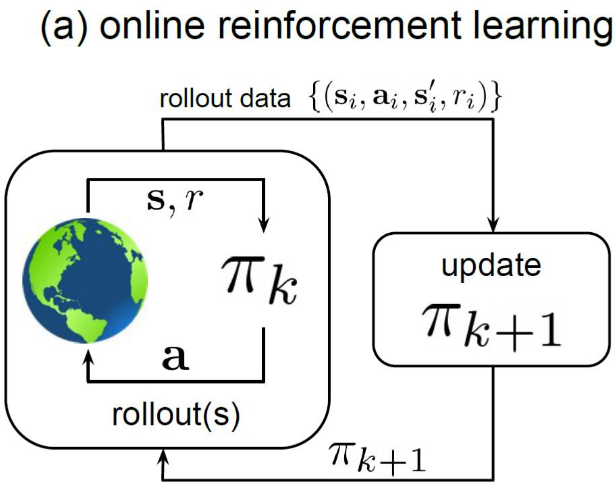
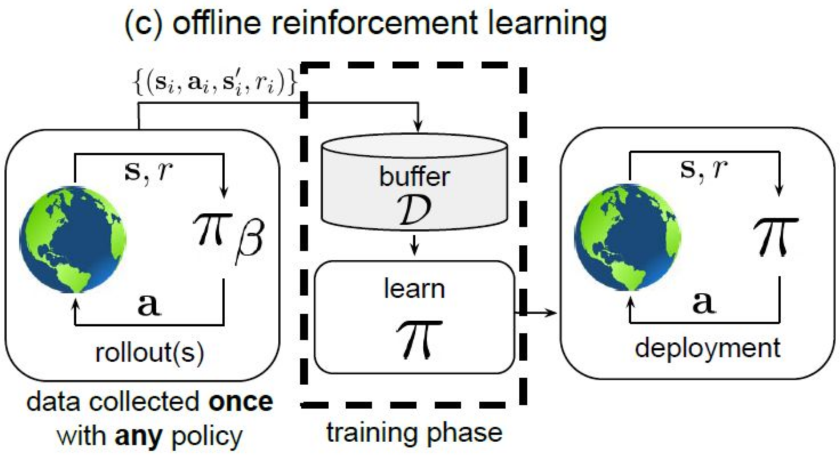
# Offline Reinforcement Learning using Models

Michael Gimelfarb

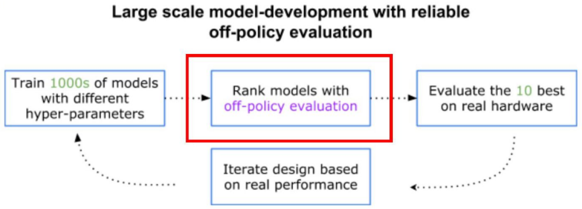
# What is Offline RL?

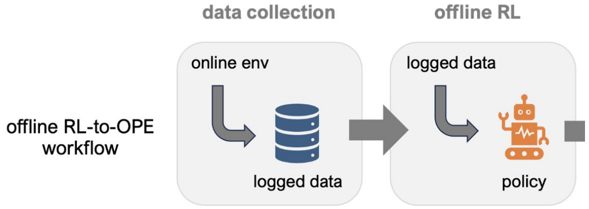


# Motivation

- **Evaluation**: Use logged data to evaluate & select the best policy obtained from a training procedure
- **Improvement**: Use logged data to learn a policy that performs better than the policy that collected the data
- **Challenges**:
    - Direct evaluation in the real world is also costly
    - Real world data is costly to obtain
    - Real world data can be limited in scale or scope

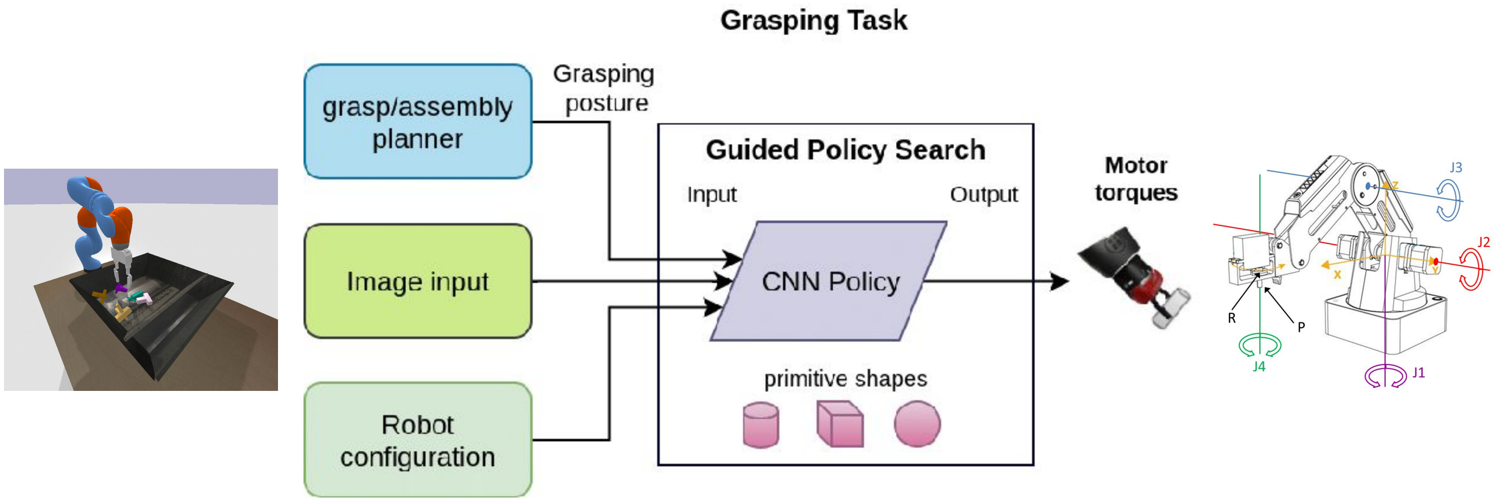


# When (and Where) to Trust the Model in Offline Policy Optimization

# A Brief Review of RL

- **Policy**: mapping from state/observation to action



# A Brief Review of RL

- **Model**: describes the dynamics of the system *P* and the immediate rewards *r*


**Markov Property**

$$P(s_{t+1} = s' \mid s_t, a_t, s_{t-1}, a_{t-1}, \dots s_1, a_1) = P(s_{t+1} = s' \mid s_t, a_t)$$

# A Brief Review of RL

- **Value**: the expected (discounted) sum of all future rewards following a given policy

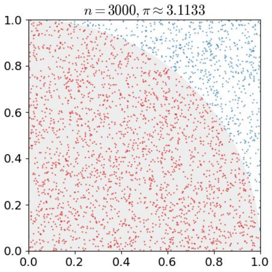
$$Q^{\pi}(s,a) = \mathbb{E}[r_1 + \gamma r_2 + \cdots + \gamma^{t-1} r_t + \ldots \mid s_1 = s,\ a_1 = a]$$

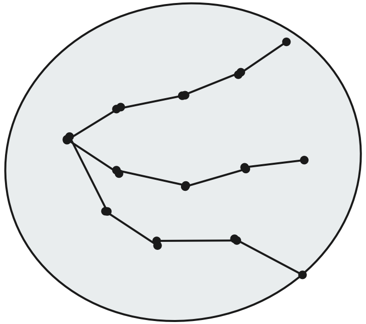
# A Brief Review of RL

- **Value**: the expected (discounted) sum of all future rewards following a given policy

$$Q^\pi(s,a) = \mathbb{E}[r_1 + \gamma r_2 + \cdots + \gamma^{t-1} r_t + \ldots \mid s_1 = s,\ a_1 = a]$$

- **Monte-Carlo policy evaluation:**



$$r_1^{(1)} + \gamma r_2^{(1)} + \cdots + \gamma^{T-1} r_T^{(1)}$$

$$r_1^{(2)} + \gamma r_2^{(2)} + \cdots + \gamma^{T-1} r_T^{(2)}$$

$$r_1^{(3)} + \gamma r_2^{(3)} + \cdots + \gamma^{T-1} r_T^{(3)}$$

# A Brief Review of RL

- **Optimal Policy:** the goal of RL is to find the policy that has the highest Q-value from some initial s, a

$$\max_\pi Q^\pi(s,a)$$

# A Brief Review of RL

- **Optimal Policy:** the goal of RL is to find the policy that has the highest Q-value from some initial s, a

$$\max_{\pi} Q^{\pi}(s, a)$$

- **Greedy policy improvement:** basis of most RL algorithms

$$\pi'(s) = \text{argmax}_a Q^{\pi}(s, a)$$

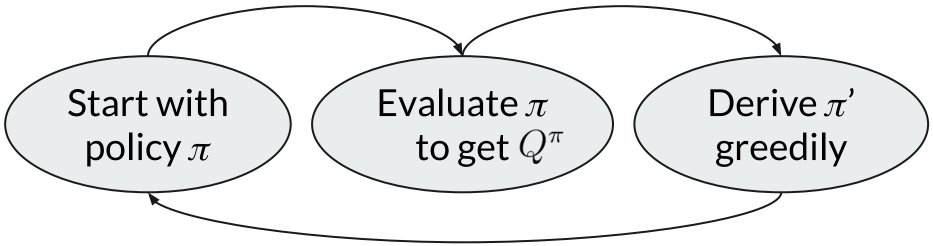
# A Brief Review of RL

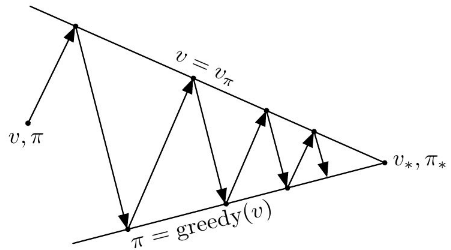
- **Optimal Policy:** the goal of RL is to find the policy that has the highest Q-value from some initial s, a

$$\max_\pi Q^\pi(s,a)$$

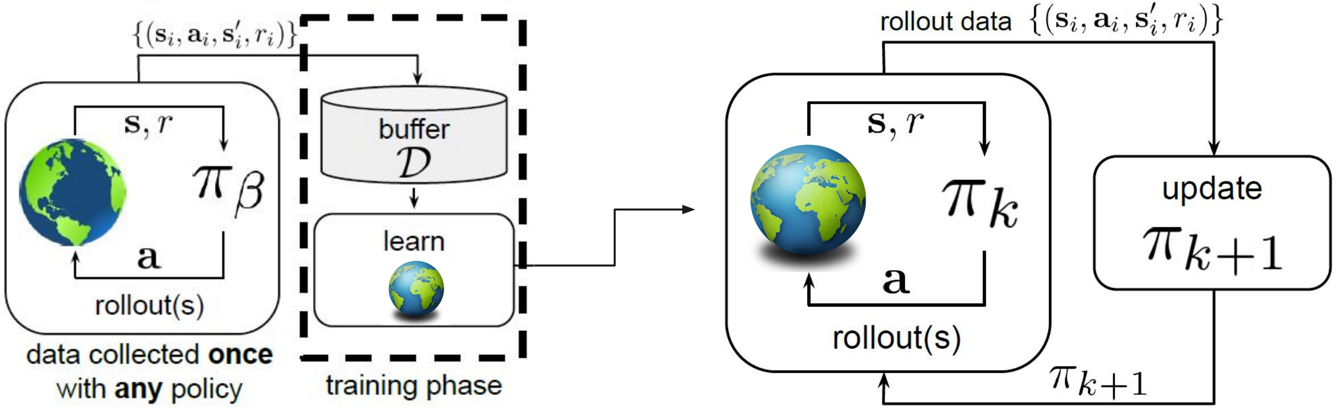
- **Greedy policy improvement:** basis of most RL algorithms

$$\pi'(s) = \text{argmax}_a Q^\pi(s,a)$$



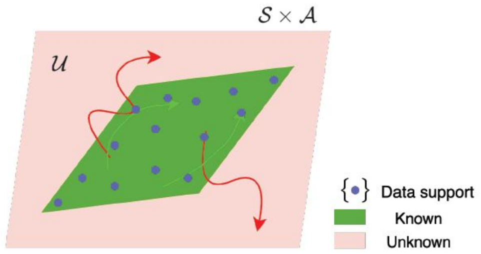



# Model-Based Offline Policy Optimization



# Model-Based Offline Policy Optimization

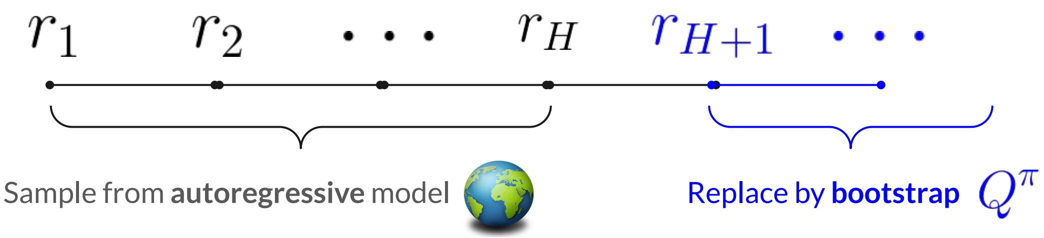
- Challenges of model-based offline RL:
    - Impossible to learn a **globally accurate model**
    - **Compounding errors** for long horizon
    - Policy **distribution shift**
- Learn both an autoregressive **<u>dynamics model and a value function</u>**:
    - How to get the **"best of both worlds"**?
    - Can we rely on the model **only where accurate**?
    - How to do this **automatically** without complicated validation?



# Model-Based Offline Policy Optimization

- Replace Monte-Carlo with **(n-step) bootstrapping**

$$
\begin{aligned}
Q^\pi(s,a) &= \mathbb{E}[r_1 + \gamma r_2 + \cdots + \gamma^{H-1} r_H + \gamma^H (r_{H+1} + \gamma r_{H+2} + \dots)] \\
&= \mathbb{E}[r_1 + \gamma r_2 + \cdots + \gamma^{H-1} r_H + \gamma^H Q^\pi(s',a')]
\end{aligned}
$$



# Model-Based Offline Policy Optimization

- Produce an **ensemble** of estimators:

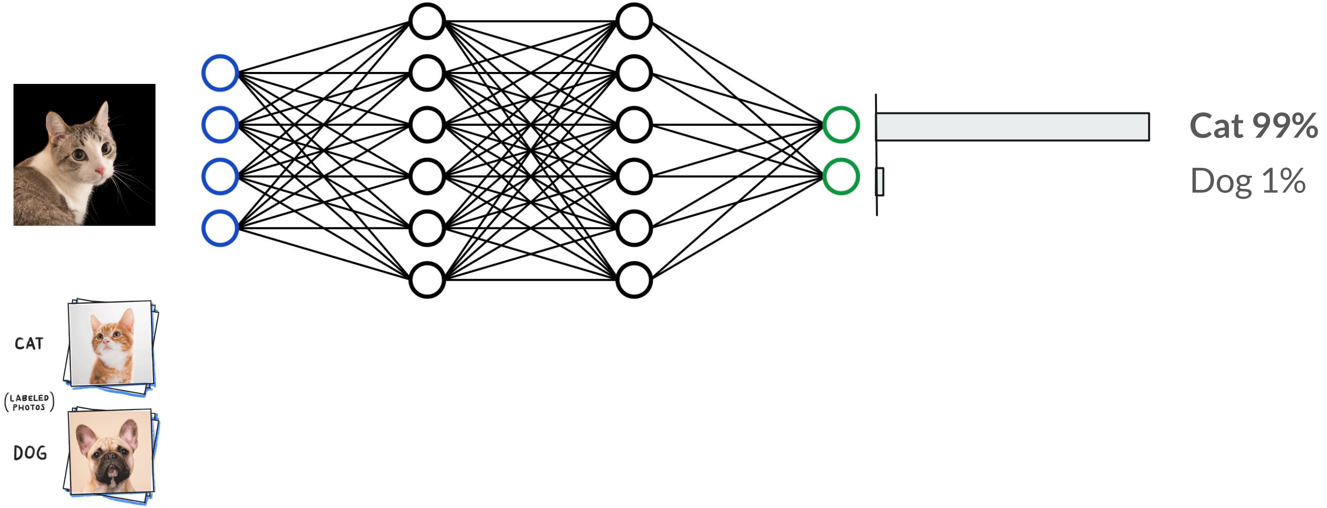
$$
\begin{aligned}
R_0 &= Q^\pi(s,a) \\
R_1 &= r_1 + \gamma Q^\pi(s',a') \\
&\vdots \\
R_H &= r_1 + \cdots + \gamma^{H-1} r_H + \gamma^H Q^\pi(s',a')
\end{aligned}
$$

$$\underbrace{r_1 \quad r_2 \quad \cdots \quad r_H}_{\text{Sample from autoregressive model}} \quad \underbrace{r_{H+1} \quad \cdots}_{\text{Replace by bootstrap } Q^\pi}$$

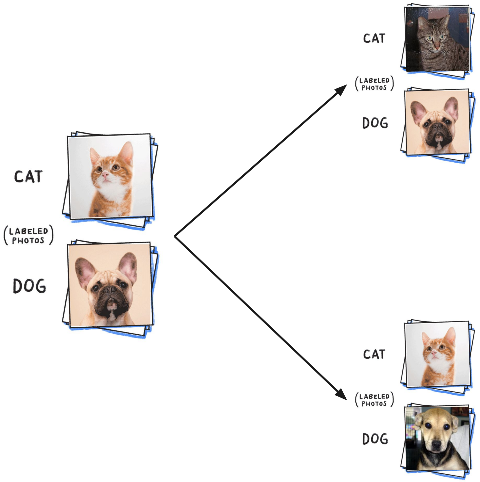
Sample from **autoregressive** model 

Replace by **bootstrap** $Q^\pi$

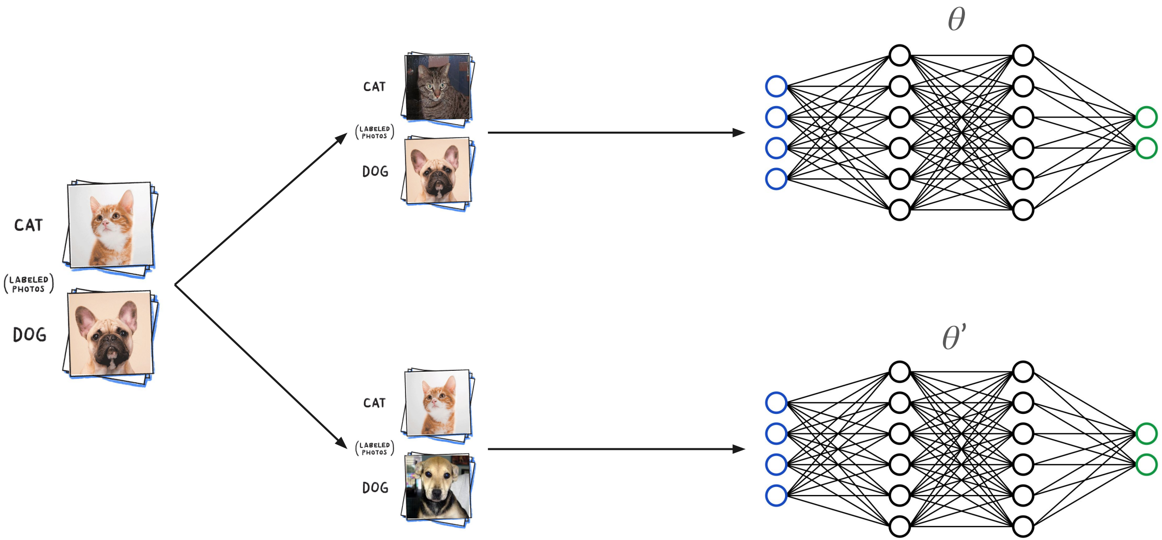
# Bootstrapping in ML



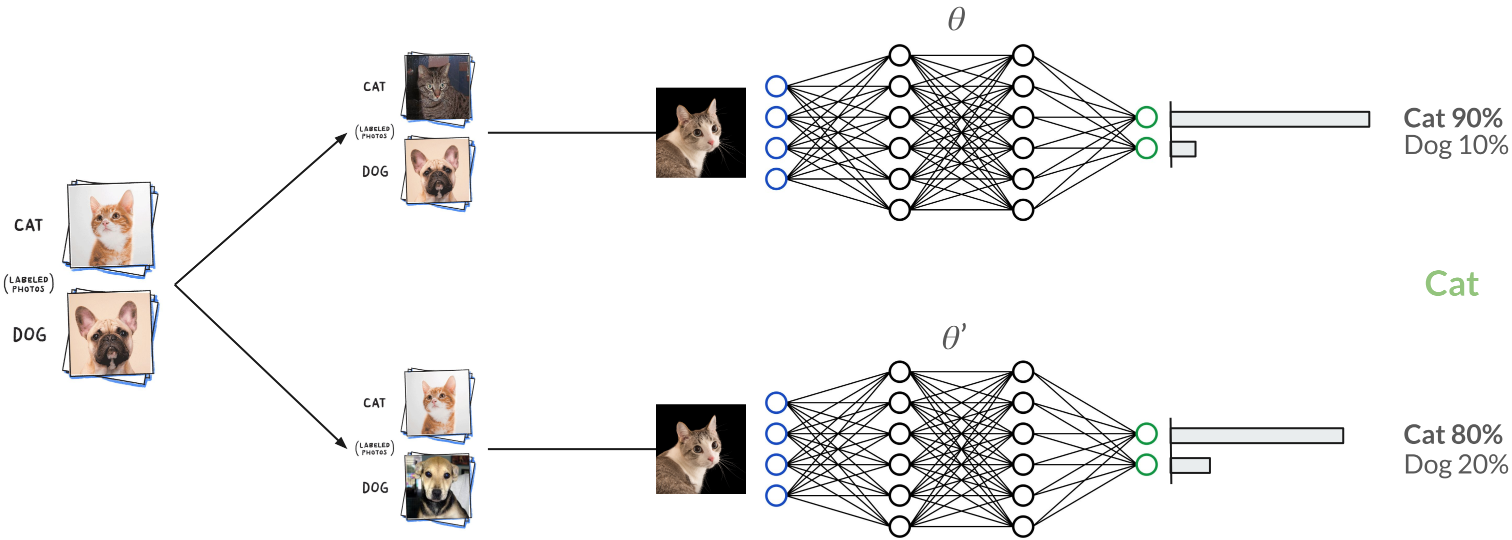
# Bootstrapping in ML



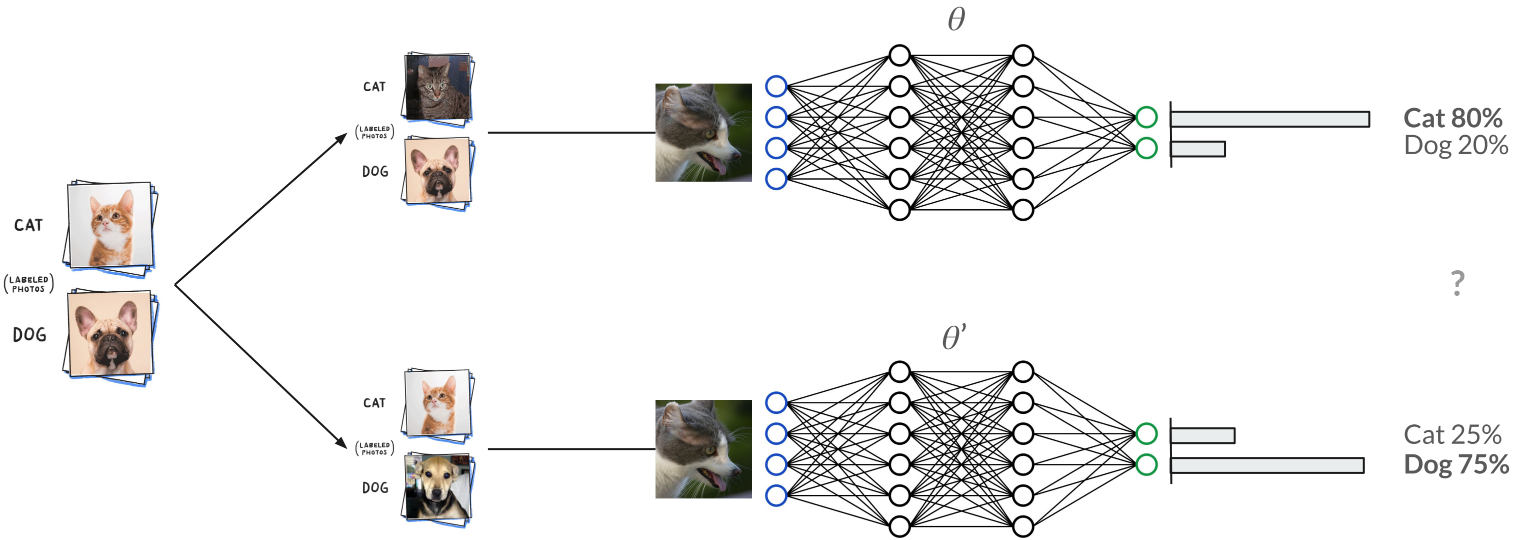
# Bootstrapping in ML



# Bootstrapping in ML



# Bootstrapping in ML



# Model-Based Offline Policy Optimization

- Produce an **ensemble** of estimators:

$$
\begin{aligned}
R_0 &= Q^\pi(s,a) \\
R_1 &= r_1 + \gamma Q^\pi(s',a') \\
&\vdots \\
R_H &= r_1 + \cdots + \gamma^{H-1} r_H + \gamma^H Q^\pi(s',a')
\end{aligned}
$$

$$\underbrace{r_1 \quad r_2 \quad \cdots \quad r_H}\quad \underbrace{r_{H+1} \quad \cdots}$$

Sample from **autoregressive** model 

Replace by **bootstrap** $Q^\pi$

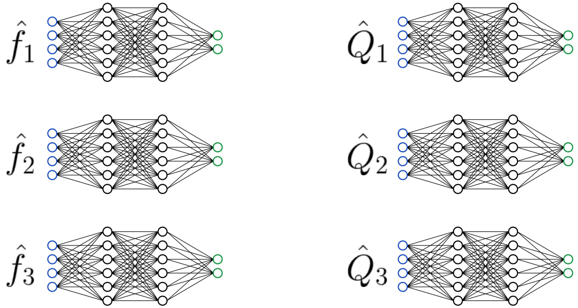
# Model-Based Offline Policy Optimization

1. Use statistical bootstrap to estimate **variation of n-step return**
2. Fit a **Gaussian** to each return estimator in the ensemble
3. Apply **Bayes' rule** to estimate total variance

# Model-Based Offline Policy Optimization

1. Use statistical bootstrap to estimate **variation of n-step return**
2. Fit a **Gaussian** to each return estimator in the ensemble
3. Apply **Bayes' rule** to estimate total variance

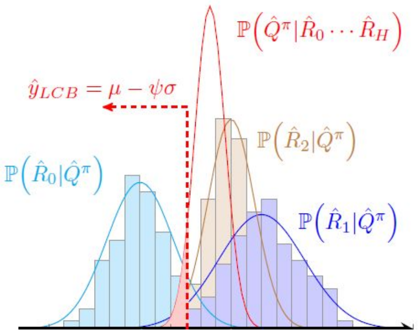
$$R_H = \underbrace{r_1 + \cdots + \gamma^{H-1} r_H}_{} + \gamma^H \underbrace{Q^\pi(s', a')}_{}$$



# Model-Based Offline Policy Optimization

1. Use statistical bootstrap to estimate **variation of n-step return**
2. Fit a **Gaussian** to each return estimator in the ensemble
3. Apply **Bayes' rule** to estimate total variance

$$\mu_H = \mathbb{E}_\pi[R_H] = \mathbb{E}_{\hat{f}_k}[\mathbb{E}_\pi[R_H|\hat{f}_k]]$$
$$\sigma_H^2 = Var_\pi[R_H] = \mathbb{E}_{\hat{f}_k}[Var_\pi[R_H|\hat{f}_k]] + Var_{\hat{f}_k}[\mathbb{E}_\pi[R_H|\hat{f}_k]]$$



# Model-Based Offline Policy Optimization

1. Use statistical bootstrap to estimate **variation of n-step return**
2. Fit a **Gaussian** to each return estimator in the ensemble
3. Apply **Bayes' rule** to estimate total variance

$$posterior = likelihood \times prior$$

Given:

- **Prior** $P(sick)$
- **Likelihood** $P(cough|sick)$ $P(cough|not\ sick)$

$$P(sick|cough) = \frac{P(cough|sick)P(sick)}{P(cough|sick)P(sick) + P(cough|not\ sick)(1 - P(sick))}$$

# Model-Based Offline Policy Optimization

1. Use statistical bootstrap to estimate **variation of n-step return**
2. Fit a **Gaussian** to each return estimator in the ensemble
3. Apply **Bayes' rule** to estimate total variance

$$\mathbb{P}\left(\hat{Q}^\pi \mid \hat{R}_0, \ldots, \hat{R}_H\right) \propto \mathbb{P}\left(\hat{R}_0, \ldots, \hat{R}_H \mid \hat{Q}^\pi\right) \, \mathbb{P}\left(\hat{Q}^\pi\right) = \mathbb{P}\left(\hat{Q}^\pi\right) \prod_{h=0}^{H} \mathbb{P}\left(\hat{R}_h \mid \hat{Q}^\pi\right)$$

Prior (improper)

Gaussian Likelihood

$$\hat{R}_h \mid \hat{Q}^\pi \sim \mathcal{N}(\mu_h, \sigma_h^2)$$

# Model-Based Offline Policy Optimization

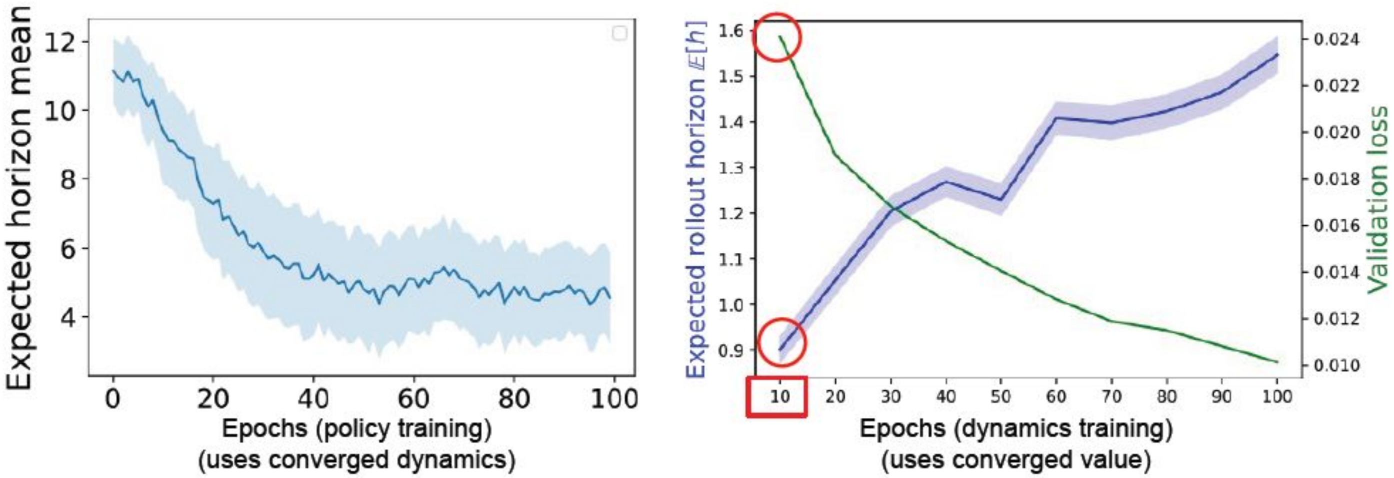
1. Use statistical bootstrap to estimate **variation of n-step return**
2. Fit a **Gaussian** to each return estimator in the ensemble
3. Apply **Bayes' rule** to estimate total variance

$$P(\hat{Q}^\pi | \hat{R}_1, \ldots \hat{R}_H) \sim \mathcal{N}(\mu, 1/\rho)$$

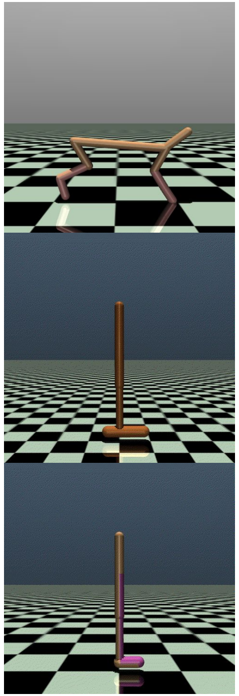
$$\mu = \frac{\sum_h \left(\frac{1}{\sigma_h^2}\right) \mu_h}{\sum_h \left(\frac{1}{\sigma_h^2}\right)} \qquad \rho = \sum_h \frac{1}{\sigma_h^2}$$

# Model-Based Offline Policy Optimization

$$\mathbb{E}[h] = \frac{\sum_h h \times \frac{1}{\sigma_h^2}}{\sum_h \frac{1}{\sigma_h^2}}$$



# Model-Based Offline Policy Optimization



$$\underset{\pi}{\operatorname{argmax}}\, \mu - k\sigma$$

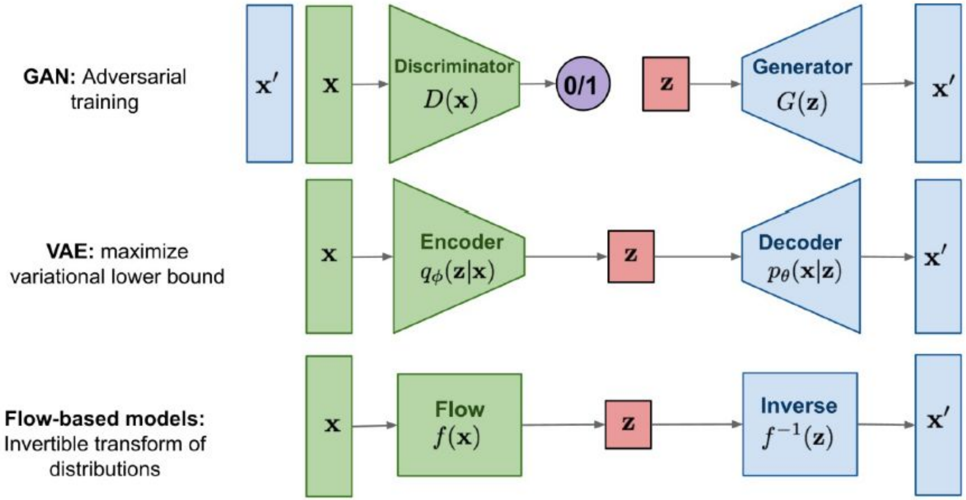
- Substantial improvements over prior SOTA MB methods
- SOTA results for 11 out of 18 benchmark datasets
- Please check out our paper!

Table 1: Normalized scores on D4RL MuJoCo Gym environments. Experiments ran with 5 seeds.

| | | MOPO | MOReL | COMBO | CQL | TD3+BC | EDAC | IQL | CBOP |
|---|---|---|---|---|---|---|---|---|---|
| random | halfcheetah | 35.4 ± 2.5 | 25.6 | **38.8** | 35.4 | 10.2 ± 1.3 | 28.4 ± 1.0 | - | 32.8 ± 0.4 |
| | hopper | 11.7 ± 0.4 | **53.6** | 17.9 | 10.8 | 11.0 ± 0.1 | 31.3 ± 0.0 | - | 31.4 ± 0.0 |
| | walker2d | 13.6 ± 2.6 | 37.3 | 7.0 | 7.0 | 1.4 ± 1.6 | **21.7 ± 0.0** | - | 17.8 ± 0.4 |
| medium | halfcheetah | 42.3 ± 1.6 | 42.1 | 54.2 | 44.4 | 42.8 ± 0.3 | 67.5 ± 1.2 | 47.4 | **74.3 ± 0.2** |
| | hopper | 28.0 ± 12.4 | 95.4 | 94.9 | 79.2 | 99.5 ± 1.0 | 101.6 ± 0.6 | 66.2 | **102.6 ± 0.1** |
| | walker2d | 17.8 ± 19.3 | 77.8 | 75.5 | 58.0 | 79.7 ± 1.8 | 92.5 ± 0.8 | 78.3 | **95.5 ± 0.4** |
| medium replay | halfcheetah | 53.1 ± 2.0 | 40.2 | 55.1 | 46.2 | 43.3 ± 0.5 | 63.9 ± 0.8 | 44.2 | **66.4 ± 0.3** |
| | hopper | 67.5 ± 24.7 | 93.6 | 73.1 | 48.6 | 31.4 ± 3.0 | 101.8 ± 0.5 | 94.7 | **104.3 ± 0.4** |
| | walker2d | 39.0 ± 9.6 | 49.8 | 56.0 | 26.7 | 25.2 ± 5.1 | 87.1 ± 2.3 | 73.8 | **92.7 ± 0.9** |
| medium expert | halfcheetah | 63.3 ± 38.0 | 53.3 | 90.0 | 62.4 | 97.9 ± 4.4 | **107.1 ± 2.0** | 86.7 | 105.4 ± 1.6 |
| | hopper | 23.7 ± 6.0 | 108.7 | 111.1 | 98.7 | **112.2 ± 0.2** | 110.7 ± 0.1 | 91.5 | 111.6 ± 0.2 |
| | walker2d | 44.6 ± 12.9 | 95.6 | 96.1 | 111.0 | 101.1 ± 9.3 | 114.7 ± 0.9 | 109.6 | **117.2 ± 0.5** |
| expert | halfcheetah | - | - | - | - | 105.7 ± 1.9 | **106.8 ± 3.4** | - | 100.4 ± 0.9 |
| | hopper | - | - | - | - | **112.2 ± 0.2** | 110.3 ± 0.3 | - | 111.4 ± 0.2 |
| | walkcr2d | - | - | - | - | 105.7 ± 2.7 | 115.1 ± 1.9 | - | **122.7 ± 0.8** |
| full replay | halfcheetah | - | - | - | - | - | 84.6 ± 0.9 | - | **85.5 ± 0.3** |
| | hopper | - | - | - | - | - | 105.4 ± 0.7 | - | **108.1 ± 0.3** |
| | walker2d | - | - | - | - | - | 99.8 ± 0.7 | - | **107.8 ± 0.2** |

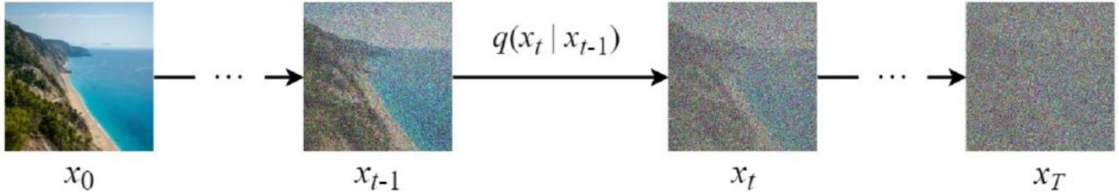
# Off-Policy Evaluation using Diffusion Models

# Generative Models




From https://lilianweng.github.io/

# Diffusion Model




Distribution of the noised images — Output — Mean $\mu_t$ — Variance $\Sigma_t$

$$\boxed{q(x_t|x_{t-1})} = \mathcal{N}(\boxed{x_t}; \boxed{\sqrt{1-\beta_t}x_{t-1}}, \boxed{\beta_t I})$$

Notations:

$t$ : time step (from 0 to $T$)

$x_0$ : a data sampled from the real data distrition $q(x)$ (i.e. $x_0 \sim q(x)$)

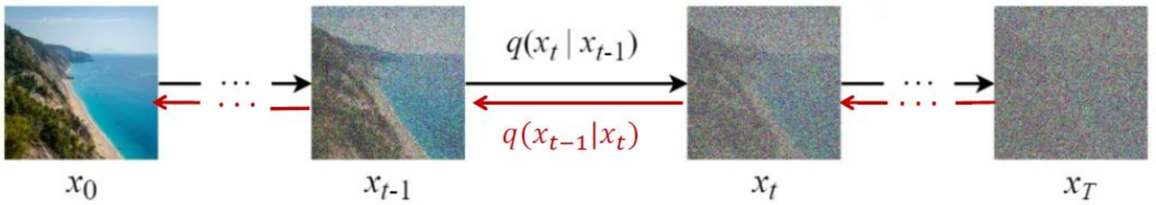
$\beta_t$ : variance schedule ($0 \leq \beta_t \leq 1$, and $\beta_0$ = small number, $\beta_T$ = large number)

$I$ : identity matrix

From Steins (medium.com)

# Diffusion Model

From Steins (medium.com)




- Forward factorization: $q(\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_T) = q(\boldsymbol{x}_0) \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$
- Reverse factorization: $q(\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_T) = \prod_{t=1}^{T} q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\, q(\boldsymbol{x}_T)$
  - Since joint distribution is Gaussian then $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is also Gaussian
  - $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = N(\boldsymbol{x}_{t-1}|\tilde{\mu}_t(\boldsymbol{x}_t, t), \sigma_t \boldsymbol{I})$

# Diffusion Model

Recall that $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\ \epsilon$ . Ho et al. NeurIPS 2020 observe that:

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

They propose to represent the mean of the denoising model using a *noise-prediction* network:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\ \epsilon_\theta(\mathbf{x}_t, t)\right)$$

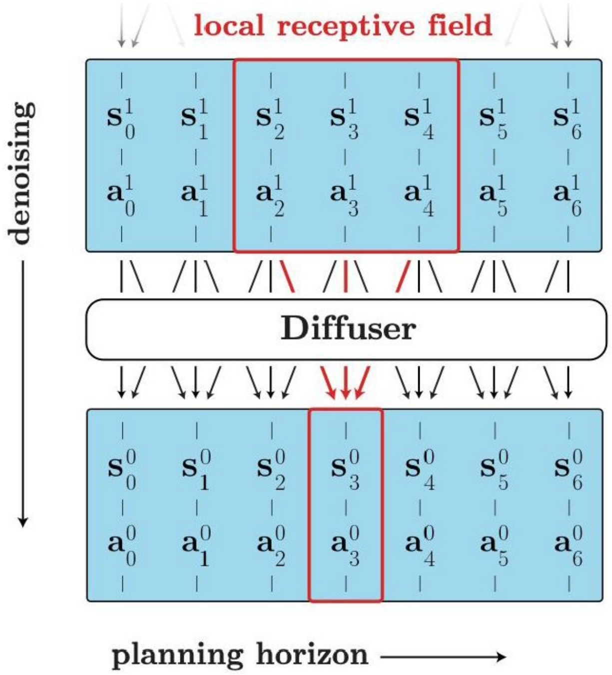
With this parameterization

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}\left[\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\ \epsilon}_{\mathbf{x}_t}, t)||^2\right] + C$$

slide from https://cvpr2022-tutorial-diffusion-models.github.io/

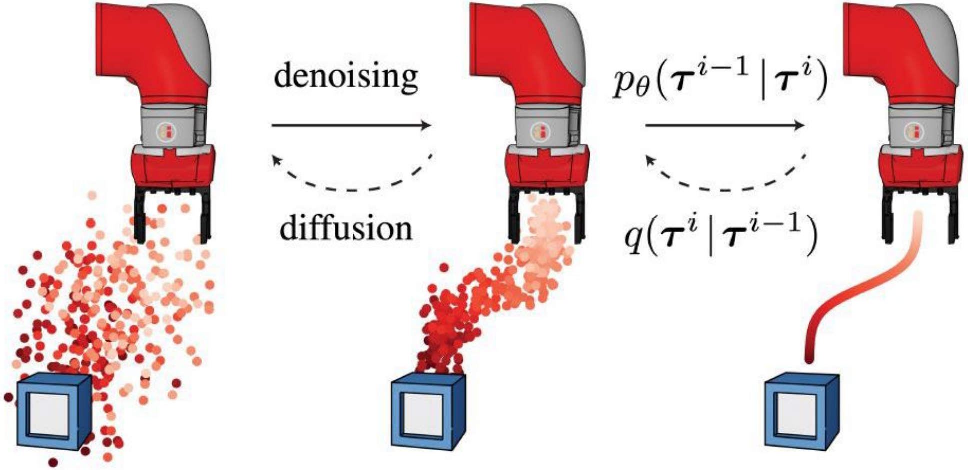
# Diffuser

- Represent trajectories as single-channel images
- Train a diffusion model to iteratively denoise entire trajectory
- Use (one-dimensional) convolutions for temporal equivariance and horizon-independence

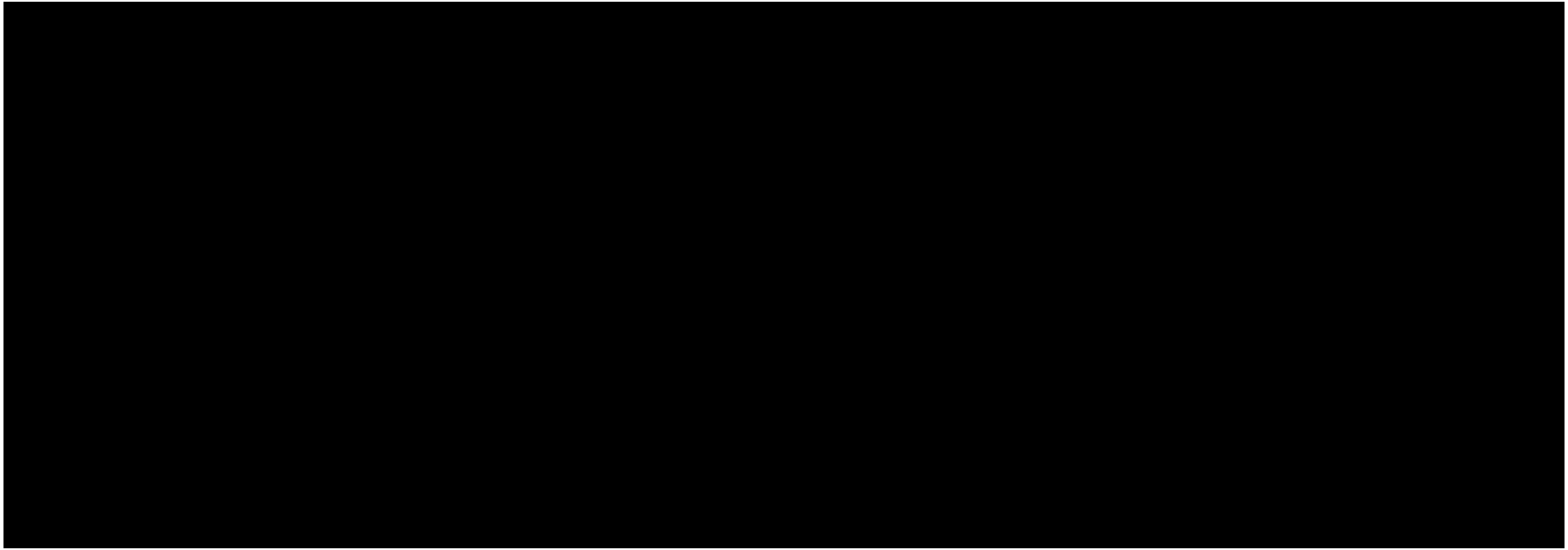


# Diffuser

- Prediction is **non-autoregressive**: entire trajectory is predicted simultaneously

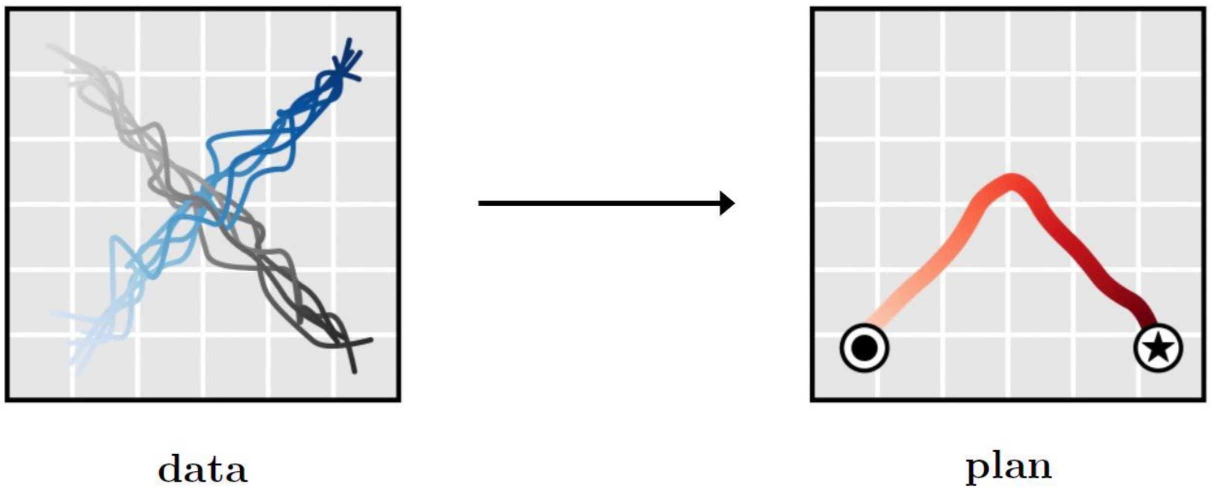


# Diffuser



# Diffuser

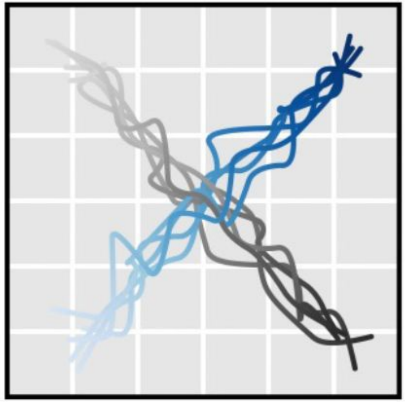
- Diffuser is non-Markovian, but still compositional due to temporal convolutions



# Diffuser

- Diffuser is non-Markovian, but still compositional due to temporal convolutions

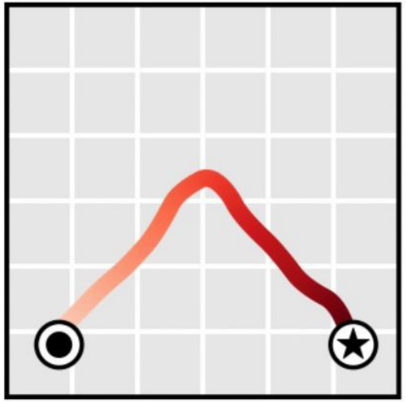
Offline data from behavior policy



**data**

Target policy

Rollout from target policy

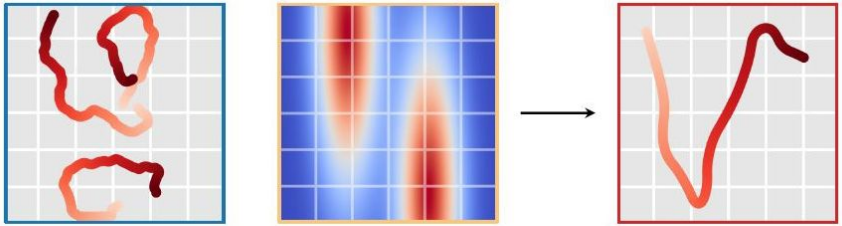


**plan**

# Diffuser

- Synthesize different behaviors through conditional trajectory synthesis with different learned guidance functions:

$$\underbrace{\tilde{p}_\theta(\boldsymbol{\tau})}_{\text{Behavior Model}} \propto \underbrace{p_\theta(\boldsymbol{\tau})}_{\text{Diffusion Model}} \underbrace{h(\boldsymbol{\tau})}_{\text{Guidance Function}}$$



# Diffuser

- How can the diffuser generate trajectories from **another policy**?
  1. **Guidance** function
  2. **Inpainting**

# Diffuser

- How can the diffuser generate trajectories from **another policy**?
  1. **Guidance** function
  2. **Inpainting**

$$\underbrace{\tilde{p}_\theta(\boldsymbol{\tau})}_{\text{Behavior Model}} \propto \underbrace{p_\theta(\boldsymbol{\tau})}_{\text{Diffusion Model}} \underbrace{h(\boldsymbol{\tau})}_{\text{Guidance Function}}$$

$$p_\theta(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i}, \mathcal{O}_{1:T}) \approx \mathcal{N}(\boldsymbol{\tau}^{i-1}; \mu + \Sigma g, \Sigma)$$

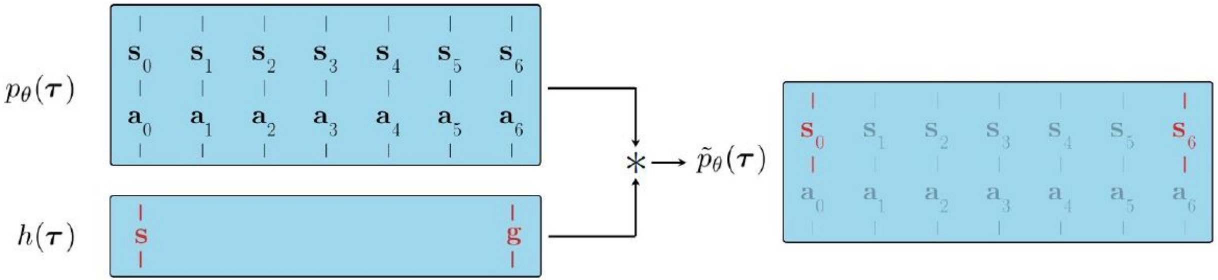
$$\begin{aligned} g &= \nabla_{\boldsymbol{\tau}} \log p(\mathcal{O}_{1:T} \mid \boldsymbol{\tau})|_{\boldsymbol{\tau}=\mu} \\ &= \sum_{t=0}^{T} \nabla_{\mathbf{s}_t, \mathbf{a}_t} r(\mathbf{s}_t, \mathbf{a}_t)|_{(\mathbf{s}_t, \mathbf{a}_t)=\mu_t} = \nabla \mathcal{J}(\mu). \end{aligned}$$

Dhariwal, Prafulla, and Alexander Nichol. "Diffusion models beat gans on image synthesis." *Advances in neural information processing systems* 34 (2021): 8780-8794.

# Diffuser

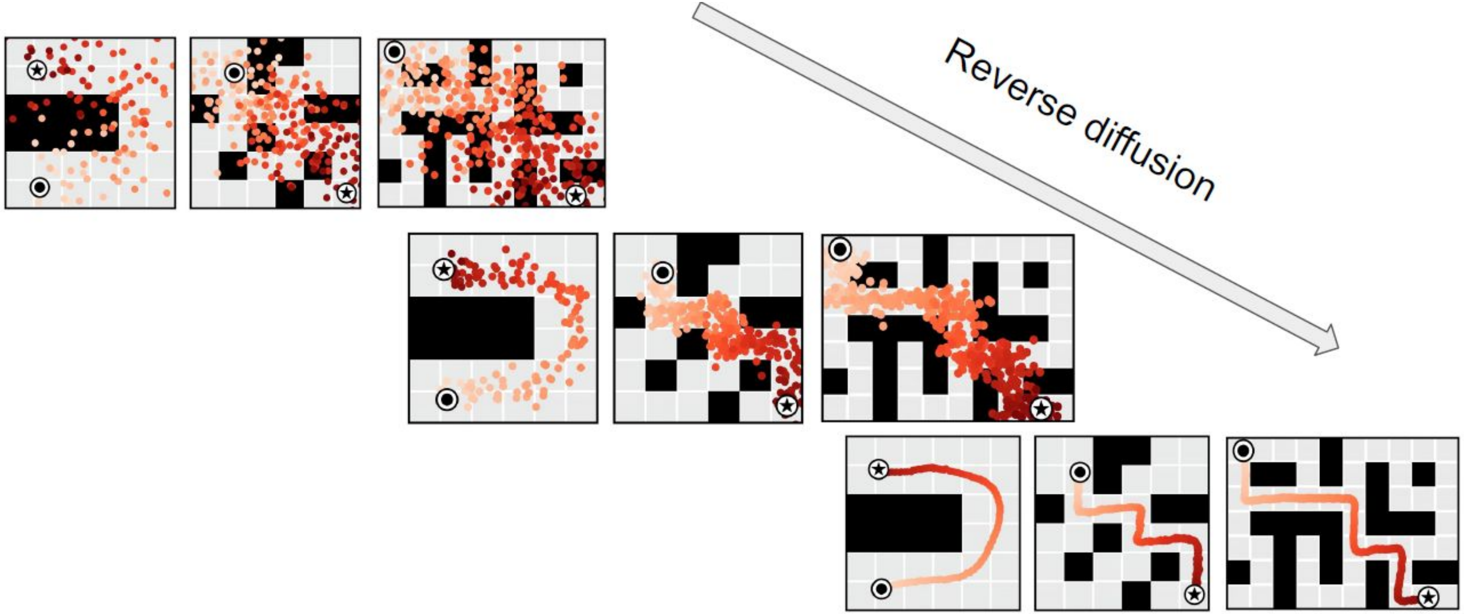
- How can the diffuser generate trajectories from **another policy**?
  1. **Guidance** function
  2. **Inpainting**

     - Specify a guidance function over the final explicit goal state of a trajectory




     - Construct a goal seeking policy through guidance

# Diffuser



# Diffuser for Off-Policy Evaluation

- Recall that we **collect data from some behavior policy**
- But we want to **evaluate some other target policy**

# Diffuser for Off-Policy Evaluation

- Recall that we **collect data from some behavior policy**
- But we want to **evaluate some other target policy**

$$\underbrace{\tilde{p}_\theta(\boldsymbol{\tau})}_{\text{Behavior Model}} \propto \underbrace{p_\theta(\boldsymbol{\tau})}_{\text{Diffusion Model}} \underbrace{h(\boldsymbol{\tau})}_{\text{Guidance Function}}$$

$$p_\theta(\boldsymbol{\tau}^{i-1} \mid \boldsymbol{\tau}^{i}, \mathcal{O}_{1:T}) \approx \mathcal{N}(\boldsymbol{\tau}^{i-1}; \mu + \Sigma g, \Sigma)$$

$$g = \nabla_{\boldsymbol{\tau}} \log p(\mathcal{O}_{1:T} \mid \boldsymbol{\tau})|_{\boldsymbol{\tau}=\mu}$$

$$\log \nabla_{s,a} \prod_t \pi(a_t|s_t) = \sum_t \log \nabla_{s,a} \pi(a_t|s_t)$$

Use the target policy probability as prior

# Summary

- Off-policy RL is an important and challenging problem
- Discussed how statistical bootstrapping can produce an estimate of uncertainty of any ML model
- Applied this idea to choose the best rollout horizon in offline policy optimization
- Introduced the diffusion model as a powerful way of simulating trajectories from any policy
- Applied diffusion as a potential way to evaluate any target policy through guidance