CSCI 699: Robotic Perception

Yue Wang

Sep 10th, 2025

Administratives

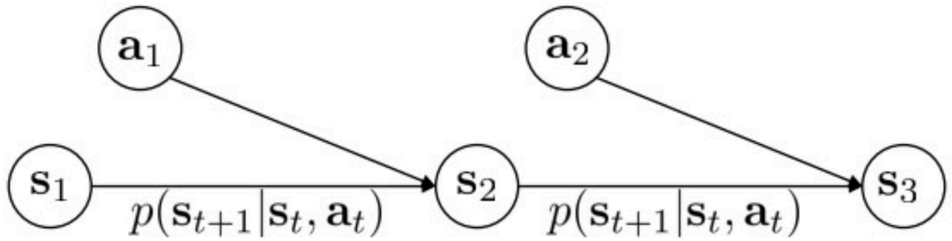
Project

- Sign-up sheet: https://docs.google.com/spreadsheets/d/1dtALsb_-Mo7E1ghzYDc-m0n6BnbyLCYyxVxvGNDcDnE/edit?usp=sharing
- Project proposal
  - 2-4 pages
    - Introduction
    - Related work
    - Proposed method / pipeline
    - (Optionally) risk mitigation
    - (Tentative) work assignment
  - Due on 9/21

Notations, MDP, POMDP

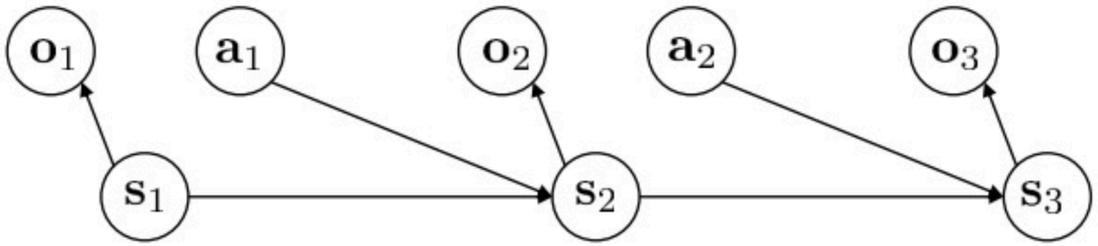
MDP

- Definitions
  - S is the state space. $s_t \in S$ is the state at time step t.
  - A is the action space. $a_t \in A$ is the action at time step t.
  - O is the observation space. $o_t \in O$ is the observation at time step t.
  - p is the one-step transition probability (a.k.a, dynamics): $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.
  - h is the observation model: $o_t \sim h(\cdot \mid s_t)$
  - $r : S \times A \to \mathbb{R}$ is the reward function.
- A Markov decision process (MDP) is a tuple (S, A, p, r)
  - Goal: learn a policy $\pi_\theta(a_t \mid s_t)$



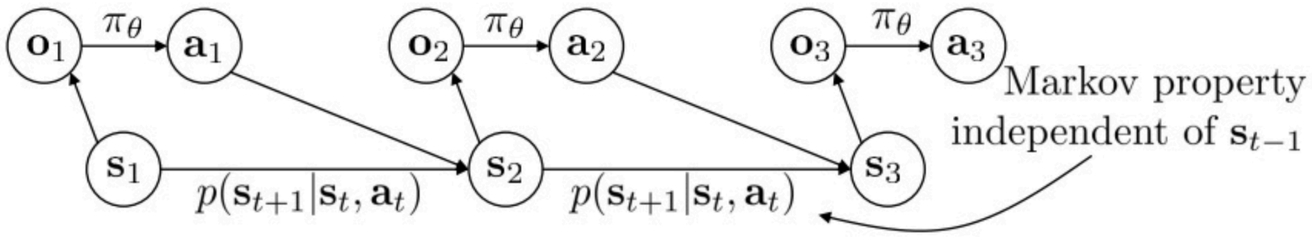
POMDP

- Definitions
  - S is the state space. $s_t \in S$ is the state at time step t.
  - A is the action space. $a_t \in A$ is the action at time step t.
  - O is the observation space. $o_t \in O$ is the observation at time step t.
  - p is the one-step transition probability (a.k.a, dynamics): $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.
  - h is the observation model: $o_t \sim h(\cdot \mid s_t)$
  - $r : S \times A \to \mathbb{R}$ is the reward function.
- A partially observed Markov decision process (POMDP) is a tuple (S, A, O, p, h, r)
  - Goal: learn a policy $\pi_\theta(a_t \mid o_t)$



POMDP

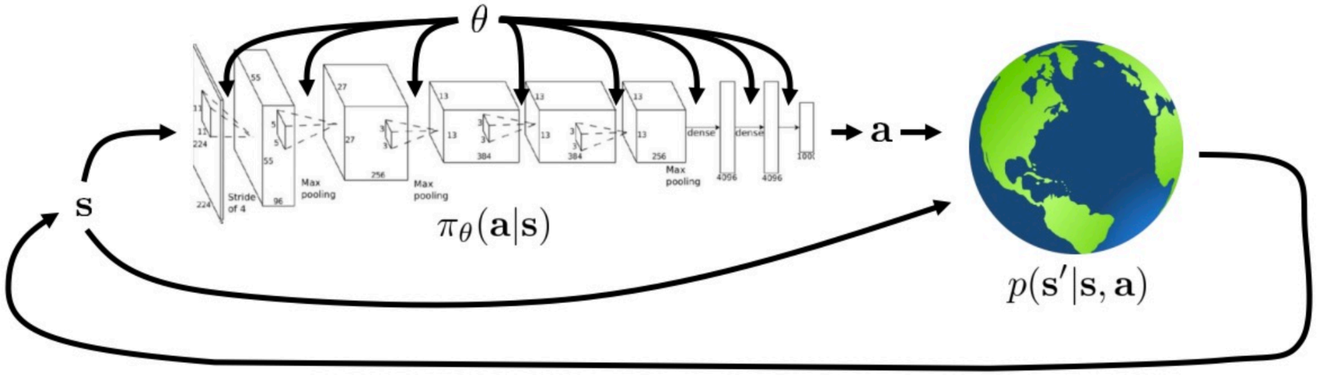
- Definitions
  - S is the state space. $s_t \in S$ is the state at time step t.
  - A is the action space. $a_t \in A$ is the action at time step t.
  - O is the observation space. $o_t \in O$ is the observation at time step t.
  - p is the one-step transition probability (a.k.a, dynamics): $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.
  - h is the observation model: $o_t \sim h(\cdot \mid s_t)$
  - $r : S \times A \to \mathbb{R}$ is the reward function.
- Why Markov?
  - Allow us to throw away history once state is known!



Andrey Markov

The Goal of Reinforcement Learning and Control

- Finite horizon case: T is finite
- Infinite horizon case: $T = \infty$
- The cumulative reward can be discounted: $\sum_t \gamma^t r(s_t, a_t)$ where $0 < \gamma \leq 1$



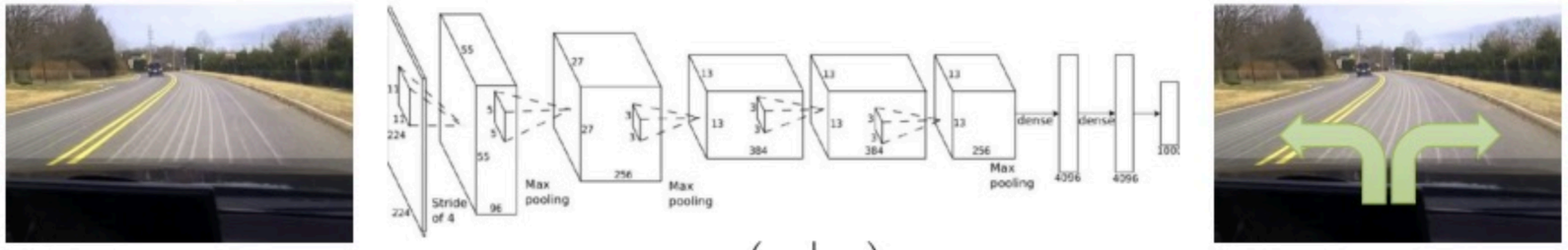
$$\underbrace{p_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

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

Imitation Learning

Imitation Learning

- Collect expert data (observation/state and action pairs)
- Train a function to map observations/states to actions

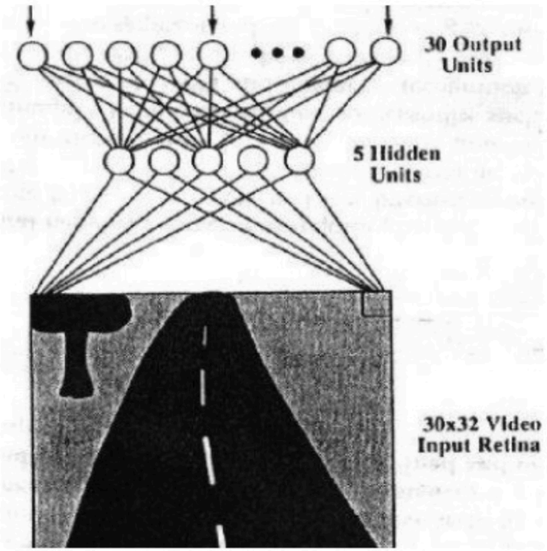


$\mathbf{o}_t$ $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ $\mathbf{a}_t$

Deep Imitation Learning in 1989

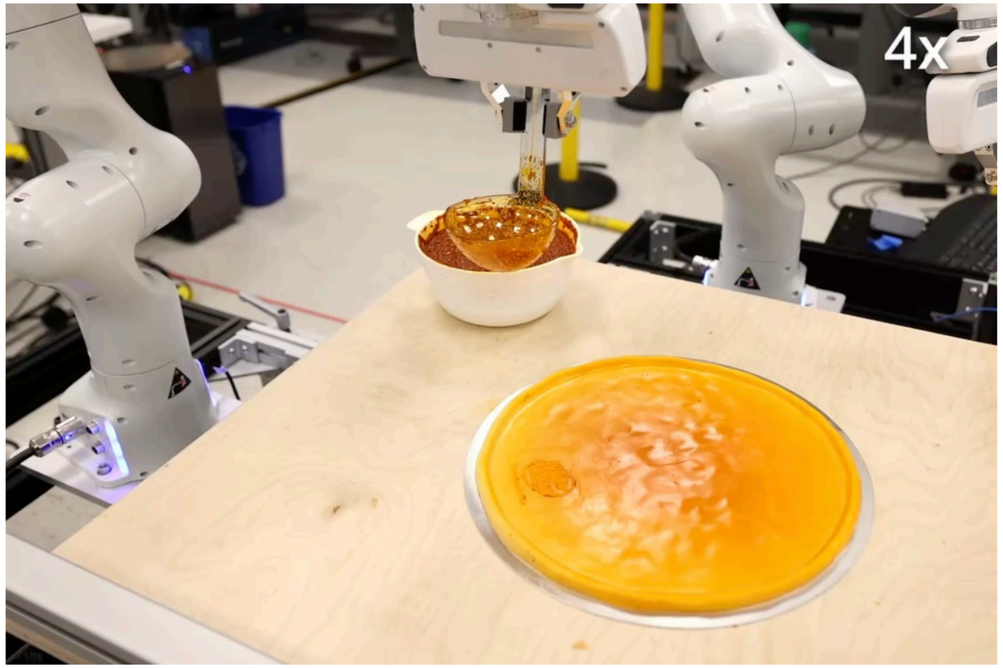
ALVINN:
AN AUTONOMOUS LAND VEHICLE IN A NEURAL NETWORK

Dean A. Pomerleau
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213



Deep Imitation Learning Today

- Left: Mobile ALOHA; Right: Diffusion Policy

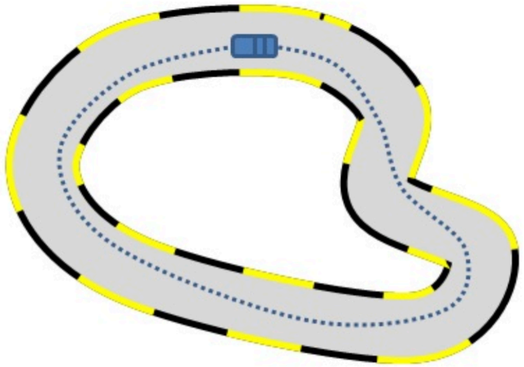


Why/how is imitation learning different from standard supervised learning?

It Sounds Like Supervised Learning

- Main idea:
  - Collect expert data (observation/state and action pairs)
  - Train a function to map observations/ states to actions
  - A.k.a "behavior cloning (BC)"

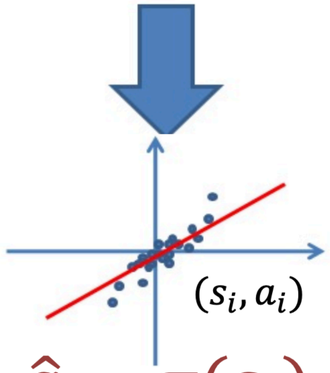
Expert Trajectories



s: state / observation

a: expert action

$$s_0, a_0, s_1, a_1, \ldots, s_T, a_T$$



$$\hat{a} = \pi(s_i)$$

Supervised Learning

Learned Policy π

Mapping from state (image) to control (steering direction)

It Sounds Like Supervised Learning

- High-dimensional observations

[Pomerleau89, Saxena05, Ross11a]

Input:

Camera Image

Policy

Output:

Steering Angle
in [-1, 1]

It Sounds Like Supervised Learning

- Does BC work?

[Pomerleau89, Saxena05, Ross11a]

Input:

Camera Image

Policy

Output:

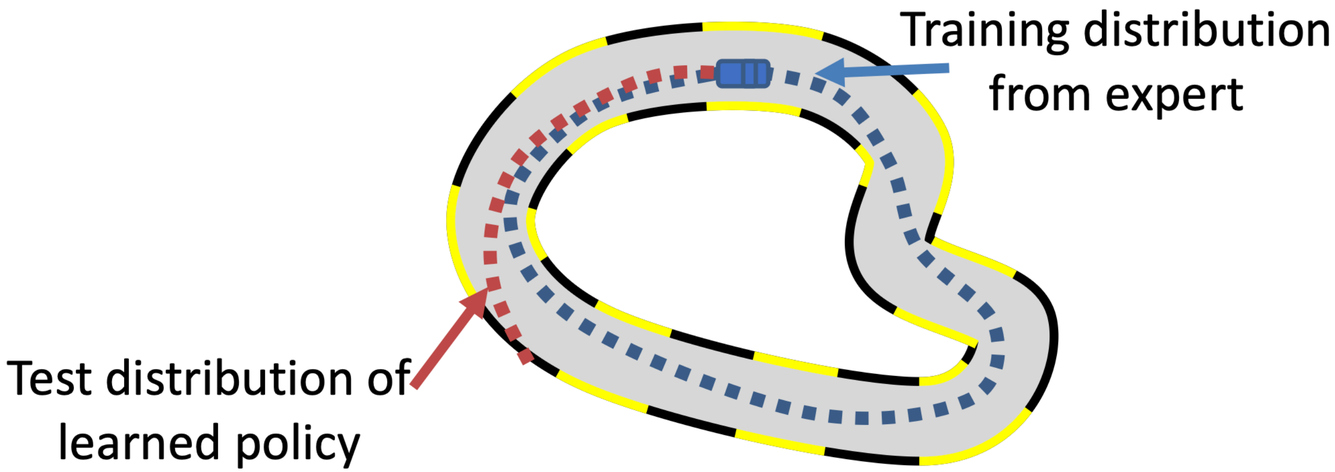
Steering Angle
in [-1, 1]

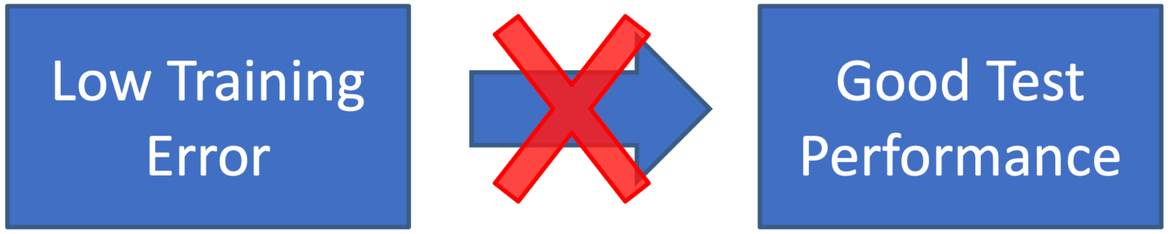
It Is NOT Supervised Learning

- The i.i.d (independent and identically distributed) assumption doesn't hold!
- (x_i, y_i) is i.i.d sampled from some distribution.
- Theory proves that "low training error" -> "good test performance"

It Is NOT Supervised Learning

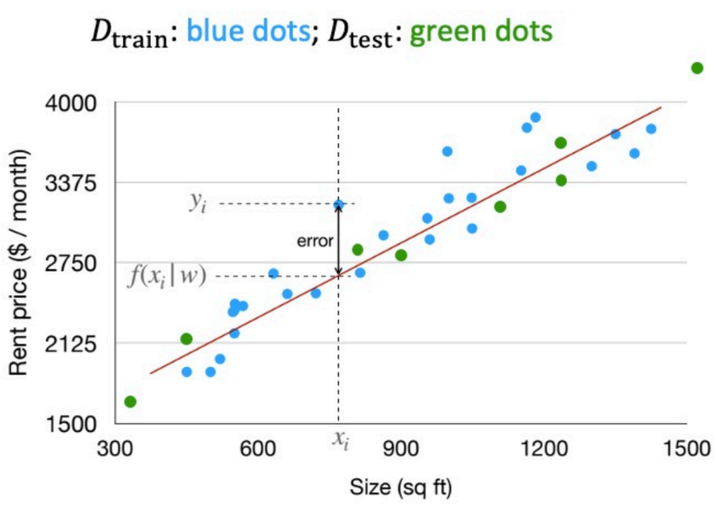
- The expert data only contains good behaviors
- Therefore, a small mistake in the test time will cause cascading failures

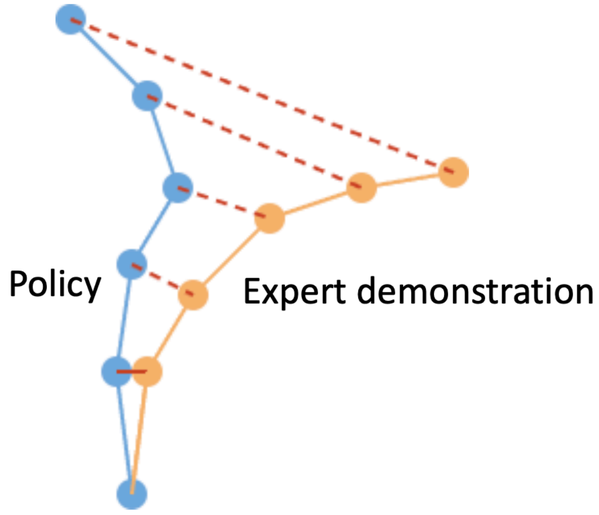




It Is NOT Supervised Learning

- Recap the domain shift problem in machine learning

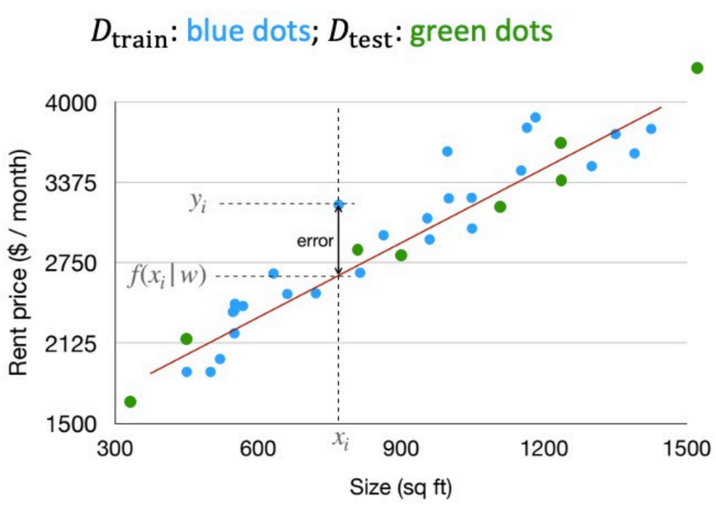


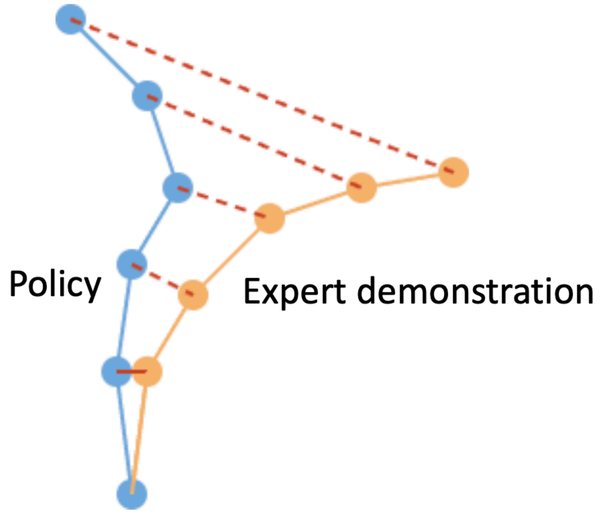


- Even worse in sequential decision making! The domain shift is dynamic and fatal
  - Policy only trained with expert data
  - Expert has almost zero chance seeing bad state
  - Statistically speaking, imitation learning is “cliff walking"
  - Once we make one mistake, we are outside of the training distribution.

It Is NOT Supervised Learning

- Recap the domain shift problem in machine learning





- Why is it so pessimistic?
  - In reality, we can often recover from mistakes
  - But this paradox still exists: IL could work better if the data has more mistakes.

The Issue Is Also In Sequential Prediction

- Natural language generation
  - If a model generates one wrong token, it drifts off distribution and errors accumulate.
- Structured prediction tasks
  - E.g. stock market prediction

How to address this issue?

How To Address This Problem?

- Be smart about how we collect (and augment) our data
- Change the algorithm (Dagger)
- Use powerful models that make few mistakes

How To Address This Problem?

- **Be smart about how we collect (and augment) our data**
- Change the algorithm (Dagger)
- Use powerful models that make few mistakes

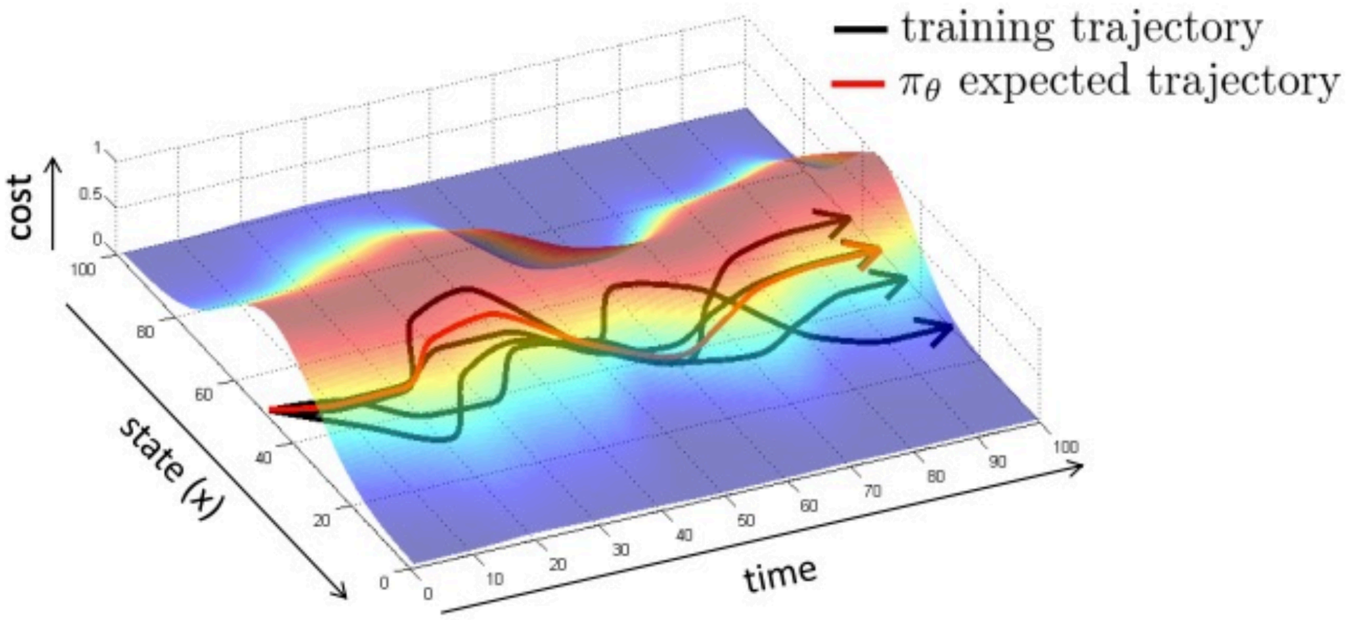
ALVINN:
AN AUTONOMOUS LAND VEHICLE IN A NEURAL NETWORK

Dean A. Pomerleau
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

"…the network must not solely be shown examples of accurate driving, but also how to recover (i.e. return to the road center) once a mistake has been made."

What Makes BC Easy / Hard?

- **Be smart about how we collect (and augment) our data**
  - Idea 1: Intentionally add mistakes and corrections
    - The mistakes hurt, but the corrections help, often more then the mistakes hurt
  - Idea 2:Use data augmentation
    - Add some “fake” or “synthetic” data that illustrates corrections

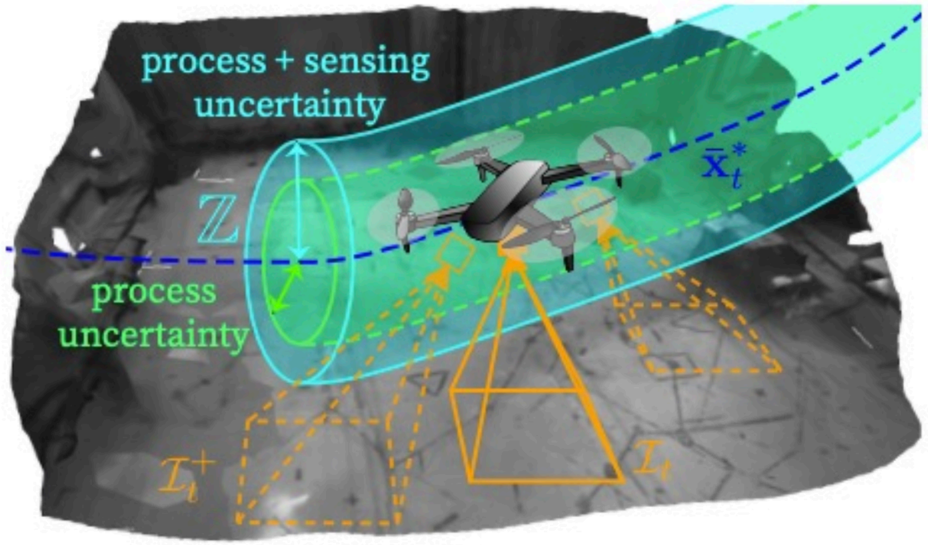


Data Augmentation Example

- Augment a policy using a tube

Demonstration-Efficient Guided Policy Search via Imitation of Robust Tube MPC

Andrea Tagliabue, Dong-Ki Kim, Michael Everett, Jonathan P. How

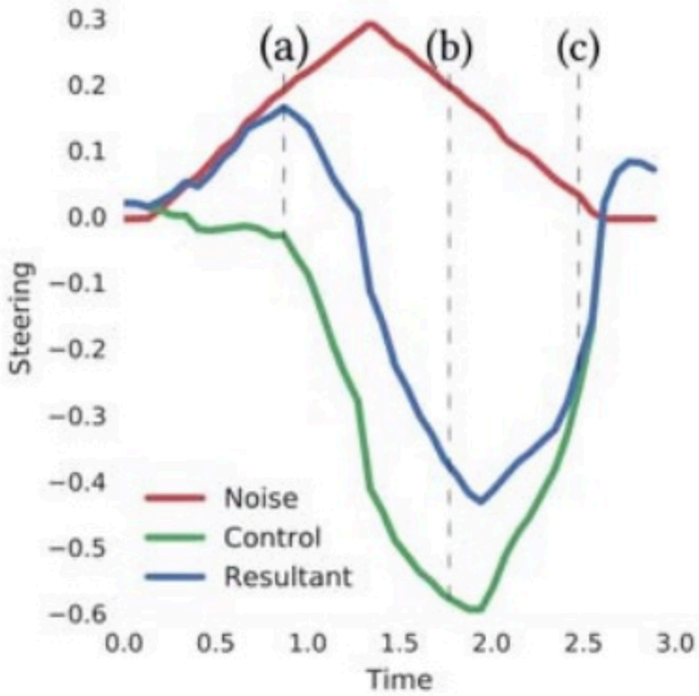


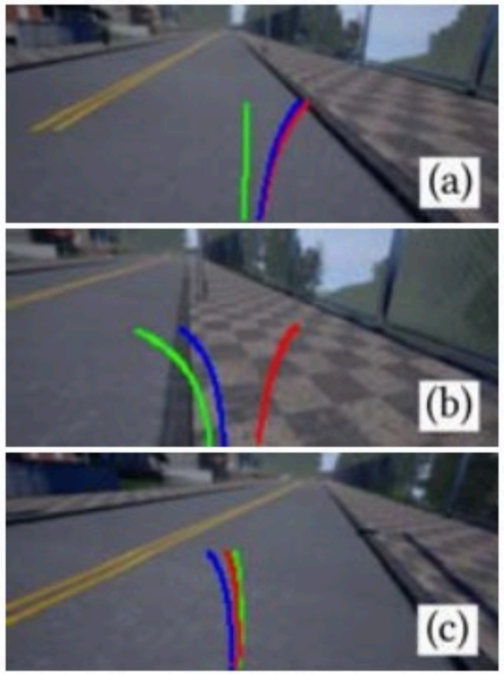
Data Augmentation Example

- Perturb the driver's steering signal

End-to-end Driving via Conditional Imitation Learning

Felipe Codevilla[1,2] Matthias Müller[1,3] Antonio López[2] Vladlen Koltun[1] Alexey Dosovitskiy[1]

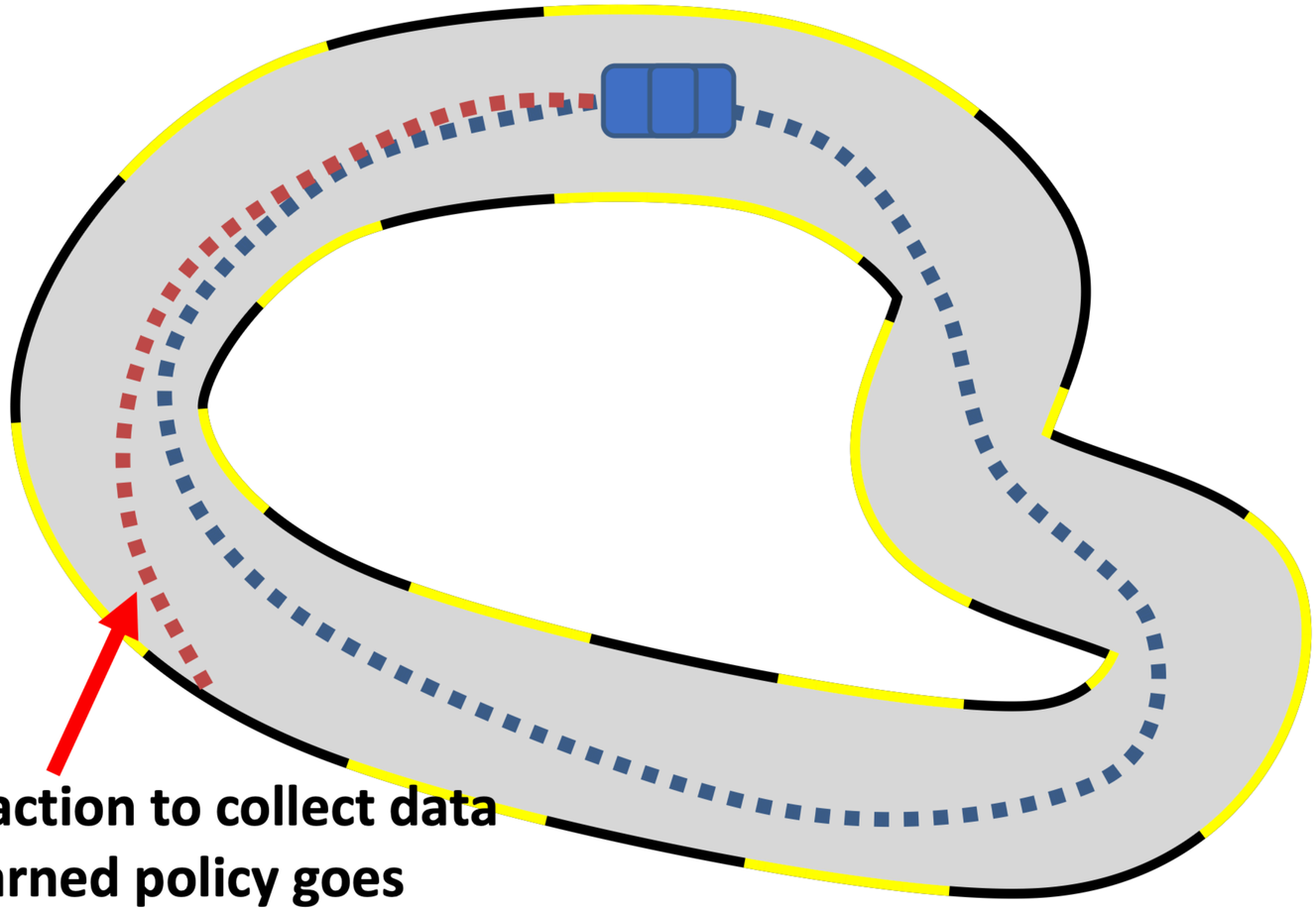




How To Address This Problem?

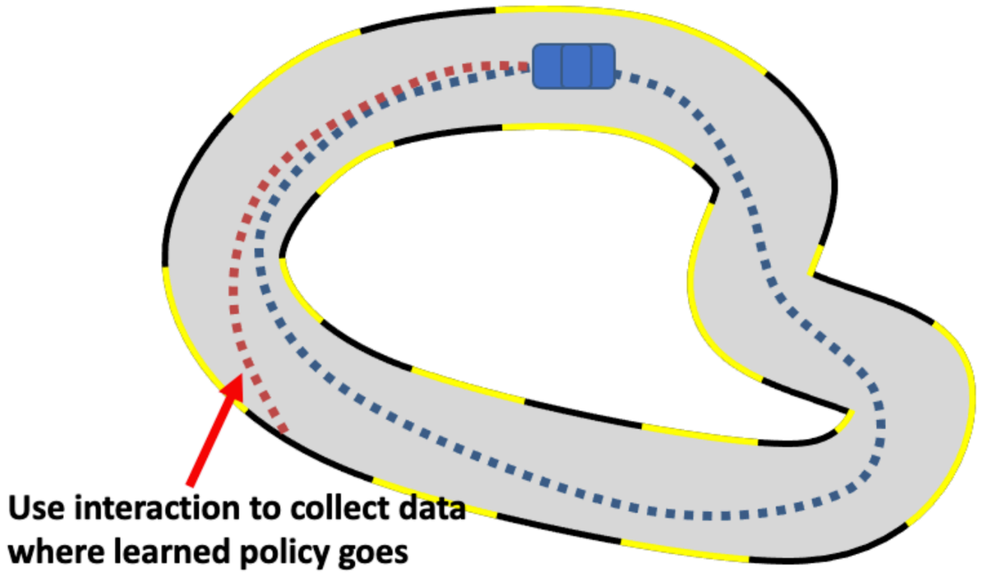
- Be smart about how we collect (and augment) our data
- **Change the algorithm (Dagger)**
- Use powerful models that make few mistakes

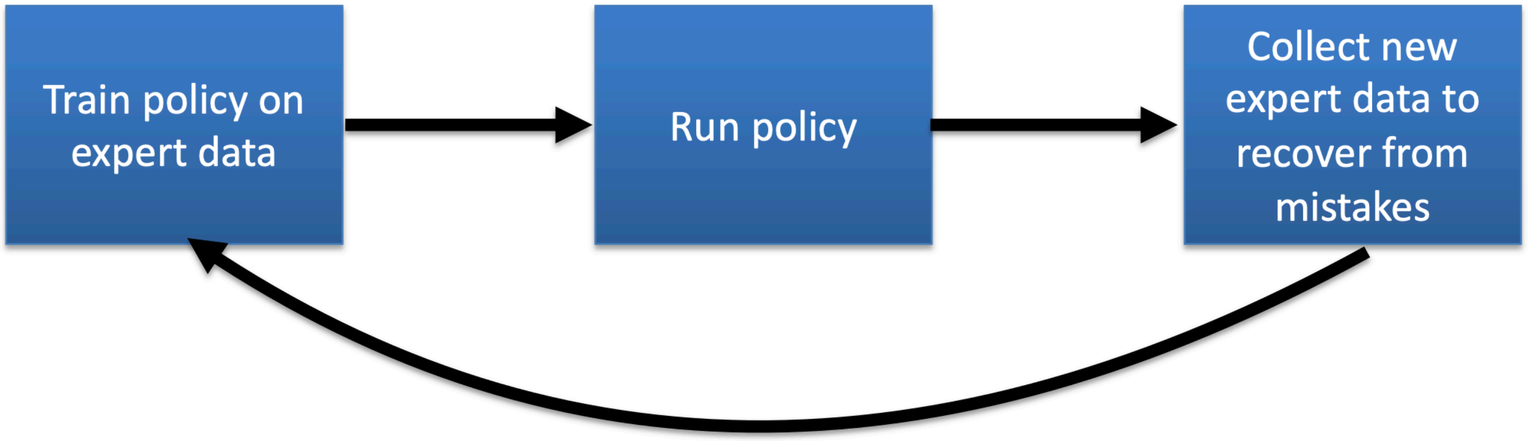
Key Idea: Interaction



Use interaction to collect data where learned policy goes

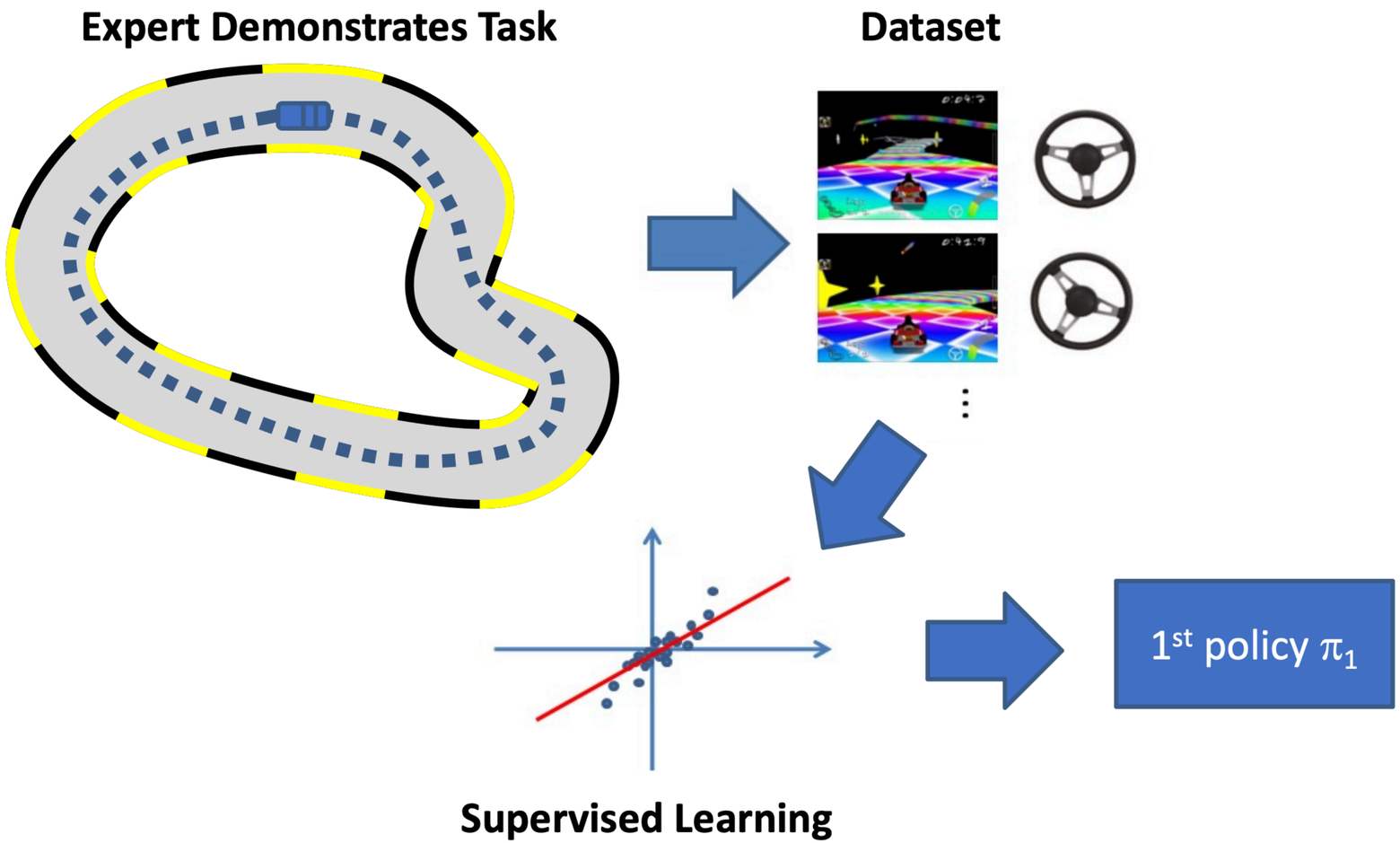
Key Idea: Interaction





Key Idea: Interaction

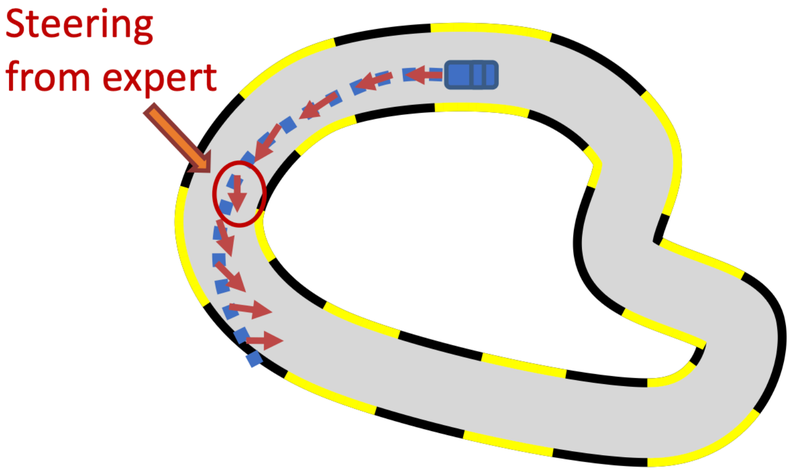
1st iteration



Key Idea: Interaction

2nd iteration

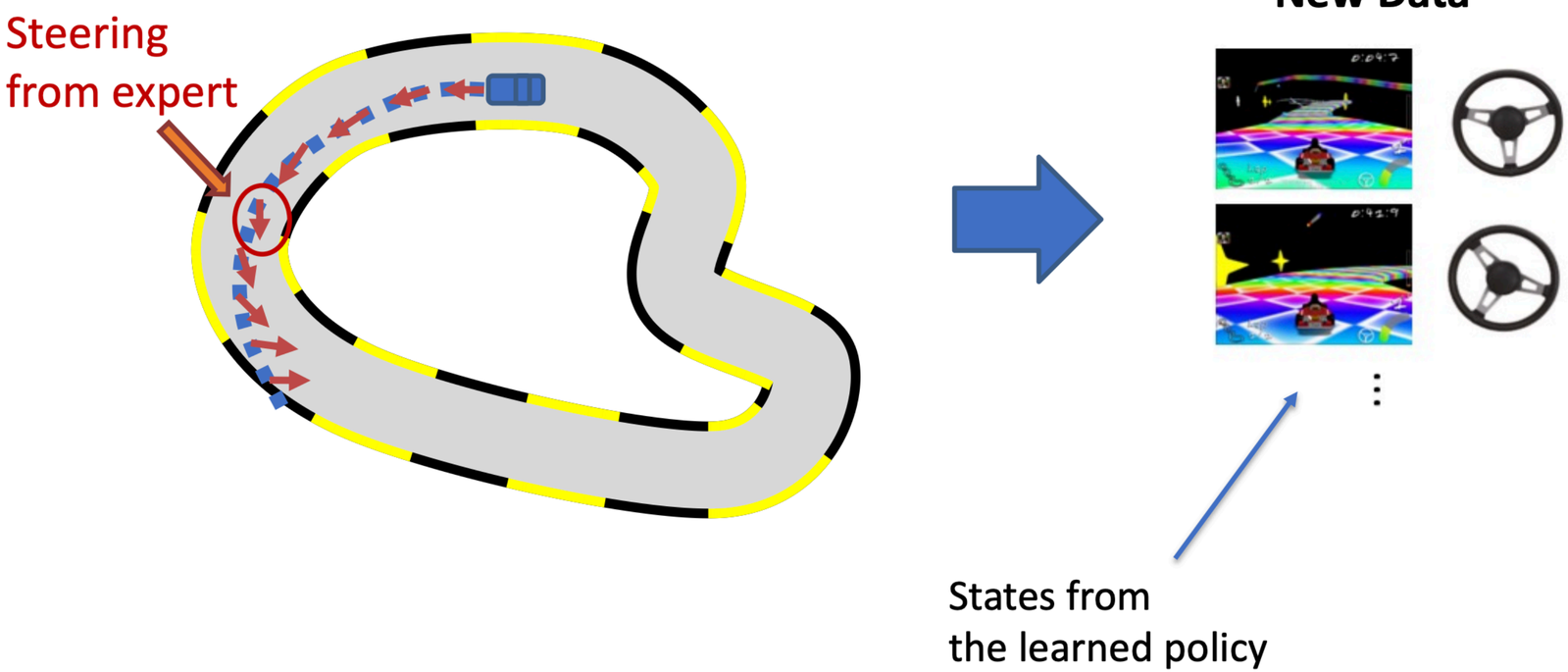
Execute π_1 and Query Expert



Key Idea: Interaction

2nd iteration

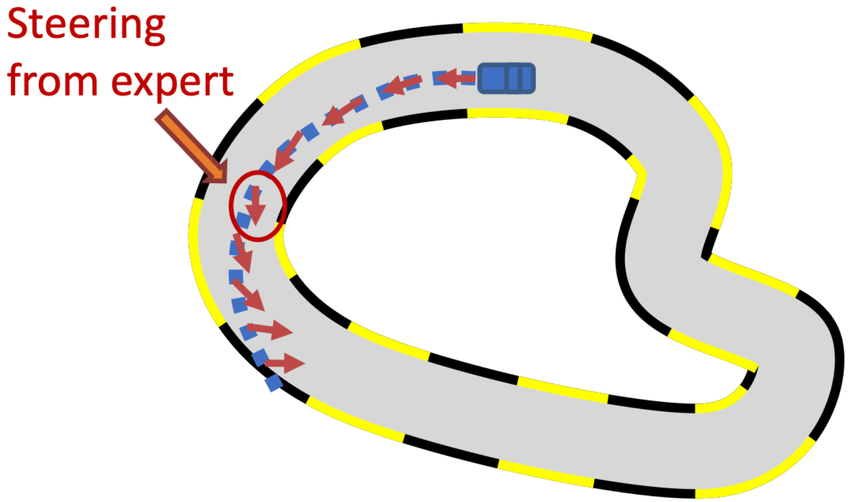
Execute π_1 and Query Expert



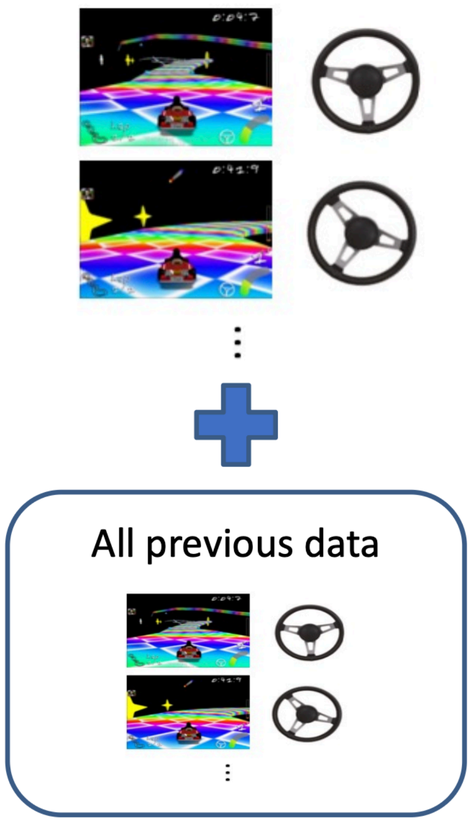
Key Idea: Interaction

2nd iteration

Execute π_1 and Query Expert

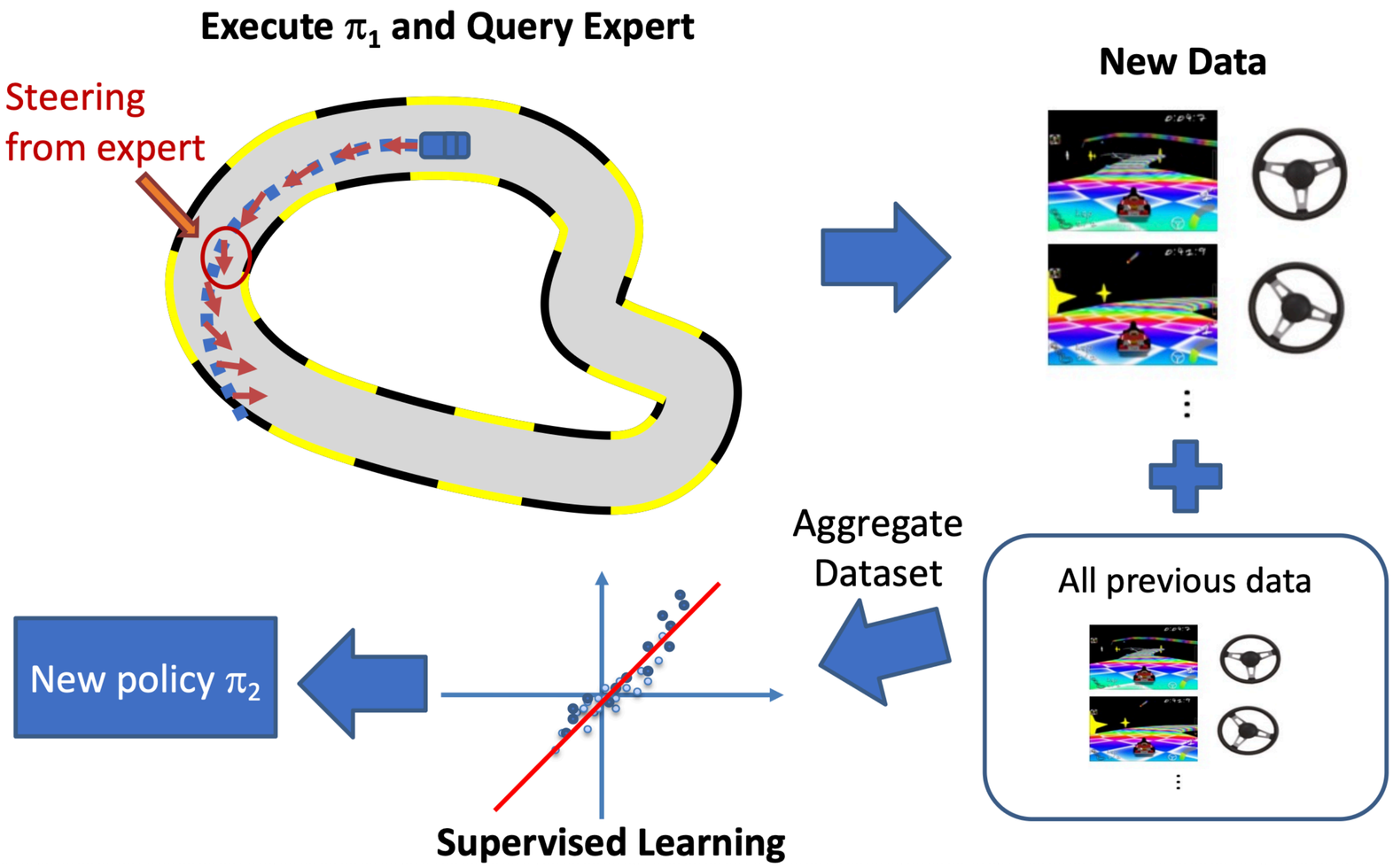


New Data



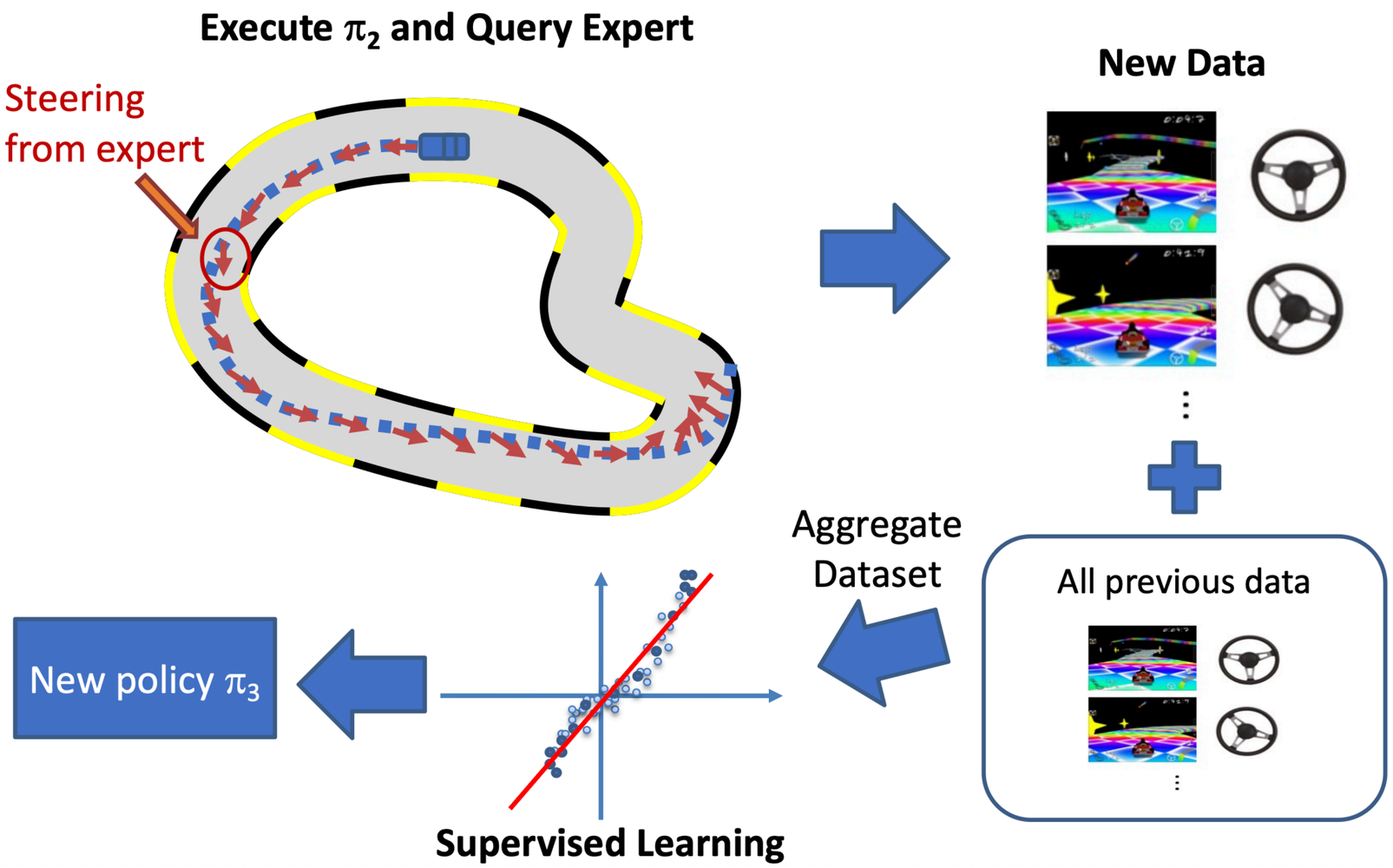
Key Idea: Interaction

2nd iteration



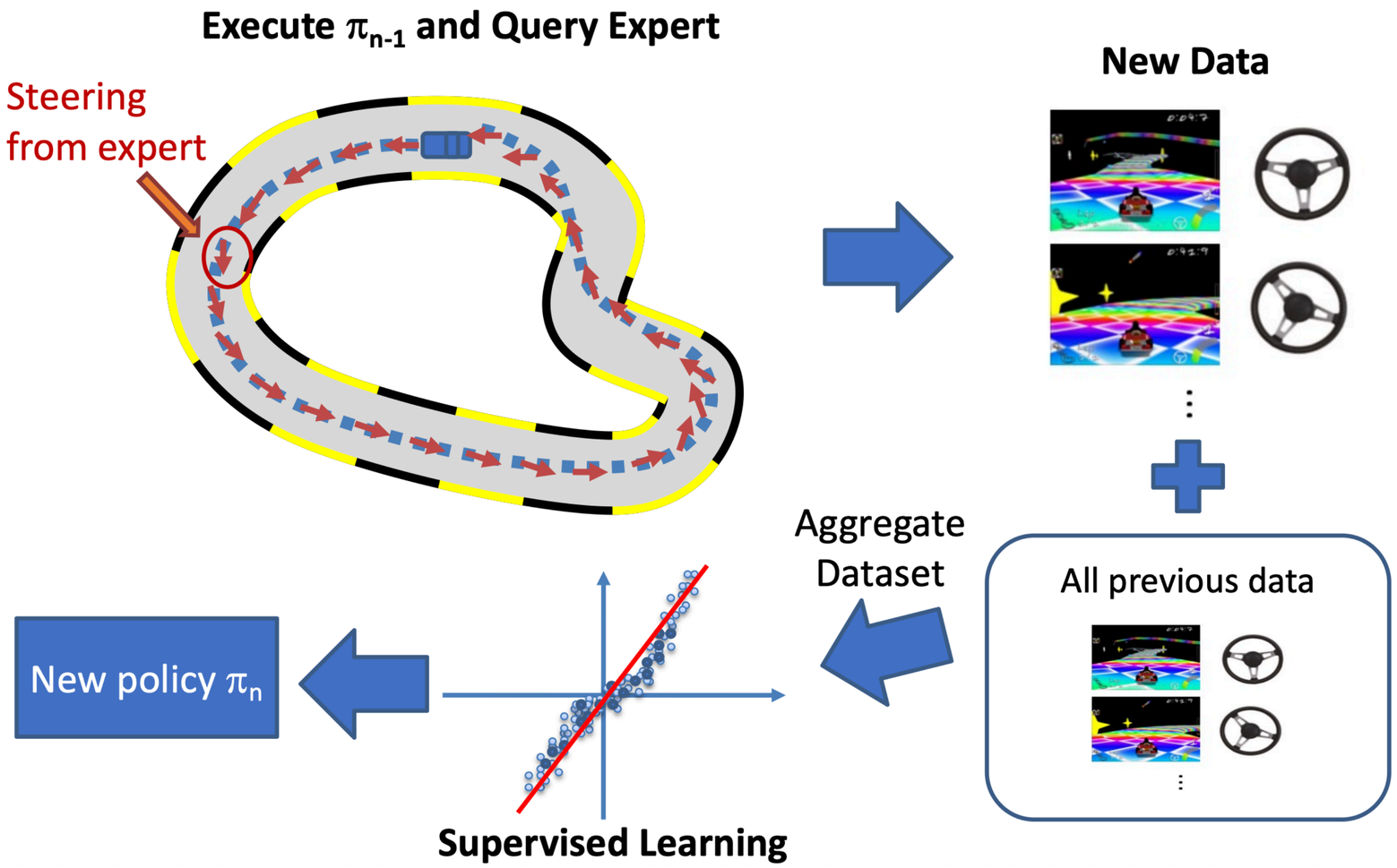
Key Idea: Interaction

3rd iteration



Key Idea: Interaction

n^{th} iteration



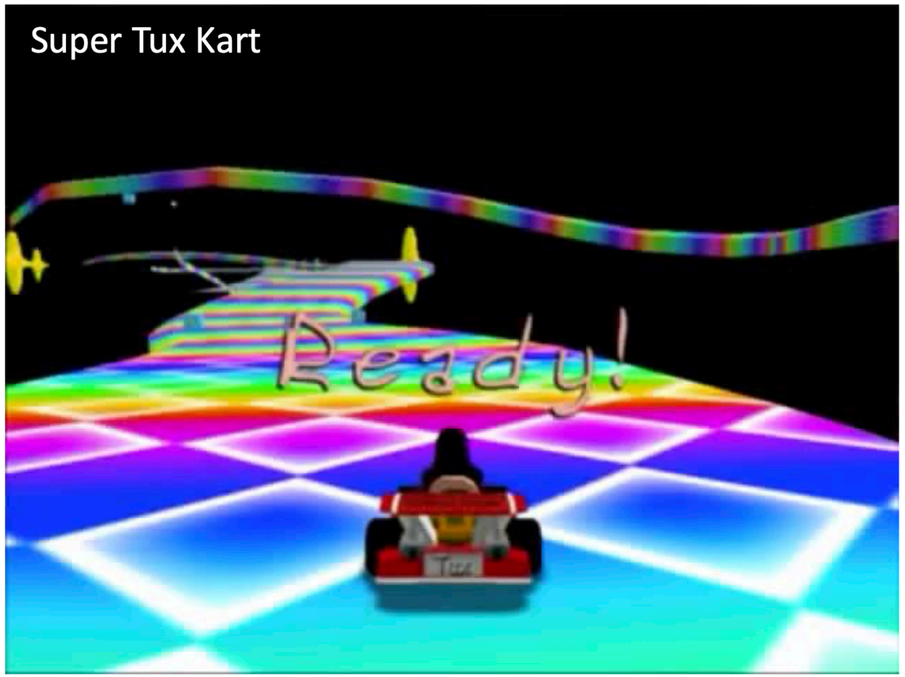
Key Idea: Interaction

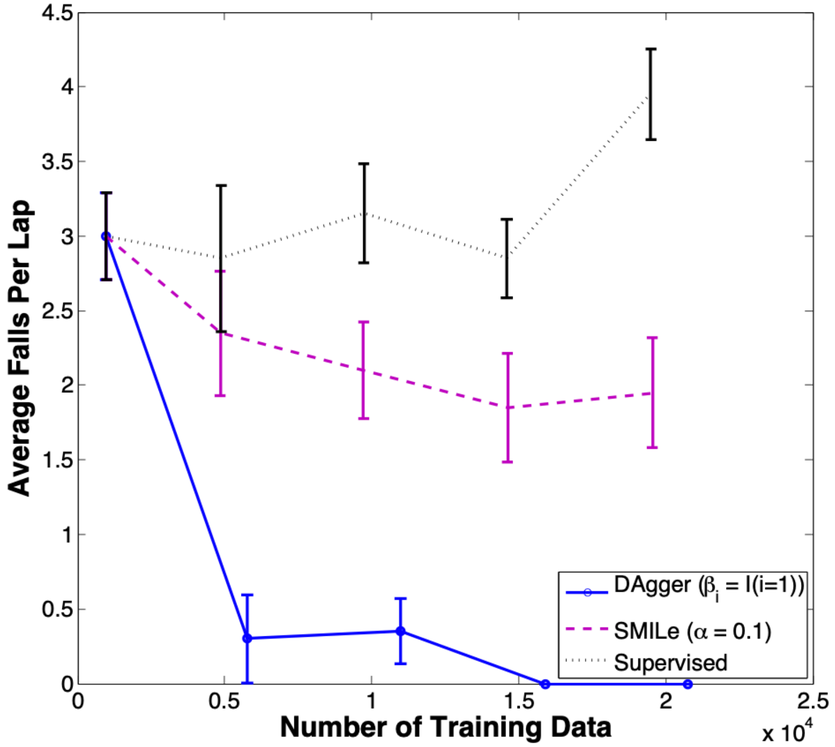
Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{\pi}_1$ to any policy in Π.
for $i = 1$ **to** N **do**
  Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$.
  Sample T-step trajectories using π_i.
  Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{D}_i$.
  Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$.
end for
Return best $\hat{\pi}_i$ on validation.

Algorithm 3.1: DAGGER Algorithm.

Does Dagger work?

- Yes!

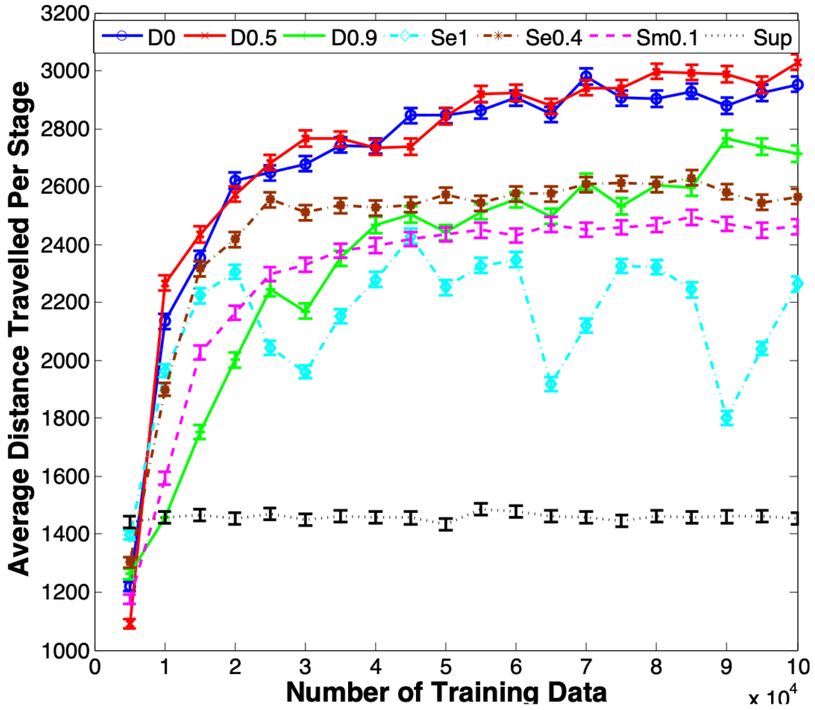




Does Dagger work?

- Yes!

Super Mario Bros



Summary of DAgger

- Essentially an online learning algorithm
  - With regret guarantees!

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

A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning

Stéphane Ross
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
stephaneross@cmu.edu

Geoffrey J. Gordon
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ggordon@cs.cmu.edu

J. Andrew Bagnell
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dbagnell@ri.cmu.edu

Problems with DAgger

- Shared control scheme can alter expert behavior
  - Human expert does not receive feedback
- β is difficult to tune
  - High beta reduces to behavior cloning
  - Low beta makes training unsafe and difficult for expert
- Expert demonstration is queried at random

Ensemble DAgger

- Novice is an ensemble of Neural Networks
- Decision rule considerations:
  - Discrepancy between novice and expert
  - Variance between networks in the ensemble (doubt)

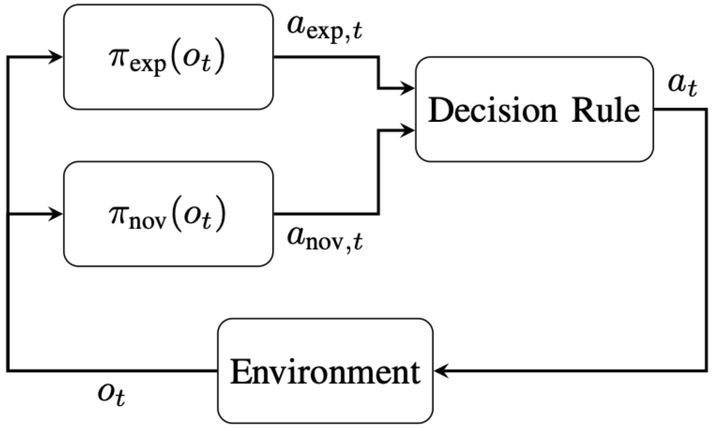


Fig. 2: Flowchart for DAgger variants, where the decision rule differs between approaches.

Algorithm 4 EnsembleDAgger Decision Rule

procedure DR(o_t, τ, χ)
  $\bar{a}_{\text{nov},t}, \sigma^2_{a_{\text{nov},t}} \leftarrow \pi_{\text{nov}}(o_t)$
  $a_{\text{exp},t} \leftarrow \pi_{\text{exp}}(o_t)$
  $\hat{\tau} \leftarrow \|\bar{a}_{\text{nov},t} - a_{\text{exp},t}\|^2$
  $\hat{\chi} \leftarrow \sigma^2_{a_{\text{nov},t}}$
  if $\hat{\tau} \leq \tau$ **and** $\hat{\chi} \leq \chi$
    return $\bar{a}_{\text{nov},t}$
  else
    return $a_{\text{exp},t}$

Problems with DAgger

- Shared control scheme can alter expert behavior
  - Human expert does not receive feedback
- ~~β is difficult to tune~~
  - ~~High beta reduces to behavior cloning~~
  - ~~Low beta makes training unsafe and difficult for expert~~
- ~~Expert demonstration is queried at random~~

Human-Gated (HG) Dagger

- Motivation: Human expert can provide better actions if given control for long stretches.
- Features
  - Expert is the gating function
  - Novice is represented by NN Ensemble
  - Computes doubt when expert takes control

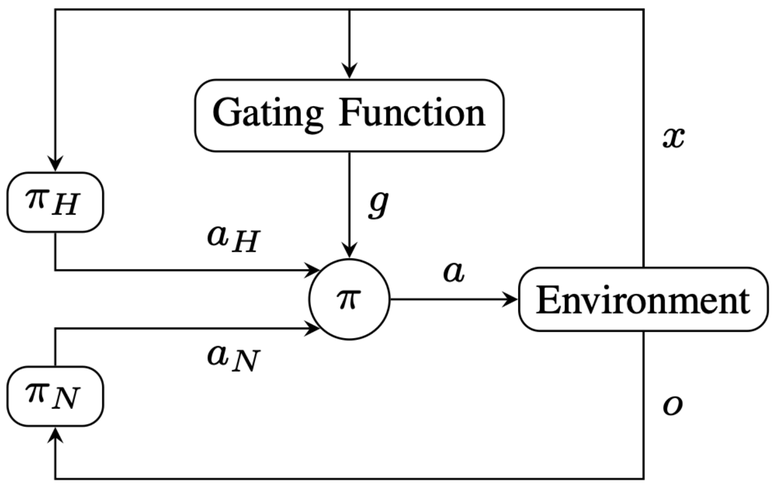


Fig. 1: Control loop for HG-DAGGER

Algorithm 1 HG-DAGGER

procedure HG-DAGGER($\pi_H, \pi_{N_1}, \mathcal{D}_{BC}$)
  $\mathcal{D} \leftarrow \mathcal{D}_{BC}$
  $\mathcal{I} \leftarrow []$
  for epoch $i = 1 : K$
    for rollout $j = 1 : M$
      for timestep $t \in T$ of rollout j
        if expert has control
          record expert labels into $\mathcal{D}_j$
        if expert is taking control
          record doubt into I_j
      $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_j$
      append $\mathcal{I}_j$ to $\mathcal{I}$
    train $\pi_{N_{i+1}}$ on $\mathcal{D}$
  $\tau \leftarrow f(\mathcal{I})$
  return $\pi_{N_{K+1}}, \tau$

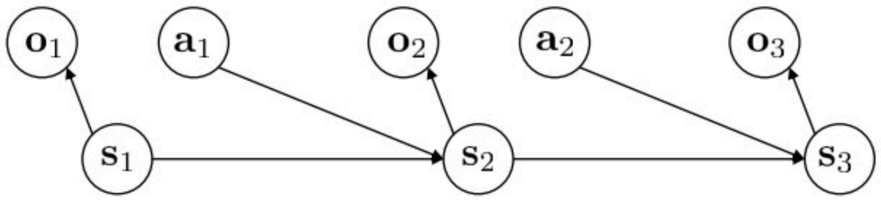
Tesla!

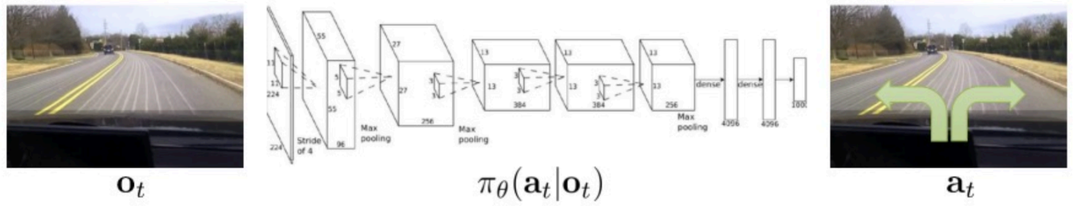
- Elon Musk quotes from 2019 interview with ARK Invest
  - *"We have just a vast amount of data on interventions. So, effectively, the customers are training the system on how to drive."*
  - *Every time somebody intervenes - takes over from Autopilot - it saves that information and uploads it to our system … And we're really starting to get quite good at not even requiring human labeling. Basically the person, say, drives the intersection and is thereby training training Autopilot what to do."*

How to further improve imitation learning with powerful models?

Use History

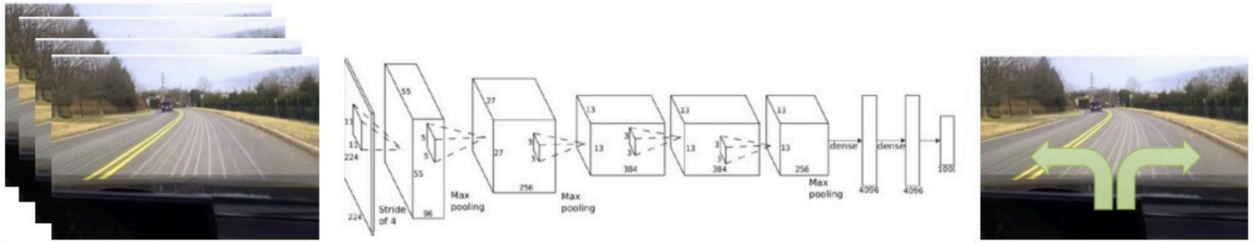
- Remember: o_t is the observation
- Idea: learn $\pi_\theta(a_t \mid o_1, \ldots, o_t)$
  - $\pi_\theta(a_t \mid o_t)$: If we see the same thing twice, we do the same thing twice, regardless of what happened before — not natural for human demonstrators!
- Question: does $\pi_\theta(a_t \mid s_1, \ldots, s_t)$ make sense?
  - From Markovian perspective: not necessarily
  - From imitation learning perspective: yes!



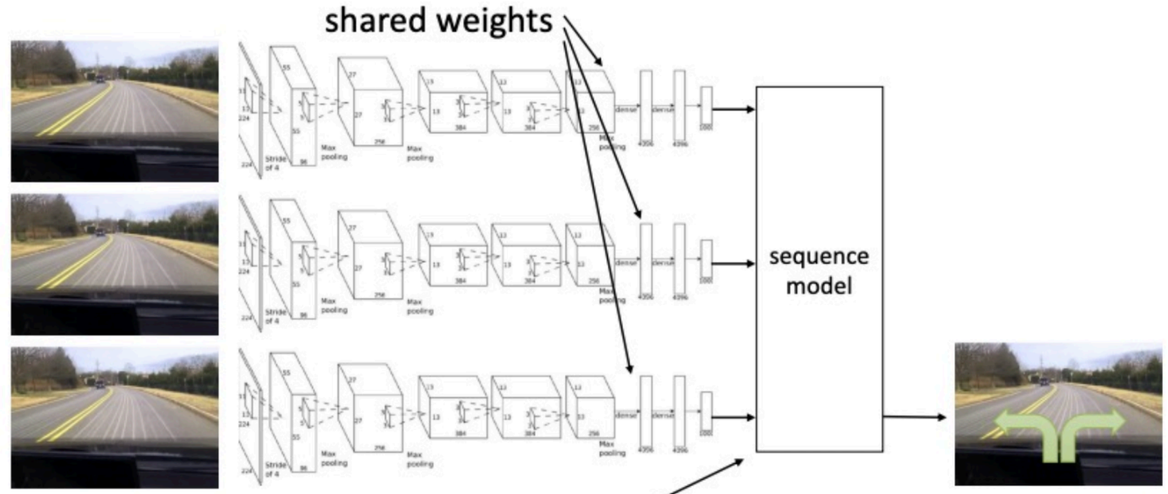


Use History

- How to represent $\pi_\theta(a_t \mid o_1, \ldots, o_t)$?
- "Fully connected"?
  - Variable number of frames & too many weights



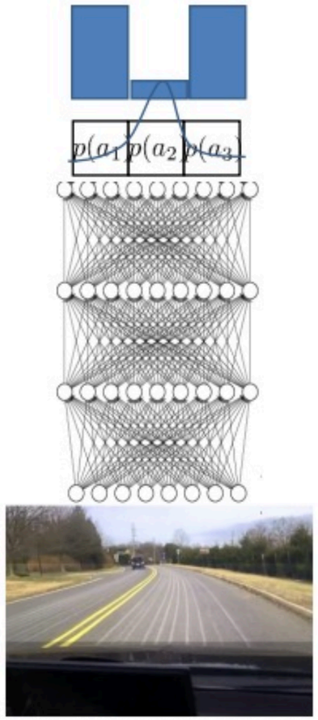
- Use sequential models

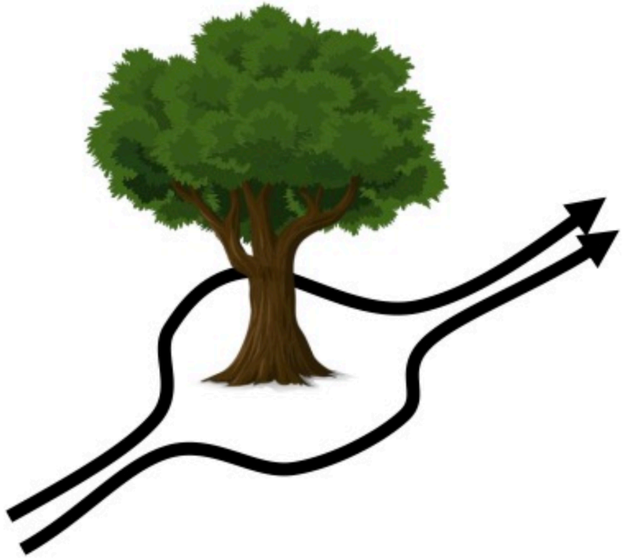


Can be done with Transformers, LSTM cells, etc.

Model Multi-Modal Behaviors

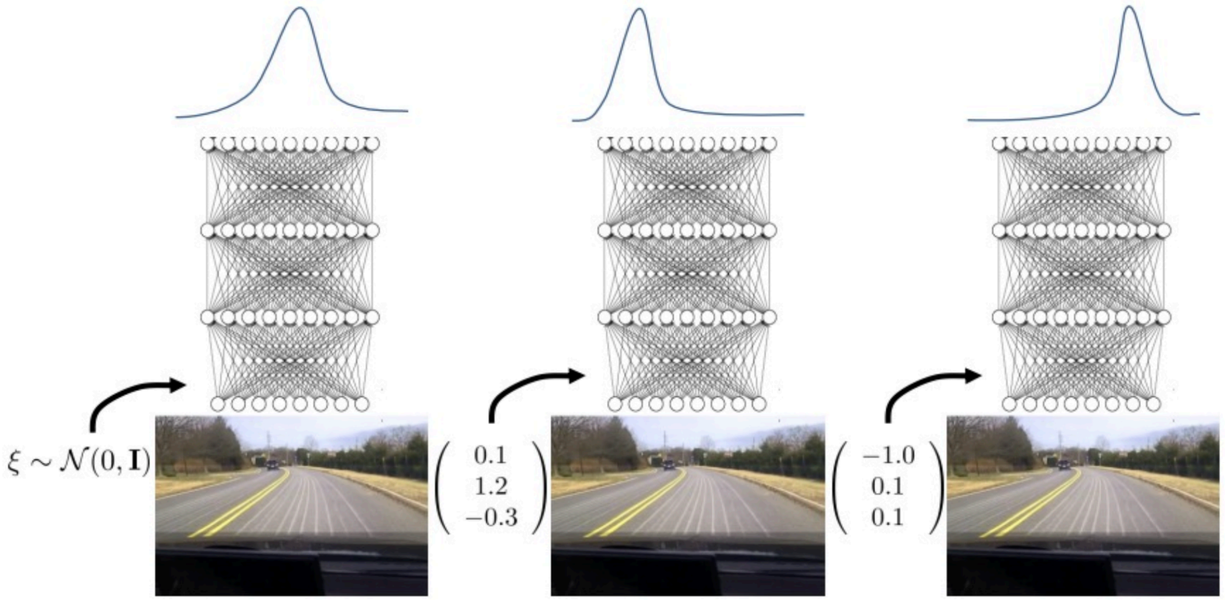
- Multi-modal behavior is common for many demonstrators
  - Human demonstrator
  - Optimal control
  - Oracle search algorithm

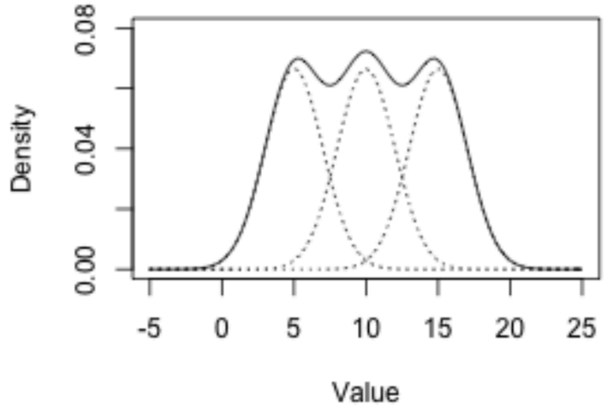




Model Multi-Modal Behaviors

- How to model multi-modal behavior?
  - Gaussian mixture model
  - Latent variable models: (conditional) VAE
  - Diffusion models
  - Other generative models (e.g. GAN)





Imitation Learning with Privileged Teachers

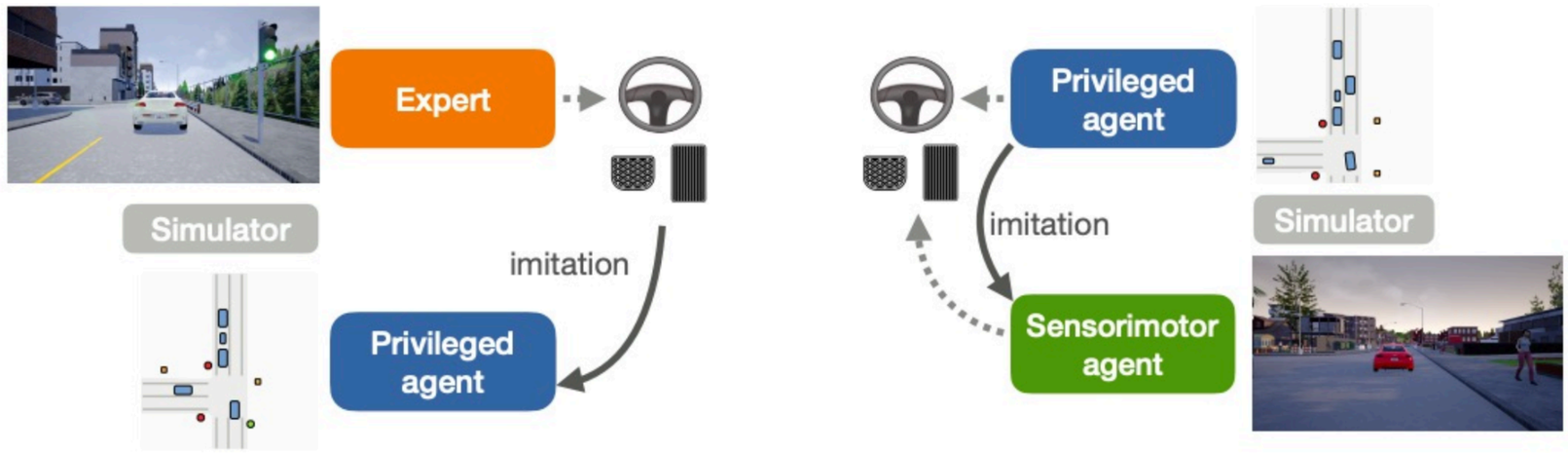
- It might be hard to directly learn $\pi_\theta(a_t | o_t)$
- o_t is high-dimensional!
- Idea: obtain a "privileged" teacher $\pi_p(a_t | p_t)$ first
  - p_t contains some "ground truth" information that is not directly available to students
  - Then use $\pi_p(a_t | p_t)$ to generate demonstrations for $\pi_\theta(a_t | o_t)$

Example of IL with Privileged Teachers: Driving

- Stage 1: Learn a "privileged agent" from expert
  - It knows ground truth state (traffic light, other vehicles' pos/vel, etc)
- Stage 2: a sensorimotor student learns from this train privileged agent

Learning by Cheating

Dian Chen UT Austin
Brady Zhou Intel Labs, UT Austin
Vladlen Koltun Intel Labs
Philipp Krähenbühl UT Austin



(a) Privileged agent imitates the expert (b) Sensorimotor agent imitates the privileged agent

Example of IL with Privileged Teachers: Driving

- Why it works (compared to direct IL)?
  - The trained "privileged agent" is white-box and can provide as much supervision as you want (e.g., running DAgger)
  - Optimization might be easier
  - Inherently offer some invariance (e.g., night/day driving, different vehicle colors)
  - Easy to do data augmentation

(a) Road map (b) Rotation and shift aug.

Example of IL with Privileged Teachers: Locomotion

- This privileged learning pipeline is even easier and more useful in simulation!

Learning quadrupedal locomotion over challenging terrain

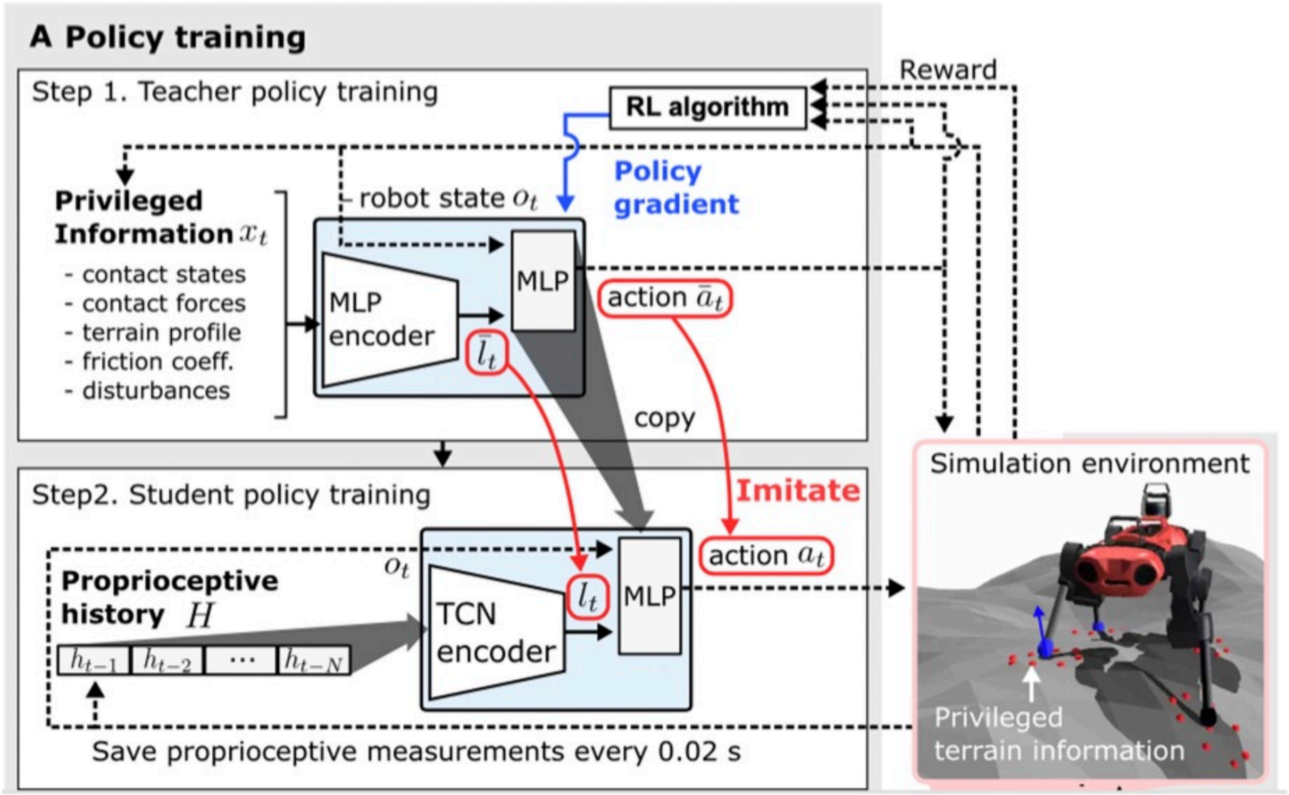
Joonho Lee[1]*, Jemin Hwangbo[1,2], Lorenz Wellhausen[1], Vladlen Koltun[3], Marco Hutter[1]

Example of IL with Privileged Teachers: Locomotion

- This privileged learning pipeline is even easier and more useful in simulation!

Example of IL with Privileged Teachers: Locomotion

- This privileged information contains almost everything available in sim (contact, terrain, fraction, disturbance, etc)
- Students only learn from proprioceptive history (IMU, joint angle, etc)
- Privileged teacher is trained using RL (PPO, different from the driving example)

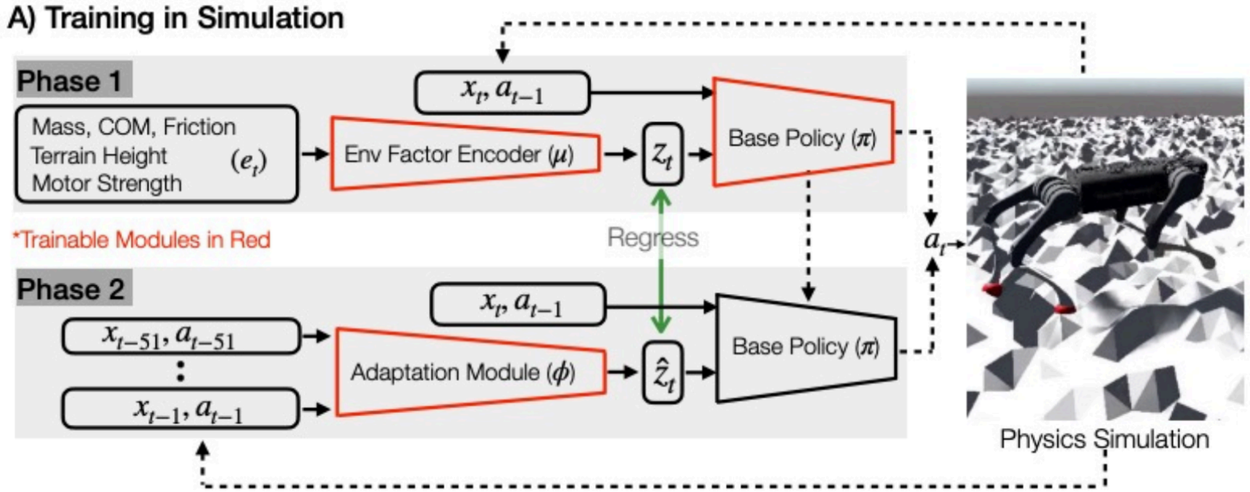


Variants: Student Learning Not in Action Space

- Student learning doesn't have to happen in the action space
  - RMA: in latent space

RMA: Rapid Motor Adaptation for Legged Robots

Ashish Kumar, UC Berkeley
Zipeng Fu, Carnegie Mellon University
Deepak Pathak, Carnegie Mellon University
Jitendra Malik, UC Berkeley, Facebook



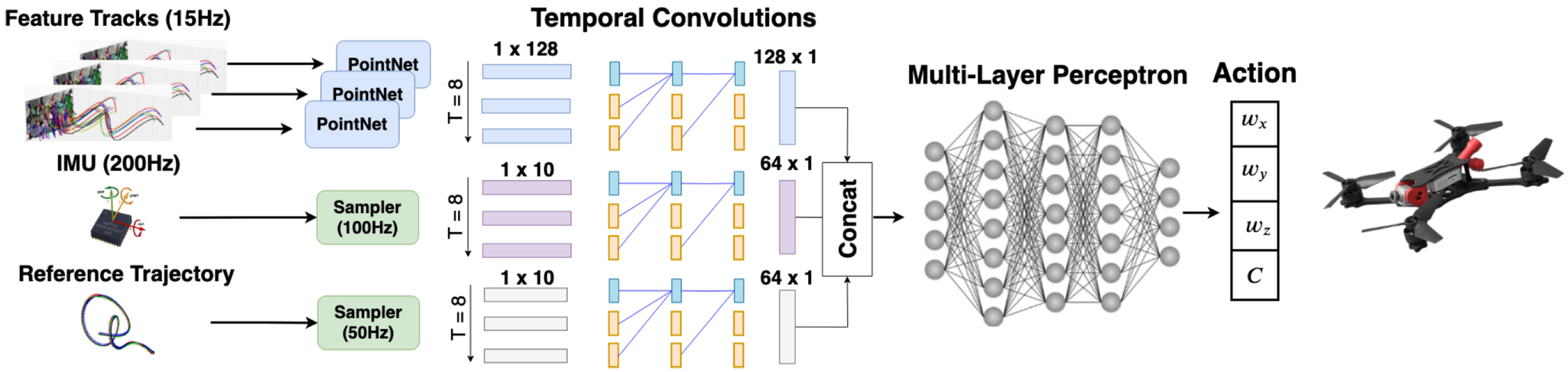
Example of IL with Privileged Teachers: Drone

Deep Drone Acrobatics

Elia Kaufmann*‡, Antonio Loquercio*‡, René Ranftl†, Matthias Müller†, Vladlen Koltun†, Davide Scaramuzza‡

Example of IL with Privileged Teachers: Drone

- Privileged teacher is an MPC controller
  - Knowing state



Deep imitation learning via generative models

Generative Models?

- What are generative models?
  - Learning: learn a distribution p_θ that "matches" p_{data}
  - Sampling: Generate novel data, $x_{\text{new}} \sim p_\theta$
- Almost all types of generative models can be integrated with IL
- p_{data} is from expert: can be action level, trajectory level, …
- Often conditional
- https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

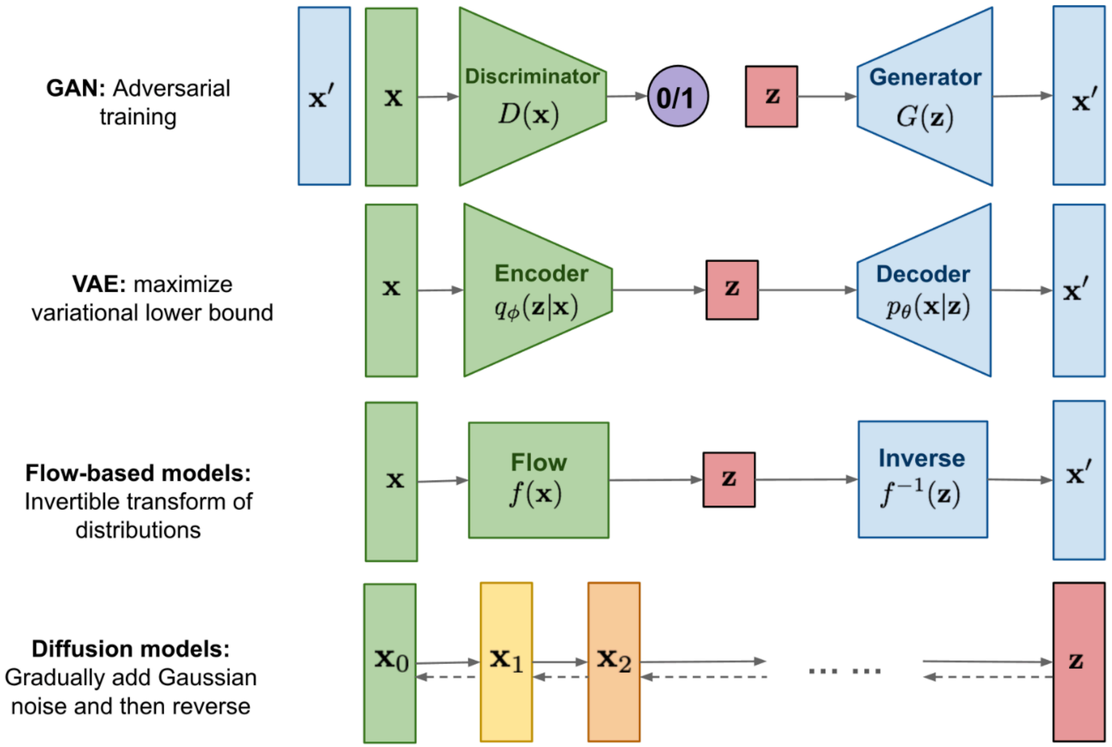


Figure 1: Overview of different types of generative models.

$$\mathbf{x}^{(j)} \sim p_{\text{data}}$$
$$j = 1, 2, \ldots, |\mathcal{D}|$$

GAN + Imitation Learning

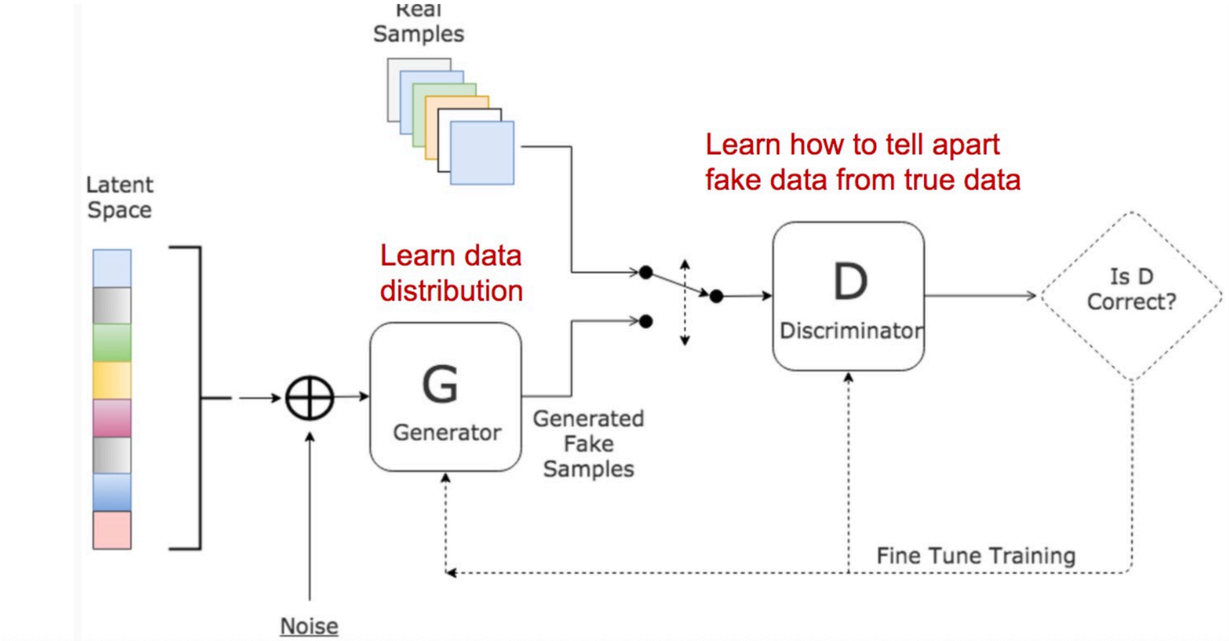
- Generative Adversarial IL (GAIL)
- GAN
- GAIL: $x = (s, a)$

Generative Adversarial Imitation Learning

Jonathan Ho
OpenAI
hoj@openai.com

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$



GAN + Imitation Learning

- GAIL's main idea (repeat the following three steps):
  - Sample trajectory from students
  - Update discriminator, which aims to classify teacher and student
  - Train student policy which aims to minimize the discriminator's accuracy
- Question: can GAIL model multi-modal behavior?

Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \tag{17}$$

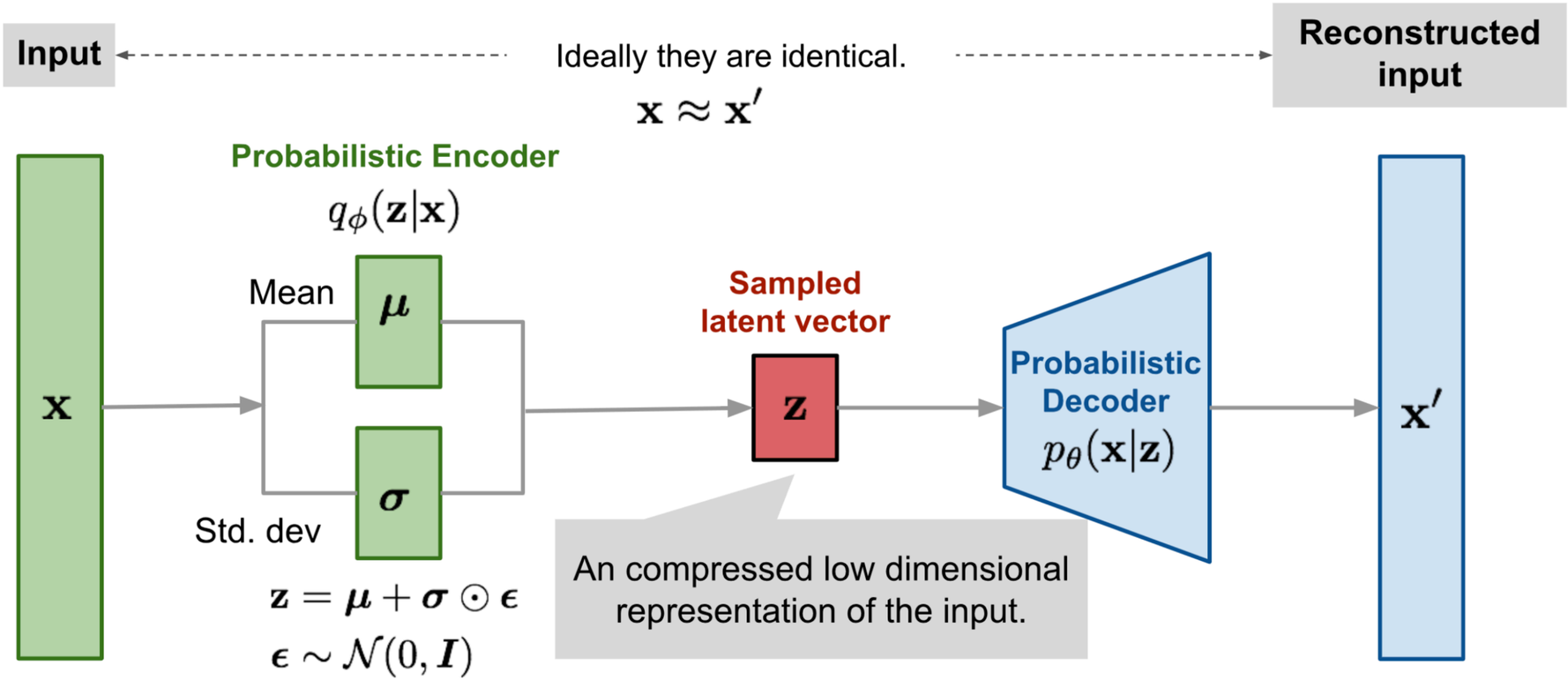
5: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

$$\begin{aligned} &\hat{\mathbb{E}}_{\tau_i}\left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right] - \lambda \nabla_\theta H(\pi_\theta), \\ &\text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \end{aligned} \tag{18}$$

6: **end for**

VAE + Imitation Learning

- Variational Autoencoder (VAE)
  - Learn a reconstruction model that has structured and smooth latent distribution.

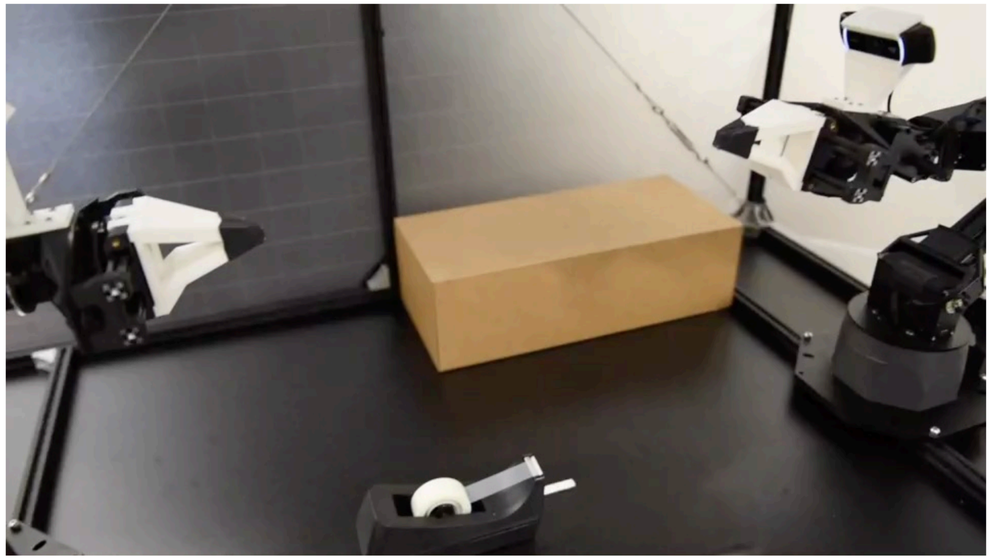


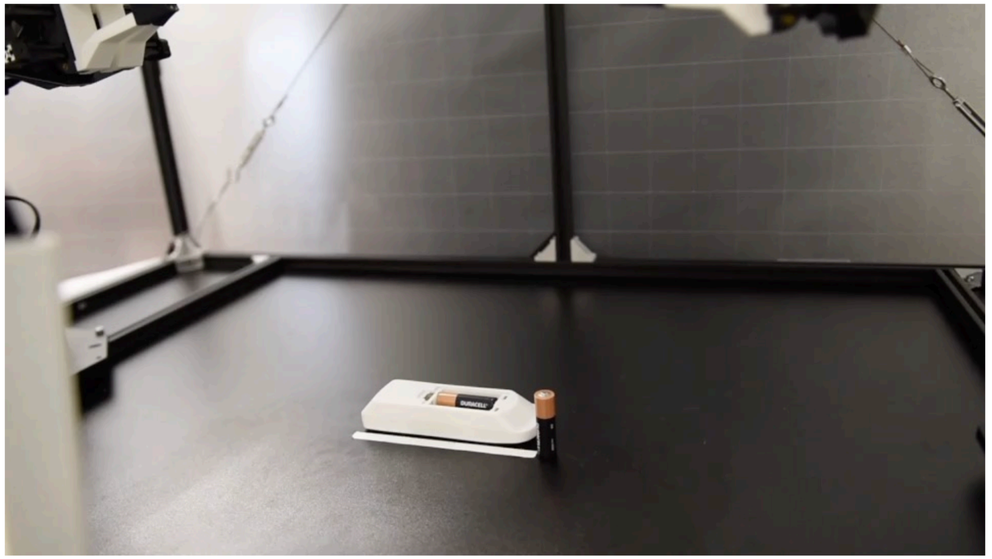
VAE + Imitation Learning

- Action Chunking with Transformers (ACT)

Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware

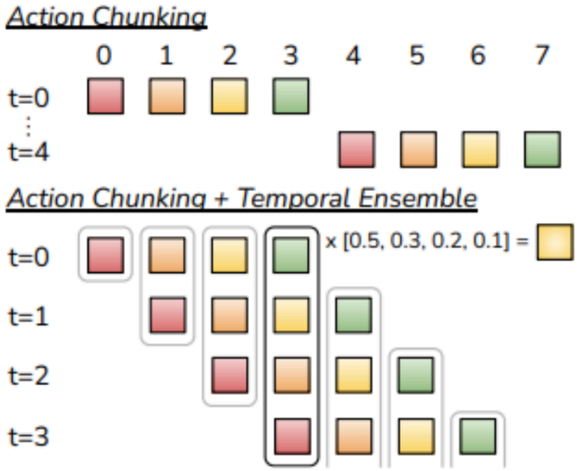
Tony Zhao Vikash Kumar Sergey Levine Chelsea Finn

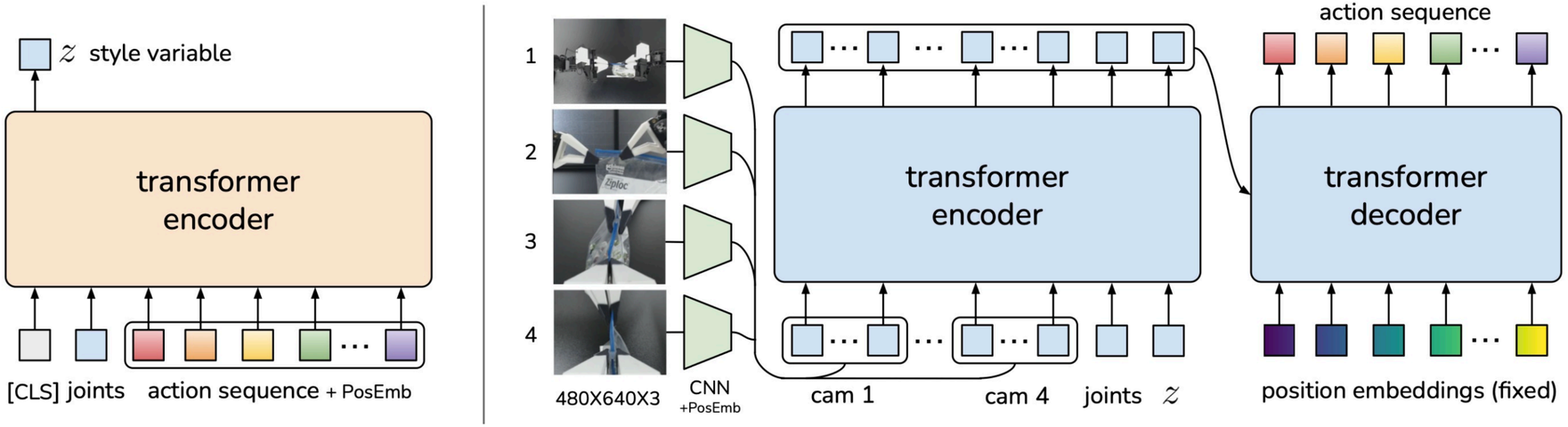




VAE + Imitation Learning

- Action Chunking with Transformers (ACT)
  - Based on CAVE (conditional VAE)
  - Encoder: expert action sequence + observation -> latent
  - Decoder: latent + more observation -> action sequence prediction
  - Key: action chunking + temporal ensemble

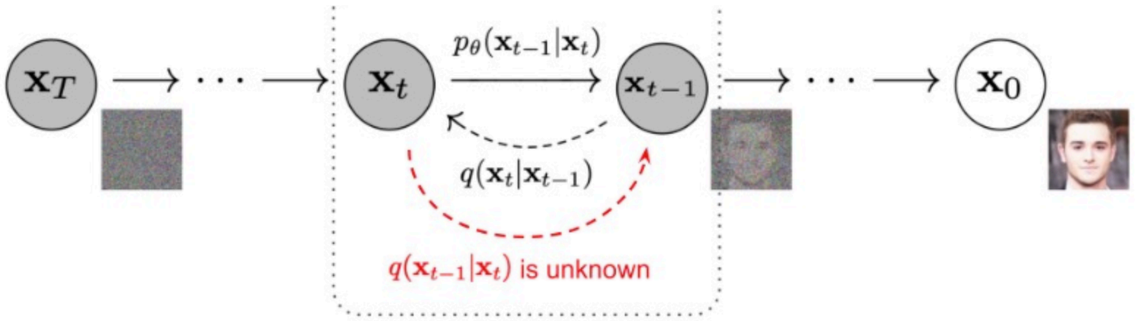




Diffusion Model + Imitation Learning

- What are diffusion models?
- Forward diffusion process: sample a real data x_0, and gradually corrupt it by iid Gaussians.

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$



- Reverse diffusion process: learn Gaussian $p_\theta(x_{t-1} \mid x_t)$ and then recreate x_0 from x_T

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

Diffusion Model + Imitation Learning

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

- Connections with stochastic gradient Langevin dynamics
  - Concept from physics. It produces samples from $p(x)$ using only its log gradient:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \frac{\delta}{2}\underbrace{\nabla_{\mathbf{x}} \log p(\mathbf{x}_{t-1})}_{\text{score}} + \sqrt{\delta}\boldsymbol{\epsilon}_t, \quad \text{where } \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

  - When $t \to \infty$, $\delta \to 0$, the distribution of x_t equals to the true $p(x)$

Diffusion Model + Imitation Learning

IMITATING HUMAN BEHAVIOUR WITH DIFFUSION MODELS

Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, Sam Devlin
Microsoft Research

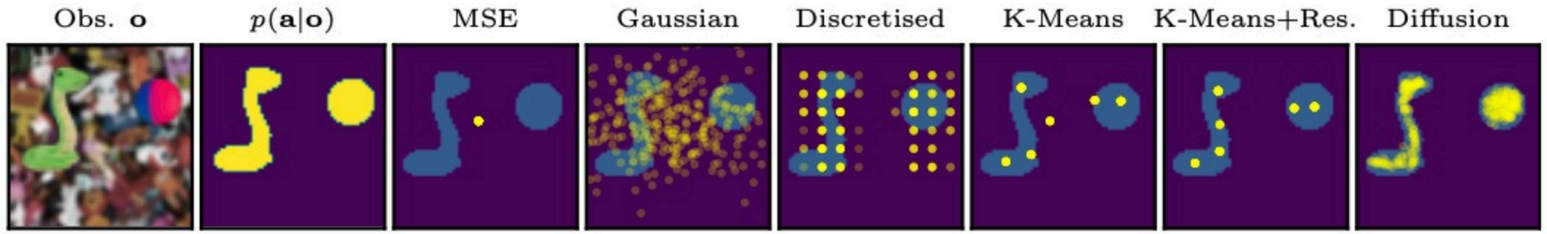
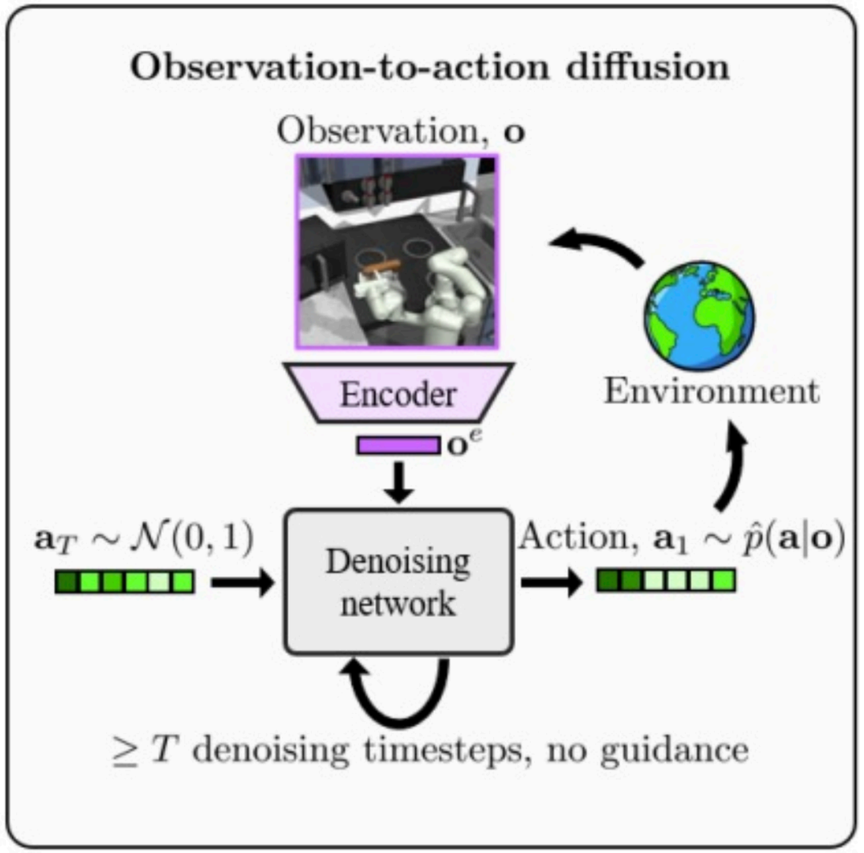


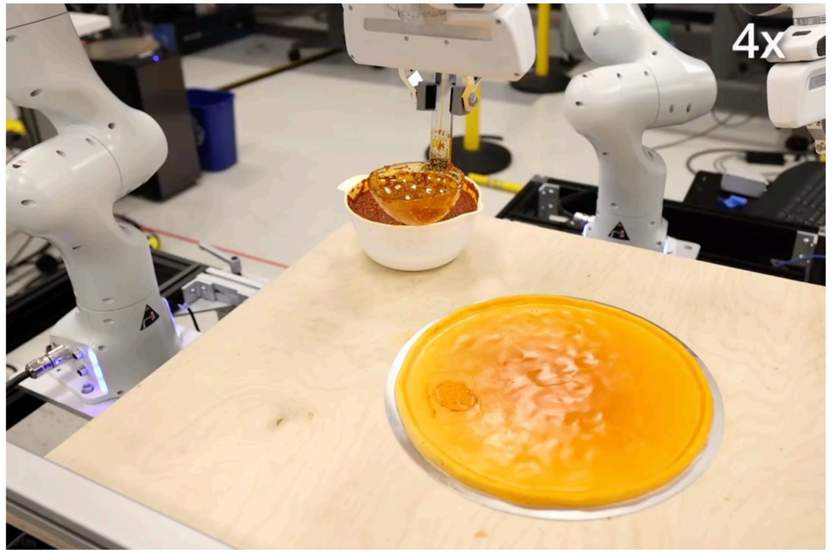
Figure 1: Expressiveness of a variety of models for behaviour cloning in a single-step, arcade claw game with two simultaneous, continuous actions. Existing methods fail to model the full action distribution, $p(\mathbf{a}|\mathbf{o})$, whilst diffusion models excel at covering multimodal & complex distributions.

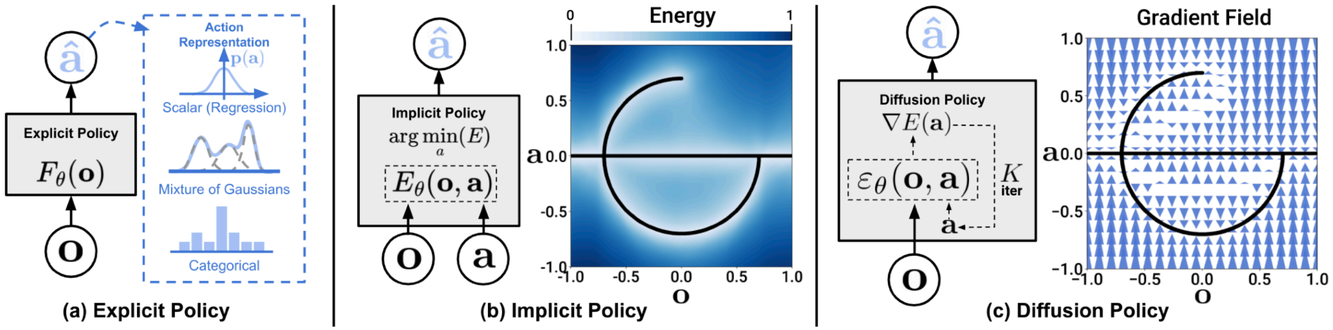


Diffusion Model + Imitation Learning

Diffusion Policy:
Visuomotor Policy Learning via Action Diffusion

Cheng Chi[1], Siyuan Feng[2], Yilun Du[3], Zhenjia Xu[1], Eric Cousineau[2], Benjamin Burchfiel[2], Shuran Song[1]
[1] Columbia University [2] Toyota Research Institute [3] MIT
https://diffusion-policy.cs.columbia.edu



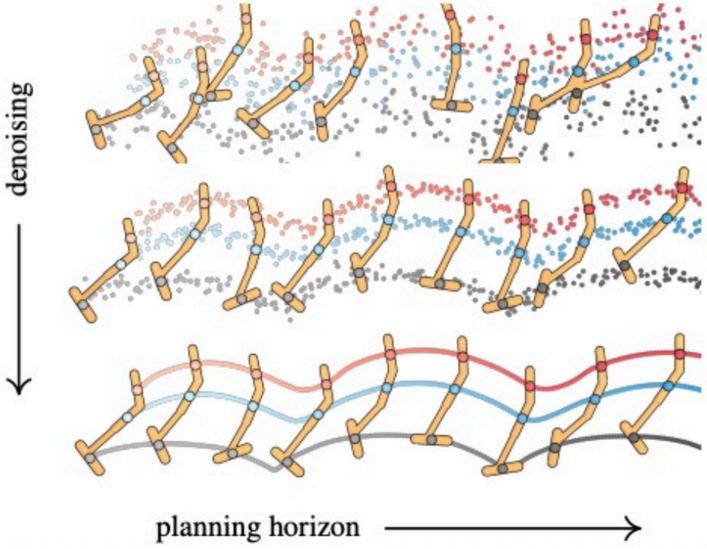


Diffusion Model + Imitation Learning

- Diffuser (trajectory-level diffusion)

Algorithm 1 Guided Diffusion Planning

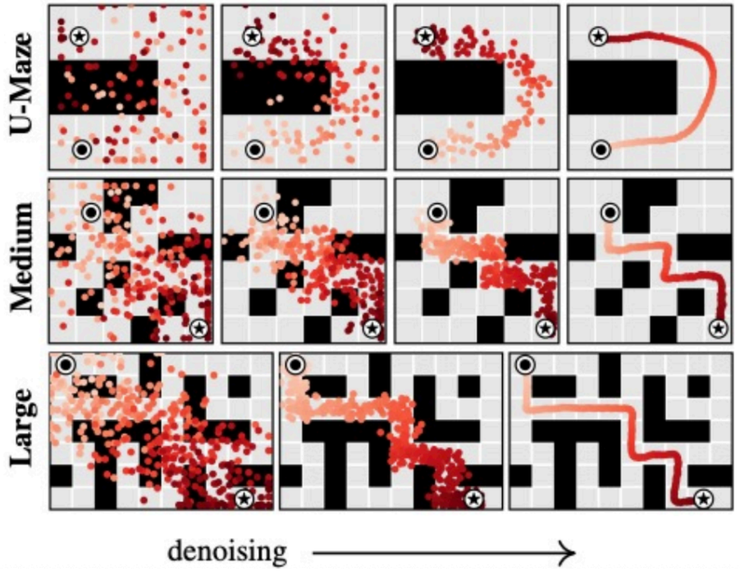
Require Diffuser μ_θ, guide $\mathcal{J}$, scale α, covariances Σ^i
while not done **do**
  Observe state $\mathbf{s}$; initialize plan $\boldsymbol{\tau}^N \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  for $i = N, \ldots, 1$ **do**
    // parameters of reverse transition
    $\mu \leftarrow \mu_\theta(\boldsymbol{\tau}^i)$
    // guide using gradients of return
    $\boldsymbol{\tau}^{i-1} \sim \mathcal{N}(\mu + \alpha\Sigma\nabla\mathcal{J}(\mu), \Sigma^i)$
    // constrain first state of plan
    $\boldsymbol{\tau}^{i-1}_{\mathbf{s}_0} \leftarrow \boldsymbol{s}$
  Execute first action of plan $\boldsymbol{\tau}^0_{\mathbf{a}_0}$



Planning with Diffusion for Flexible Behavior Synthesis

Michael Janner [*1] Yilun Du [*2] Joshua B. Tenenbaum [2] Sergey Levine [1]

$$\boldsymbol{\tau} = \begin{bmatrix} \mathbf{s}_0 & \mathbf{s}_1 & \cdots & \mathbf{s}_T \\ \mathbf{a}_0 & \mathbf{a}_1 & & \mathbf{a}_T \end{bmatrix}$$



Summary: Generative Models + IL

- IL can be integrated with various generative models:
  - GAN
  - VAE
  - Diffusion models
- The learned distribution could be
  - Action level (conditional on observation) $p(a_t | o_t)$
  - State-action (or observation-action) pair $p(s_t, a_t)$
  - A sequence of actions
  - A full state-action trajectory

Acknowledgement

- Many contents are taken from UT Austin CS391, CMU 16-831