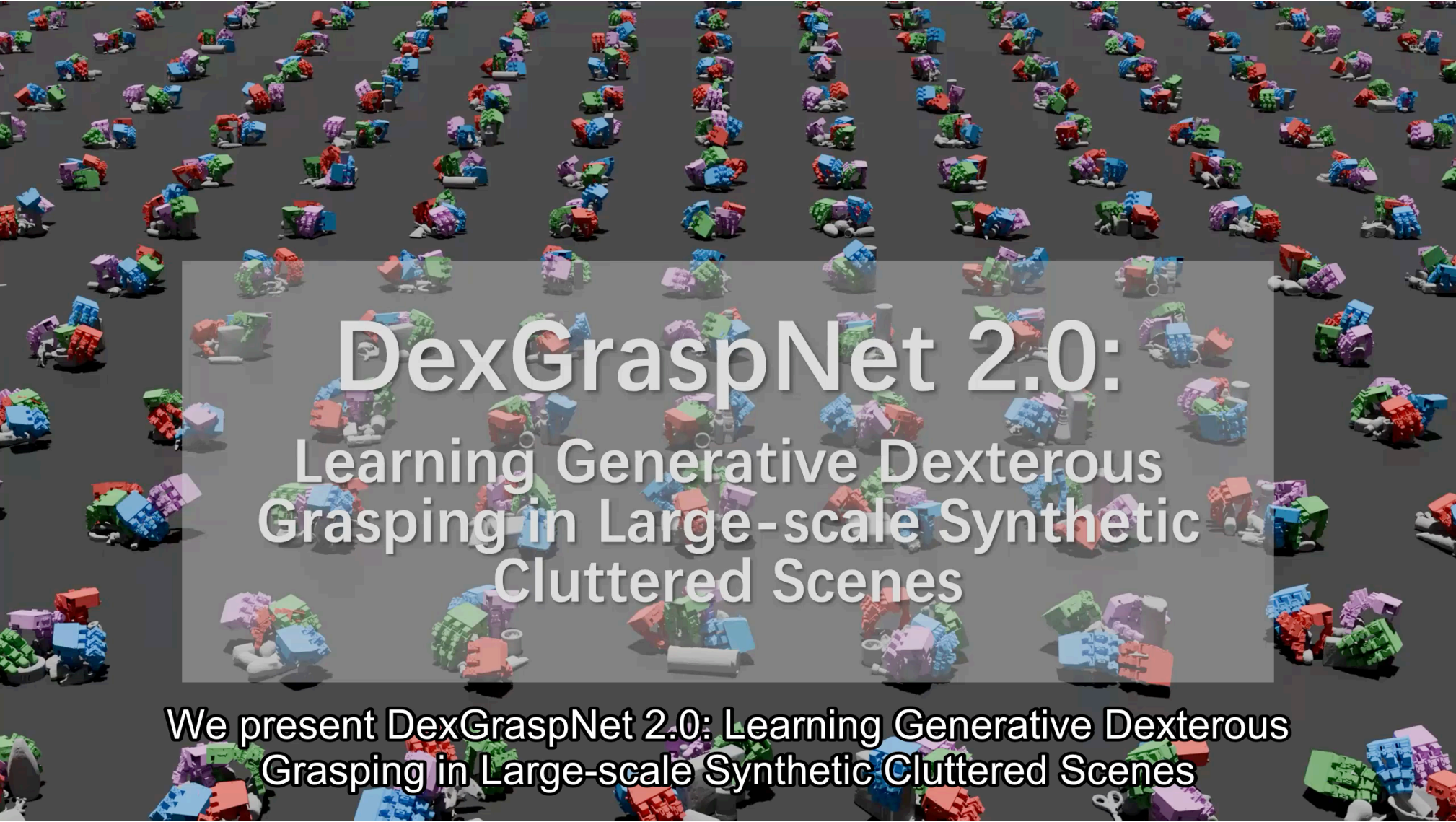
- Stage 2: How to grasp
 - Use only the **local feature** in stage 1
 - Predict residual position T , rotation R , finger joint angles θ
 - To handle the multi-modal grasping pose distribution, we use diffusion model for T and R
 - Then predict θ directly



Dexterous Grasping Pose Diffusion Model



Real Experiments



DexGraspNet 2.0: Learning Generative Dexterous Grasping in Large-scale Synthetic Cluttered Scenes

We present DexGraspNet 2.0: Learning Generative Dexterous Grasping in Large-scale Synthetic Cluttered Scenes

Experiments

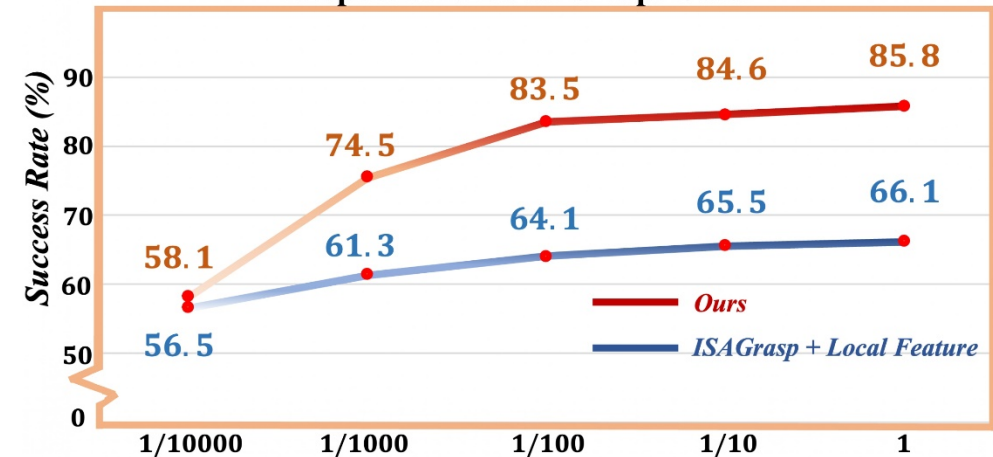
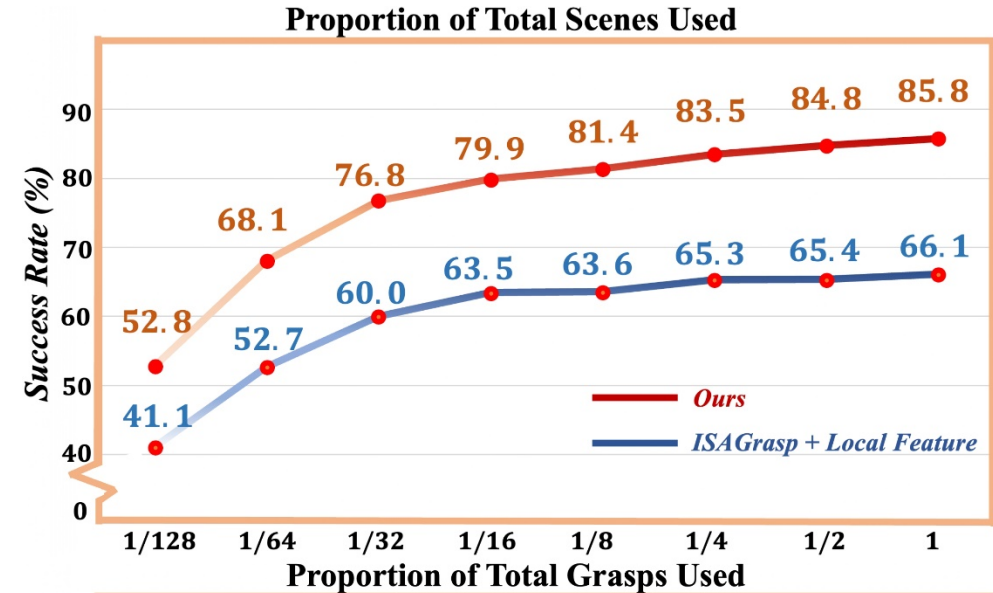
- Achieve SOTA on DexGraspNet 2.0 benchmark

	Method	GraspNet-1Billion			ShapeNet		
		Dense	Random	Loose	Dense	Random	Loose
Baseline	HGC-Net [8]	46.0	37.8	26.7	46.4	44.8	30.4
	GraspTTA [†] [11]	62.5	54.1	42.8	56.6	57.8	46.4
	ISAGrasp [†] [5]	63.4	60.7	51.4	64.0	56.3	52.7
Ablation	local feature	16.8	10.9	4.8	21.3	17.6	10.9
	decomposed model	84.2	80.7	71.3	74.9	72.5	66.4
	random scene	90.0	84.1	68.2	78.9	78.8	71.3
	Ours	90.6	83.7	73.2	81.0	85.4	74.2

Table 1: **Baseline and ablation studies for dexterous grasping.** Modified baseline methods are indicated with [†]. Ablation studies are conducted on three aspects as shown in the lower half of the table. Each **Dense** scene contains 8-11 objects, and each **Random** scene contains 1-10 objects, obtained by deleting objects from Dense scenes, and each **Loose** scene contains 1-2 objects.

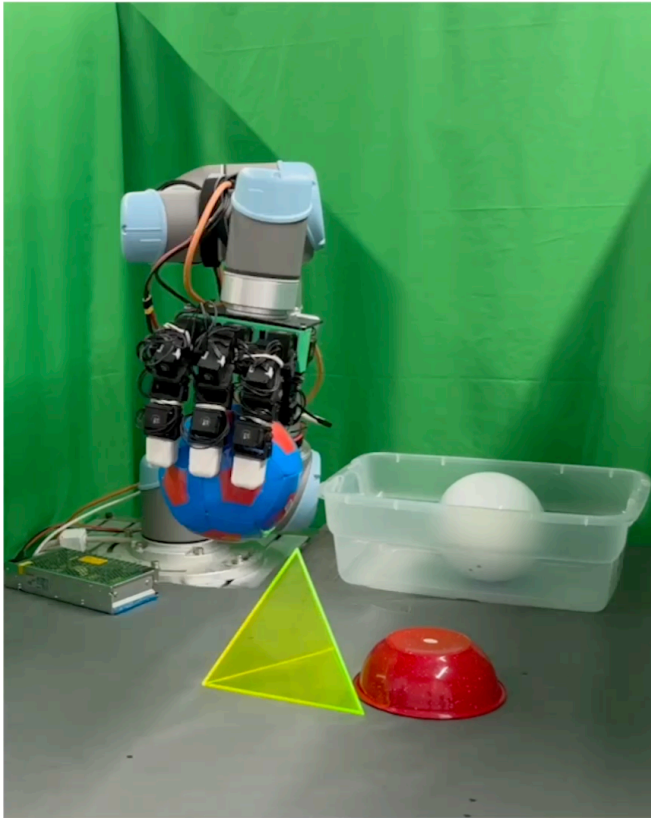
Scaling Law of Grasp Data

- We explore the impact of scaling the number of grasping poses and scenes
- Significant performance degradation with less poses/scenes



Real Experiments

- Compare to gripper
- The dexterous hand has stronger wrapping ability



End Effector	Normal	Large
Parallel Gripper	92.4	0.0
Dexterous Hand	81.5	100.0

Summary

➤ Main contribution:

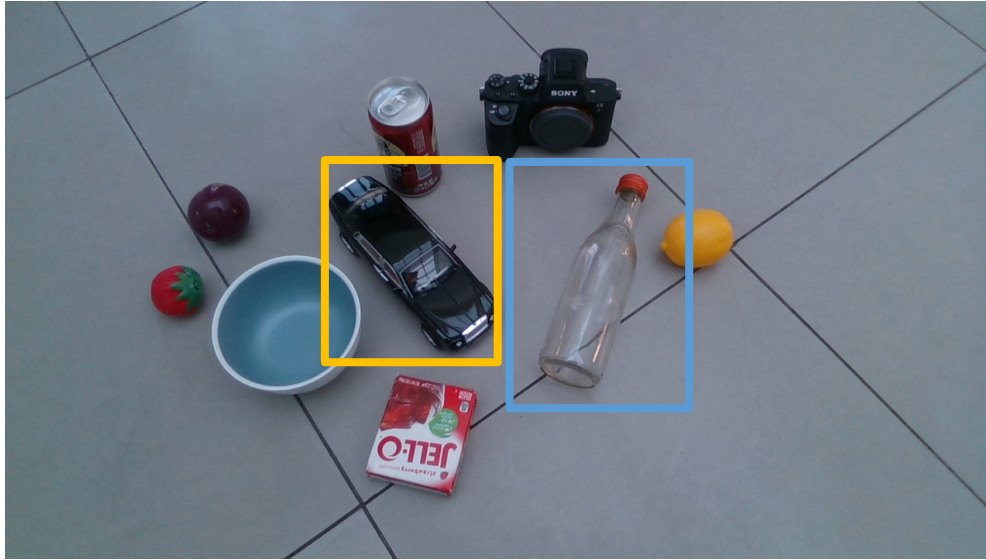
- **Large-scale Benchmark** for dexterous grasping in cluttered scenes
7500 synthetic training scenes and 670 test scenes.
- Propose an **end-to-end generative grasp prediction pipeline**
generates wrist 6D pose via diffusion model.
regress hand joint poses conditioned on wrist pose.

➤ Limitations:

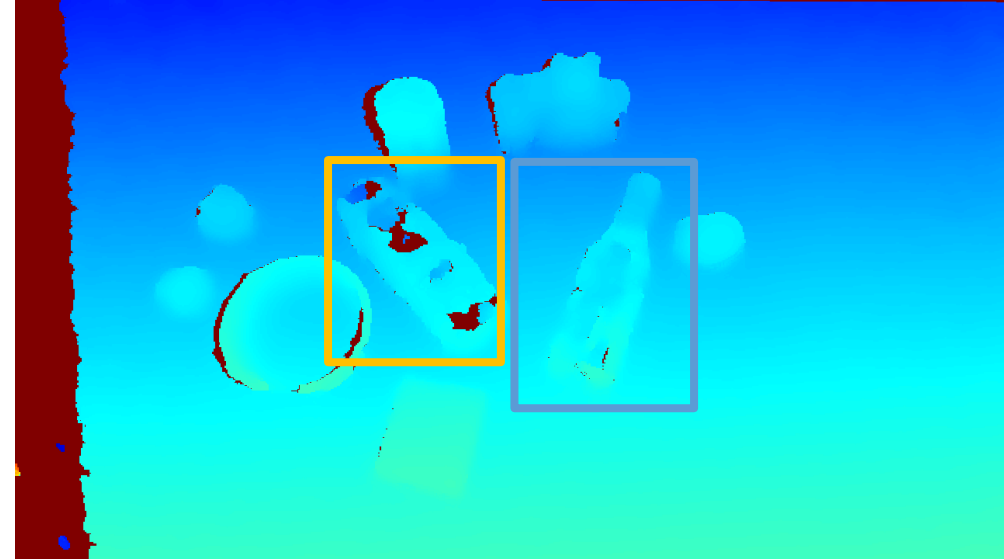
- **Limited to power grasping**
- **Data only covers force closure cases.**

Depth Sensing Problem Leading to Failure in Transparent Objects

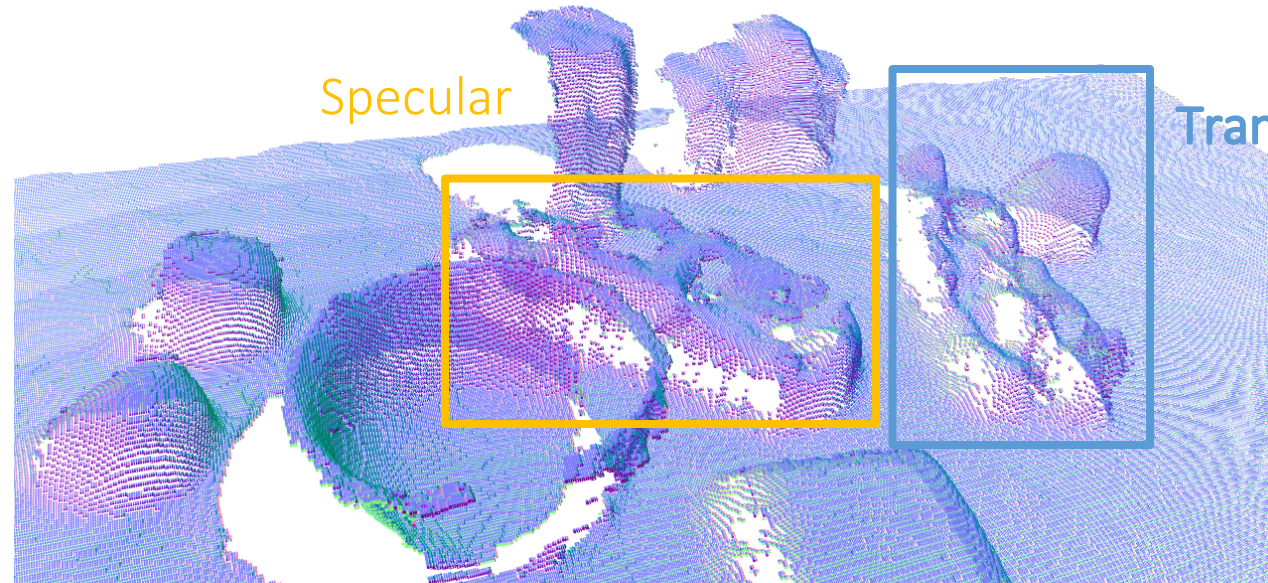
RGB



Depth



Point cloud



Specular

Transparent

Demo: Robotic Grasping

GraspNet-baseline
only





ICRA2024
YOKOHAMA | JAPAN

ASGrasp: Generalizable Transparent Object Reconstruction and 6-DoF Grasp Detection from RGB-D Active Stereo Camera

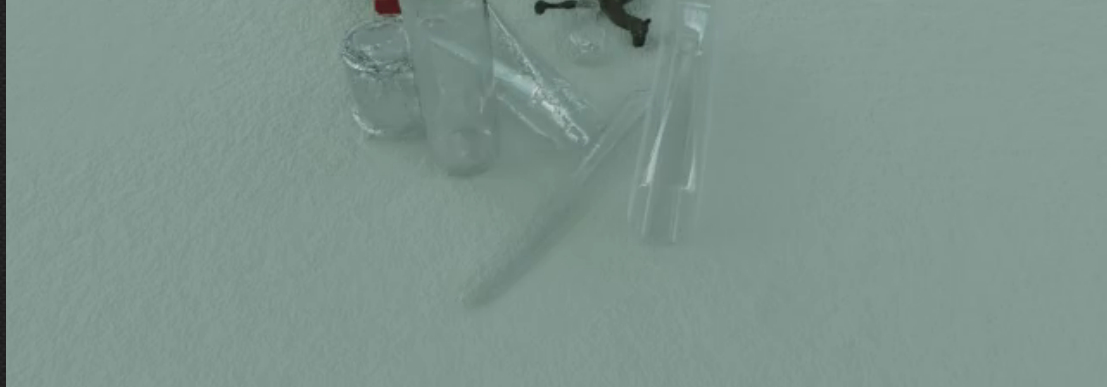
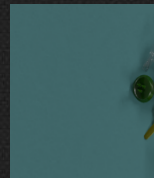
Jun Shi¹, Yong A¹, Yixiang Jin¹, Dingzhe Li¹, Haoyu Niu¹, Zhezhu Jin¹, He Wang^{†2,3,4}

¹ Samsung R&D Institute China, Beijing (SRC-B)

² CFCS, School of Computer Science, Peking University ³ Galbot

⁴ Beijing Academy of Artificial Intelligence (BAAI)

†: corresponding author



DREDS: Domain Randomization

- Randomization of object layout: object instances, arrangement



DREDS: Domain Randomization

- Randomization of object material: BSDF



DREDS: Domain Randomization

- Randomization of background



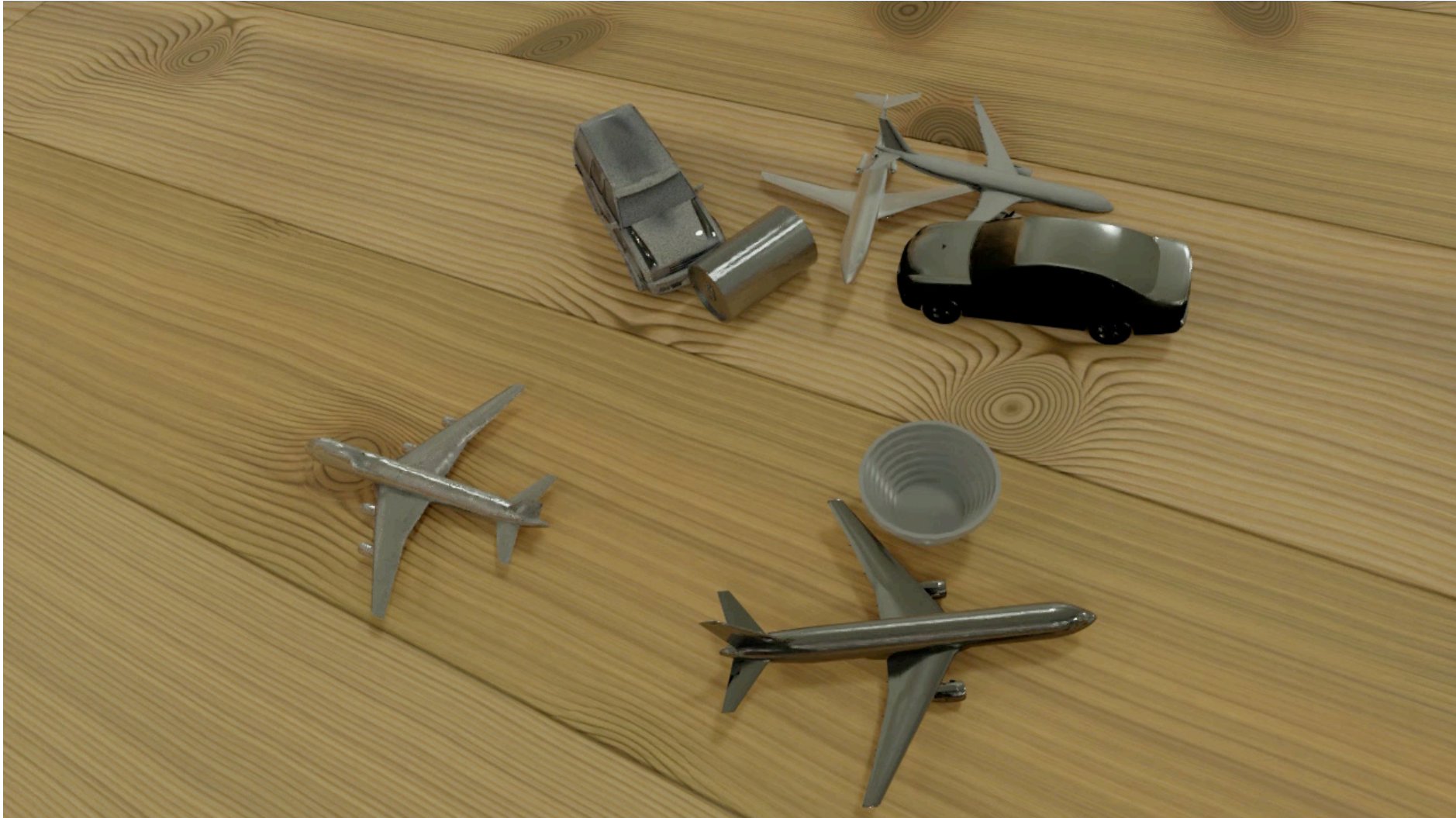
DREDS: Domain Randomization

- Randomization of illumination

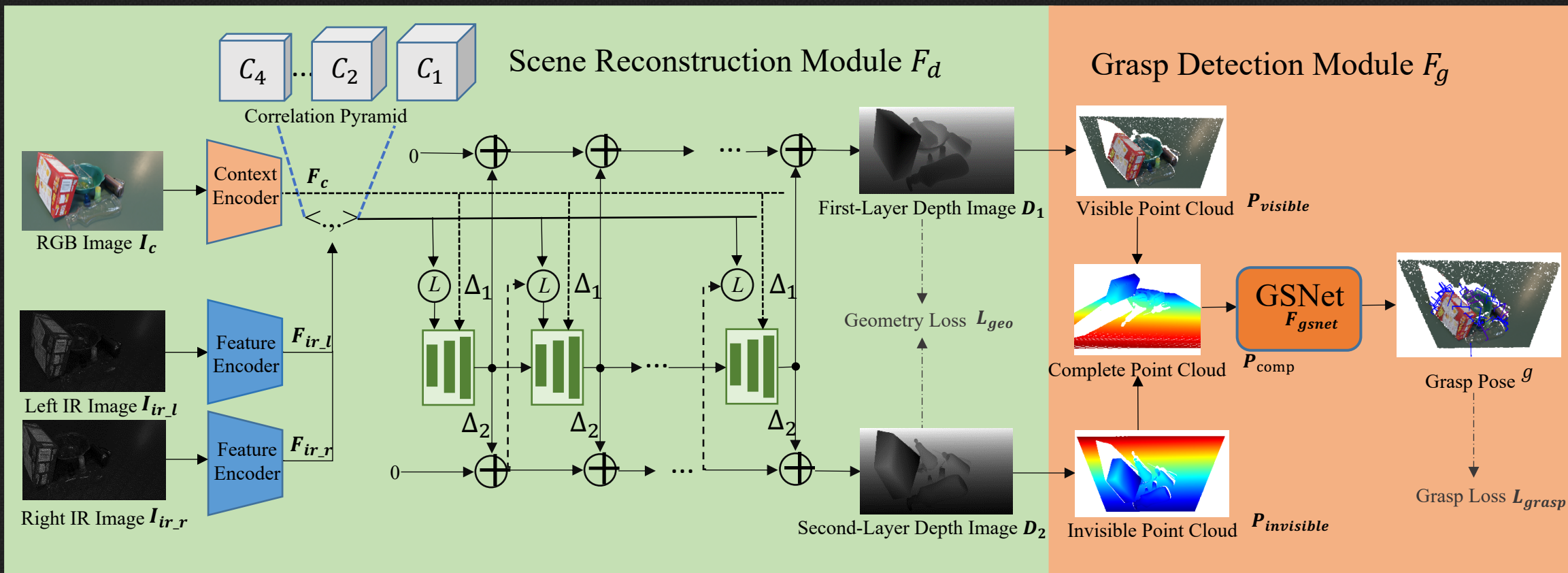


DREDS: Domain Randomization

- Randomization of camera viewpoint



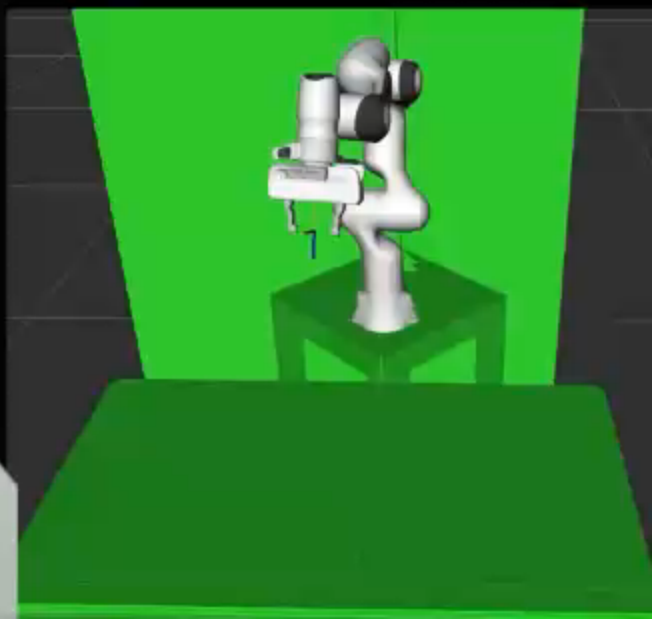
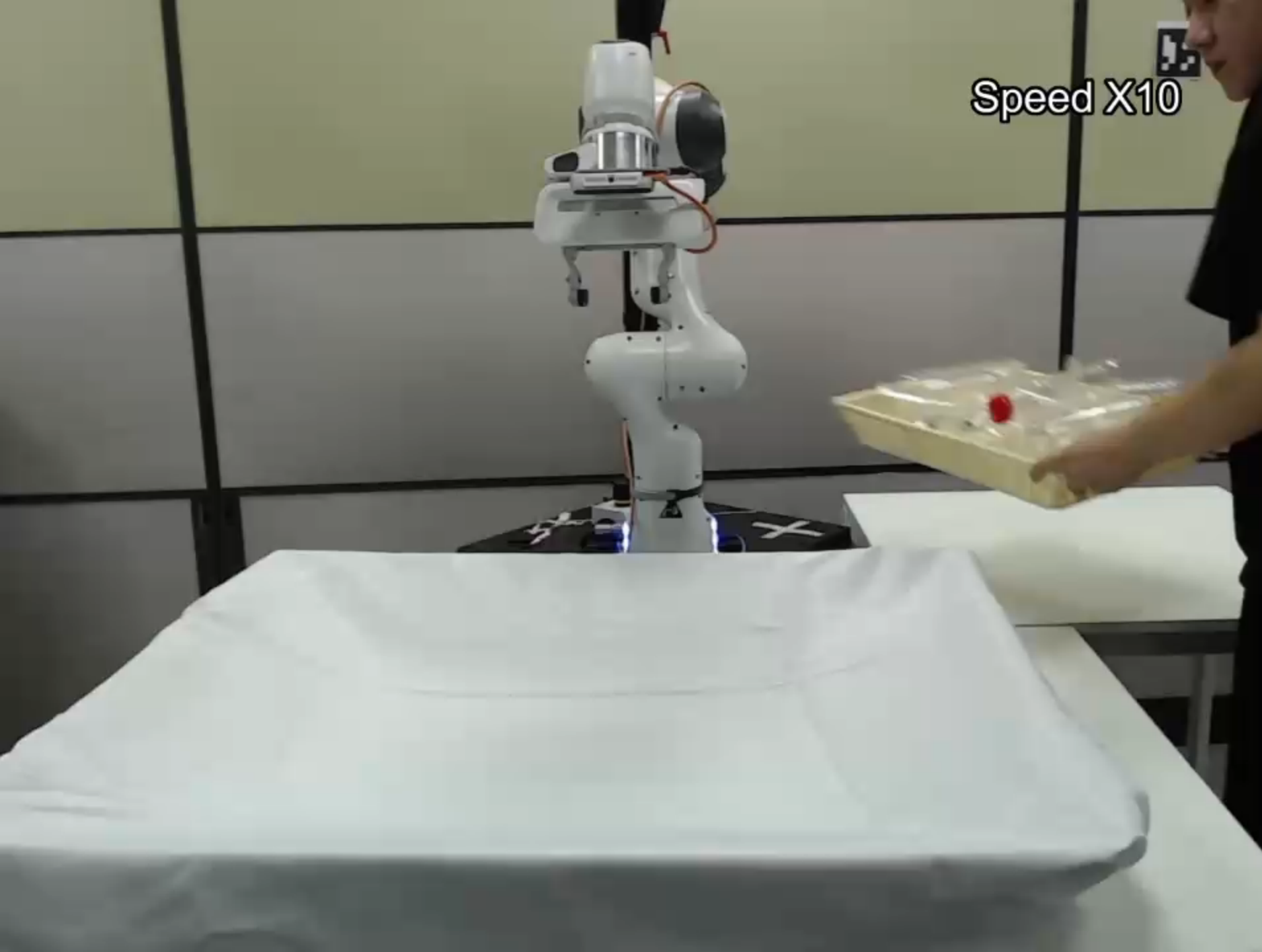
ASGrasp Pipeline



Speed X10

Grasp Poses

RViz Visualization





With a success rate > 95%



Summary of 3D Vision based Grasping

➤ Main contribution:

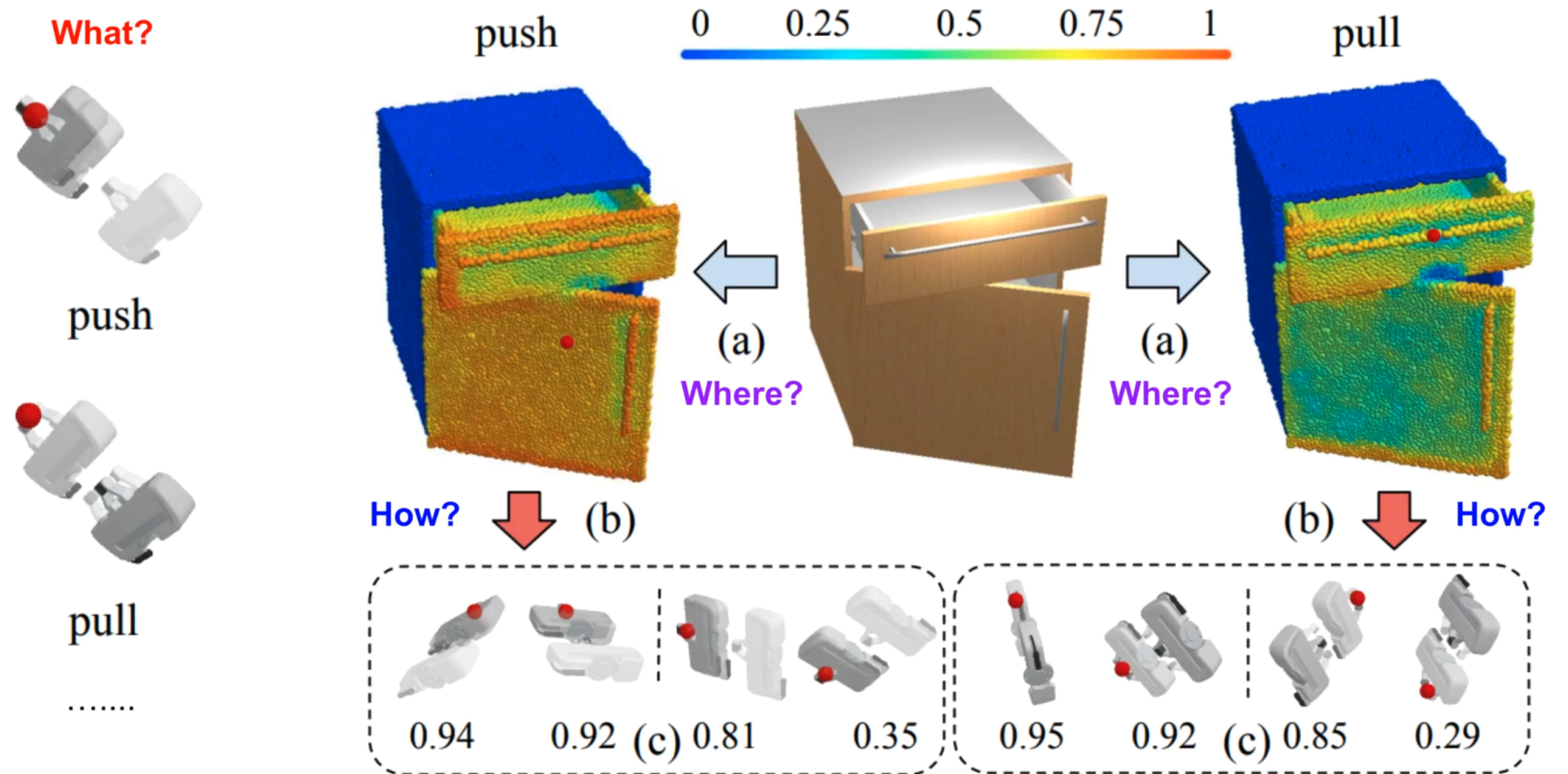
- **Large-scale Benchmark** for dexterous grasping in cluttered scenes
7500 synthetic training scenes and 670 test scenes.
- Propose an **end-to-end generative grasp prediction pipeline**
generates wrist 6D pose via diffusion model.
regress hand joint poses conditioned on wrist pose.

➤ Limitations:

- **Limited to power grasping**
- **Data only covers force closure cases.**

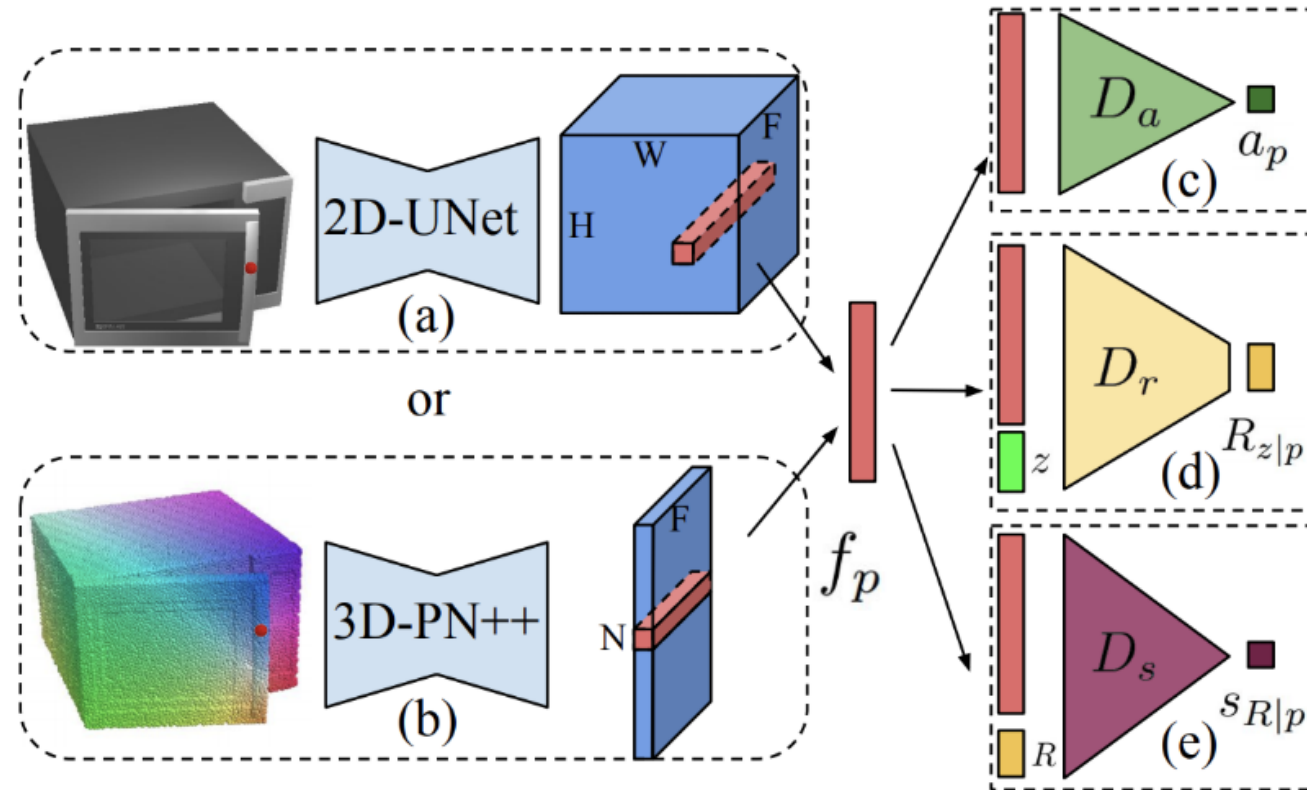
Towards Manipulation: Affordance

- Where2Act



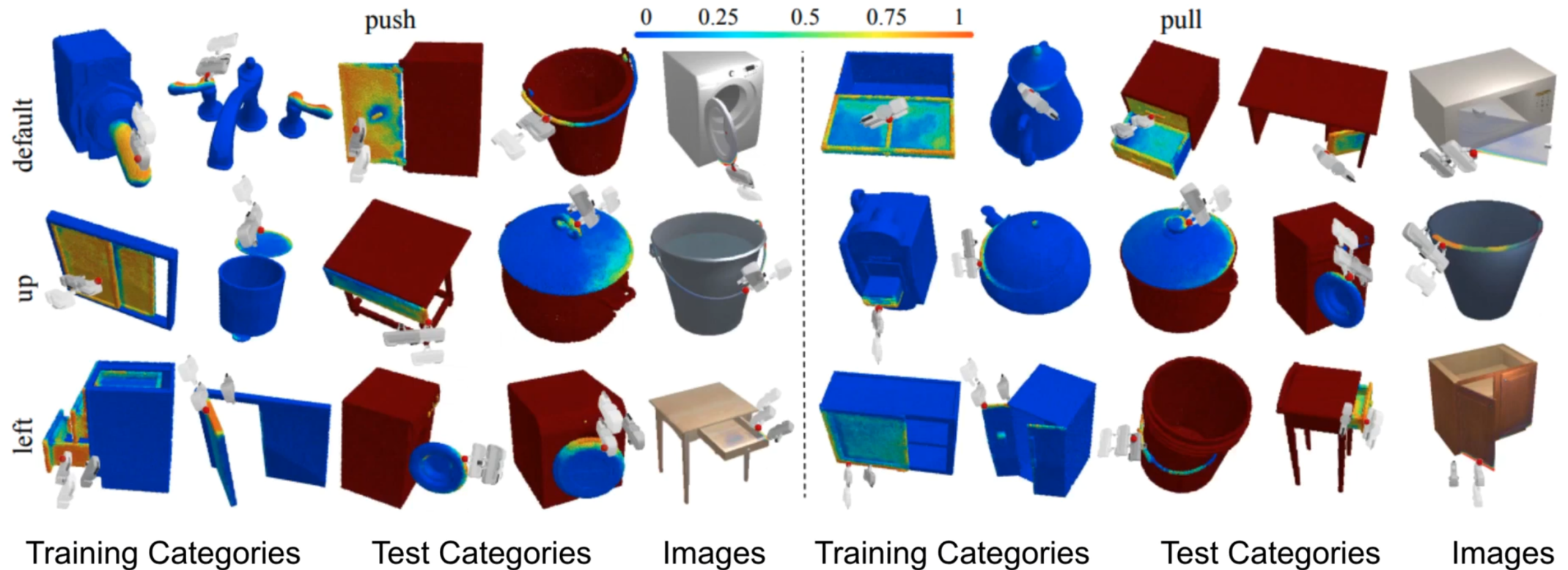
Another Vision-based Approach: Affordance

- Where2Act Pipeline



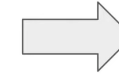
Another Vision-based Approach: Affordance

- Where2Act Result



Another Vision-based Approach: Affordance

- VAT-Mart



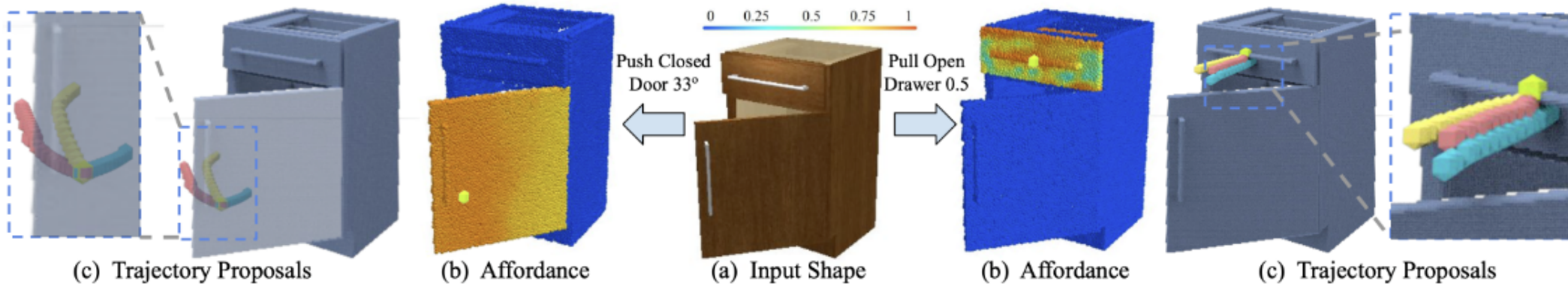
Perception:

- Detect Door Handle
- Propose Grasp Candidates
- Estimate Door Joint Parameters

Manipulation:

- Grasp the Handle
- Drive to follow a Trajectory

Open Door 30°



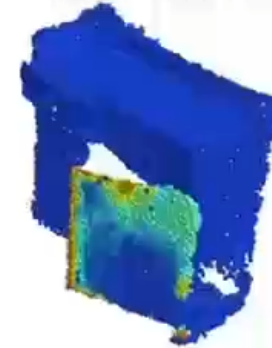
Another Vision-based Approach: Affordance

Real Robot Experiments

Task: Open Door 30°
Contact: Edge at top



Affordance



Trajectory Proposal



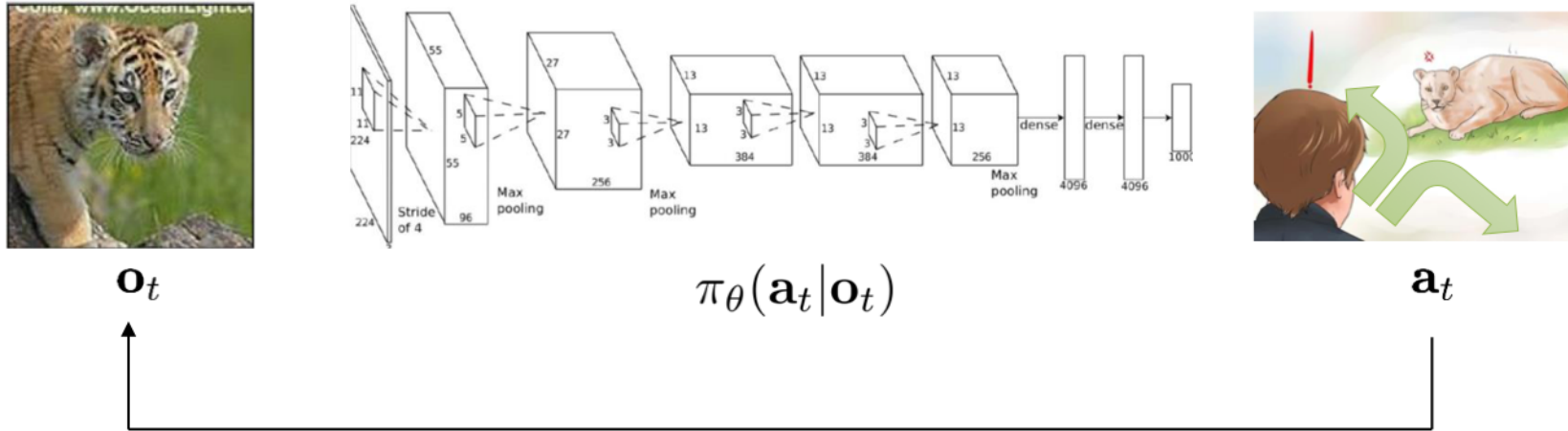
Summary of Vision-based Open-Looped Approaches

- Use vision inputs to predict
 - object pose (CAD model needed, grasp annotation needed)
 - or grasp pose (no CAD model and grasp anno.)
 - or affordance (slightly going beyond grasping)
- Motion planning to reach, heuristic to grasp/manipulate
- Limited to certain predefined manipulation (bottleneck is the heuristics)
- Usually not closed-loop, but can be closed-loop

Policy Learning

Some slides are borrowed and modified from Berkeley CS285.

Terminology & Notation



$\mathbf{s}_t$ – state

$\mathbf{o}_t$ – observation

$\mathbf{a}_t$ – action

$\pi_{\theta}(\mathbf{a}_t | \mathbf{o}_t)$ – policy

$\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)$ – policy (fully observed)

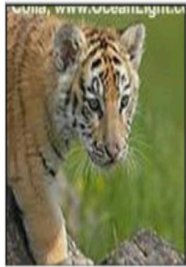


$\mathbf{o}_t$ – observation

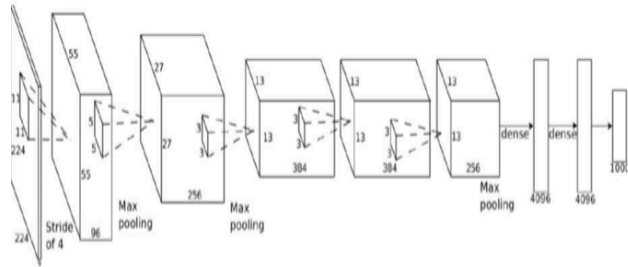


$\mathbf{s}_t$ – state

Policy Learning



o_t



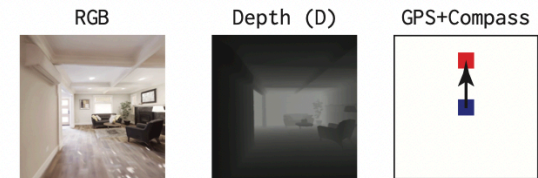
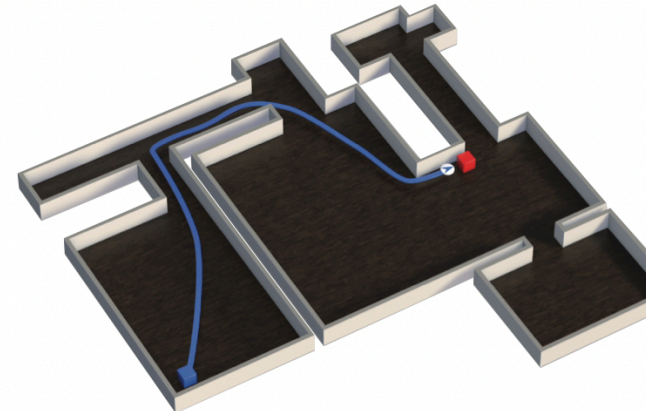
$\pi_{\theta}(a_t|o_t)$



a_t

How to learn the policy?

Supervised policy learning

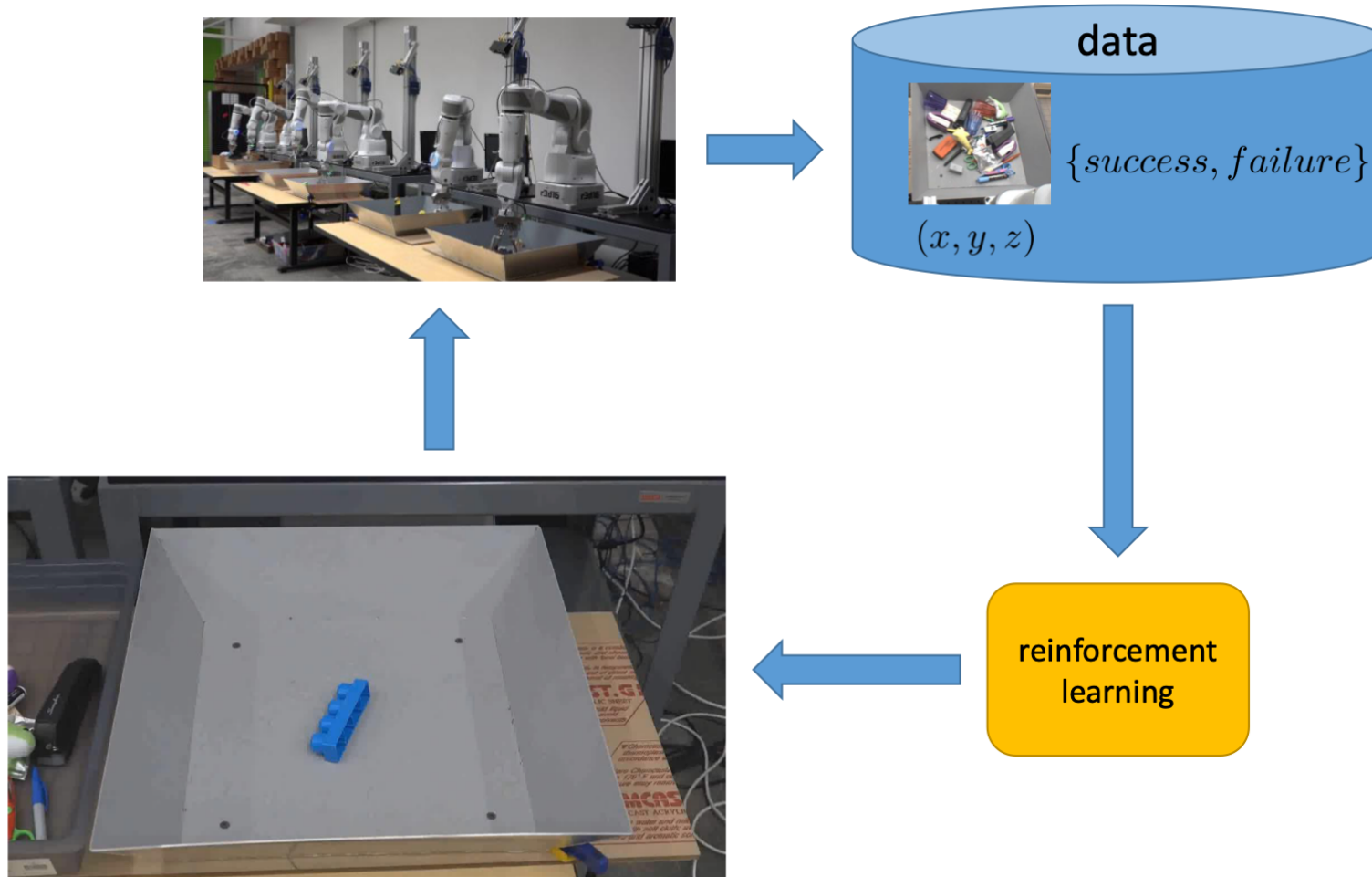


PointGoal Navigation

When an optimal policy or expert demonstrations are available, one could consider supervised policy learning, e.g. imitation learning (IL).

Reinforcement Learning

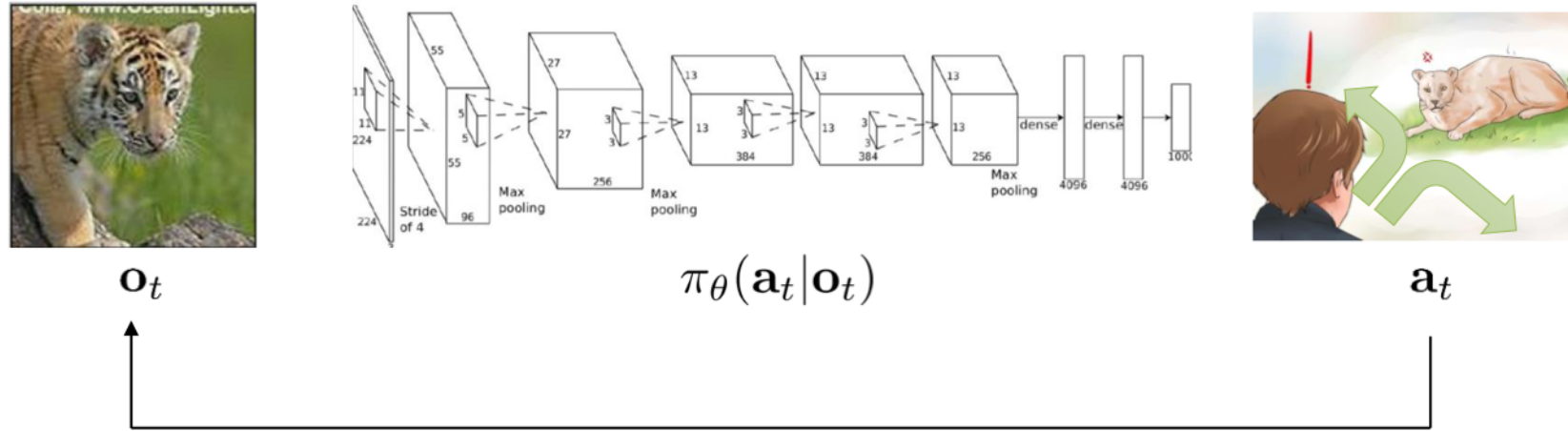
When an optimal policy is not available, trial and error!



Imitation Learning

Some slides are borrowed from Berkeley CS285.

Terminology & notation



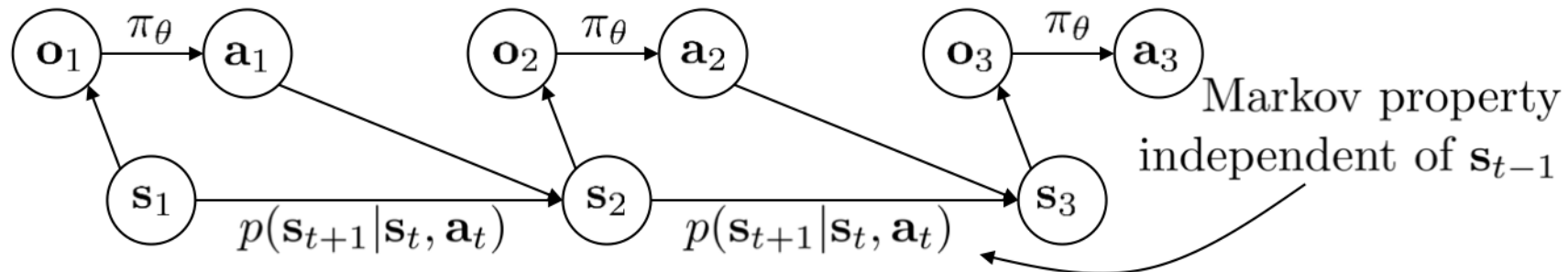
$\mathbf{s}_t$ – state

$\mathbf{o}_t$ – observation

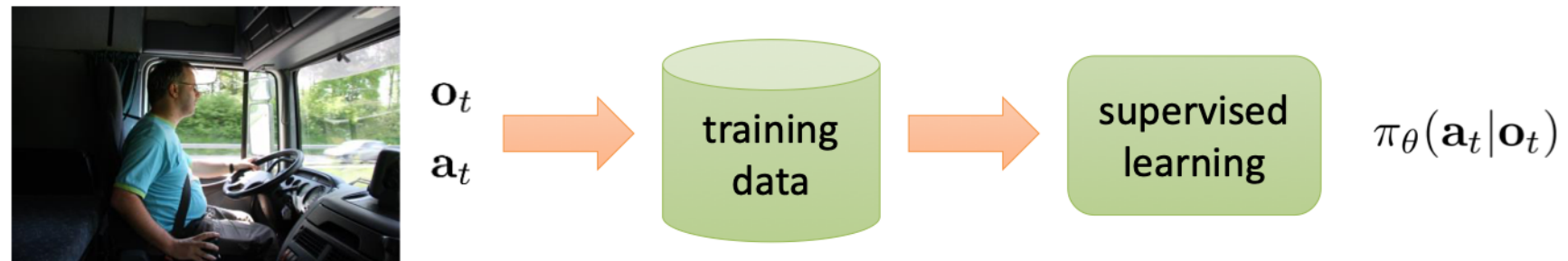
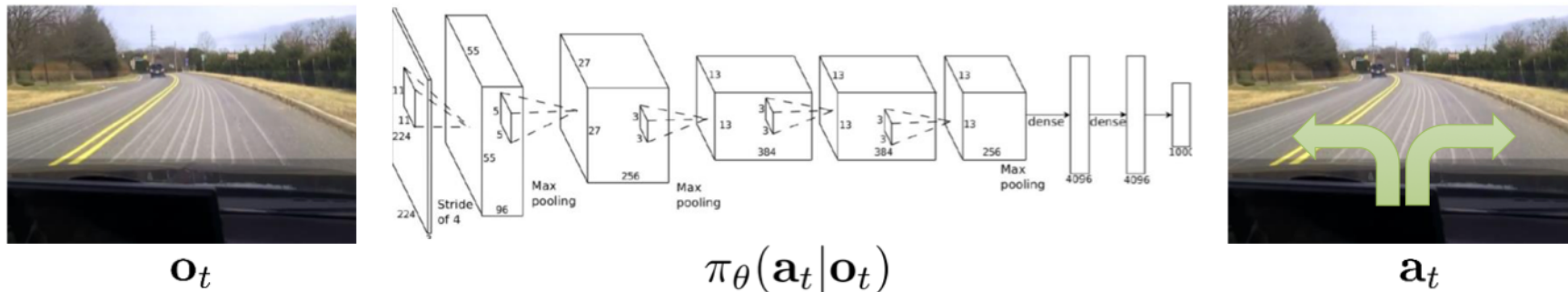
$\mathbf{a}_t$ – action

$\pi_{\theta}(\mathbf{a}_t | \mathbf{o}_t)$ – policy

$\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)$ – policy (fully observed)



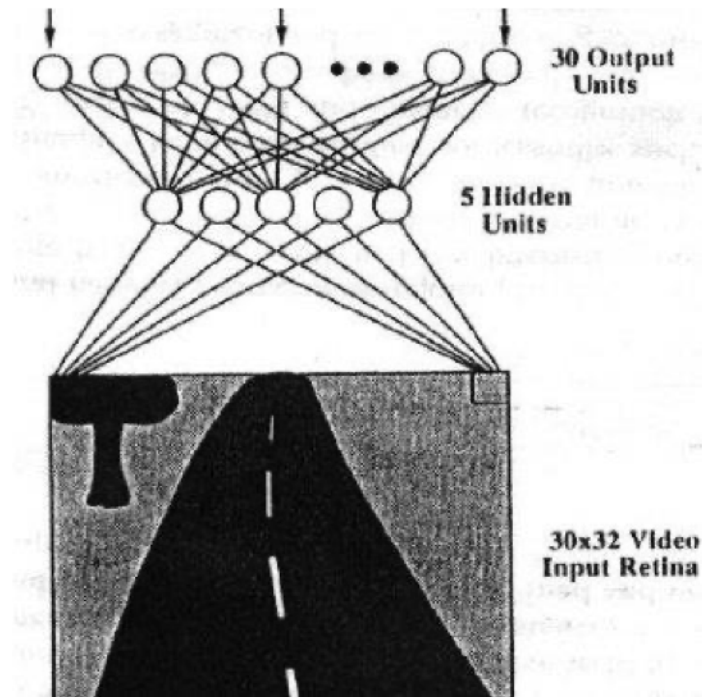
Imitation Learning



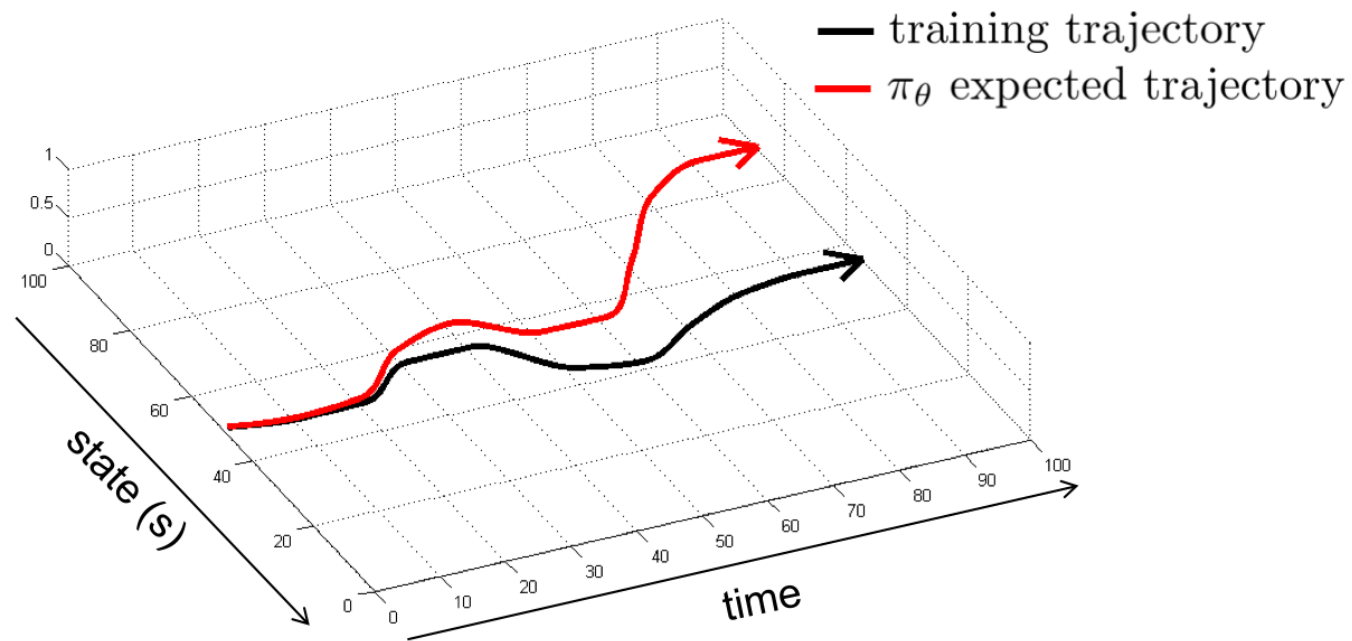
behavioral cloning

The Original Deep Imitation Learning System

ALVINN: Autonomous Land Vehicle In a Neural Network
1989

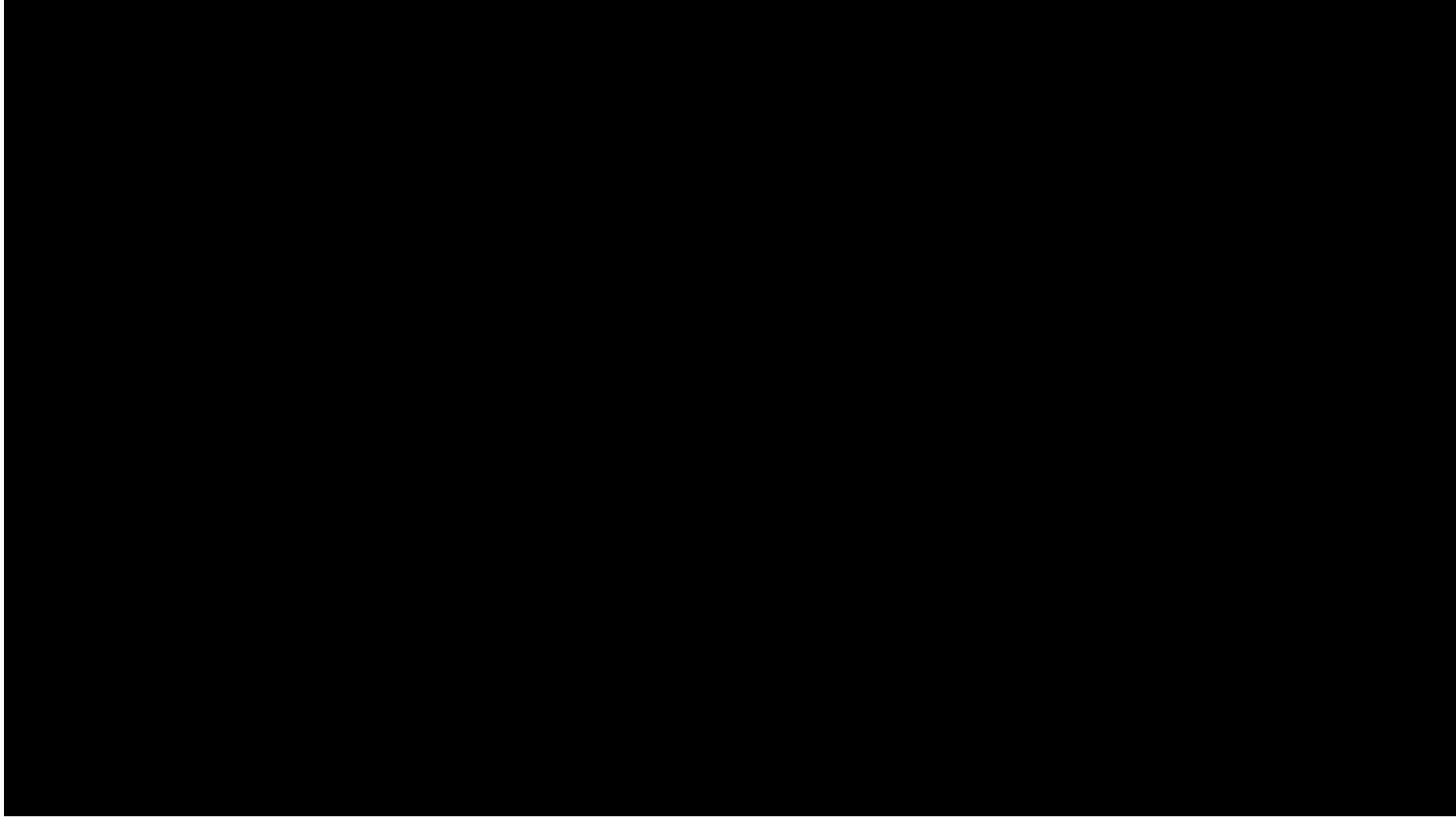


Does it work?



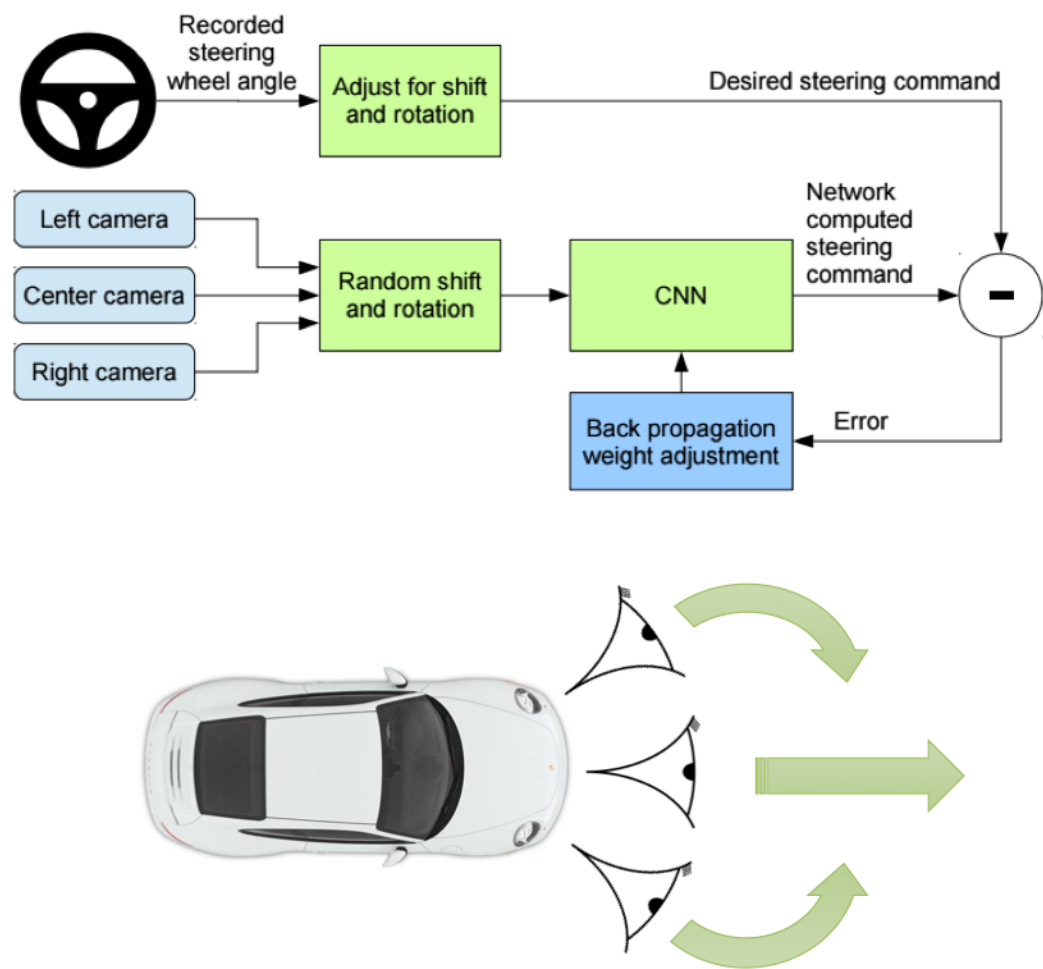
No, because of distribution drift!

Does it work?



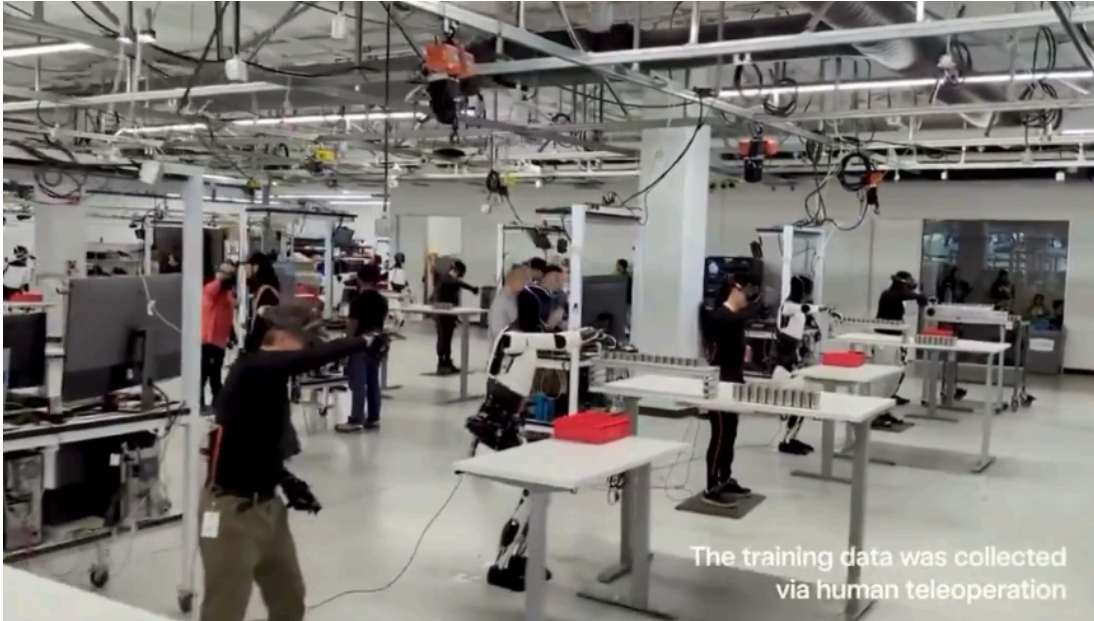
Video: Bojarski et al. '16, NVIDIA

Why did that work?



Robot Demonstration Collection: Teleoperation

Teleoperation System used by Tesla



ALOHA system from Stanford



Abramson, Norman. "The ALOHA system: Another alternative for computer communications." *Proceedings of the November 17-19, 1970, fall joint computer conference*. 1970.

Tesla Optimus: Battery Sorting and Correction Demo Collection

特斯拉人形机器人进厂了
分拣4680电池还能自我纠错
行走速度提高30%
马斯克：今年晚些时候
手臂工作方式会像人类一样



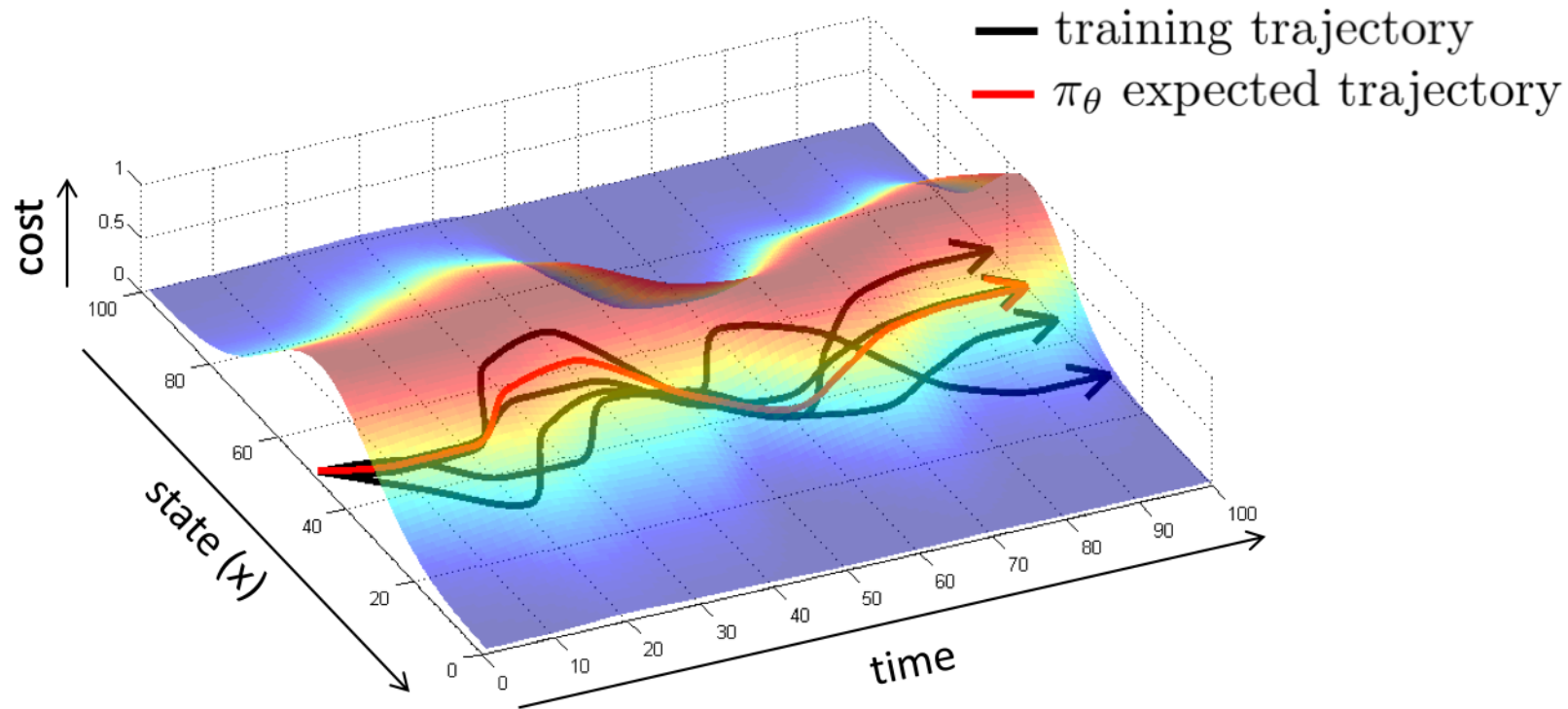
5月5日 特斯拉Optimus（擎天柱）
人形机器人发布最新视频
利用端到端的神经网络
已在工厂进行4680电池分拣等工作

澎湃
新闻

Human-in-the-Loop Imitation Learning using Remote Teleoperation



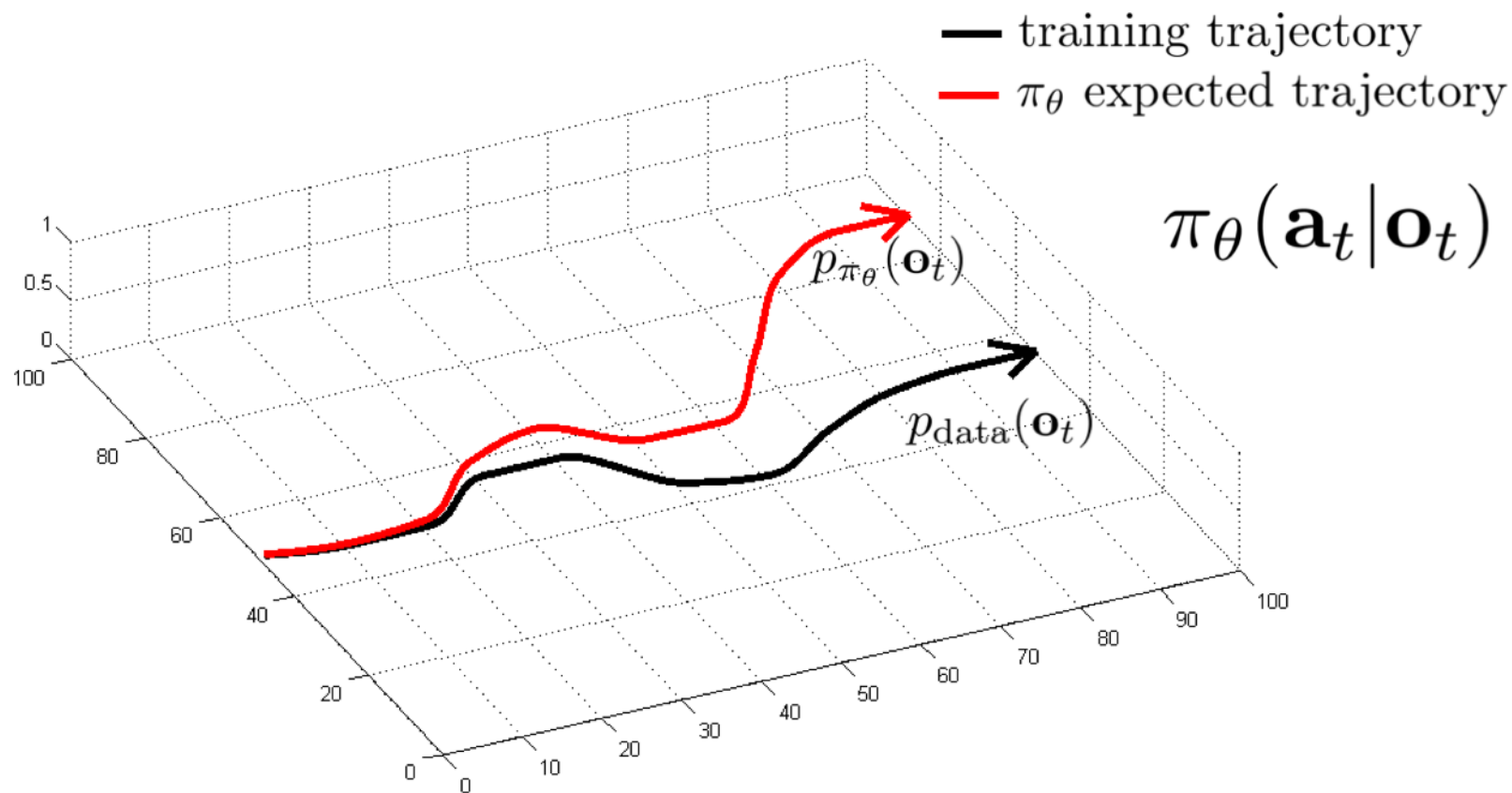
Can we make it work more often?



stability

(more on this later)

Can we make it work more often?



can we make $p_{\text{data}}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$?

Can we make it work more often?

can we make $p_{\text{data}}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$?

idea: instead of being clever about $p_{\pi_\theta}(\mathbf{o}_t)$, be clever about $p_{\text{data}}(\mathbf{o}_t)$!

DAgger: Dataset Aggregation

goal: collect training data from $p_{\pi_\theta}(\mathbf{o}_t)$ instead of $p_{\text{data}}(\mathbf{o}_t)$

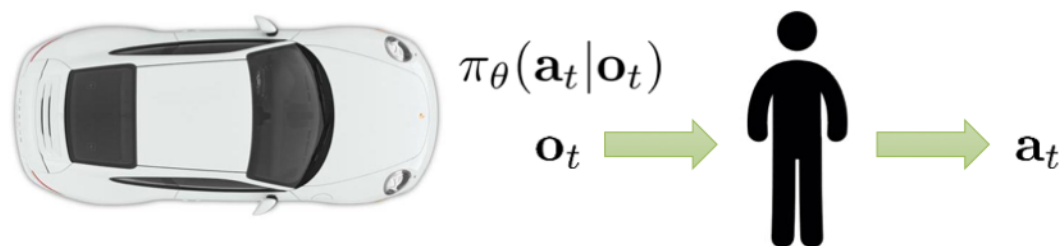
how? just run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

but need labels $\mathbf{a}_t$!

- 
1. train $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \dots, \mathbf{o}_N, \mathbf{a}_N\}$
 2. run $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
 3. Ask human to label $\mathcal{D}_\pi$ with actions $\mathbf{a}_t$
 4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

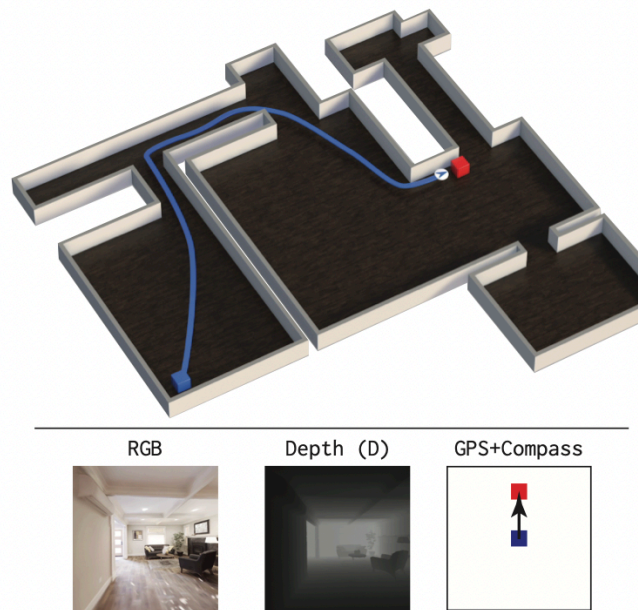
What's the problem?

1. train $\pi_{\theta}(\mathbf{a}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{a}_1, \dots, \mathbf{o}_N, \mathbf{a}_N\}$
2. run $\pi_{\theta}(\mathbf{a}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_{\pi} = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
3. Ask human to label $\mathcal{D}_{\pi}$ with actions $\mathbf{a}_t$
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{\pi}$



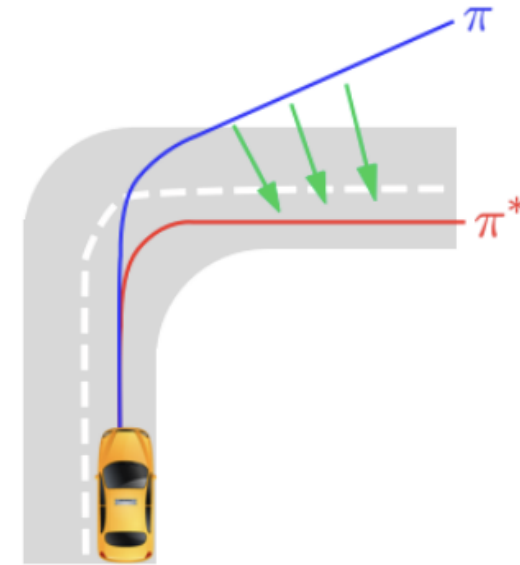
Other Sources of Action Labels

1. From optimal solution



Directly compute the shortest path and get action label

2. From a teacher policy

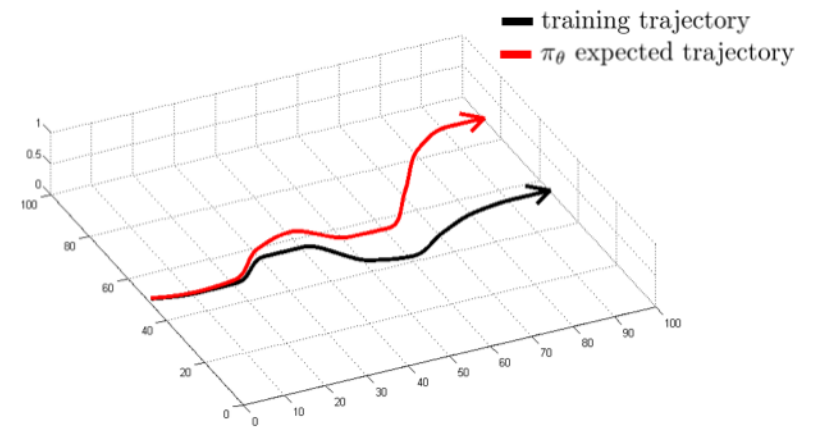


Roll out the student policy (blue), and let the teacher policy (red) correct it online

Can we make it work without more data?

Can we make it work without more data?

- DAgger addresses the problem of distributional “drift”
- What if our model is so good that it doesn’t drift?
- Need to mimic expert behavior very accurately
- But don’t overfit!



Why might we fail to fit the expert?

- 
1. Non-Markovian behavior
 2. Multimodal behavior

$$\pi_{\theta}(\mathbf{a}_t | \mathbf{o}_t)$$

behavior depends only
on current observation

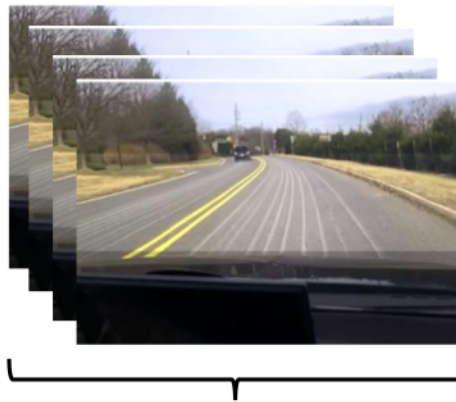
$$\pi_{\theta}(\mathbf{a}_t | \mathbf{o}_1, \dots, \mathbf{o}_t)$$

behavior depends on
all past observations

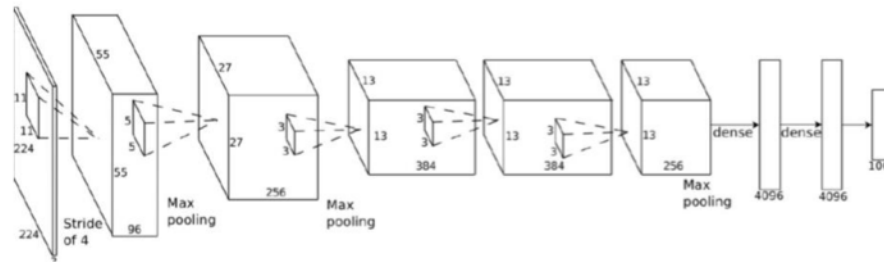
If we see the same thing
twice, we do the same thing
twice, regardless of what
happened before

Often very unnatural for
human demonstrators

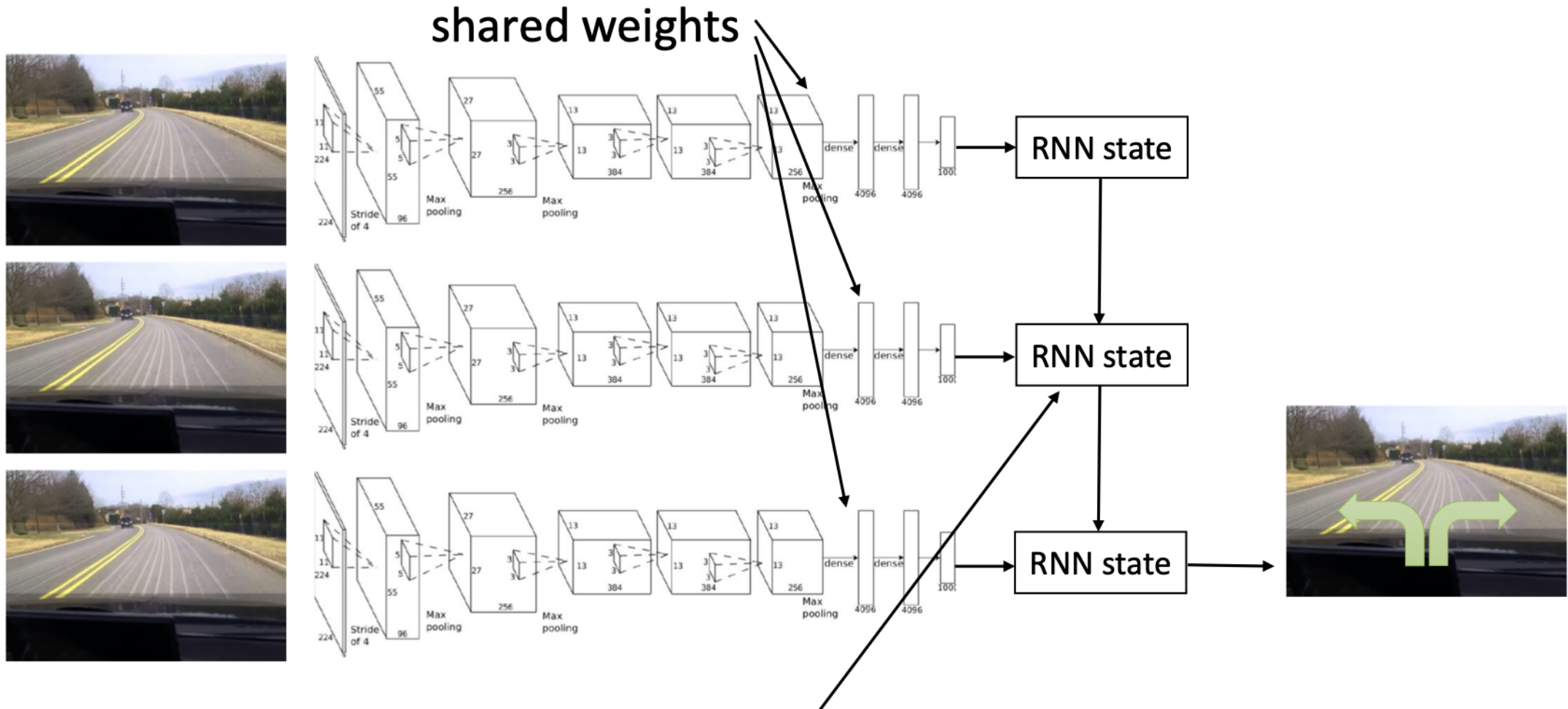
How can we use the whole history?



variable number of frames,
too many weights

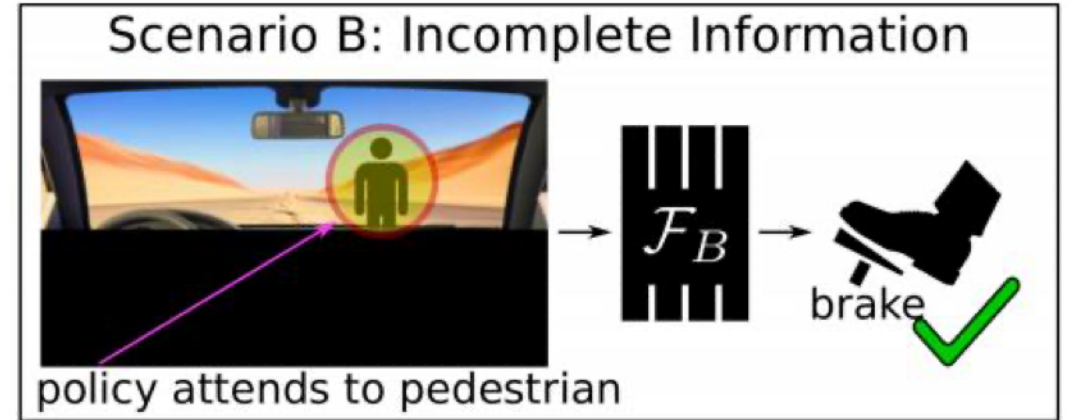
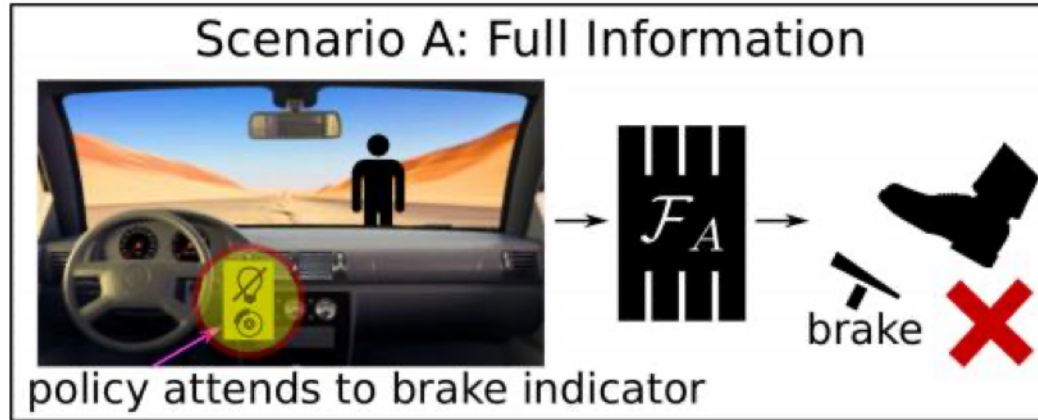


How can we use the whole history?



Typically, LSTM cells work better here

Aside: why might this work poorly?



“causal confusion”

see: de Haan et al., “Causal Confusion in Imitation Learning”

Question 1: Does including history mitigate causal confusion?

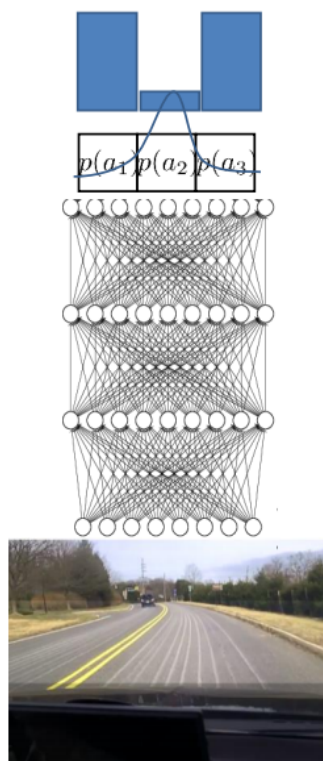
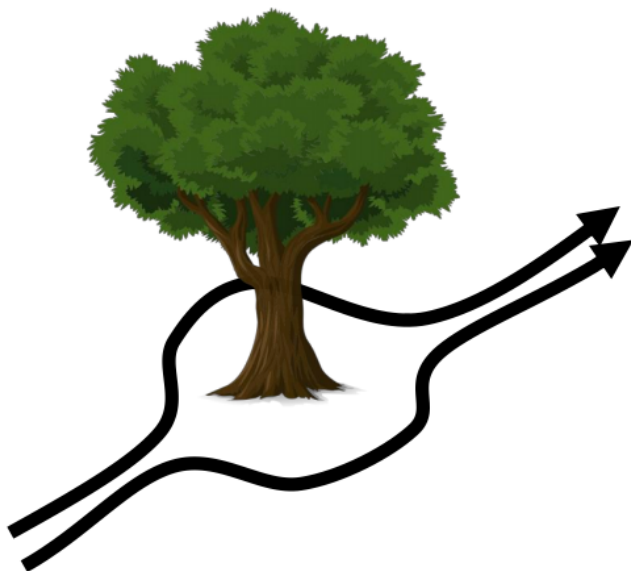
Question 2: Can DAgger mitigate causal confusion?

Why might we fail to fit the expert?

1. Non-Markovian behavior



2. Multimodal behavior



1. Output mixture of Gaussians

2. Latent variable models

3. Diffusion models

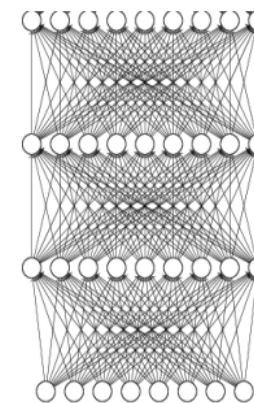
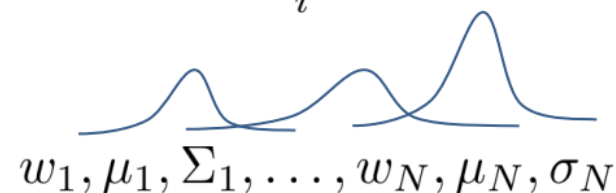
4. Autoregressive discretization



Why might we fail to fit the expert?

- ➡ 1. Output mixture of Gaussians
- 2. Latent variable models
- 3. Diffusion models
- 4. Autoregressive discretization

$$\pi(\mathbf{a}|\mathbf{o}) = \sum_i w_i \mathcal{N}(\mu_i, \Sigma_i)$$

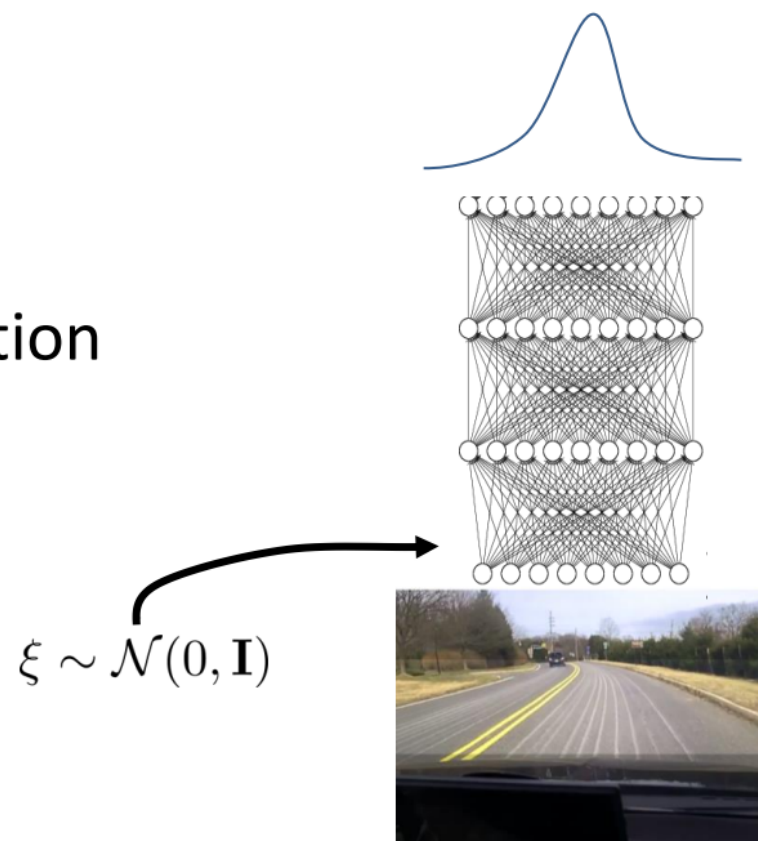


Why might we fail to fit the expert?

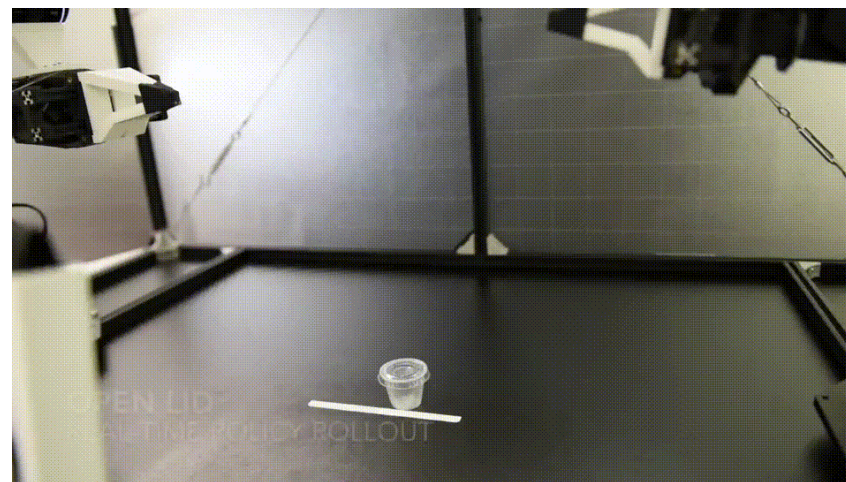
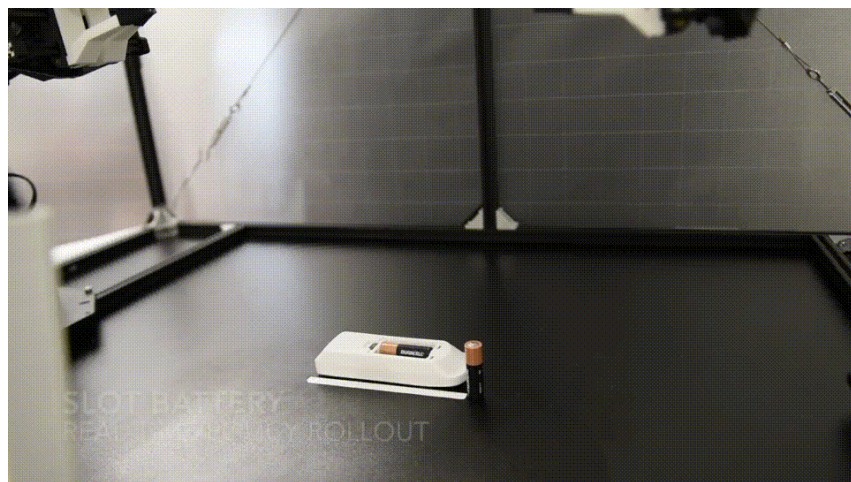
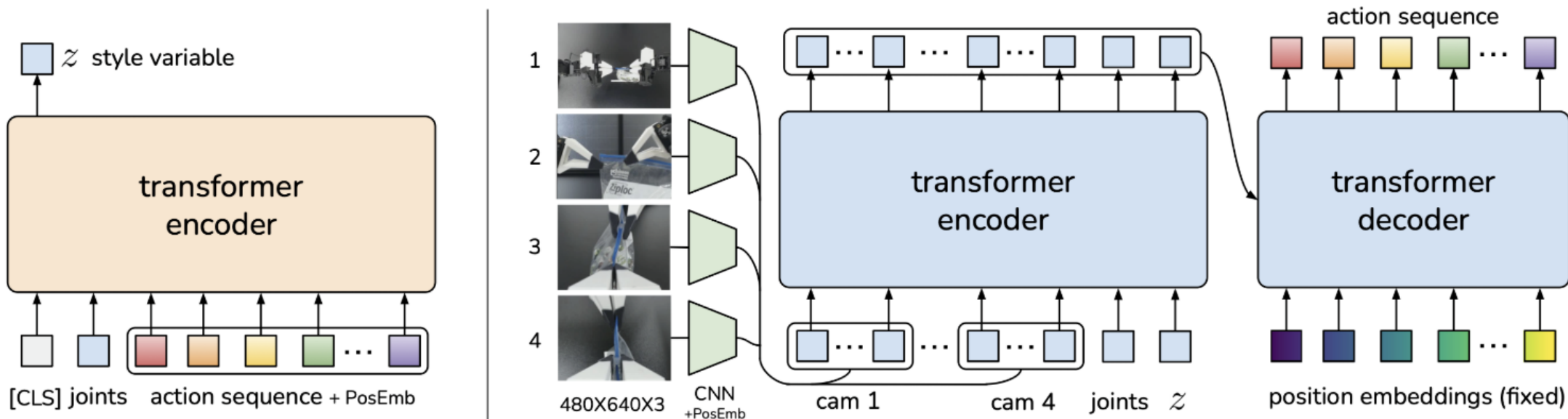
1. Output mixture of Gaussians
- ➡ 2. Latent variable models
3. Diffusion models
4. Autoregressive discretization

Look up some of these:

- Conditional variational autoencoder
- Normalizing flow/realNVP
- Stein variational gradient descent



Case study: imitation with CVAE



Why might we fail to fit the expert?

1. Output mixture of Gaussians
2. Latent variable models
- ➔ 3. Diffusion models
4. Autoregressive discretization

$\mathbf{x}_0 = \text{true image}$

$\mathbf{x}_{i+1} = \mathbf{x}_i + \text{noise}$

Learned network: $f(\mathbf{x}_i) = \mathbf{x}_{i-1}$

(actually use $f(\mathbf{x}_i) = \text{noise}$)

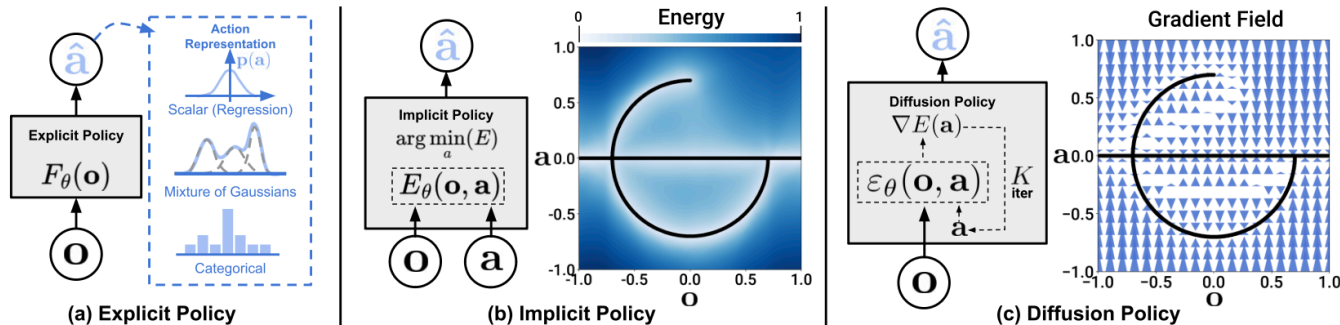
$\mathbf{x}_{i-1} = \mathbf{x}_i - f(\mathbf{x}_i)$



Diffusion based Policy

Diffusion Policy

Visuomotor Policy Learning via Action Diffusion



This paper introduces Diffusion Policy, a new way of generating robot behavior by representing a robot's visuomotor policy as a conditional denoising diffusion process. We benchmark Diffusion Policy across 12 different tasks from 4 different robot manipulation benchmarks and find that it consistently outperforms existing state-of-the-art robot learning methods with an average improvement of 46.9%. Diffusion Policy learns the gradient of the action-distribution score function and iteratively optimizes with respect to this gradient field during inference via a series of stochastic Langevin dynamics steps. We find that the diffusion formulation yields powerful advantages when used for robot policies, including gracefully handling multimodal action distributions, being suitable for high-dimensional action spaces, and exhibiting impressive training stability. To fully unlock the potential of diffusion models for visuomotor policy learning on physical robots, this paper presents a set of key technical contributions including the incorporation of receding horizon control, visual conditioning, and the time-series diffusion transformer. We hope this work will help motivate a new generation of policy learning techniques that are able to leverage the powerful generative modeling capabilities of diffusion models.

<https://diffusion-policy.cs.columbia.edu/>

Why might we fail to fit the expert?

1. Output mixture of Gaussians
2. Latent variable models
- ➔ 3. Diffusion models
4. Autoregressive discretization

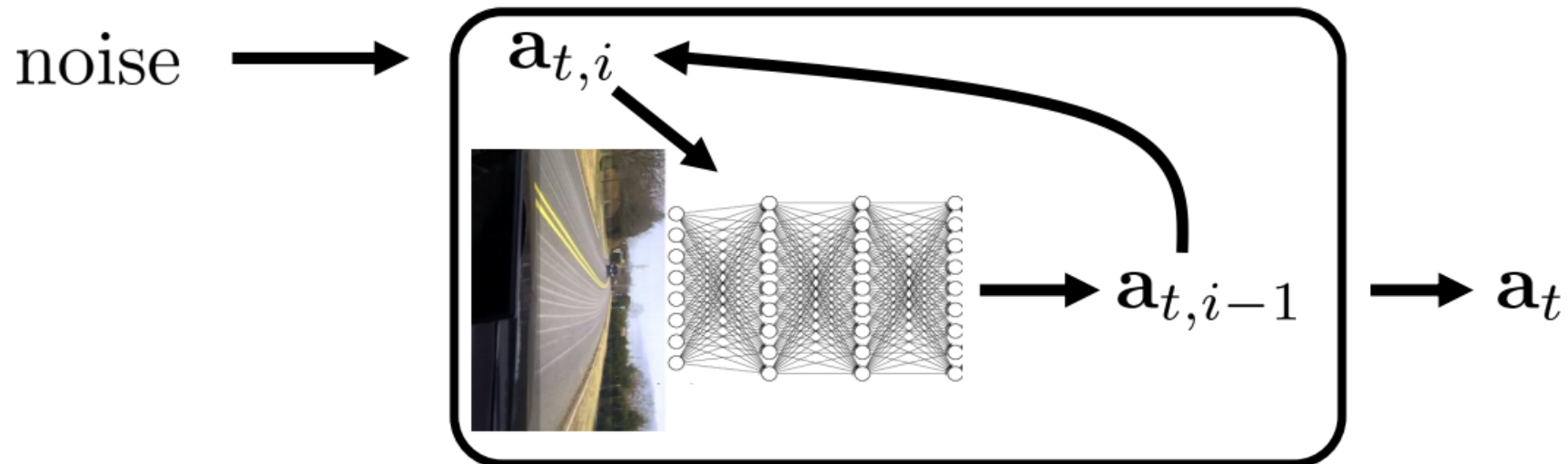
$\mathbf{x}_0 = \text{true image}$

$\mathbf{x}_{i+1} = \mathbf{x}_i + \text{noise}$

Learned network: $f(\mathbf{x}_i) = \mathbf{x}_{i-1}$

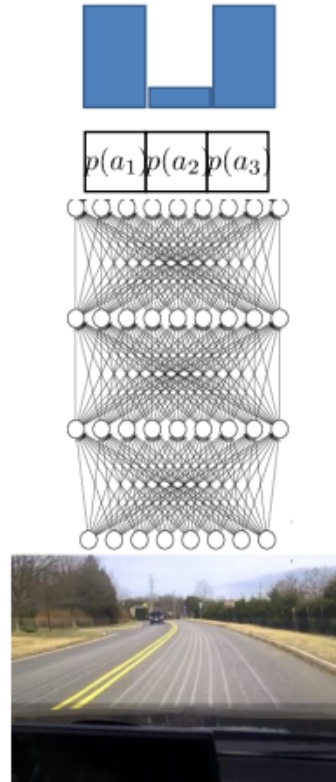
(actually use $f(\mathbf{x}_i) = \text{noise}$)

$\mathbf{x}_{i-1} = \mathbf{x}_i - f(\mathbf{x}_i)$



Why might we fail to fit the expert?

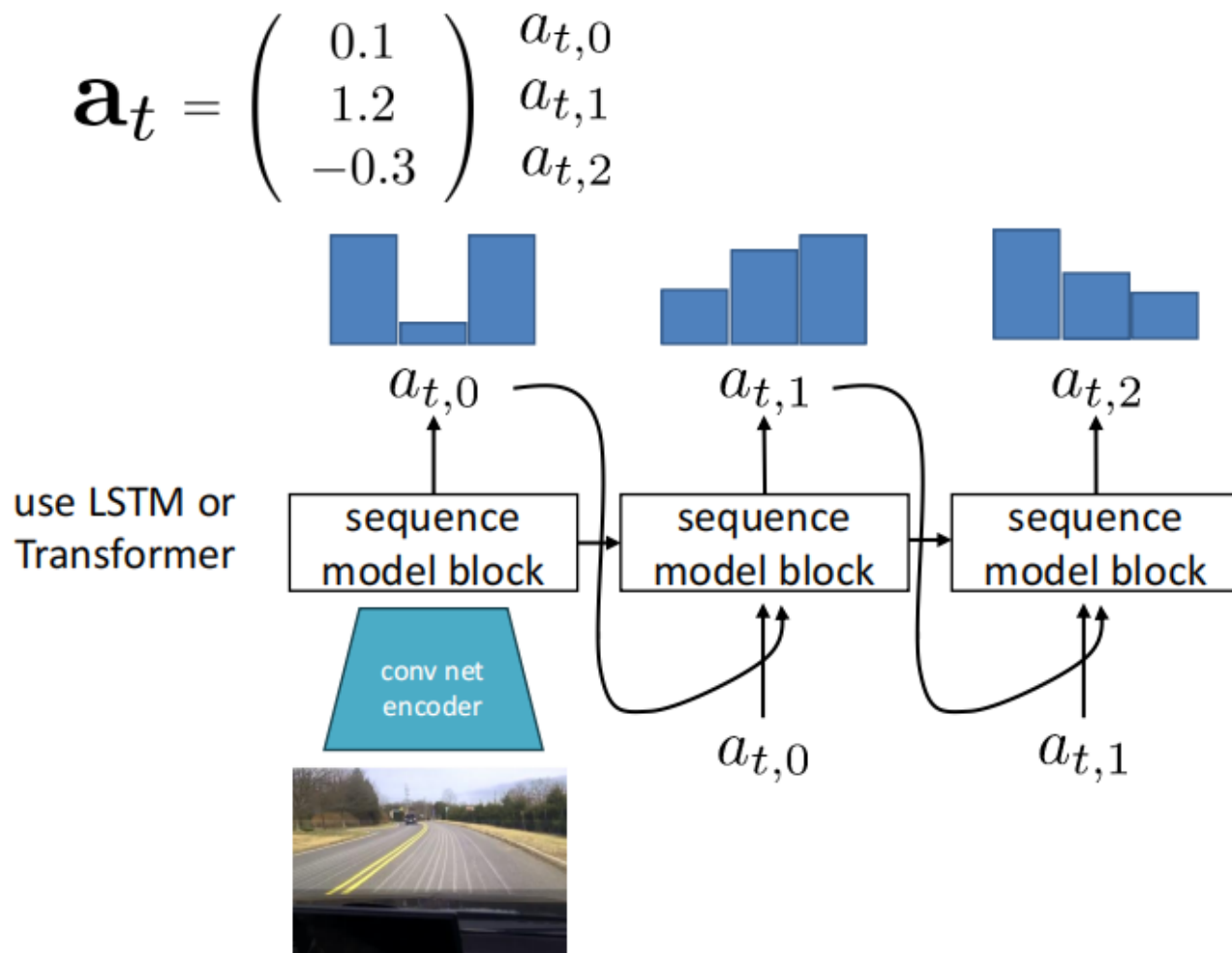
1. Output mixture of Gaussians
2. Latent variable models
3. Diffusion models
- ➔ 4. Autoregressive discretization



Problem: this is great for 1D actions, but in higher dimensions, discretizing the full space is impractical

Solution: discretize one dimension at a time

Autoregressive discretization



Why does this work?

first step: $p(a_{t,0}|\mathbf{s}_t)$

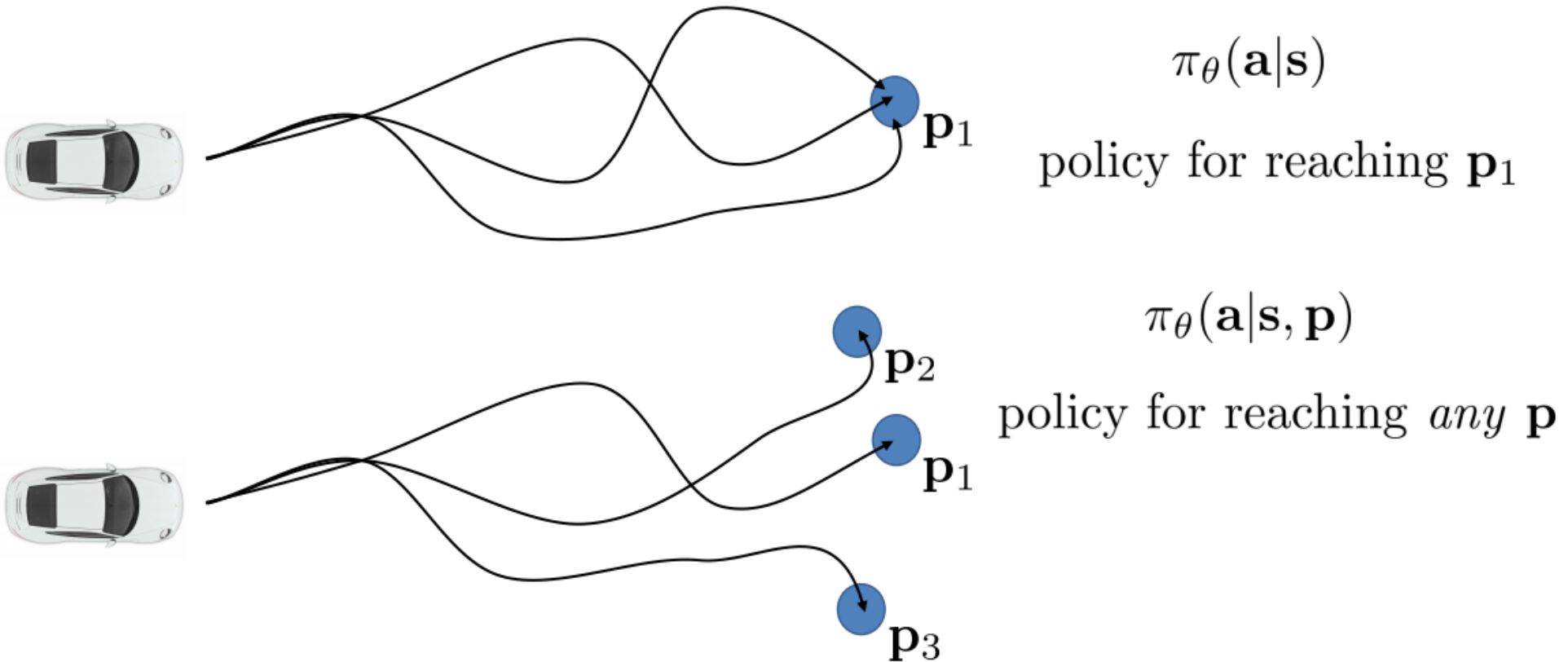
second step: $p(a_{t,1}|\mathbf{s}_t, a_{t,0})$

third step: $p(a_{t,2}|\mathbf{s}_t, a_{t,0}, a_{t,1})$

$$\begin{aligned} & p(a_{t,2}|\mathbf{s}_t, a_{t,0}, a_{t,1})p(a_{t,1}|\mathbf{s}_t, a_{t,0})p(a_{t,0}|\mathbf{s}_t) \\ &= p(a_{t,0}, a_{t,1}, a_{t,2}|\mathbf{s}_t) \\ &= p(\mathbf{a}_t|\mathbf{s}_t) \end{aligned}$$

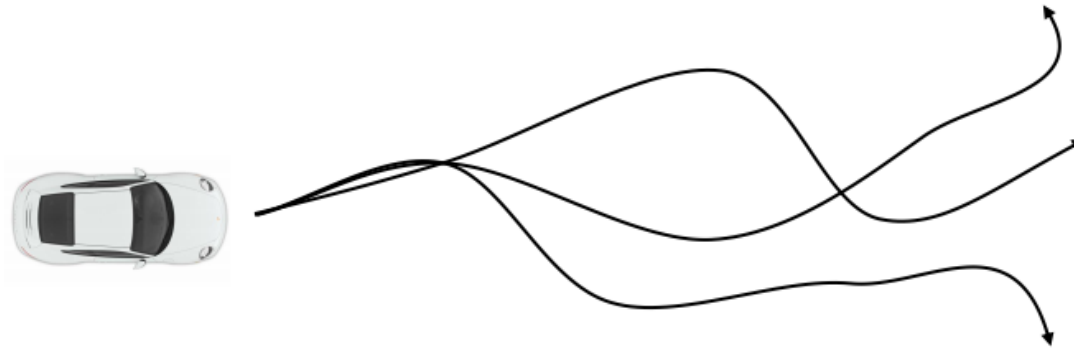
Does learning many tasks become easier?

Multi-task learning



Does learning many tasks become easier?

Goal-conditioned behavioral cloning



training time:

demo 1: $\{\mathbf{s}_1, \mathbf{a}_1, \dots, \mathbf{s}_{T-1}, \mathbf{a}_{T-1}, \mathbf{s}_T\}$ ← successful demo for reaching $\mathbf{s}_T$

demo 2: $\{\mathbf{s}_1, \mathbf{a}_1, \dots, \mathbf{s}_{T-1}, \mathbf{a}_{T-1}, \mathbf{s}_T\}$

demo 3: $\{\mathbf{s}_1, \mathbf{a}_1, \dots, \mathbf{s}_{T-1}, \mathbf{a}_{T-1}, \mathbf{s}_T\}$

for each demo $\{\mathbf{s}_1^i, \mathbf{a}_1^i, \dots, \mathbf{s}_{T-1}^i, \mathbf{a}_{T-1}^i, \mathbf{s}_T^i\}$

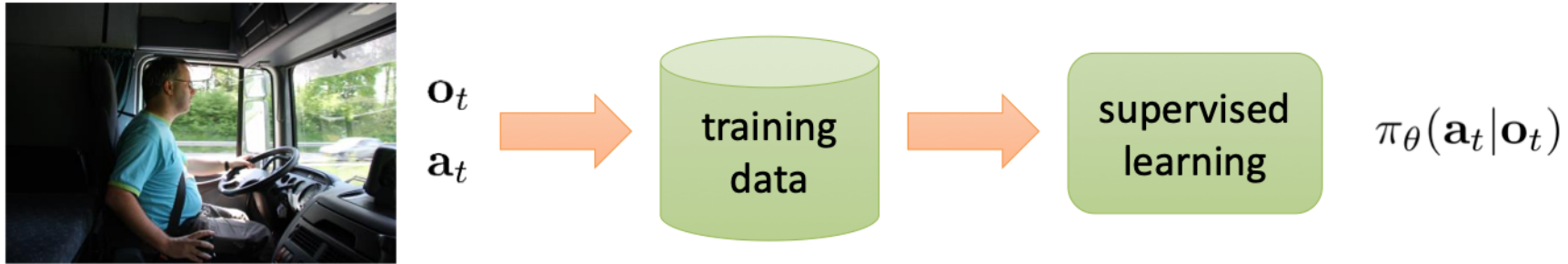
maximize $\log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i, \mathbf{g} = \mathbf{s}_T^i)$

learn $\pi_\theta(\mathbf{a} | \mathbf{s}, \mathbf{g})$ ← goal state

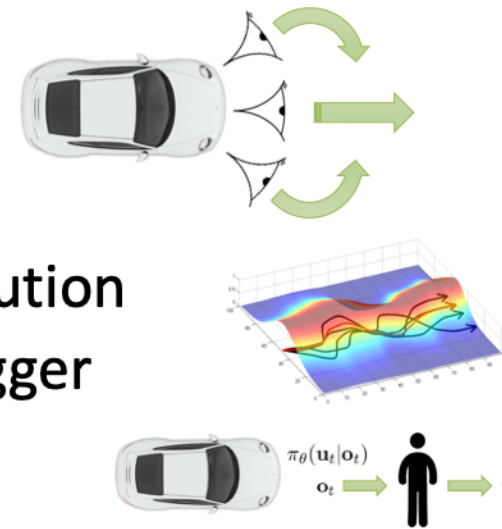
We see distributional shift in **two** places here!

Can you figure out what the second place is?

Summary: Imitation learning



- Often (but not always) insufficient by itself
 - Distribution mismatch problem
- Sometimes works well
 - Hacks (e.g. left/right images)
 - Samples from a stable trajectory distribution
 - Add more **on-policy** data, e.g. using Dagger
 - Better models that fit more accurately



Reinforcement Learning

Some slides are borrowed from Berkeley CS285.

What is reinforcement learning?

Introduction to RL

What is reinforcement learning?

- Mathematical formalism for learning-based decision making
- Approach for learning decision making and control from experience

Introduction to RL

How is this different from other machine learning topics?

Standard (supervised)
machine learning:

given $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$

learn to predict y from $\mathbf{x}$ $f(\mathbf{x}) \approx y$

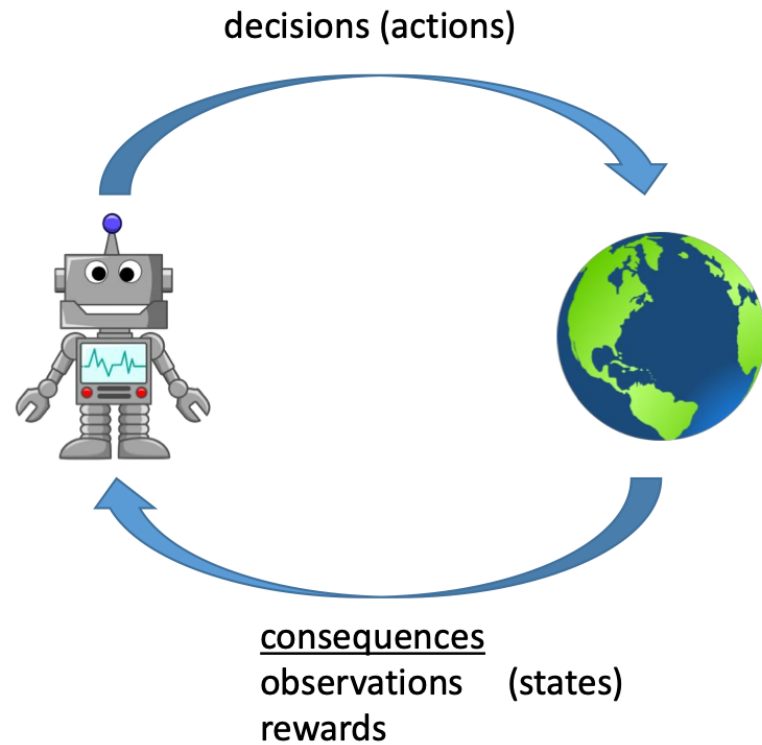
Usually assumes:

- i.i.d. data
- known ground truth outputs in training

Reinforcement learning:

- Data is **not** i.i.d.: previous outputs influence future inputs!
- Ground truth answer is not known, only know if we succeeded or failed
 - more generally, we know the reward

Introduction to RL



Actions: muscle contractions
Observations: sight, smell
Rewards: food

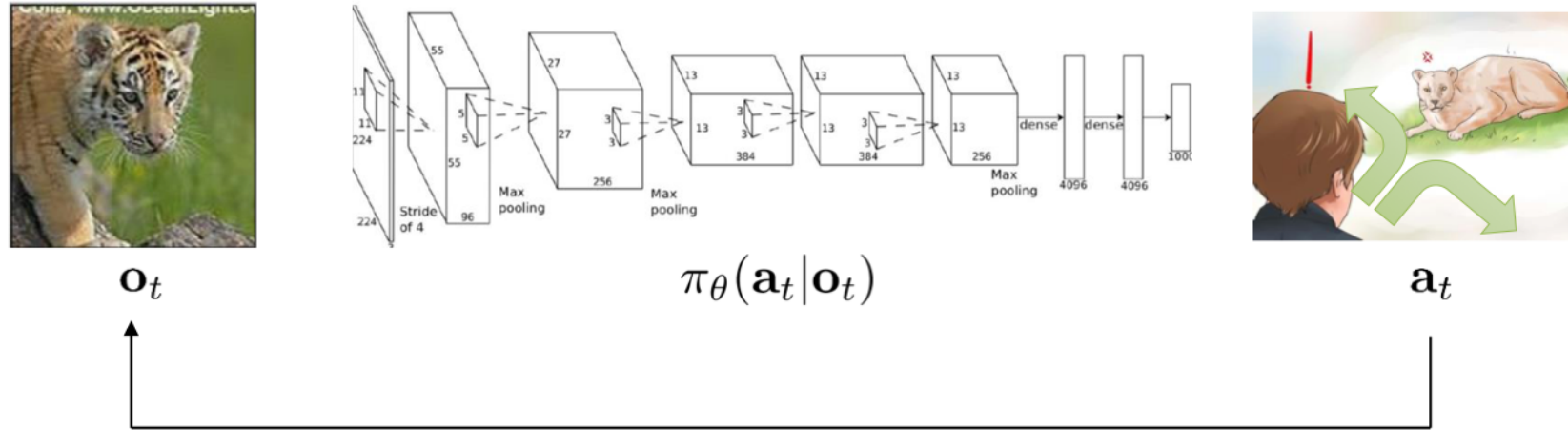


Actions: motor current or torque
Observations: camera images
Rewards: task success measure (e.g., running speed)



Actions: what to purchase
Observations: inventory levels
Rewards: profit

Terminology & notation



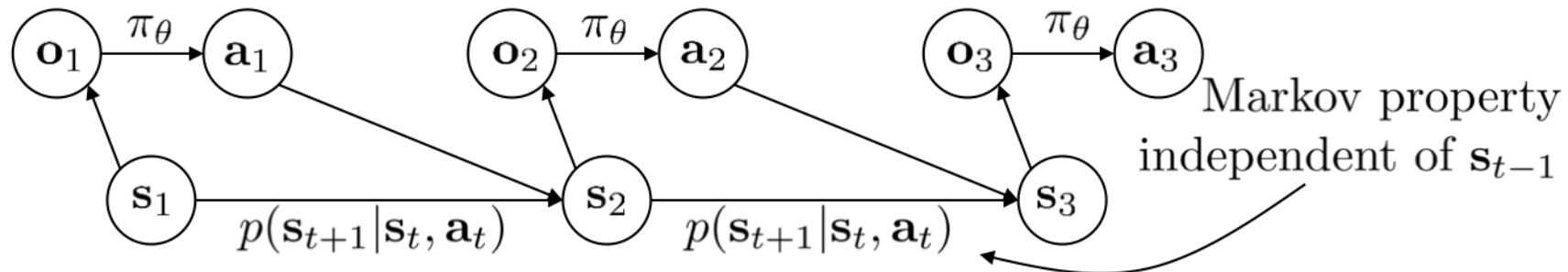
$\mathbf{s}_t$ – state

$\mathbf{o}_t$ – observation

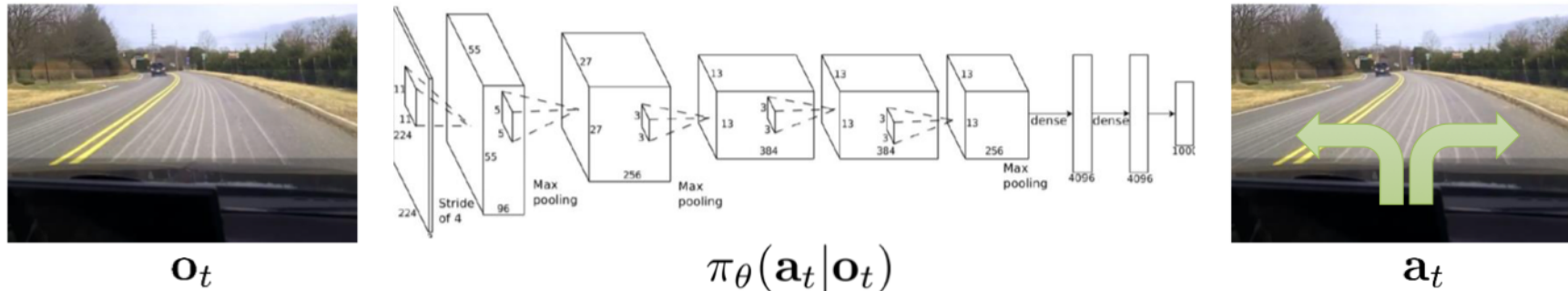
$\mathbf{a}_t$ – action

$\pi_{\theta}(\mathbf{a}_t | \mathbf{o}_t)$ – policy

$\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)$ – policy (fully observed)



Reward functions



which action is better or worse?

$r(\mathbf{s}, \mathbf{a})$: reward function

tells us which states and actions are better

$\mathbf{s}$, $\mathbf{a}$, $r(\mathbf{s}, \mathbf{a})$, and $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ define

Markov decision process



high reward



low reward

Definitions

Markov chain

$$\mathcal{M} = \{\mathcal{S}, \mathcal{T}\}$$

$\mathcal{S}$ – state space

states $s \in \mathcal{S}$ (discrete or continuous)

$\mathcal{T}$ – transition operator

$$p(s_{t+1}|s_t)$$

why “operator”?

$$\text{let } \mu_{t,i} = p(s_t = i)$$

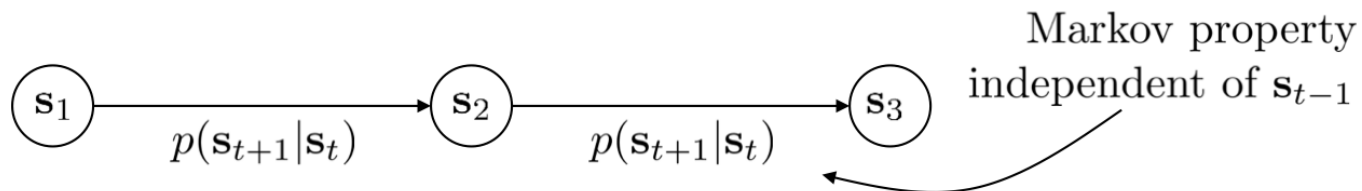
$\vec{\mu}_t$ is a vector of probabilities

$$\text{let } \mathcal{T}_{i,j} = p(s_{t+1} = i | s_t = j)$$

$$\text{then } \vec{\mu}_{t+1} = \mathcal{T} \vec{\mu}_t$$



Andrey Markov



Definitions

Markov decision process

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$$

$\mathcal{S}$ – state space

states $s \in \mathcal{S}$ (discrete or continuous)

$\mathcal{A}$ – action space

actions $a \in \mathcal{A}$ (discrete or continuous)

$\mathcal{T}$ – transition operator (now a tensor!)

let $\mu_{t,j} = p(s_t = j)$

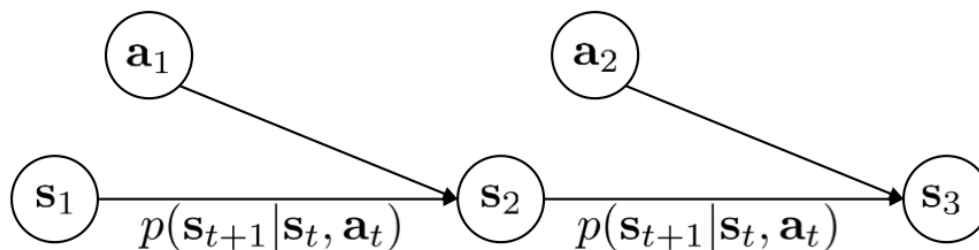
let $\xi_{t,k} = p(a_t = k)$

let $\mathcal{T}_{i,j,k} = p(s_{t+1} = i | s_t = j, a_t = k)$

$$\mu_{t+1,i} = \sum_{j,k} \mathcal{T}_{i,j,k} \mu_{t,j} \xi_{t,k}$$



Richard Bellman



Definitions

Markov decision process

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$$

$\mathcal{S}$ – state space

states $s \in \mathcal{S}$ (discrete or continuous)

$\mathcal{A}$ – action space

actions $a \in \mathcal{A}$ (discrete or continuous)

$\mathcal{T}$ – transition operator (now a tensor!)

r – reward function

$$r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$$

$r(s_t, a_t)$ – reward

Definitions

partially observed Markov decision process $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{E}, r\}$

$\mathcal{S}$ – state space states $s \in \mathcal{S}$ (discrete or continuous)

$\mathcal{A}$ – action space actions $a \in \mathcal{A}$ (discrete or continuous)

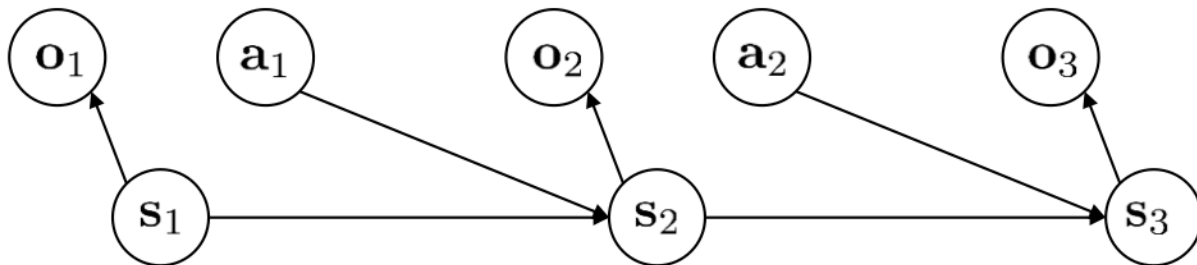
$\mathcal{O}$ – observation space observations $o \in \mathcal{O}$ (discrete or continuous)

$\mathcal{T}$ – transition operator (like before)

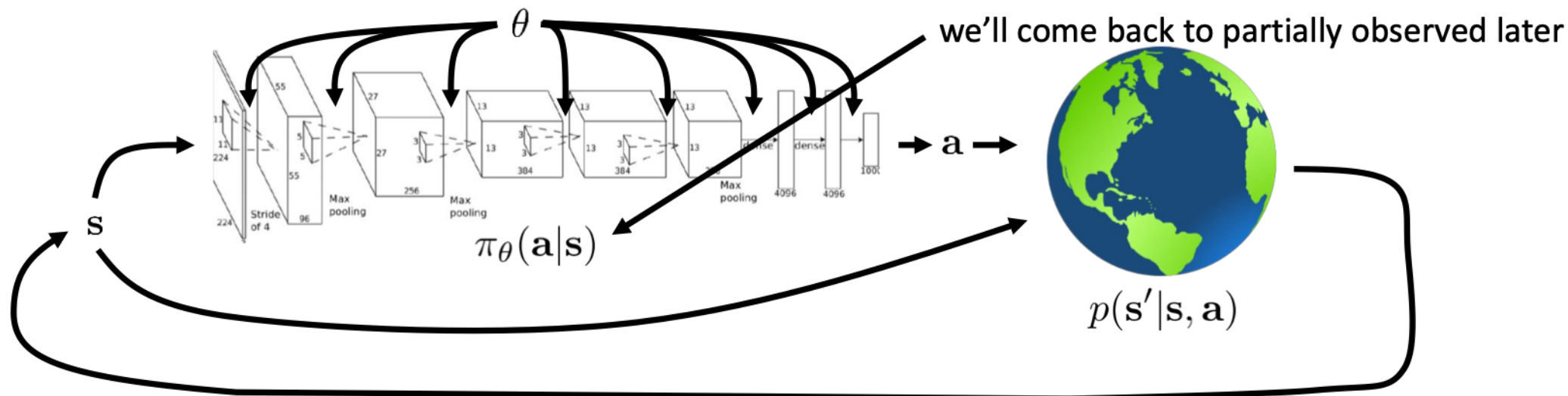
$\mathcal{E}$ – emission probability $p(o_t|s_t)$

r – reward function

$r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$



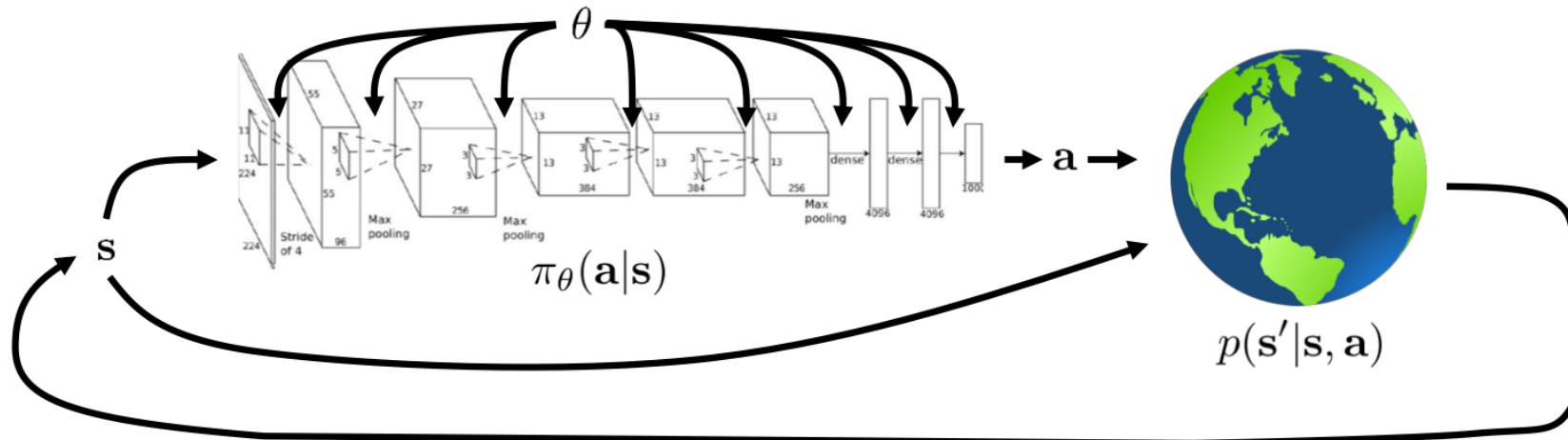
The goal of reinforcement learning



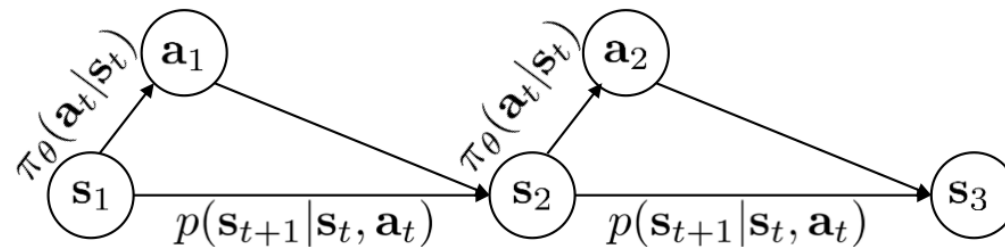
$$\underbrace{p_\theta(s_1, \mathbf{a}_1, \dots, s_T, \mathbf{a}_T)}_{p_\theta(\tau)} = p(s_1) \prod_{t=1}^T \pi_\theta(\mathbf{a}_t | s_t) p(s_{t+1} | s_t, \mathbf{a}_t)$$

$$\theta^* = \arg \max_{\theta} E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(s_t, \mathbf{a}_t) \right]$$

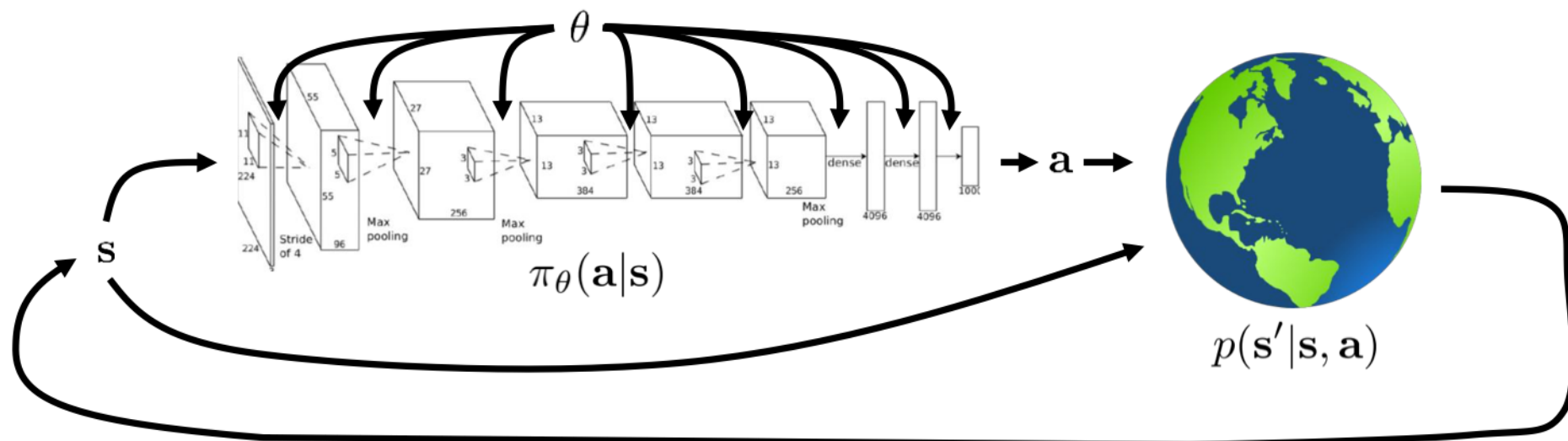
The goal of reinforcement learning



$$\underbrace{p_\theta(s_1, a_1, \dots, s_T, a_T)}_{p_\theta(\tau)} = p(s_1) \prod_{t=1}^T \underbrace{\pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)}_{\text{Markov chain on } (s, a)}$$

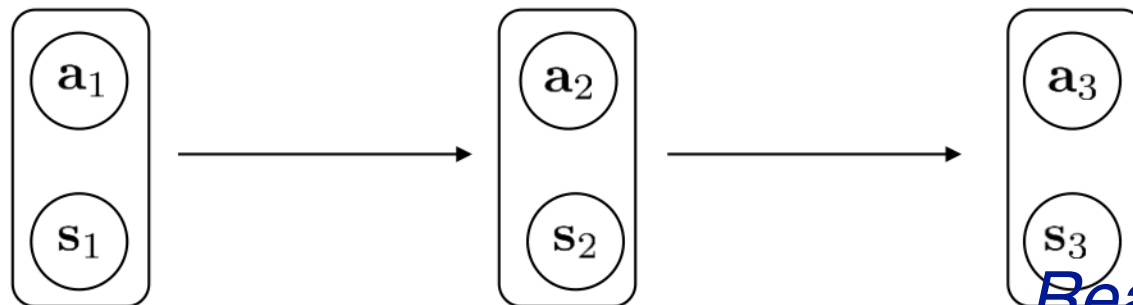


The goal of reinforcement learning



$$\underbrace{p(s_1, a_1, \dots, s_T, a_T)}_{p(\tau)} = p(s_1) \prod_{t=1}^T \underbrace{\pi_{\theta}(a_t|s_t)p(s_{t+1}|s_t, a_t)}_{\text{Markov chain on } (s, a)}$$

$$p((s_{t+1}, a_{t+1})|(s_t, a_t)) = p(s_{t+1}|s_t, a_t)\pi_{\theta}(a_{t+1}|s_{t+1})$$

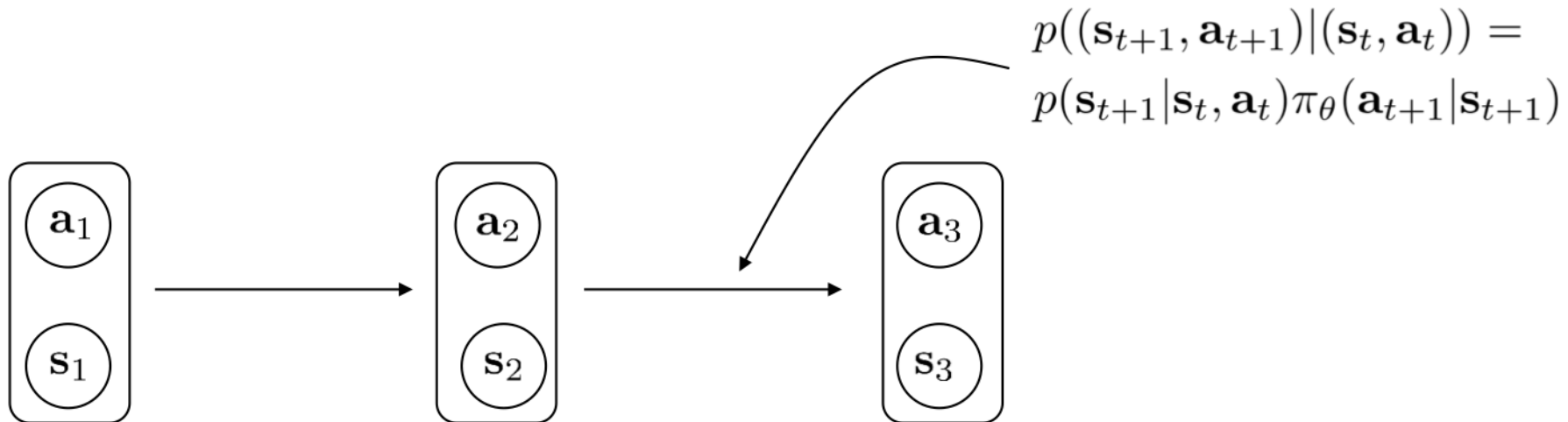


Read by Yourself

Finite horizon case: state-action marginal

$$\theta^* = \arg \max_{\theta} E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

$$= \arg \max_{\theta} \sum_{t=1}^T E_{(\mathbf{s}_t, \mathbf{a}_t) \sim p_{\theta}(\mathbf{s}_t, \mathbf{a}_t)} [r(\mathbf{s}_t, \mathbf{a}_t)] \quad p_{\theta}(\mathbf{s}_t, \mathbf{a}_t) \quad \text{state-action marginal}$$



Infinite horizon case: stationary distribution

$$\theta^* = \arg \max_{\theta} \sum_{t=1}^T E_{(\mathbf{s}_t, \mathbf{a}_t) \sim p_{\theta}(\mathbf{s}_t, \mathbf{a}_t)} [r(\mathbf{s}_t, \mathbf{a}_t)]$$

what if $T = \infty$?

does $p(\mathbf{s}_t, \mathbf{a}_t)$ converge to a *stationary* distribution?

$$\mu = \mathcal{T} \mu$$

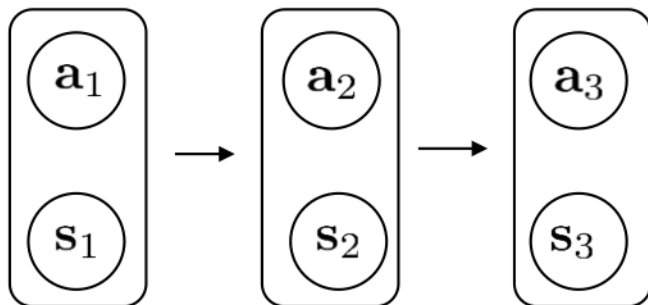
$$(\mathcal{T} - \mathbf{I})\mu = 0$$

$$\mu = p_{\theta}(\mathbf{s}, \mathbf{a}) \quad \text{stationary distribution}$$

stationary = the
same before and
after transition

μ is eigenvector of $\mathcal{T}$ with eigenvalue 1!

(always exists under some regularity conditions)



state-action transition operator

$$\begin{pmatrix} \mathbf{s}_{t+1} \\ \mathbf{a}_{t+1} \end{pmatrix} = \mathcal{T} \begin{pmatrix} \mathbf{s}_t \\ \mathbf{a}_t \end{pmatrix} \quad \begin{pmatrix} \mathbf{s}_{t+k} \\ \mathbf{a}_{t+k} \end{pmatrix} = \mathcal{T}^k \begin{pmatrix} \mathbf{s}_t \\ \mathbf{a}_t \end{pmatrix}$$

Infinite horizon case: stationary distribution

$$\theta^* = \arg \max_{\theta} \frac{1}{T} \sum_{t=1}^T E_{(\mathbf{s}_t, \mathbf{a}_t) \sim p_{\theta}(\mathbf{s}_t, \mathbf{a}_t)} [r(\mathbf{s}_t, \mathbf{a}_t)] \rightarrow E_{(\mathbf{s}, \mathbf{a}) \sim p_{\theta}(\mathbf{s}, \mathbf{a})} [r(\mathbf{s}, \mathbf{a})]$$

(in the limit as $T \rightarrow \infty$)

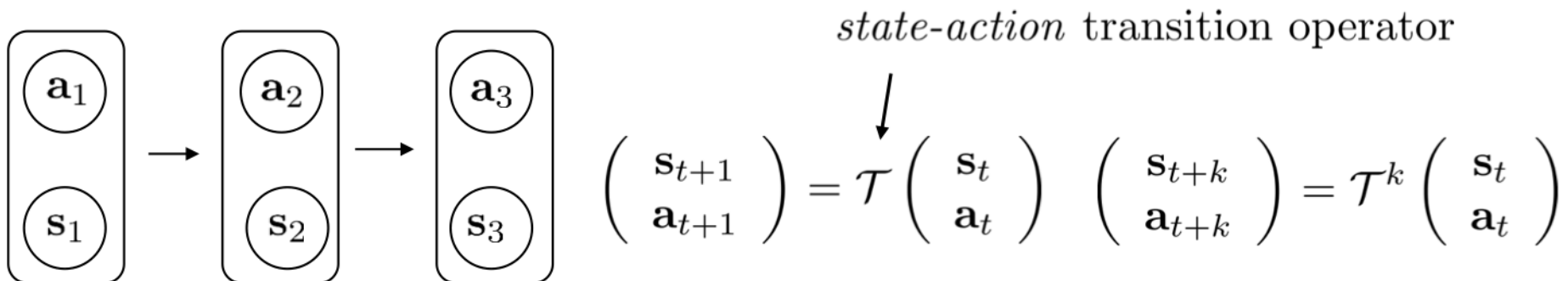
what if $T = \infty$?

does $p(\mathbf{s}_t, \mathbf{a}_t)$ converge to a *stationary* distribution?

$$\mu = \mathcal{T} \mu \quad (\mathcal{T} - \mathbf{I})\mu = 0 \quad \mu = p_{\theta}(\mathbf{s}, \mathbf{a}) \quad \text{stationary distribution}$$

stationary = the same before and after transition

μ is eigenvector of $\mathcal{T}$ with eigenvalue 1!
(always exists under some regularity conditions)



Expectations and stochastic systems

$$\theta^* = \arg \max_{\theta} E_{(\mathbf{s}, \mathbf{a}) \sim p_{\theta}(\mathbf{s}, \mathbf{a})} [r(\mathbf{s}, \mathbf{a})]$$

infinite horizon case

$$\theta^* = \arg \max_{\theta} \sum_{t=1}^T E_{(\mathbf{s}_t, \mathbf{a}_t) \sim p_{\theta}(\mathbf{s}_t, \mathbf{a}_t)} [r(\mathbf{s}_t, \mathbf{a}_t)]$$

finite horizon case

In RL, we almost always care about *expectations*



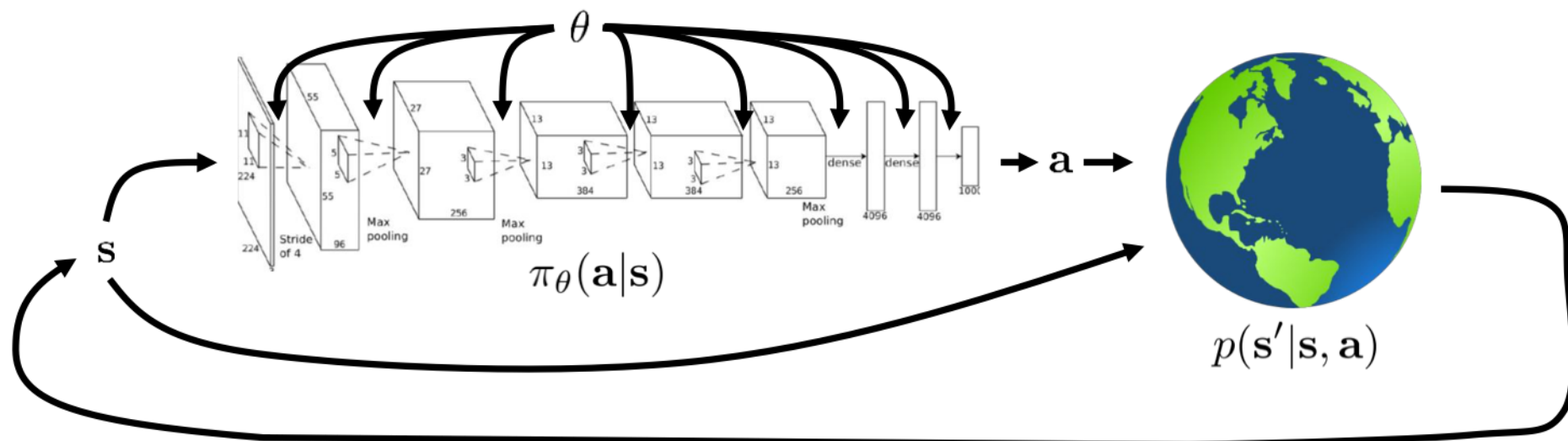
$r(\mathbf{x})$ – *not* smooth

$\pi_{\theta}(\mathbf{a} = \text{fall}) = \theta$

$E_{\pi_{\theta}}[r(\mathbf{x})]$ – *smooth* in θ !

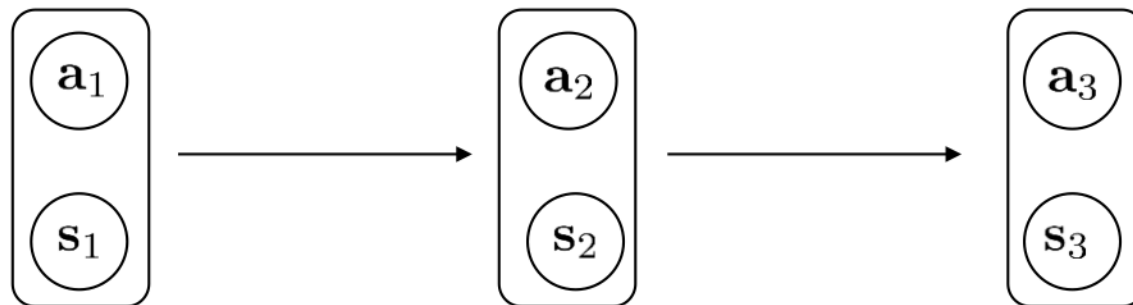
Policy Gradient

The goal of reinforcement learning



$$\underbrace{p(s_1, a_1, \dots, s_T, a_T)}_{p(\tau)} = p(s_1) \prod_{t=1}^T \underbrace{\pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)}_{\text{Markov chain on } (s, a)}$$

$$p((s_{t+1}, a_{t+1})|(s_t, a_t)) = p(s_{t+1}|s_t, a_t)\pi_\theta(a_{t+1}|s_{t+1})$$



The goal of reinforcement learning

$$\theta^* = \arg \max_{\theta} E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

$$\theta^* = \arg \max_{\theta} E_{(\mathbf{s}, \mathbf{a}) \sim p_{\theta}(\mathbf{s}, \mathbf{a})} [r(\mathbf{s}, \mathbf{a})]$$

infinite horizon case

$$\theta^* = \arg \max_{\theta} \sum_{t=1}^T E_{(\mathbf{s}_t, \mathbf{a}_t) \sim p_{\theta}(\mathbf{s}_t, \mathbf{a}_t)} [r(\mathbf{s}_t, \mathbf{a}_t)]$$

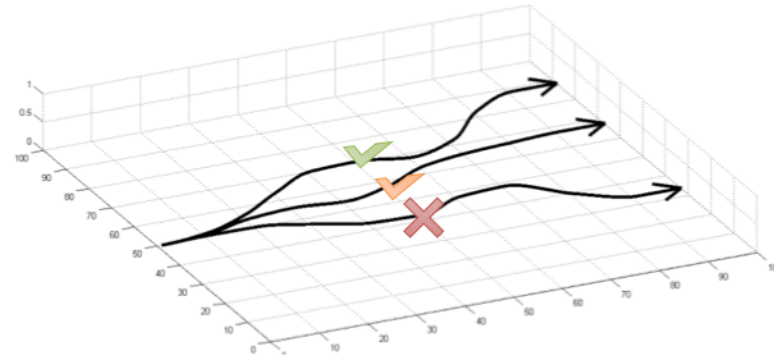
finite horizon case

Evaluating the objective

$$\theta^* = \arg \max_{\theta} \underbrace{E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]}_{J(\theta)}$$

$$J(\theta) = E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

sum over samples from π_{θ}



Direct policy differentiation

$$\theta^* = \arg \max_{\theta} \underbrace{E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]}_{J(\theta)}$$

a convenient identity

$$\underline{p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau)} = p_{\theta}(\tau) \frac{\nabla_{\theta} p_{\theta}(\tau)}{p_{\theta}(\tau)} = \underline{\nabla_{\theta} p_{\theta}(\tau)}$$

$$J(\theta) = E_{\tau \sim p_{\theta}(\tau)} \left[\underbrace{r(\tau)}_{\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t)} \right] = \int p_{\theta}(\tau) r(\tau) d\tau$$

$$\nabla_{\theta} J(\theta) = \int \underline{\nabla_{\theta} p_{\theta}(\tau)} r(\tau) d\tau = \int \underline{p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau)} r(\tau) d\tau = E_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)]$$

Direct policy differentiation

$$\begin{aligned}
 \theta^* &= \arg \max_{\theta} J(\theta) \\
 J(\theta) &= E_{\tau \sim p_{\theta}(\tau)}[r(\tau)] \\
 \nabla_{\theta} J(\theta) &= E_{\tau \sim p_{\theta}(\tau)}[\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)]
 \end{aligned}$$

log of both sides

$$\underbrace{p_{\theta}(\mathbf{s}_1, \mathbf{a}_1, \dots, \mathbf{s}_T, \mathbf{a}_T)}_{p_{\theta}(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$$

$$\log p_{\theta}(\tau) = \log p(\mathbf{s}_1) + \sum_{t=1}^T \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) + \log p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$$

$$\nabla_{\theta} \left[\cancel{\log p(\mathbf{s}_1)} + \sum_{t=1}^T \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) + \cancel{\log p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)} \right]$$

$$\nabla_{\theta} J(\theta) = E_{\tau \sim p_{\theta}(\tau)} \left[\left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

Evaluating the policy gradient

$$\text{recall: } J(\theta) = E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

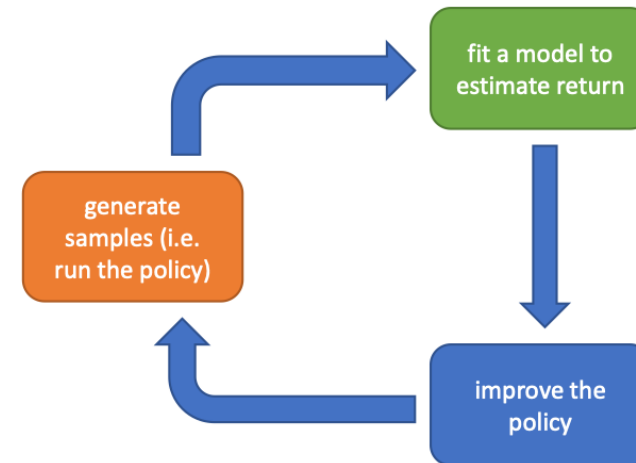
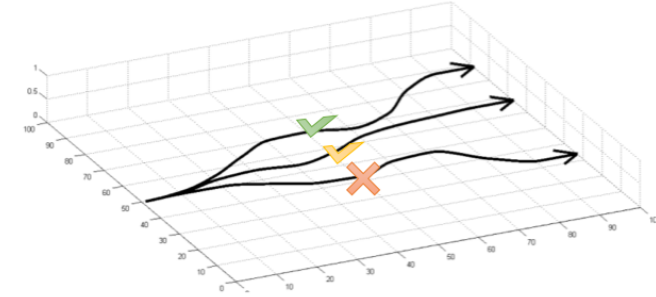
$$\nabla_{\theta} J(\theta) = E_{\tau \sim p_{\theta}(\tau)} \left[\left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^T r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$$

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)$ (run the policy)
2. $\nabla_{\theta} J(\theta) \approx \sum_i \left(\sum_t \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t^i | \mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$



Evaluating the policy gradient

recall: $J(\theta) = E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(s_t, \mathbf{a}_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(s_{i,t}, \mathbf{a}_{i,t})$

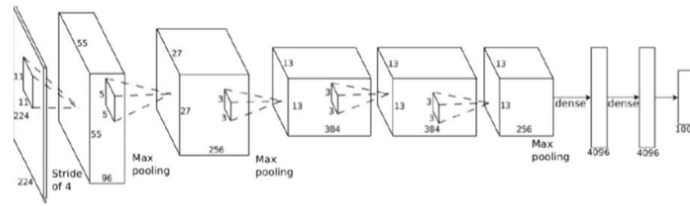
$$\nabla_{\theta} J(\theta) = E_{\tau \sim p_{\theta}(\tau)} \left[\left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t | s_t) \right) \left(\sum_{t=1}^T r(s_t, \mathbf{a}_t) \right) \right]$$

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | s_{i,t}) \right) \left(\sum_{t=1}^T r(s_{i,t}, \mathbf{a}_{i,t}) \right)$$

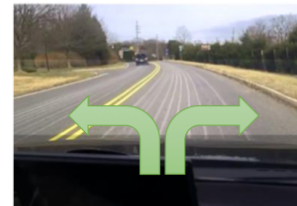
what is this?



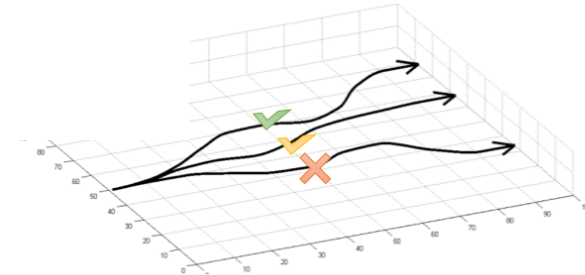
s_t



$\pi_{\theta}(\mathbf{a}_t | s_t)$



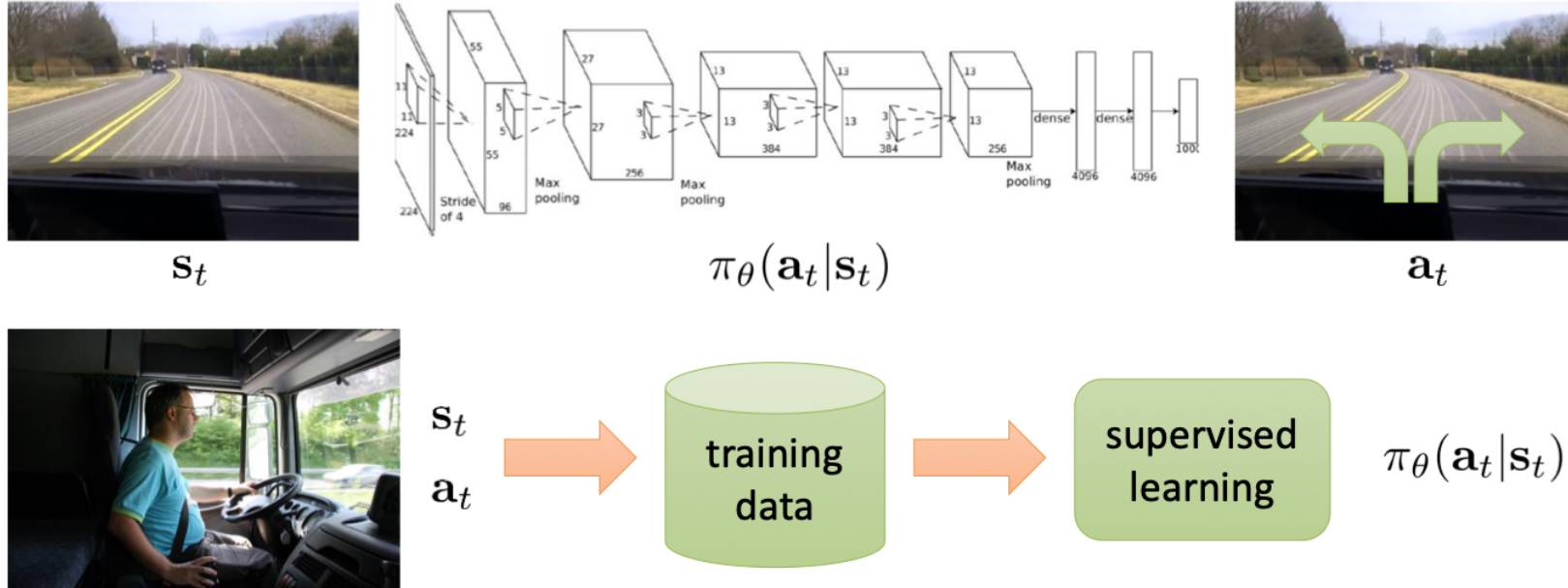
$\mathbf{a}_t$



Comparison to maximum likelihood

policy gradient:
$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^T r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

maximum likelihood:
$$\nabla_{\theta} J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right)$$



Example: Gaussian policies

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^T r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

example: $\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) = \mathcal{N}(f_{\text{neural network}}(\mathbf{s}_t); \Sigma)$

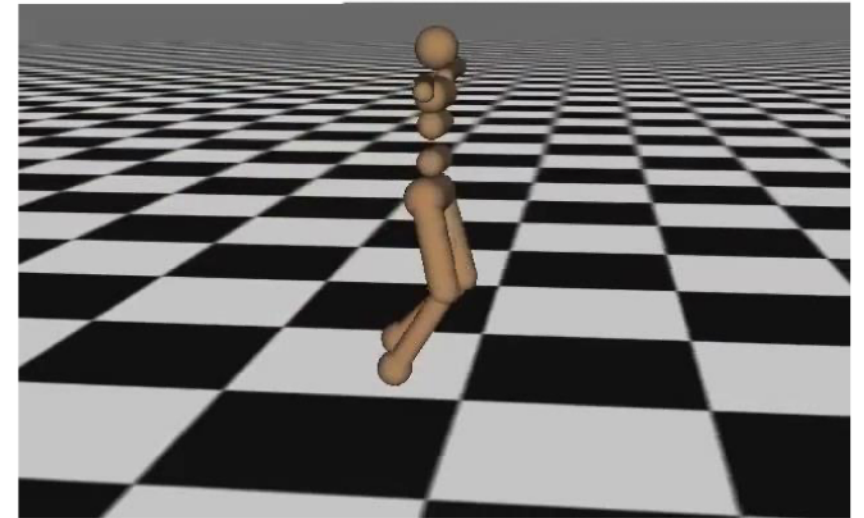
$$\log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) = -\frac{1}{2} \|f(\mathbf{s}_t) - \mathbf{a}_t\|_{\Sigma}^2 + \text{const}$$

$$\nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) = -\frac{1}{2} \Sigma^{-1} (f(\mathbf{s}_t) - \mathbf{a}_t) \frac{df}{d\theta}$$

REINFORCE algorithm:

- 1. sample $\{\tau^i\}$ from $\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)$ (run it on the robot)
- 2. $\nabla_{\theta} J(\theta) \approx \sum_i \left(\sum_t \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t^i | \mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
- 3. $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$

Iteration 2000



What did we just do?

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^T r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \underbrace{\nabla_{\theta} \log \pi_{\theta}(\tau_i)}_{\sum_{t=1}^T \nabla_{\theta} \log_{\theta} \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t})} r(\tau_i)$$

maximum likelihood: $\nabla_{\theta} J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log \pi_{\theta}(\tau_i)$

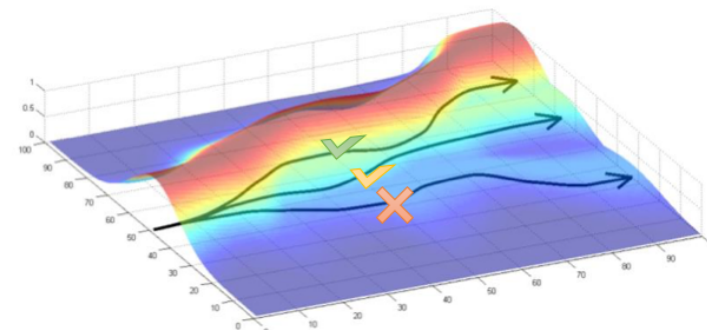
good stuff is made more likely

bad stuff is made less likely

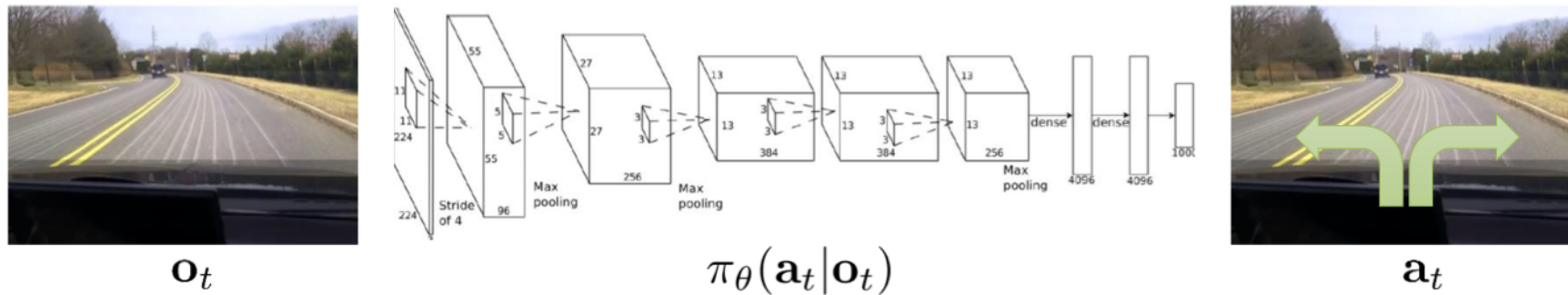
simply formalizes the notion of “trial and error”!

REINFORCE algorithm:

- 1. sample $\{\tau^i\}$ from $\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)$ (run it on the robot)
- 2. $\nabla_{\theta} J(\theta) \approx \sum_i \left(\sum_t \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t^i | \mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
- 3. $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$



Partial observability



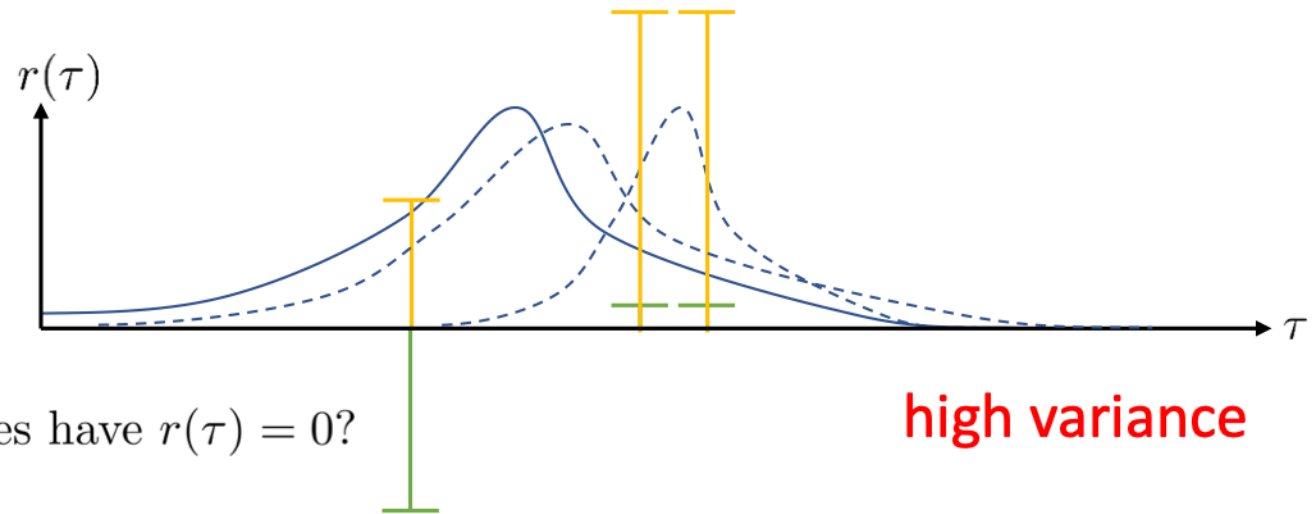
$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{o}_{i,t}) \right) \left(\sum_{t=1}^T r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

Markov property is not actually used!

Can use policy gradient in partially observed MDPs without modification

What is wrong with the policy gradient?

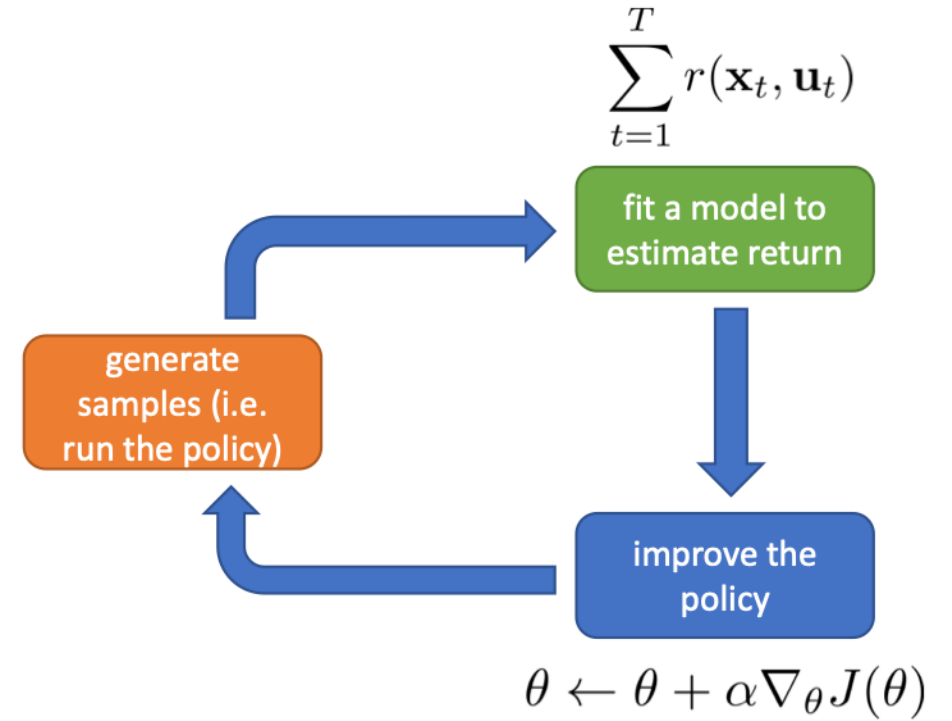
$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau)$$



even worse: what if the two “good” samples have $r(\tau) = 0$?

Review

- Evaluating the RL objective
 - Generate samples
- Evaluating the policy gradient
 - Log-gradient trick
 - Generate samples
- Understanding the policy gradient
 - Formalization of trial-and-error
- Partial observability
 - Works just fine
- What is wrong with policy gradient?





Introduction to Embodied AI

Next week: Lecture 9, Policy II

