DeepRob

[Group 8] Lecture 6

Diffusion Models and Policy Learning

by Nirshal Chandra Sekar and Mohit Yadav

University of Minnesota

Overview

- Motivation behind diffusion models
- Denoising Diffusion Probabilistic Models
  - Diffusion Process and Reverse Diffusion Process
  - Training and Sampling a Diffusion Model
  - Results from a Diffusion Model that we trained
- Conditioned v/s Unconditioned Diffusion
- Policy Learning
- Toy Problem
- Diffusion Policy on Real Robots
  - Advantages
  - Network Architect
  - Evaluation & Results

Motivation Behind Diffusion Model

Text Input ~ *"Generate an image of me delivering a presentation on diffusion models."*

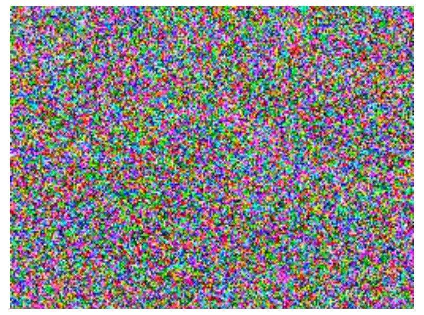
Image Output ~ Generated using ChatGPT - 4o

Motivation Behind Diffusion Model

Text Input ~ "Generate an image where the attendees are getting bored during my presentation on diffusion policies."

Image Output ~ Generated using ChatGPT - 4o

Motivation Behind Diffusion Model

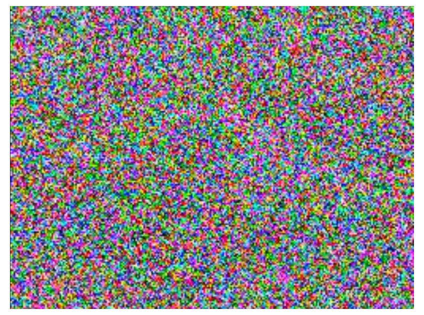


Diffusion Model

Image Source: ChatGPT - 4o

Text Input ~ *'Generate an image of me delivering a presentation on diffusion models.'*

Motivation Behind Diffusion Model



Diffusion Model

Image Source: ChatGPT - 4o

Text Input ~ *'Generate an image of Porsche 911'*

Overview

- Motivation behind diffusion models ✓
- Denoising Diffusion Probabilistic Models
  - Diffusion Process and Reverse Diffusion Process
  - Training and Sampling a Diffusion Model
  - Results from a Diffusion Model that we trained
- Conditioned v/s Unconditioned Diffusion
- Policy Learning
- Toy Problem
- Diffusion Policy on Real Robots
  - Advantages
  - Network Architect
  - Evaluation & Results

Denoising Diffusion Probabilistic Models

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

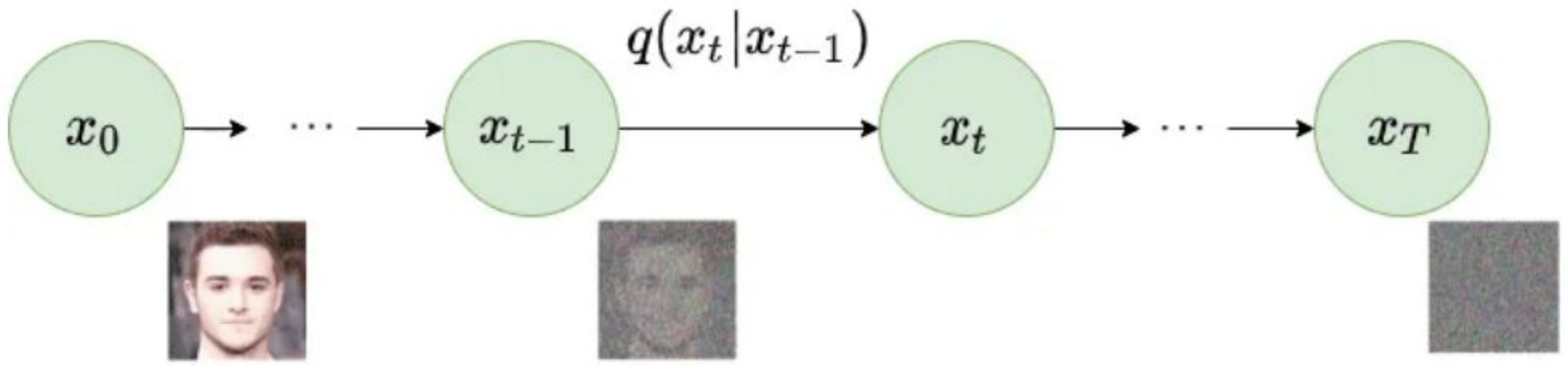
Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

Ref: Ho, J., Jain, A. and Abbeel, P., 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, *33*, pp.6840-6851.

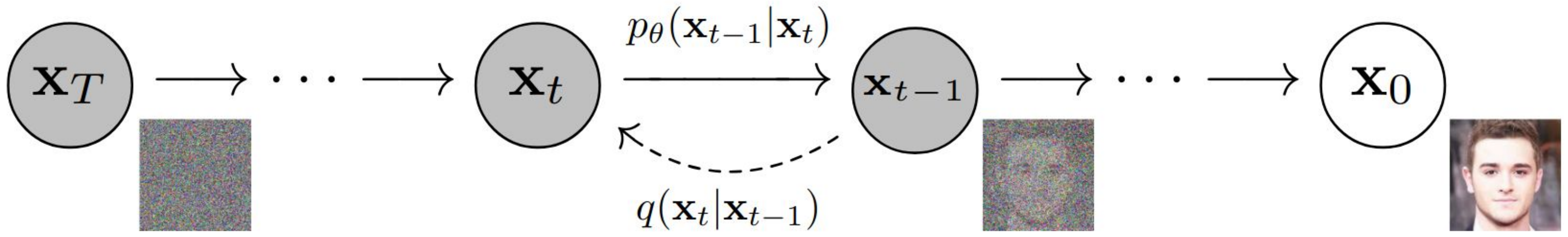
Denoising Diffusion Probabilistic Models

$q \}$



Diffusion Process / Adding Noise

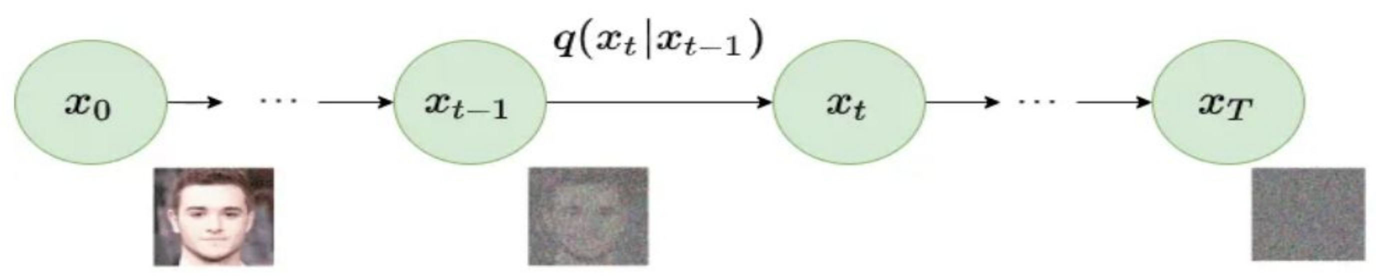
$p \}$



Reverse Diffusion Process / Removing Noise

Ref: Ho, J., Jain, A. and Abbeel, P., 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, *33*, pp.6840-6851.

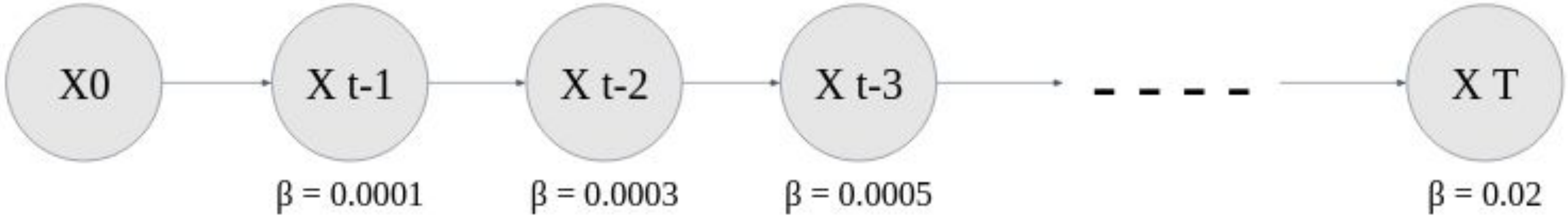
Diffusion Process / Adding Noise



Diffusion Process / Adding Noise

Joint probability of the states x1, x2, ..., xT in a Markov chain, given the initial state x0.

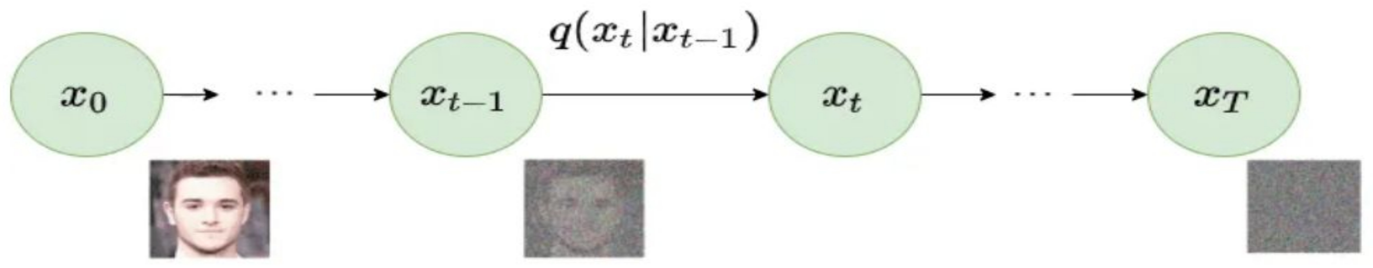
$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \qquad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t} \cdot \text{noise}$$

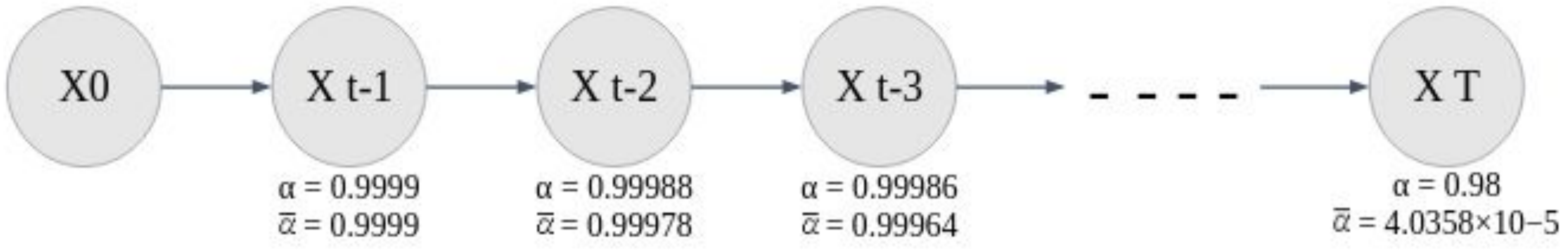
Ref: Ho, J., Jain, A. and Abbeel, P., 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, *33*, pp.6840-6851.

Diffusion Process / Adding Noise



Diffusion Process / Adding Noise

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



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \sqrt{1-\beta_t} \cdot \mathbf{x}_{t-1} + \sqrt{\beta_t} \cdot \text{noise}$$

$$q(\mathbf{x}_t|\mathbf{x}_0) = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \cdot \text{noise}$$

Where,

$\alpha = 1 - \beta$

$\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$

For more information please refer the paper

Ref: Ho, J., Jain, A. and Abbeel, P., 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, *33*, pp.6840-6851.

Diffusion Process / Adding Noise

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \sqrt{1-\beta_1} \cdot x_0 + \sqrt{\beta_1} \cdot \text{noise}$$

- β1 = 0.0001, β2 = 0.0003, β3 = 0.0005
- noise = 5
- input = 10

q(X1|X0) = (√(1-0.0001)*10) + (√0.0001*5) = 10.0494
q(X2|X1) = (√(1-0.0003)*9.9999) + (√0.0003*5) = 10.1344
q(X3|X2) = (√(1-0.0005)*1.0012 + (√0.0005*5) = **10.24**

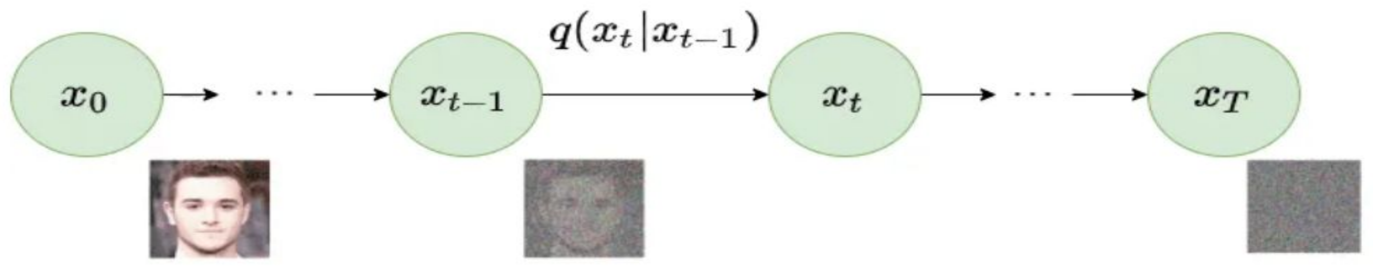
$$q(\mathbf{x}_t|\mathbf{x}_0) = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{\mathbf{1}-\bar{\alpha}_t} \cdot \mathbf{noise}$$

- α = 0.9999, α= 0.9997, α = 0.9995
- α’ = 0.9999, α’ = 0.9996, α’ = 0.9991
- noise = 5
- input = 10

q(X3|X0) = √(0.9991)*10 + √(1-0.9991)*5 = **10.14**

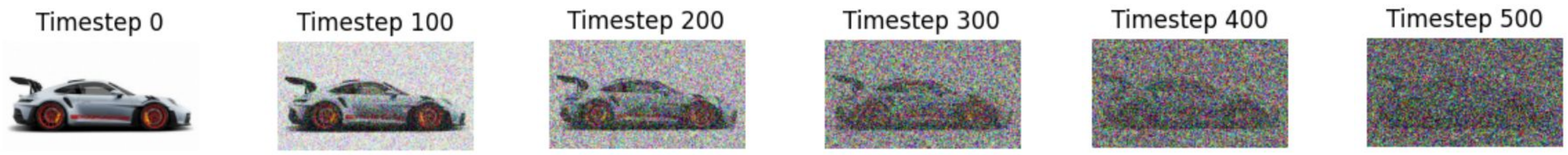
Ref: Ho, J., Jain, A. and Abbeel, P., 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, *33*, pp.6840-6851.

Diffusion Process / Adding Noise



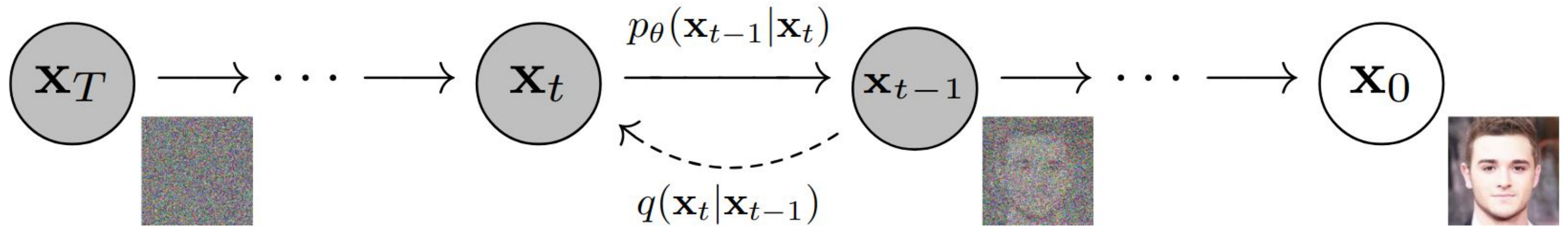
Diffusion Process / Adding Noise

$$q(\mathbf{x}_t|\mathbf{x}_0) = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \cdot \text{noise}$$



Ref: Ho, J., Jain, A. and Abbeel, P., 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, *33*, pp.6840-6851.

Reverse Diffusion / Removing Noise



Reverse Diffusion Process / Removing Noise

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

This is what the model is trying to learn

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \boxed{\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}\right) + \sigma_t \cdot Z$$

Where,

$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\sigma_t^2 = \beta_t$

For more information please refer the paper

Ref: Ho, J., Jain, A. and Abbeel, P., 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, *33*, pp.6840-6851.

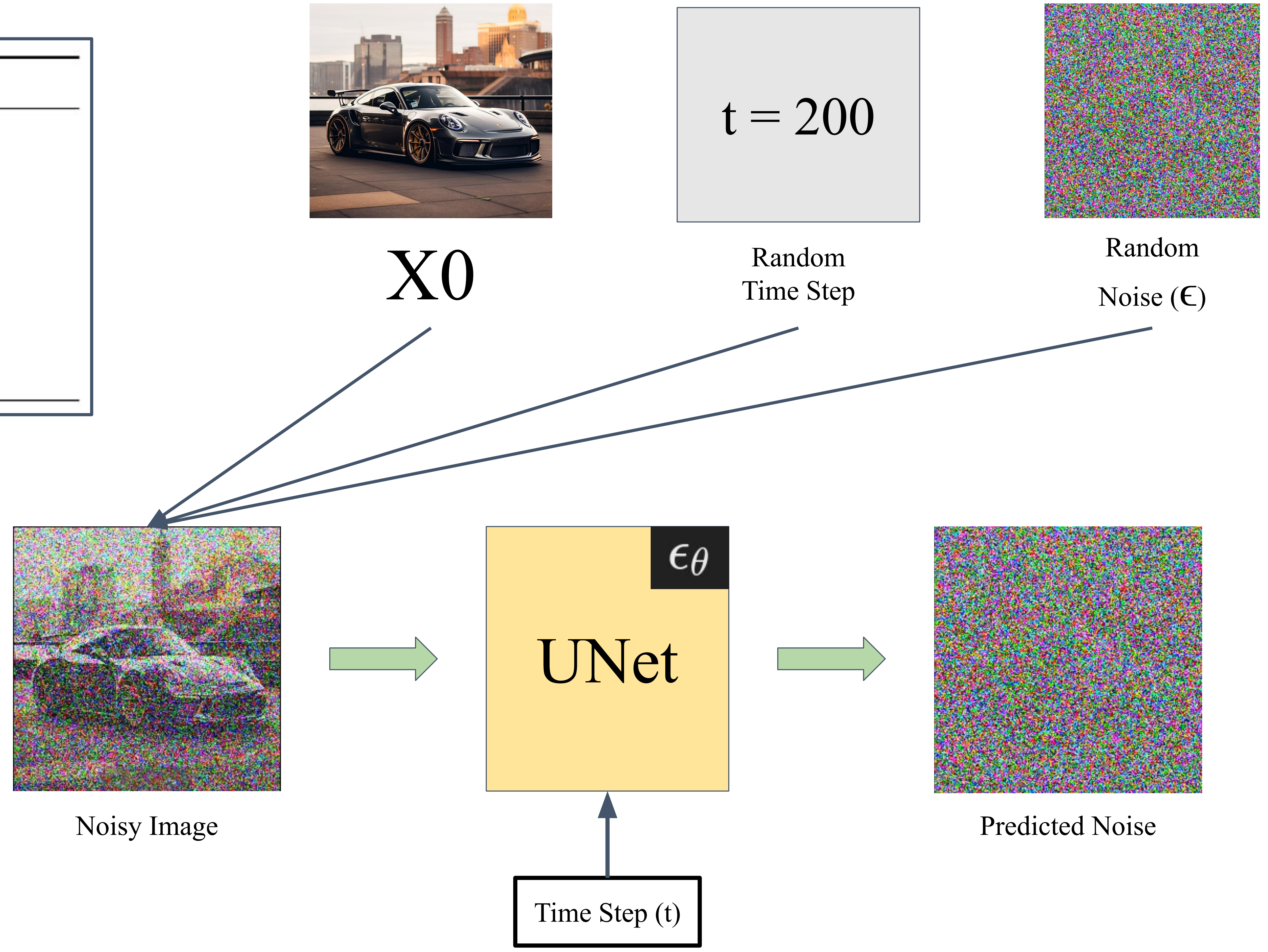
Overview

- Motivation behind diffusion models ✓
- Denoising Diffusion Probabilistic Models
  - Diffusion Process and Reverse Diffusion Process ✓
  - Training and Sampling a Diffusion Model
  - Results from a Diffusion Model that we trained
- Conditioned v/s Unconditioned Diffusion
- Policy Learning
- Toy Problem
- Diffusion Policy on Real Robots
  - Advantages
  - Network Architect
  - Evaluation & Results

Training a Diffusion Model

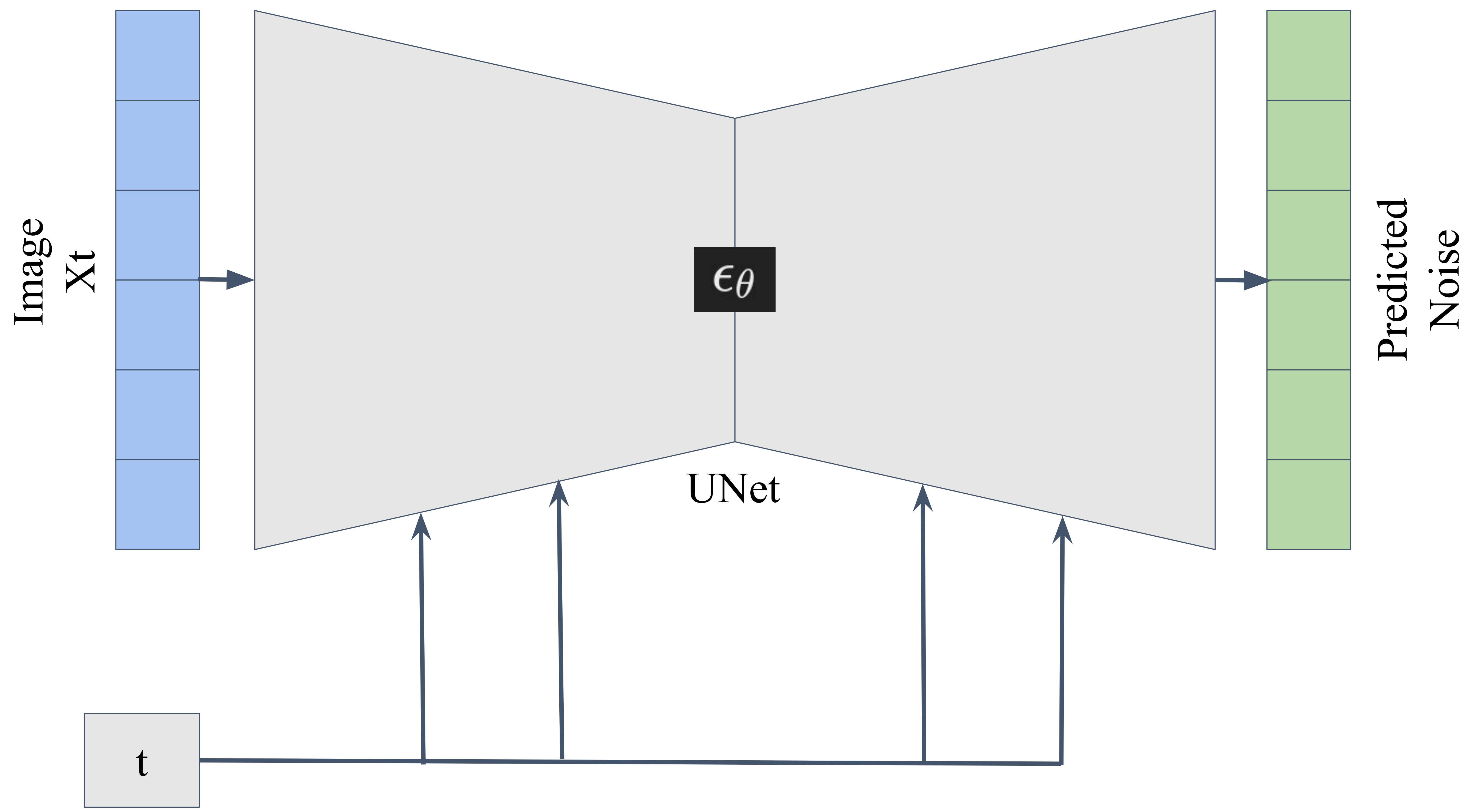
Algorithm 1 Training

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



Ref: Ho, J., Jain, A. and Abbeel, P., 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, *33*, pp.6840-6851.

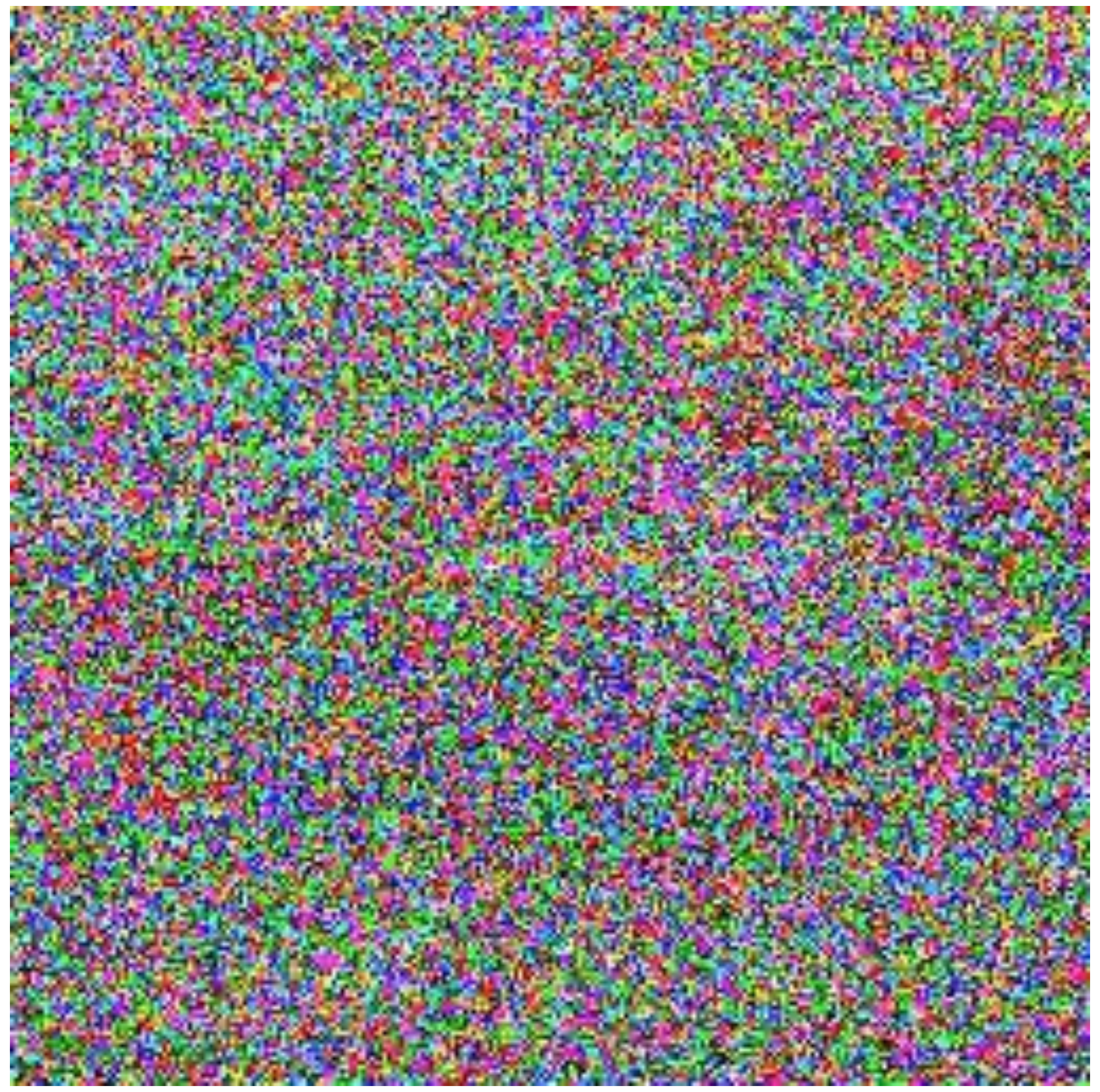
Training a Diffusion Model

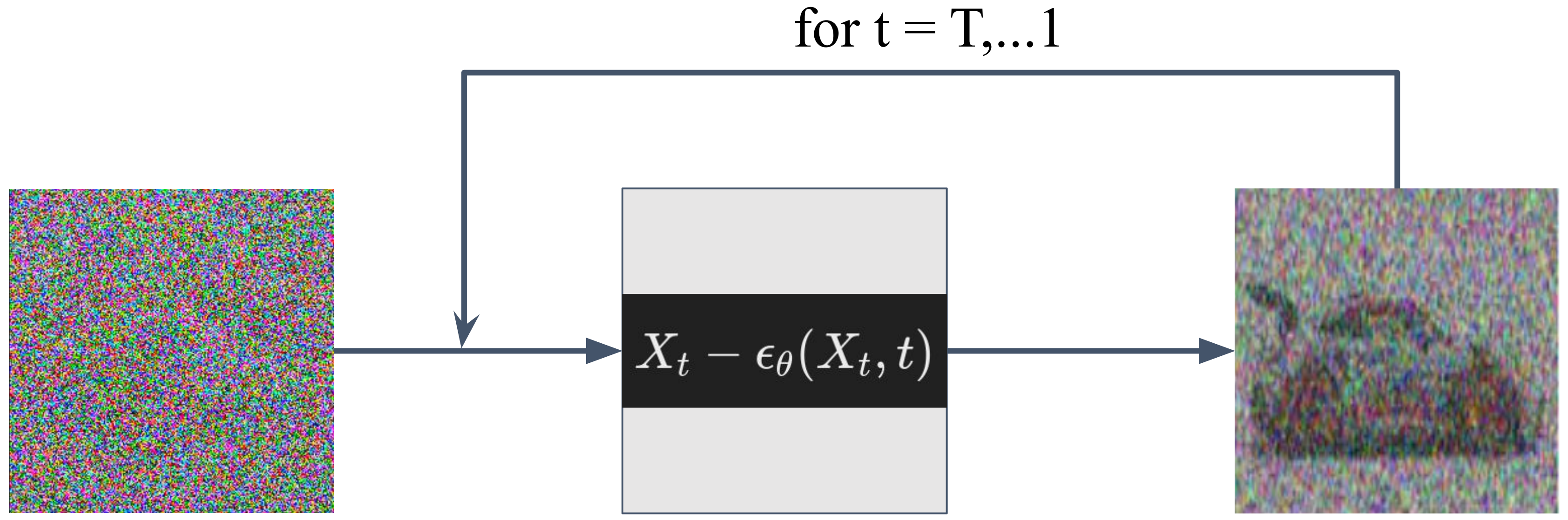


Sampling from a Diffusion Model

Algorithm 2 Sampling

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





Ref: Ho, J., Jain, A. and Abbeel, P., 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, *33*, pp.6840-6851.

Overview

- Motivation behind diffusion models ✓
- Denoising Diffusion Probabilistic Models
  - Diffusion Process and Reverse Diffusion Process ✓
  - Training and Sampling a Diffusion Model ✓
  - Results from a Diffusion Model that we trained
- Conditioned v/s Unconditioned Diffusion
- Policy Learning
- Toy Problem
- Diffusion Policy
  - Advantages
  - Network Architect
  - Evaluation & Results

Denoising Diffusion Probabilistic Models

Stanford Cars Image Dataset

Diffusion Model

Hardware: RTX 3090 Ti
Train Time: 1.5 days
Epochs: 500

Dataset Source: https://www.kaggle.com/datasets/jessicali9530/stanford-cars-dataset

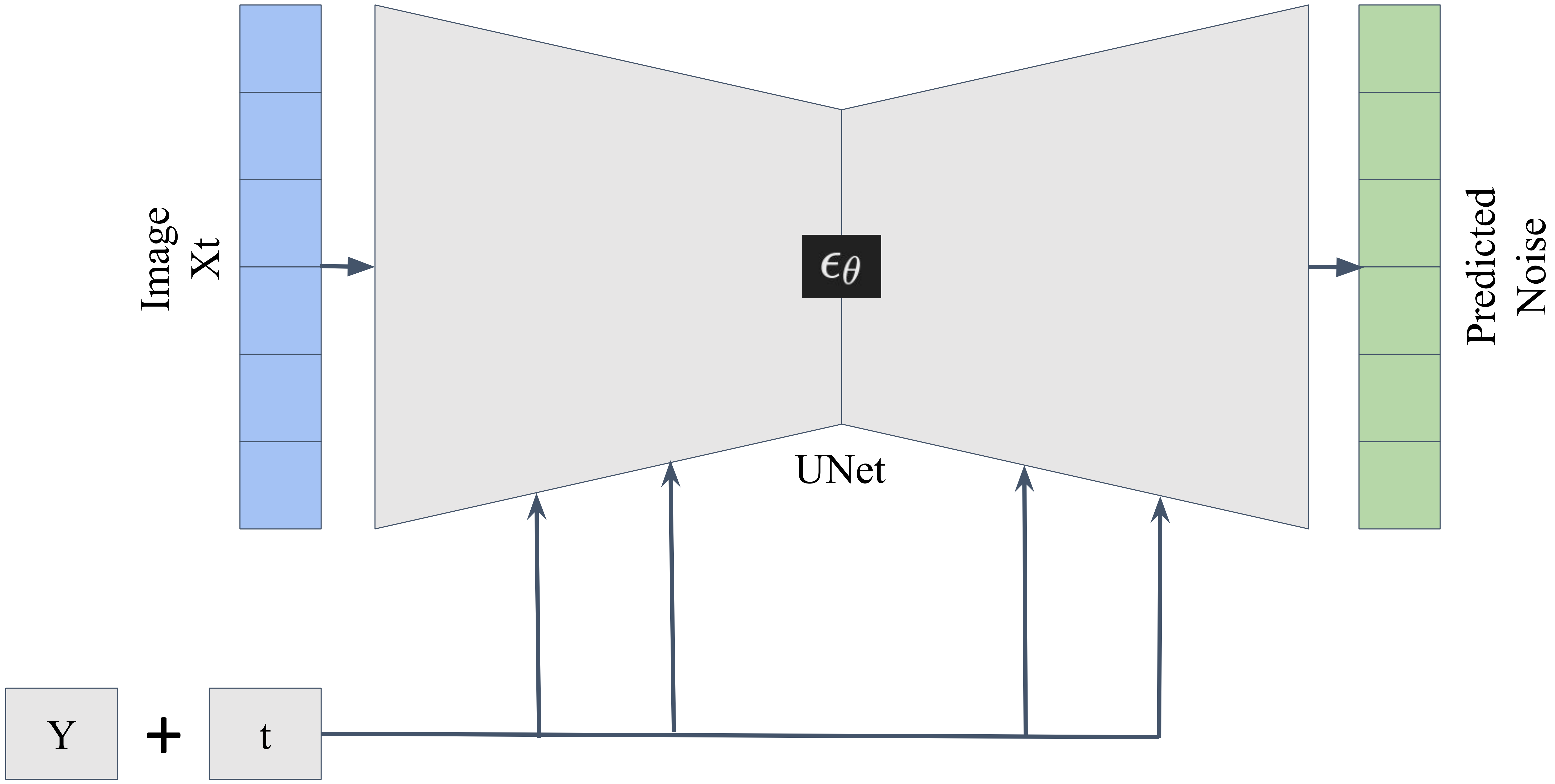
Denoising Diffusion Probabilistic Models

Diffused Cars!

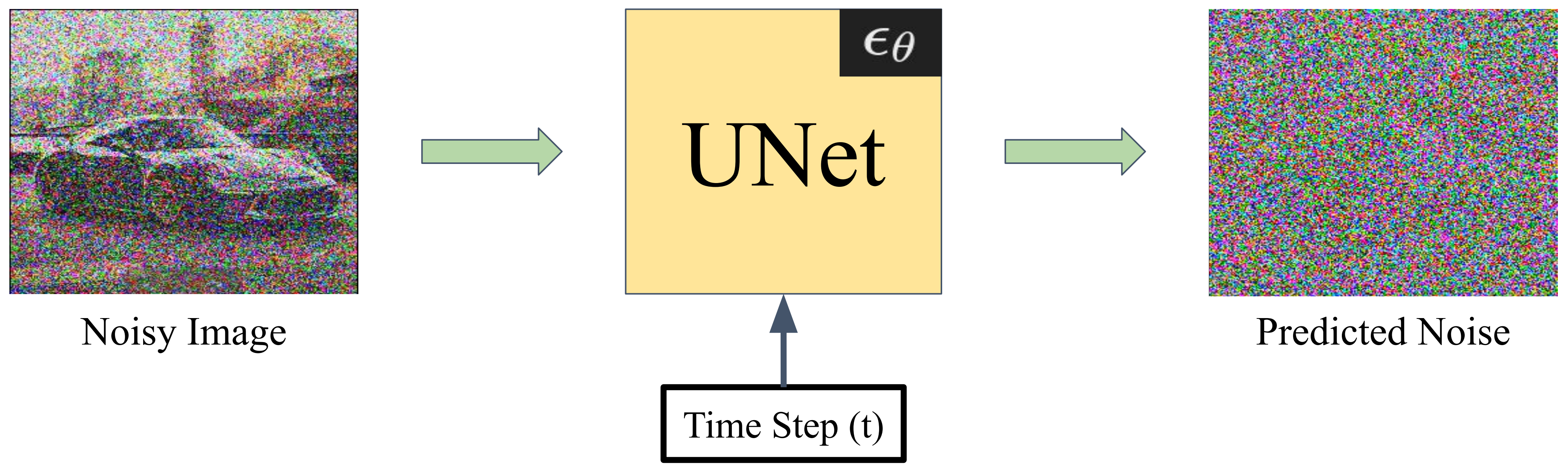
Overview

- Motivation behind diffusion models ✓
- Denoising Diffusion Probabilistic Models ✓
  - Diffusion Process and Reverse Diffusion Process ✓
  - Training and Sampling a Diffusion Model ✓
  - Results from a Diffusion Model that we trained ✓
- Conditioned v/s Unconditioned Diffusion
- Policy Learning
- Toy Problem
- Diffusion Policy on Real Robots
  - Advantages
  - Network Architect
  - Evaluation & Results

Training a Diffusion Model

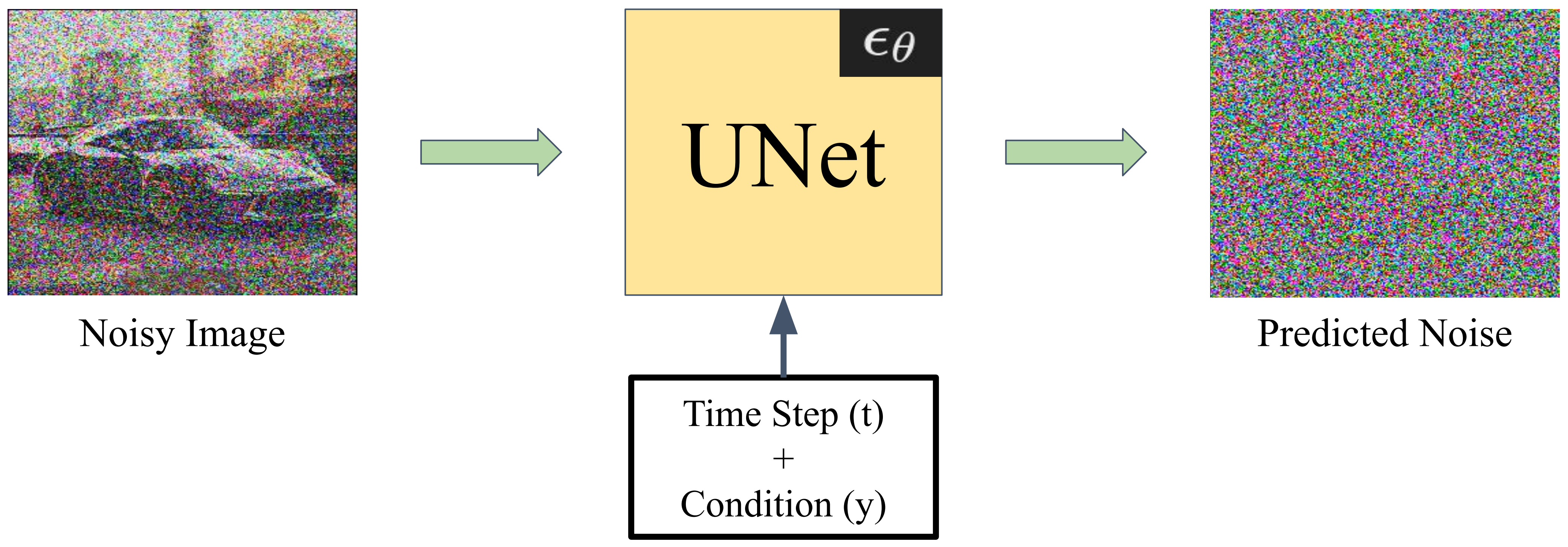


Unconditioned v/s Conditioned



$$\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$$

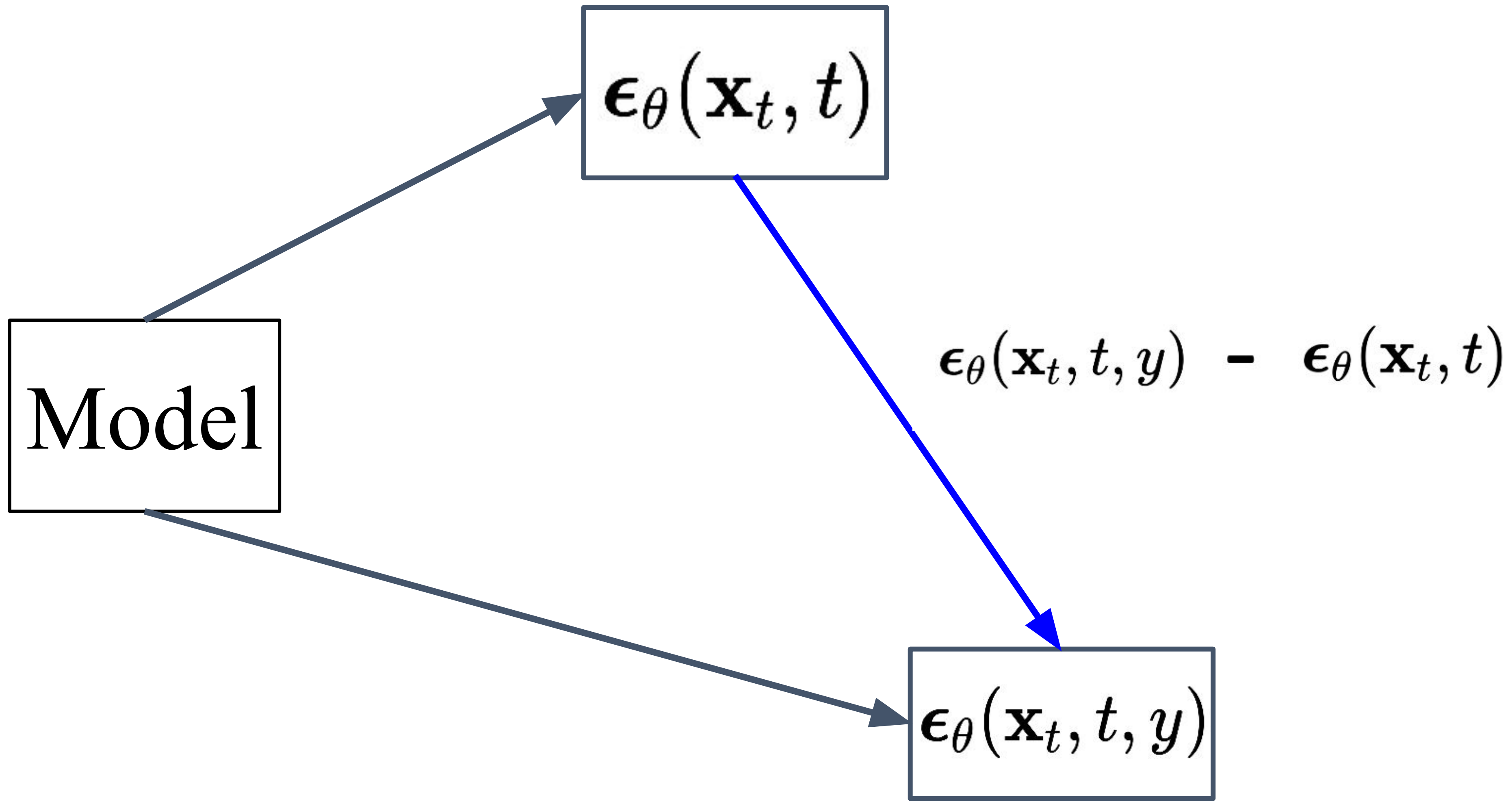
$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \cdot Z$$



$$\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y)$$

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \tilde{\epsilon}_\theta(x_t, t, y)\right) + \sigma_t \cdot Z$$

Conditioned Diffusion



$$\tilde{\epsilon}_\theta(\mathbf{x}_t, t, y) = \epsilon_\theta(\mathbf{x}_t, t) + \lambda(\epsilon_\theta(\mathbf{x}_t, t, y) - \epsilon_\theta(\mathbf{x}_t, t))$$

Ref: Ho, J. and Salimans, T., 2022. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.
University of Michigan - Diffusion Models Lecture: https://www.eecs.umich.edu/courses/eecs442-ahowens/fa23/slides/lec11-diffusion.pdf

Overview

- Motivation behind diffusion models ✓
- Denoising Diffusion Probabilistic Models ✓
  - Diffusion Process and Reverse Diffusion Process ✓
  - Training and Sampling a Diffusion Model ✓
  - Results from a Diffusion Model that we trained ✓
- Conditioned v/s Unconditioned Diffusion ✓
- Policy Learning
- Toy Problem
- Diffusion Policy on Real Robots
  - Advantages
  - Network Architect
  - Evaluation & Results

Policy Learning

Policy Learning

Policy: Given current observations, Policy tells a robot what to do next.

Policy Learning

Policy: Given current observations, Policy tells a robot what to do next.

$$A = \pi(O)$$

Policy Learning

Policy: Given current observations, Policy tells a robot what to do next.

$$A = \pi(O)$$

$$O$$

Image Source: https://www.lakeelsinorehonda.com/blogs/5482/9-things-you-didnt-know-about-self-driving-cars

Policy Learning

Policy: Given current observations, Policy tells a robot what to do next.

$$A = \pi(O)$$

O

π

Image Source: https://www.lakeelsinorehonda.com/blogs/5482/9-things-you-didnt-know-about-self-driving-cars

Policy Learning

Policy: Given current observations, Policy tells a robot what to do next.

$$A = \pi(O)$$

O → Policy → "Turn Left"

O π A

Image Source: https://www.lakeelsinorehonda.com/blogs/5482/9-things-you-didnt-know-about-self-driving-cars

Policy Learning

Policy: Given current observations, Policy tells a robot what to do next.

$$A = \pi(O)$$

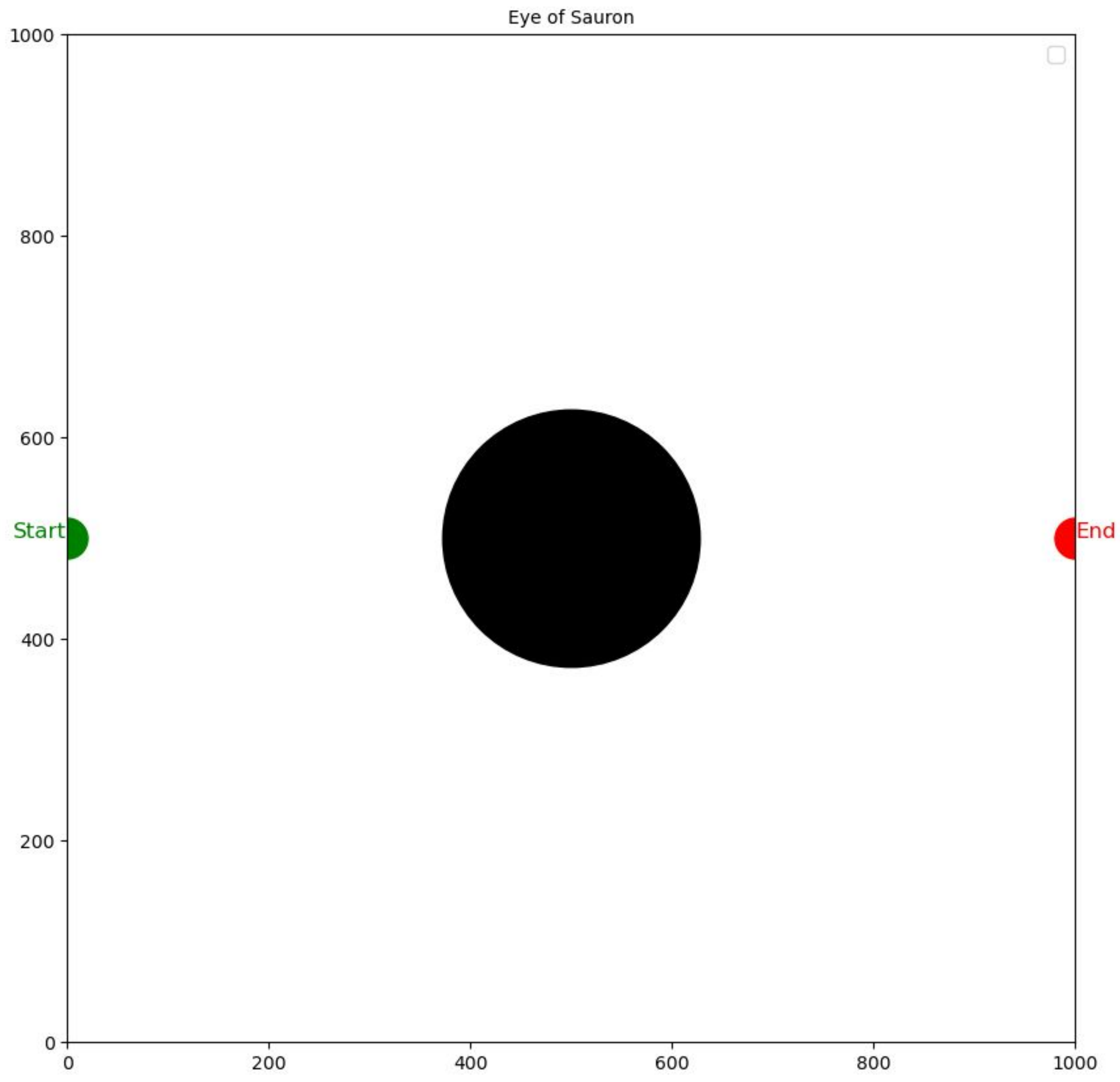
O → Policy π → "Turn Left" A

Policy can be hardcoded or can be learned through data(**Policy Learning**)

Image Source: https://www.lakeelsinorehonda.com/blogs/5482/9-things-you-didnt-know-about-self-driving-cars

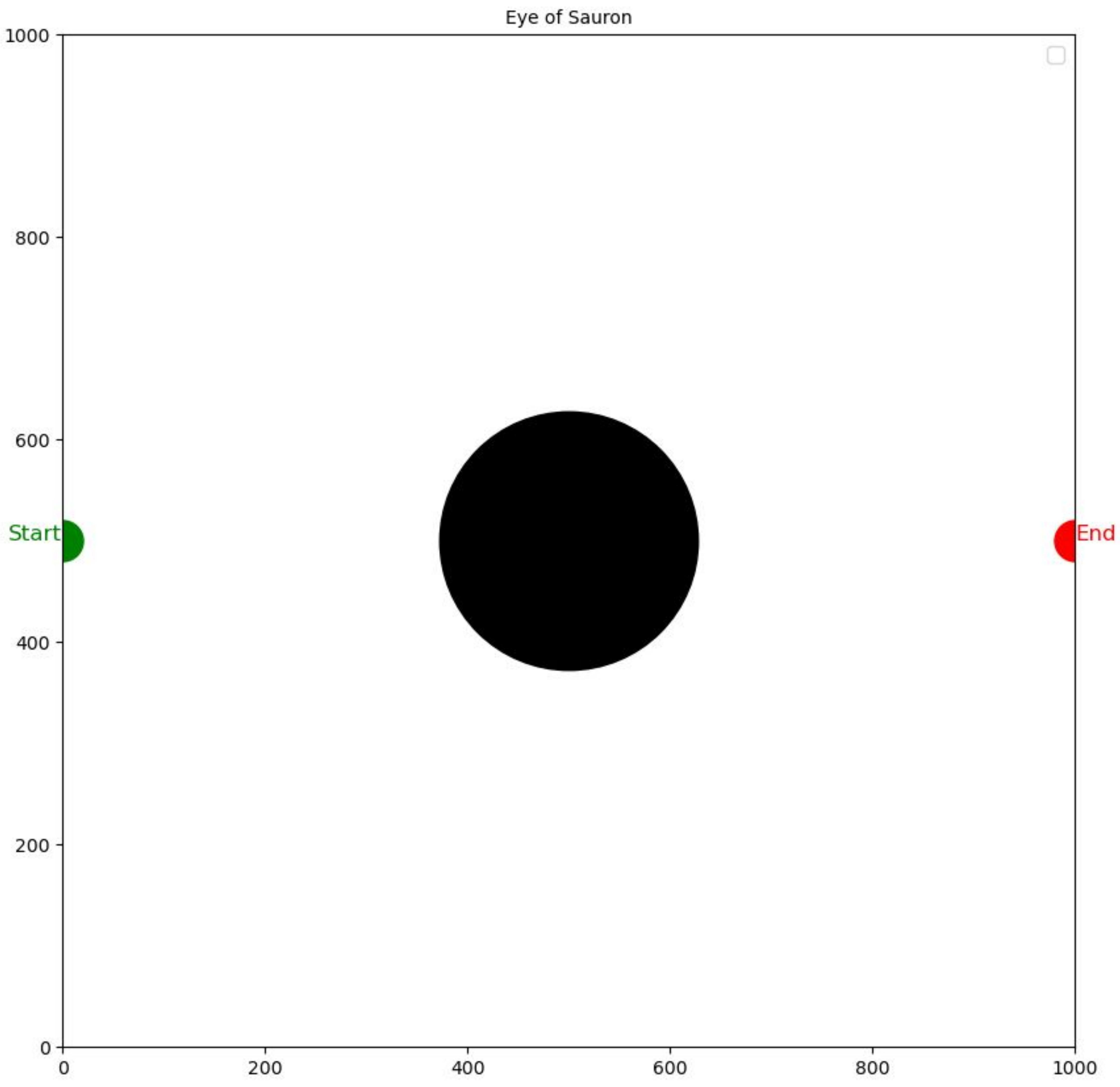
Toy Problem Definition

Toy Problem Definition



Toy Problem Definition

- Robot End Effector need to move from Start to the End

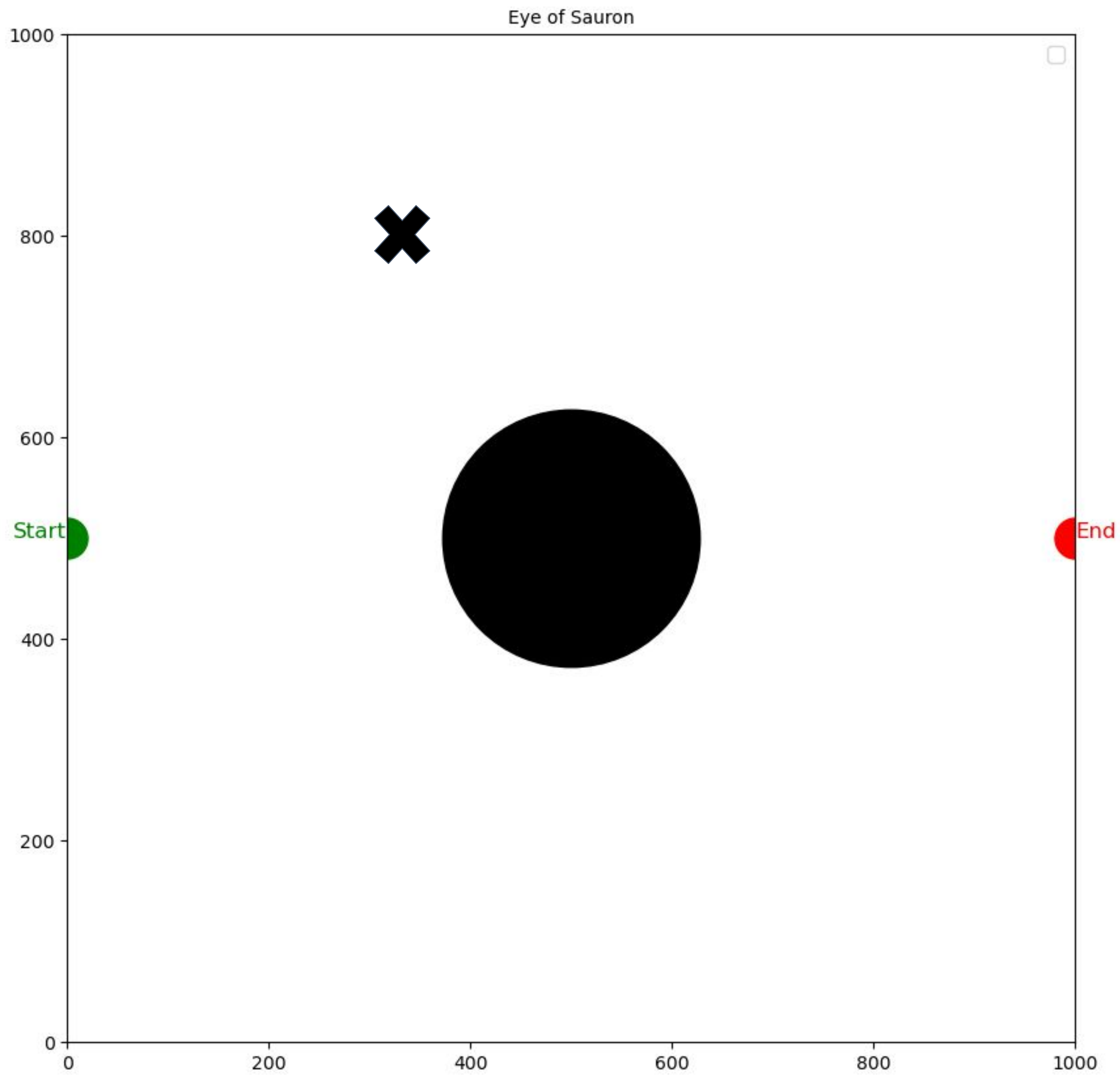


Toy Problem Definition

- Robot End Effector need to move from Start to the End .

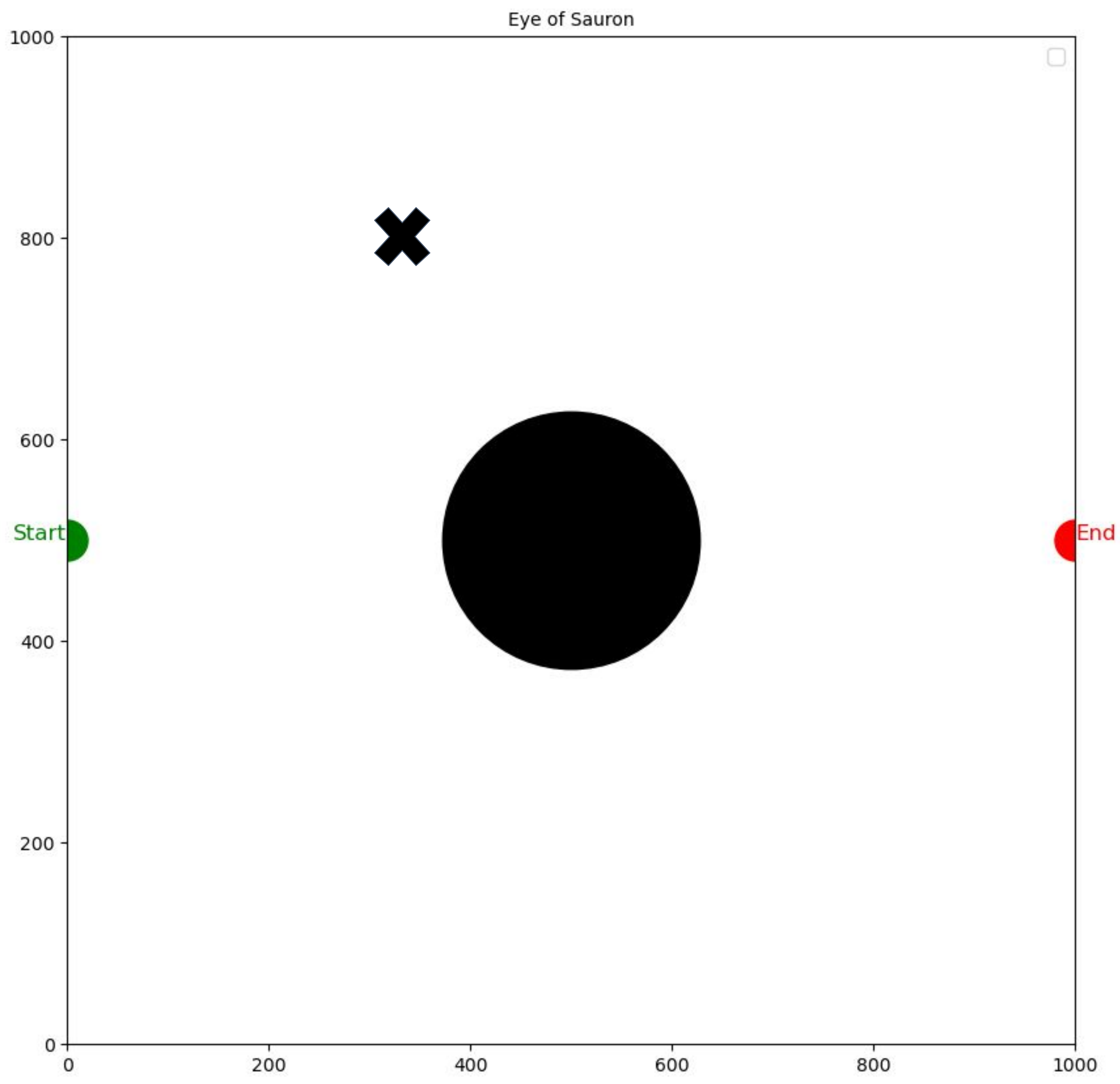
- End Effector ✖ location(*State*) is defined by two variable (x, y) : x, y ∈ [0, 1000]

    - Example: EFF is at (400, 800)



Toy Problem Definition

- Robot End Effector need to move from Start to the End .
- End Effector ✖ location(*State*) is defined by two variable (x, y) : x, y ∈ [0, 1000]
    - Example: EFF is at (400, 800)
- Action is defined by two variables (x, y) : x, y ∈ [0, 1000]
    - Example: Go to (405, 790)



Toy Problem Data

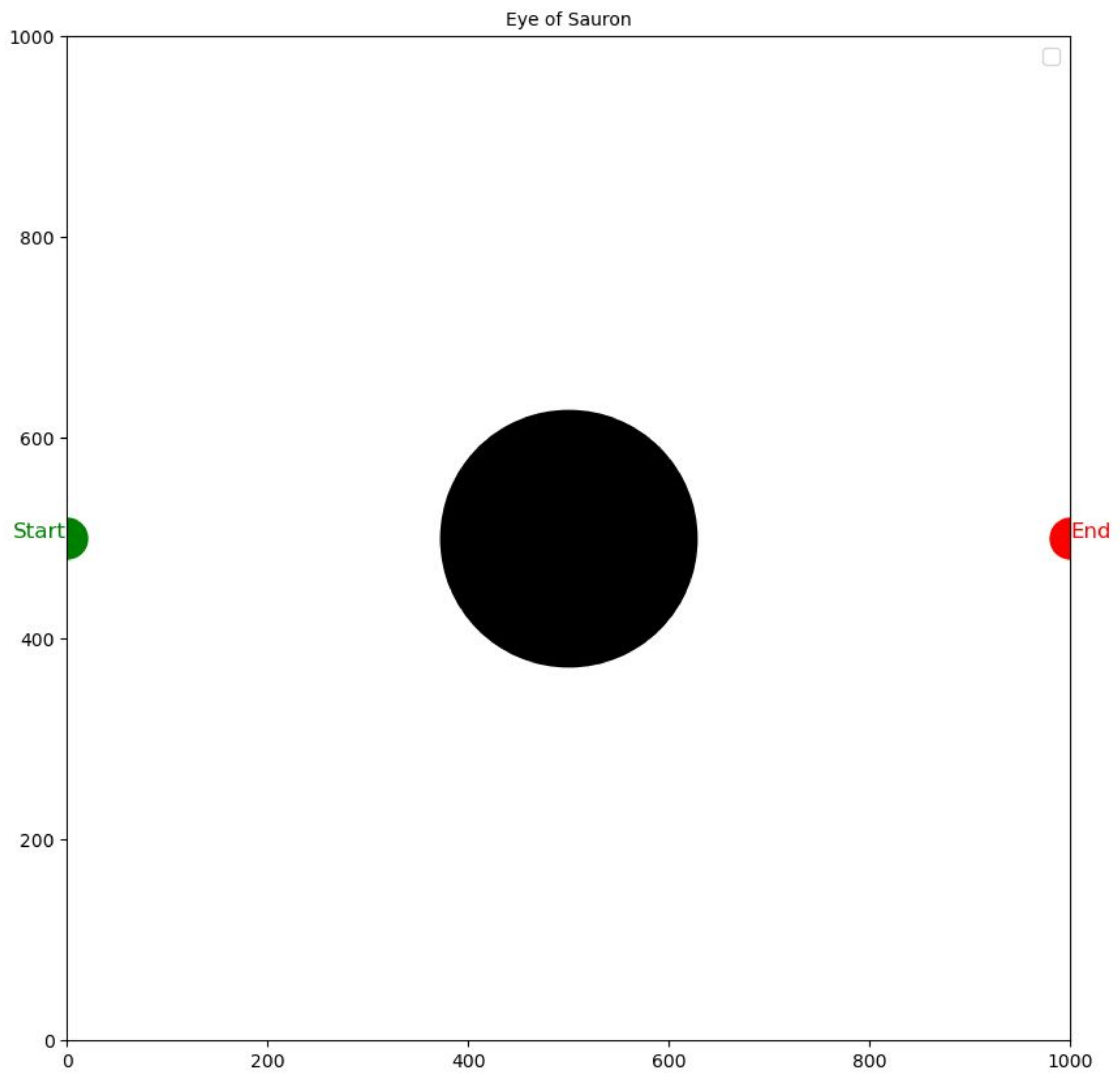
Toy Problem Data

- We formulate our problem as a Supervised Learning problem.

Toy Problem Data

- We formulate our problem as a Supervised Learning problem.

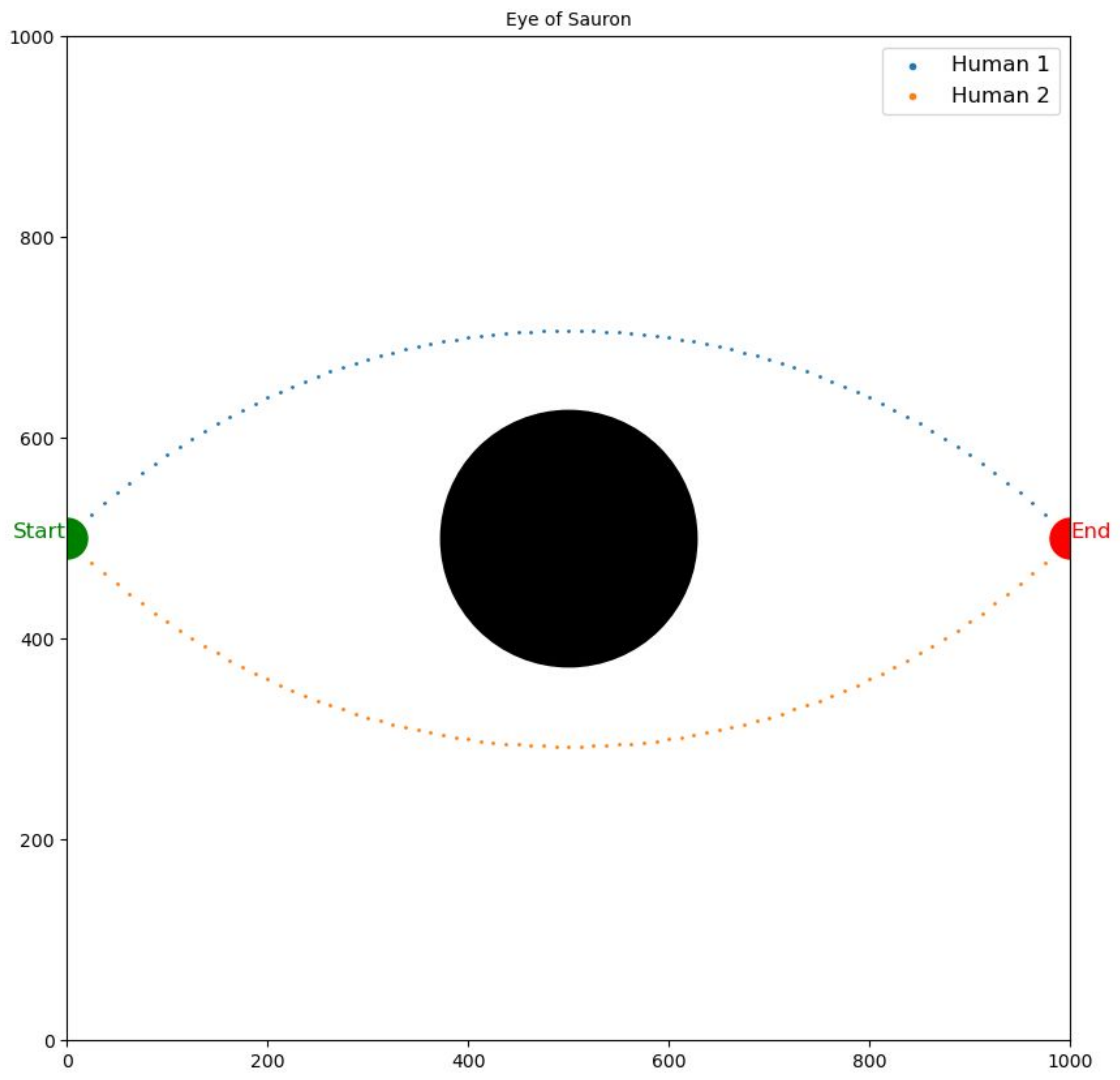
- Let's say we collected data from human demonstration to complete the task



Toy Problem Data

- We formulate our problem as a Supervised Learning problem.

- Let's say we collected data from human demonstration to complete the task***



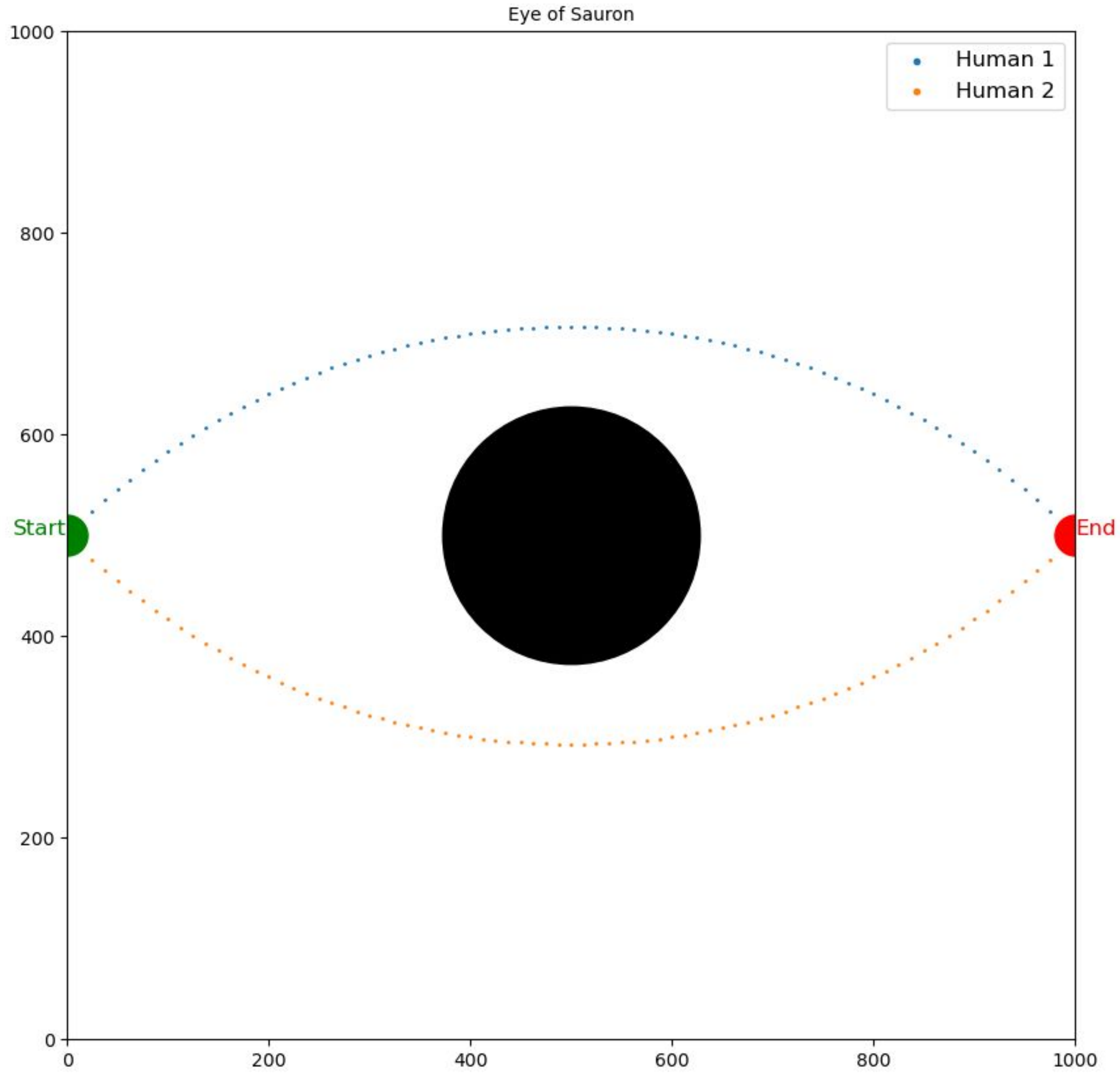
*** toy data was generated using two equations of circle centered at (500,0) and (500,1000).

Toy Problem Data

- We formulate our problem as a Supervised Learning problem.

- Let's say we collected data from human demonstration to complete the task***

- Every dot in the plot represents action taken by robot's end effector



*** toy data was generated using two equations of circle centered at (500,0) and (500,1000).

Toy Problem Data

Now we have “State” to “Action” mapping to train our Model.

Toy Problem Data

Now we have "State" to "Action" mapping to train our Model.

State/
Observation

X	0	12	25	38	50	62	75	88	100	112	125	138	150	162	175
Y	500	512	524	535	545	556	565	574	583	591	599	607	614	621	628

Toy Problem Data

Now we have “State” to “Action” mapping to train our Model.

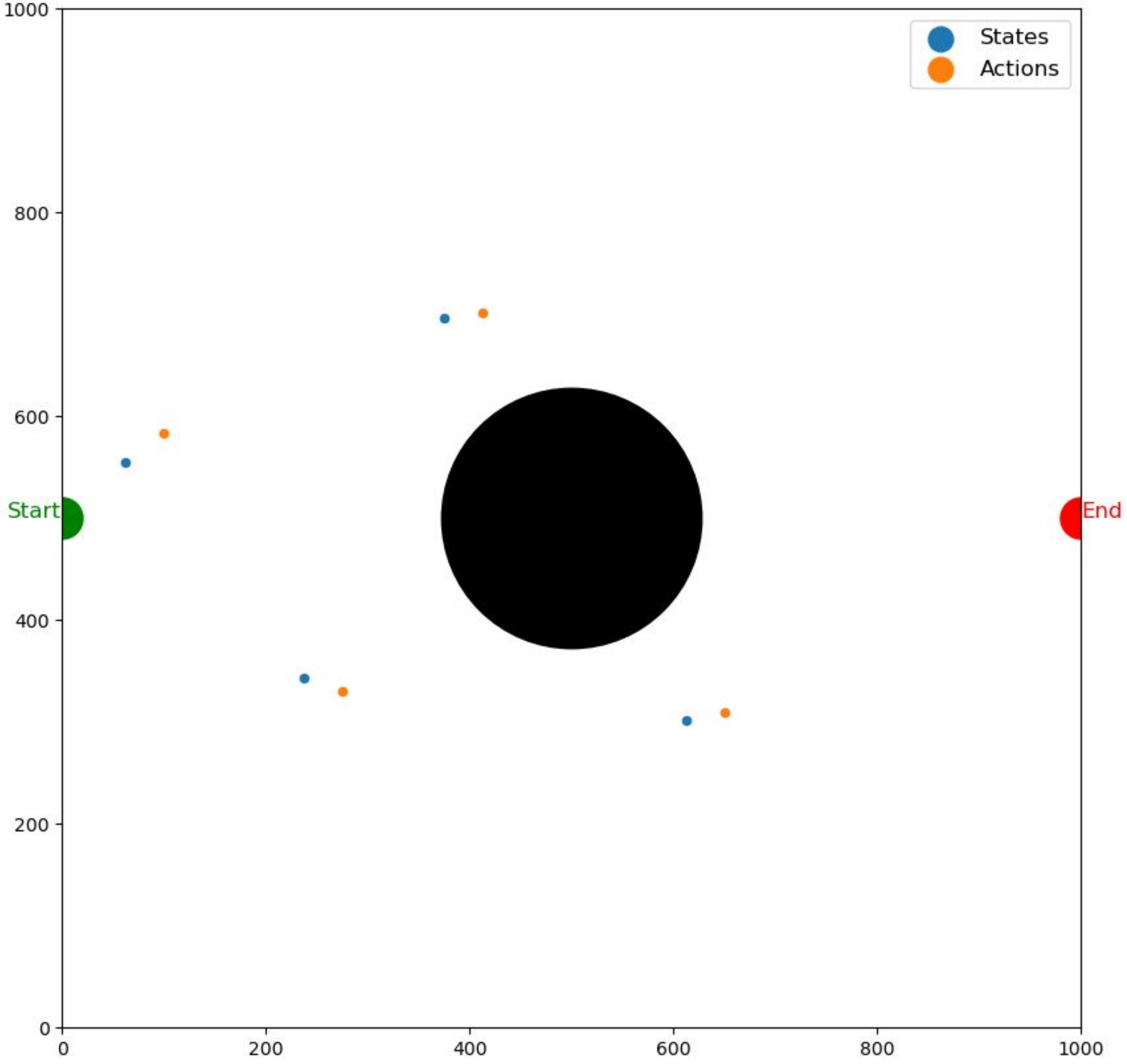
State/ Observation

X	0	12	25	38	50	62	75	88	100	112	125	138	150	162	175
Y	500	512	524	535	545	556	565	574	583	591	599	607	614	621	628

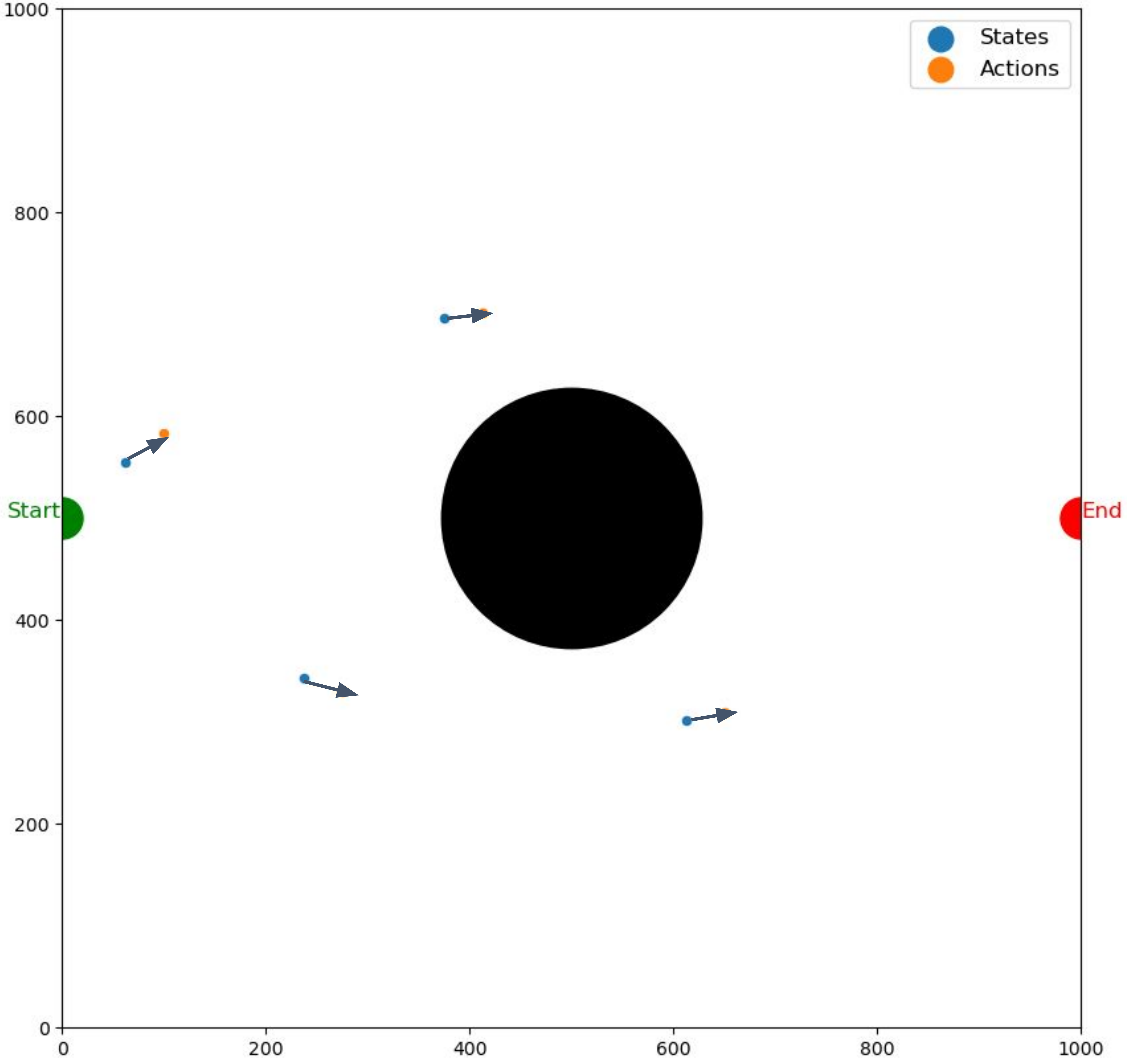
Action

X	38	50	62	75	88	100	112	125	138	150	162	175	188	200	212
Y	535	545	556	565	574	583	591	599	607	614	621	628	634	640	646

Toy Problem Data



Toy Problem Data



Some Nuances before Building a Model

Some Nuances before Building a Model

- It takes time to infer from the model, robot will be idle for a this time?

Some Nuances before Building a Model

- It takes time to infer from the model, robot will be idle for a this time?
- **Move the robot while model is predicting in parallel.**

Some Nuances before Building a Model

- It takes time to infer from the model, robot will be idle for a this time?
- **Move the robot while model is predicting in parallel.**

- Only having information about current location is enough??

Some Nuances before Building a Model

- It takes time to infer from the model, robot will be idle for a this time?
- **Move the robot while model is predicting in parallel.**

- Only having information about current location is enough??
- **Provide history of states**

Some Nuances before Building a Model

- It takes time to infer from the model, robot will be idle for a this time?
- **Move the robot while model is predicting in parallel.**

- Only having information about current location is enough??
- **Provide history of states**

- Only predicting the next Action is not efficient, motion will be Zig-Zaggy

Some Nuances before Building a Model

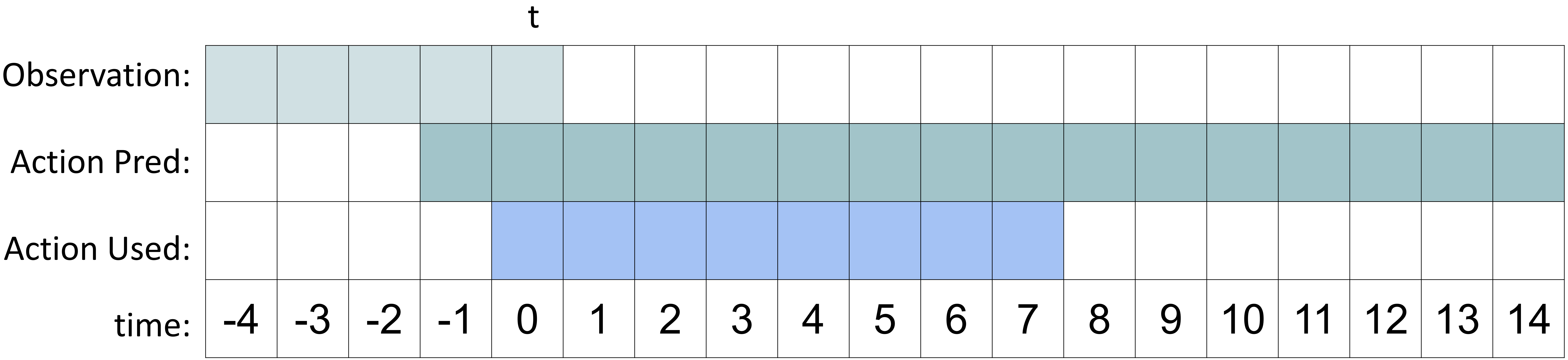
- It takes time to infer from the model, robot will be idle for a this time?
- **Move the robot while model is predicting in parallel.**

- Only having information about current location is enough??
- **Provide history of states**

- Only predicting the next Action is not efficient, motion will be Zig-Zaggy
- **Predict sequence of actions**

Some Nuances before Building a Model

- For Toy Example:
  - Observation Horizon: 5
  - Prediction Horizon: 16
  - Action Horizon: 8 (used when policy is rolled out)

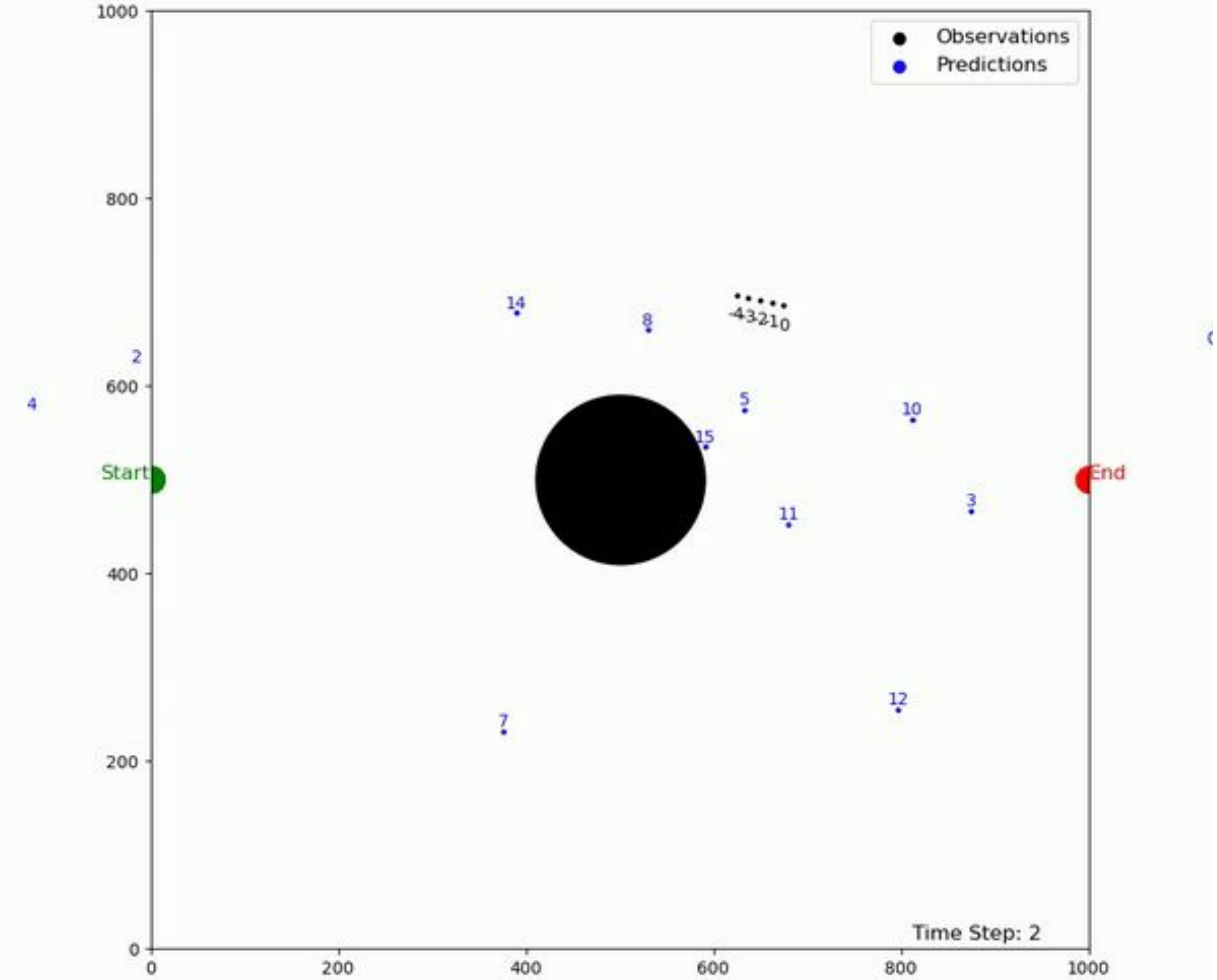


Toy Model Denoising

With the setup discuss we train a Diffusion Model to predict next 16 actions conditioned upon the last 5 states.

The output from the model is visualized in next slide.

Toy Model Denoising



Overview

- Motivation behind diffusion models ✓
- Denoising Diffusion Probabilistic Models ✓
  - Diffusion Process and Reverse Diffusion Process ✓
  - Training and Sampling a Diffusion Model ✓
  - Results from a Diffusion Model that we trained ✓
- Conditioned v/s Unconditioned Diffusion ✓
- Policy Learning ✓
- Toy Problem ✓
- Diffusion Policy on Real Robots
  - Advantages
  - Network Architect
  - Evaluation & Results

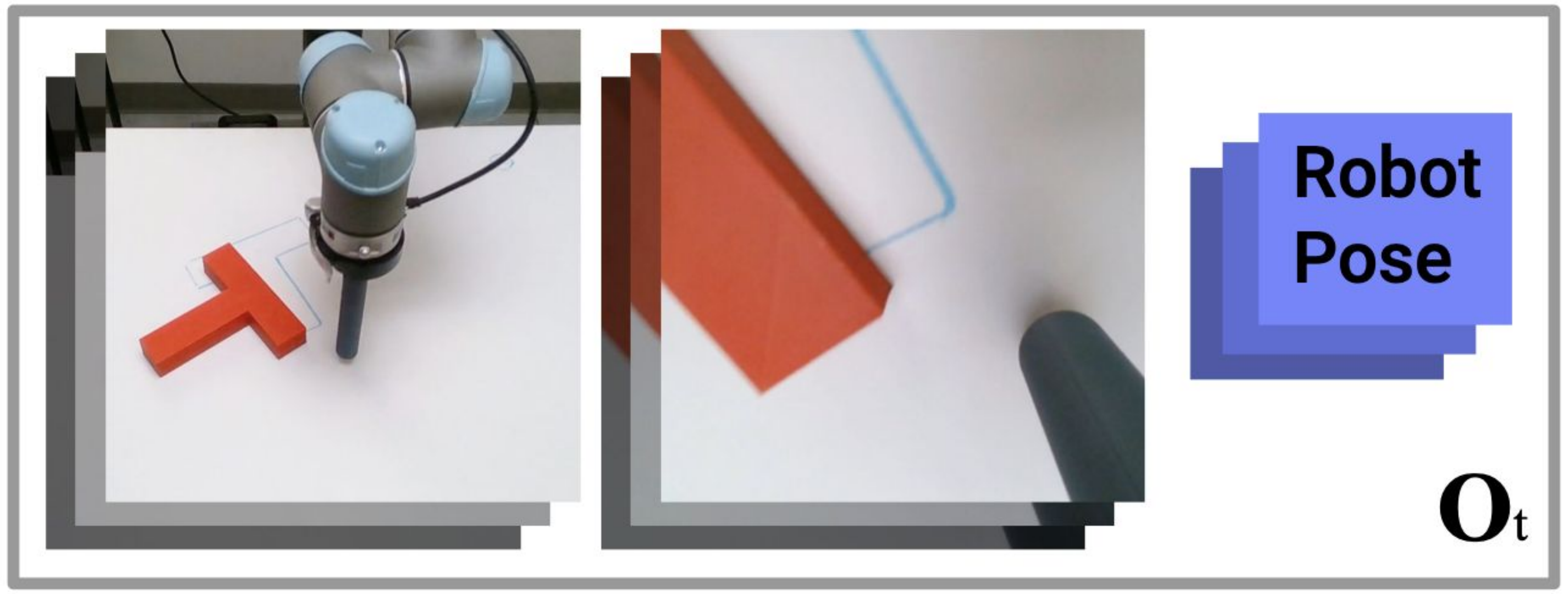
Transfer to Real Robot for Manipulation

Transfer to Real Robot for Manipulation

- Observation could include
  - Robot EEF location & orientation
  - Robot Joint Configuration
  - Image Input
  - Other information that you want your model to have….

- Action could be:
  - Robot EFF location & orientation
  - Robot Joint Configuration
  - Joint Velocities, etc.

Image Observation Sequence



C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

Diffusion Policy Paper

How does the Maths transfer from Image Space?

How does Maths Transfer from Image Space?

Unconditional Image generation

$$\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, k) + \mathcal{N}(0, \sigma^2 I))$$

C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

How does Maths Transfer from Image Space?

Unconditional Image generation

$$\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, k) + \mathcal{N}(0, \sigma^2 I))$$

Conditional Image Generation

$$\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, \mathbf{y}, k) + \mathcal{N}(0, \sigma^2 I))$$

C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

How does Maths Transfer from Image Space?

Two Modifications:

Unconditional Image generation

$$\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, k) + \mathcal{N}(0, \sigma^2 I))$$

Conditional Image Generation

$$\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, \mathbf{y}, k) + \mathcal{N}(0, \sigma^2 I))$$

C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

How does Maths Transfer from Image Space?

Two Modifications:

1. Image(x) -> Action(A)

Unconditional Image generation $$\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, k) + \mathcal{N}(0, \sigma^2 I))$$

Conditional Image Generation $$\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, \mathbf{y}, k) + \mathcal{N}(0, \sigma^2 I))$$

C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

How does Maths Transfer from Image Space?

Two Modifications:

1. Image(x) -> Action(A)
2. Denoising conditioned on observation(O)

Unconditional Image generation

$$\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, k) + \mathcal{N}(0, \sigma^2 I))$$

Conditional Image Generation

$$\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, \mathbf{y}, k) + \mathcal{N}(0, \sigma^2 I))$$

C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

How does Maths Transfer from Image Space?

Two Modifications:

1. Image(x) -> Action(A)
2. Denoising conditioned on observation(O)

Unconditional Image generation

$$\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, k) + \mathcal{N}(0, \sigma^2 I))$$

Conditional Image Generation

$$\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, \mathbf{y}, k) + \mathcal{N}(0, \sigma^2 I))$$

Action Prediction

$$\mathbf{A}_t^{k-1} = \alpha(\mathbf{A}_t^k - \gamma\varepsilon_\theta(\mathbf{O}_t, \mathbf{A}_t^k, k) + \mathcal{N}(0, \sigma^2 I))$$

C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

How does Maths Transfer from Image Space?

Two Modifications:

1. Image(x) -> Action(A)
2. Denoising conditioned on observation(O)

Unconditional Image generation	$\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, k) + \mathcal{N}(0, \sigma^2 I))$
Conditional Image Generation	$\mathbf{x}^{k-1} = \alpha(\mathbf{x}^k - \gamma\varepsilon_\theta(\mathbf{x}^k, \mathbf{y}, k) + \mathcal{N}(0, \sigma^2 I))$
Action Prediction	$\mathbf{A}_t^{k-1} = \alpha(\mathbf{A}_t^k - \gamma\varepsilon_\theta(\mathbf{O}_t, \mathbf{A}_t^k, k) + \mathcal{N}(0, \sigma^2 I))$

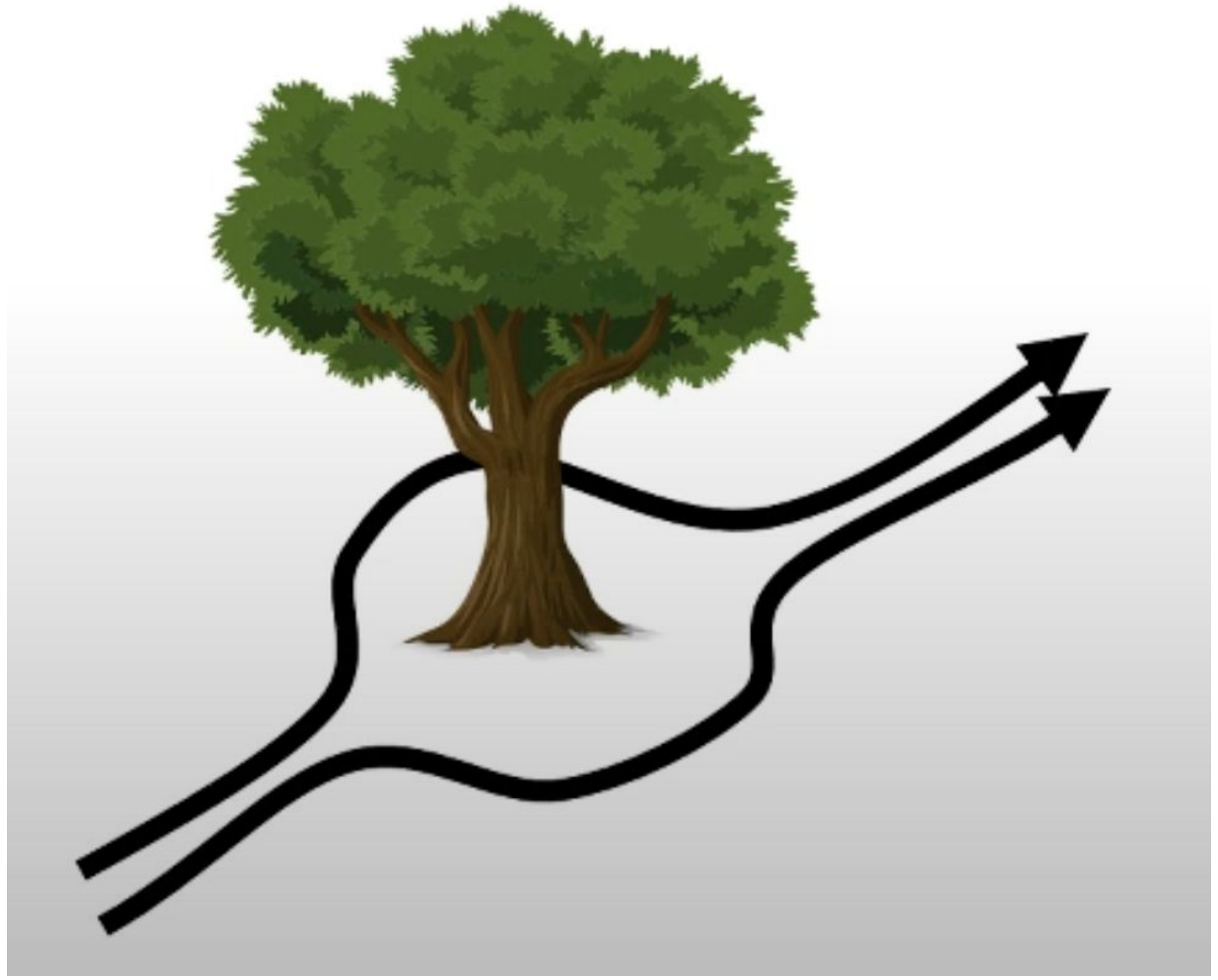
...suomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

Diffusion Policy Paper

What advantages does using diffusion for learning policy have?

1. MultiModal Action Distribution

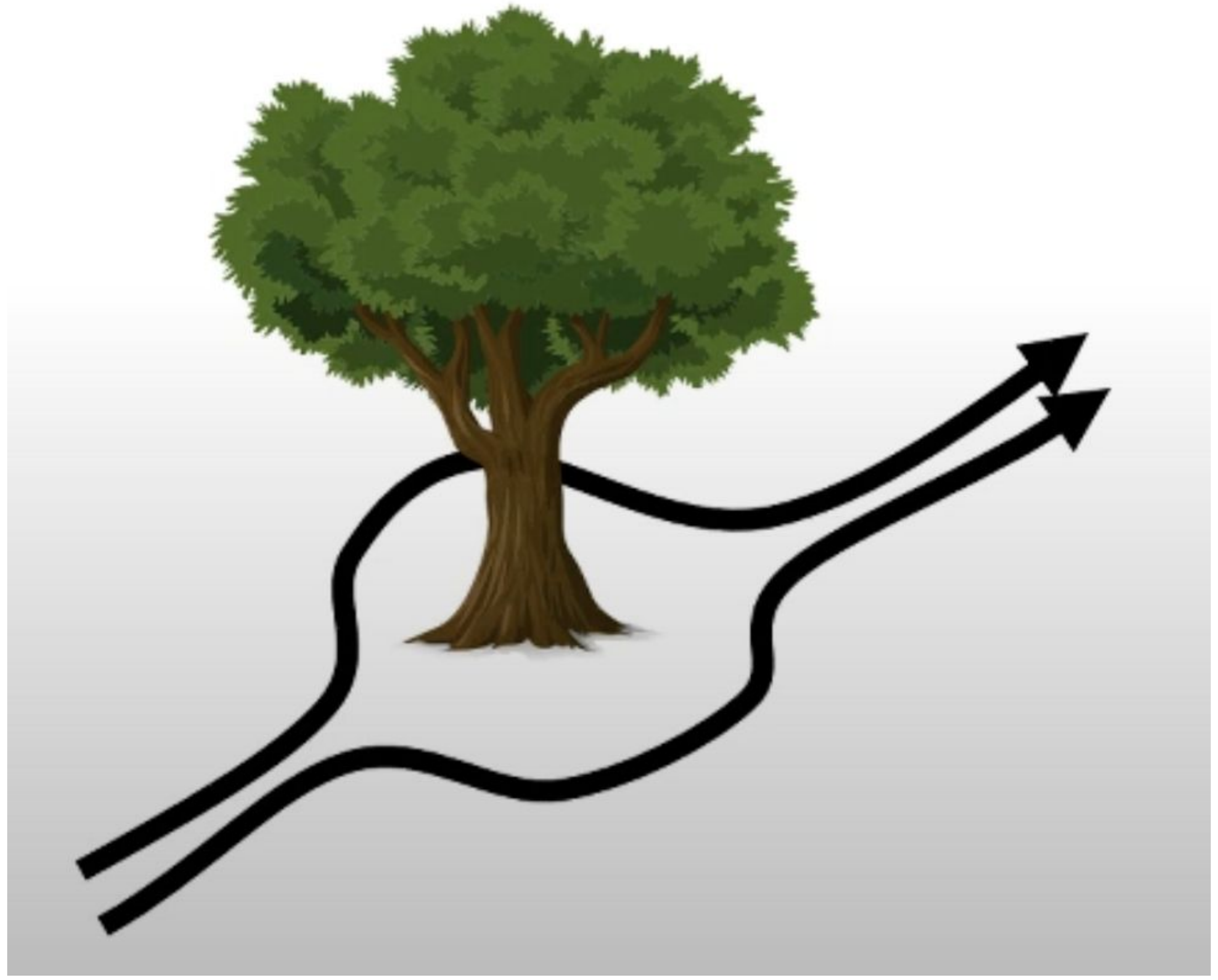
If half of the data say move right and the other half says move left:



C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

1. MultiModal Action Distribution

If half of the data say move right and the other half says move left:
Simple MLPs -> Move Center



C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

1. MultiModal Action Distribution

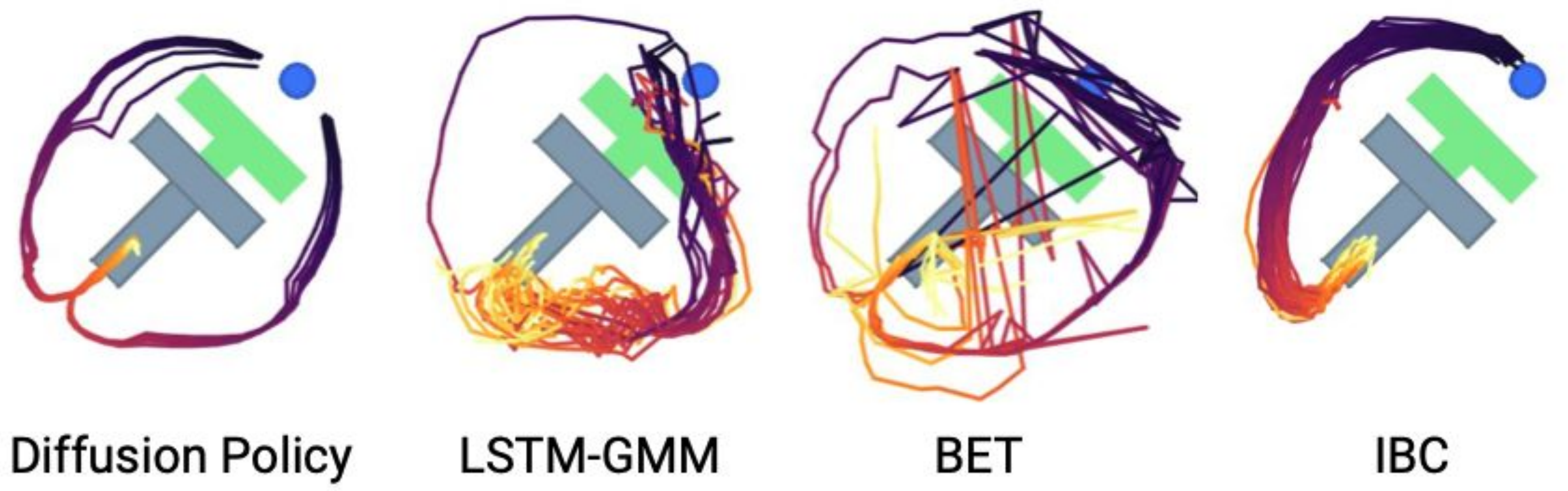
If half of the data say move right and the other half says move left:
Simple MLPs -> Move Center

Diffusion Policy

C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

1. MultiModal Action Distribution

If half of the data say move right and the other half says move left:
Simple MLPs -> Move Center



C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

1. MultiModal Action Distribution

If half of the data say move right and the other half says move left:
Simple MLPs -> Move Center

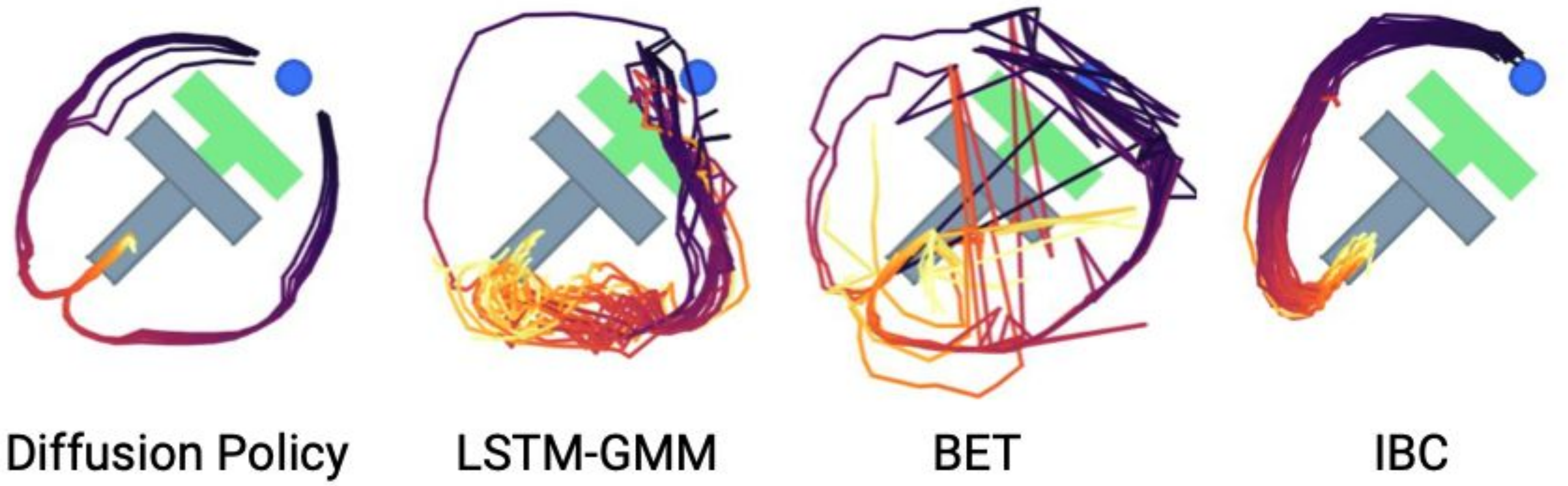
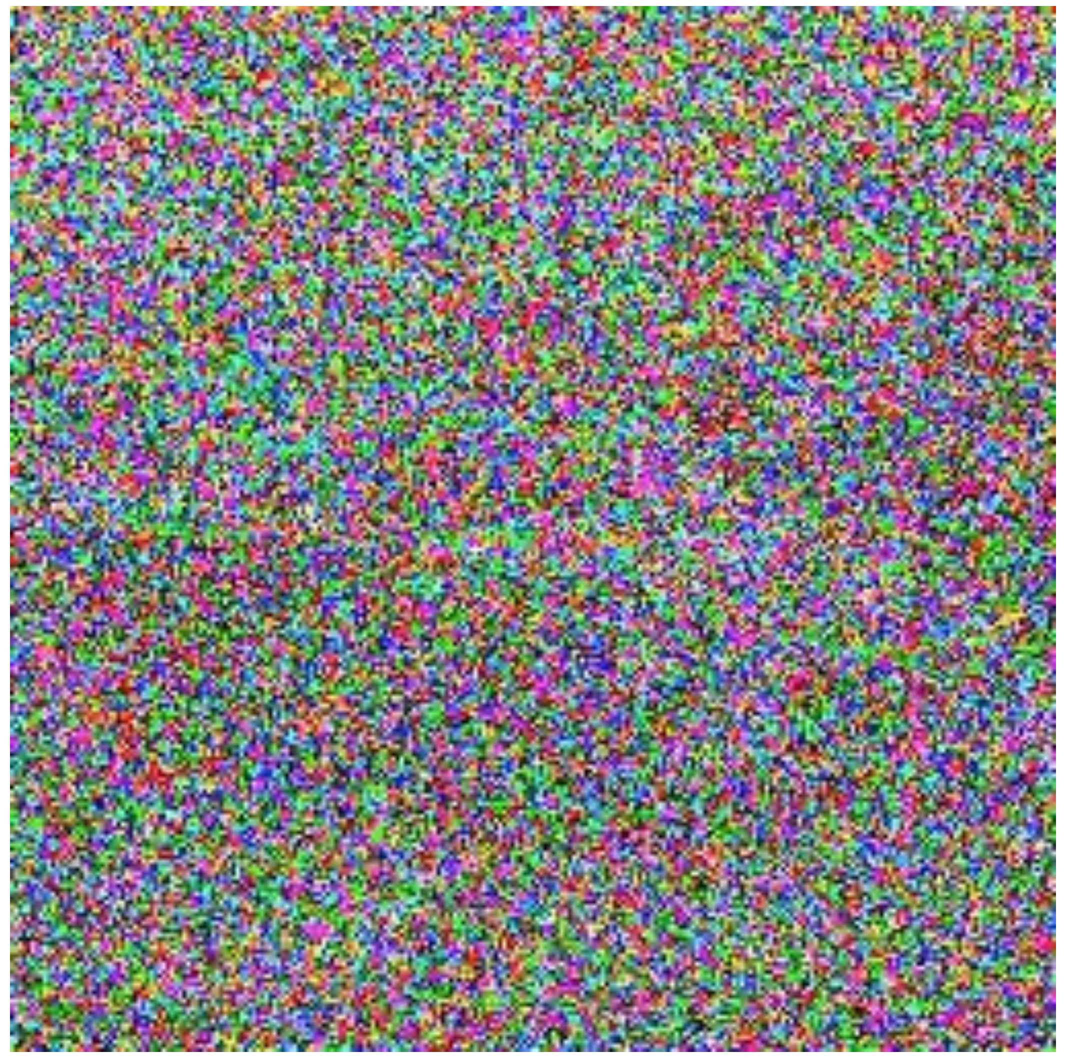


Figure 3. Multimodal behavior. At the given state, the end-effector (blue) can either go left or right to push the block. **Diffusion Policy** learns both modes and commits to only one mode within each rollout. In contrast, both **LSTM-GMM** Mandlekar et al. (2021) and **IBC** Florence et al. (2021) are biased toward one mode, while **BET** Shafiullah et al. (2022) fails to commit to a single mode due to its lack of temporal action consistency. Actions generated by rolling out 40 steps for the best-performing checkpoint.

C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

2. High Dimensional Output Space



For a Image size of: 64*64

Output Dimensions: 12288

2. High Dimensional Output Space

Allows joint inference of sequence of Actions.

X	807	33	274	458	-629	1062	-413	3	698	735	262	293	1603	339	492	571
Y	671	483	551	404	514	765	517	545	348	492	493	425	519	494	234	842

Toy example: (2 * 16) -> 32 dimensional output

2. High Dimensional Output Space

Allows joint inference of sequence of Actions.

X	807	33	274	458	-629	1062	-413	3	698	735	262	293	1603	339	492	571
Y	671	483	551	404	514	765	517	545	348	492	493	425	519	494	234	842

Toy example: (2 * 16) -> 32 dimensional output

6-DOF Manipulator: (6 * 16) -> 96 dimensional output

2. High Dimensional Output Space

Allows joint inference of sequence of Actions.

X	807	33	274	458	-629	1062	-413	3	698	735	262	293	1603	339	492	571
Y	671	483	551	404	514	765	517	545	348	492	493	425	519	494	234	842

Toy example: (2 * 16) -> 32 dimensional output

6-DOF Manipulator: (6 * 16) -> 96 dimensional output

Dual 6-DOF arm Manipulator: with Gripper Control (14 * 16)-> 224 dimensional output

3. Stable Training

Because diffusion model directly predict the gradient of energy function

C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

3. Stable Training

Because diffusion model directly predict the gradient of energy function

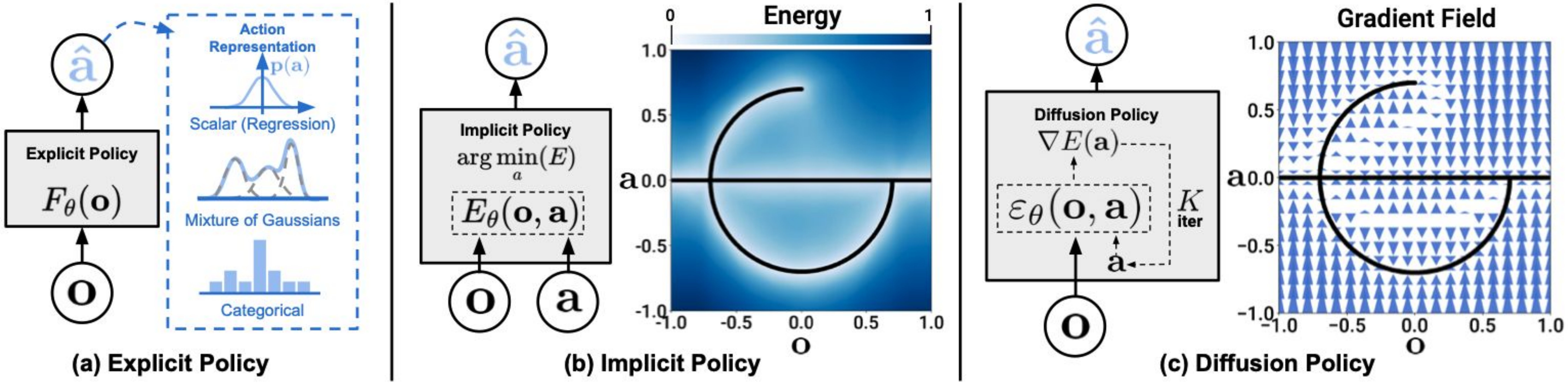


Figure 1. Policy Representations. a) Explicit policy with different types of action representations. b) Implicit policy learns an energy function conditioned on both action and observation and optimizes for actions that minimize the energy landscape c) Diffusion policy refines noise into actions via a learned gradient field. This formulation provides stable training, allows the learned policy to accurately model multimodal action distributions, and accommodates high-dimensional action sequences.

C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

3. Stable Training

Because diffusion model directly predict the gradient of energy function

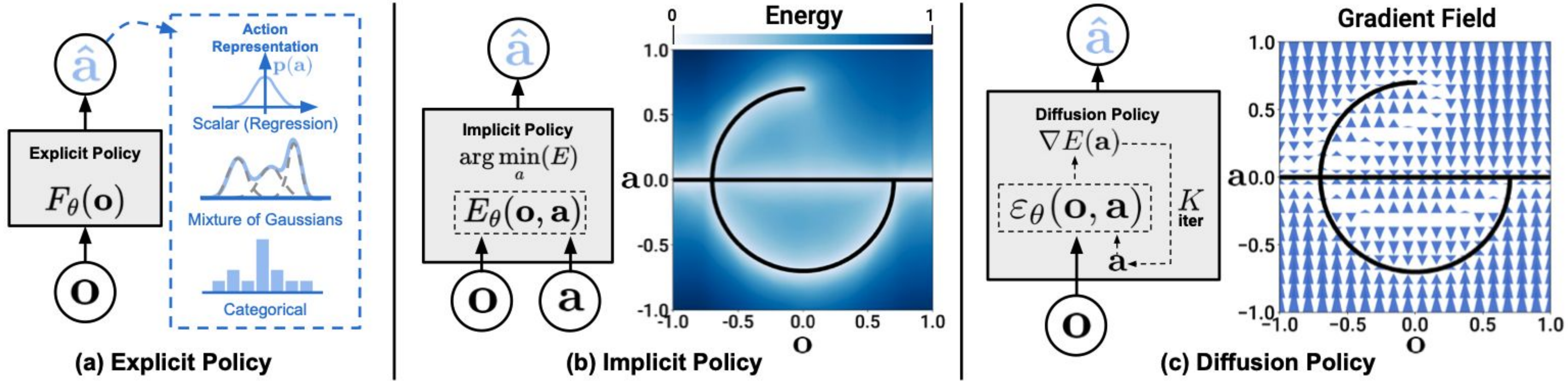
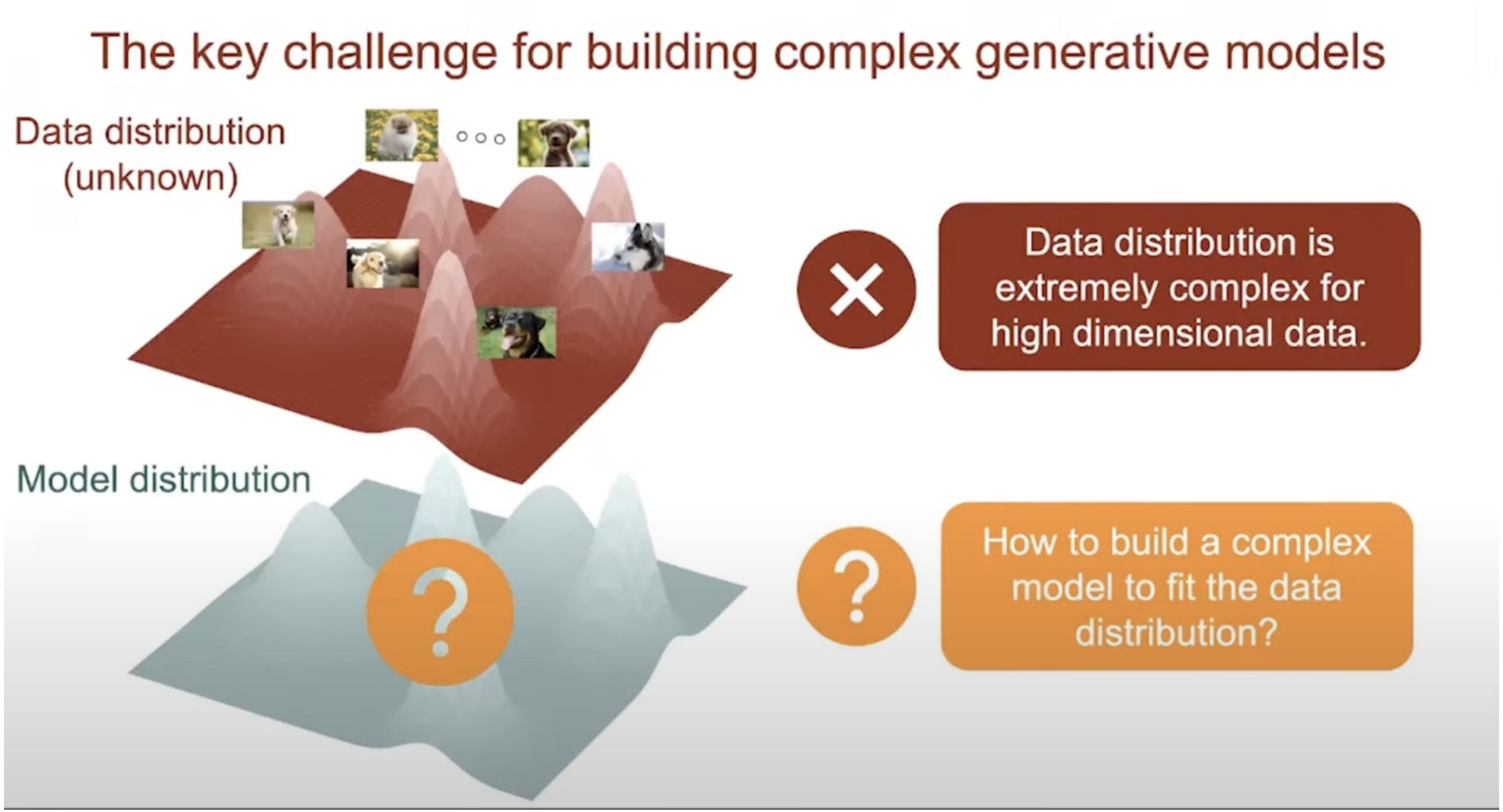


Figure 1. Policy Representations. a) Explicit policy with different types of action representations. b) Implicit policy learns an energy function conditioned on both action and observation and optimizes for actions that minimize the energy landscape c) Diffusion policy refines noise into actions via a learned gradient field. This formulation provides stable training, allows the learned policy to accurately model multimodal action distributions, and accommodates high-dimensional action sequences.

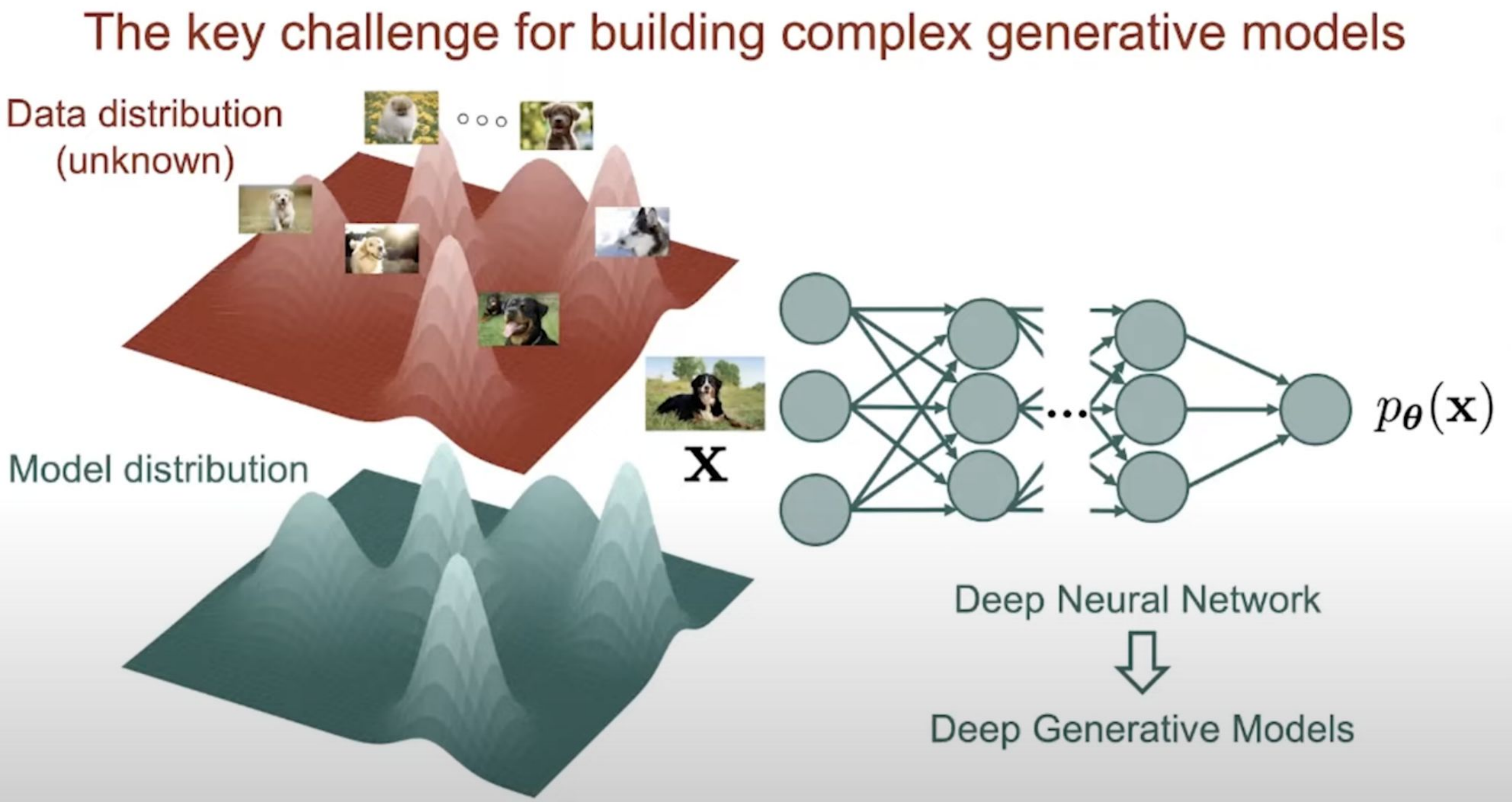
C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [O[illegible]]. Available: https://arxiv.org/abs/2303.04137

Small Maths Detour



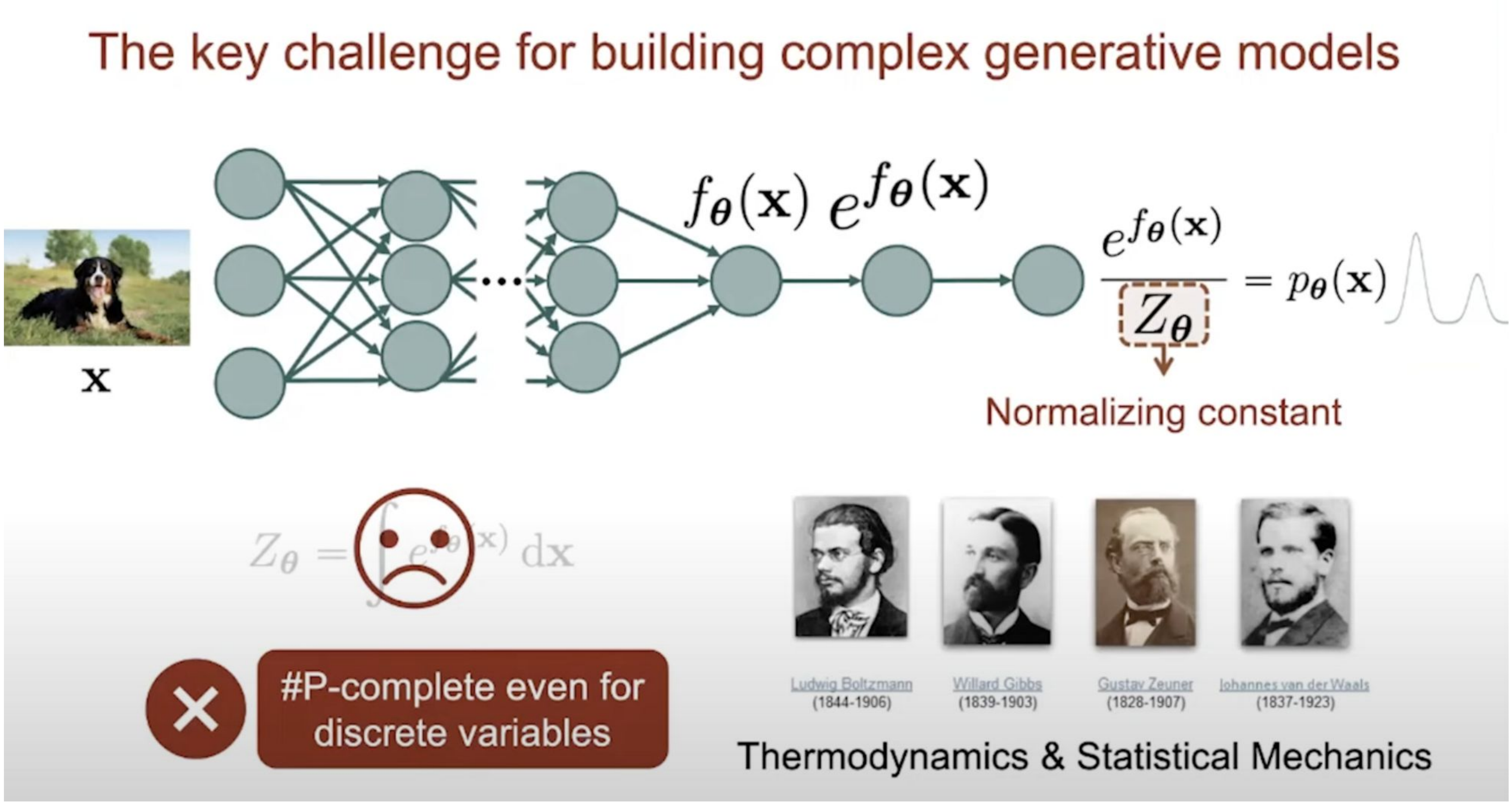
Slides by Prof. Yang Song: https://www.youtube.com/watch?v=wMmqCMwuM2Q&t=4141s

Small Maths Detour



Slides by Prof. Yang Song: https://www.youtube.com/watch?v=wMmqCMwuM2Q&t=4141s

Small Maths Detour



Slides by Prof. Yang Song: https://www.youtube.com/watch?v=wMmqCMwuM2Q&t=4141s

Small Maths Detour

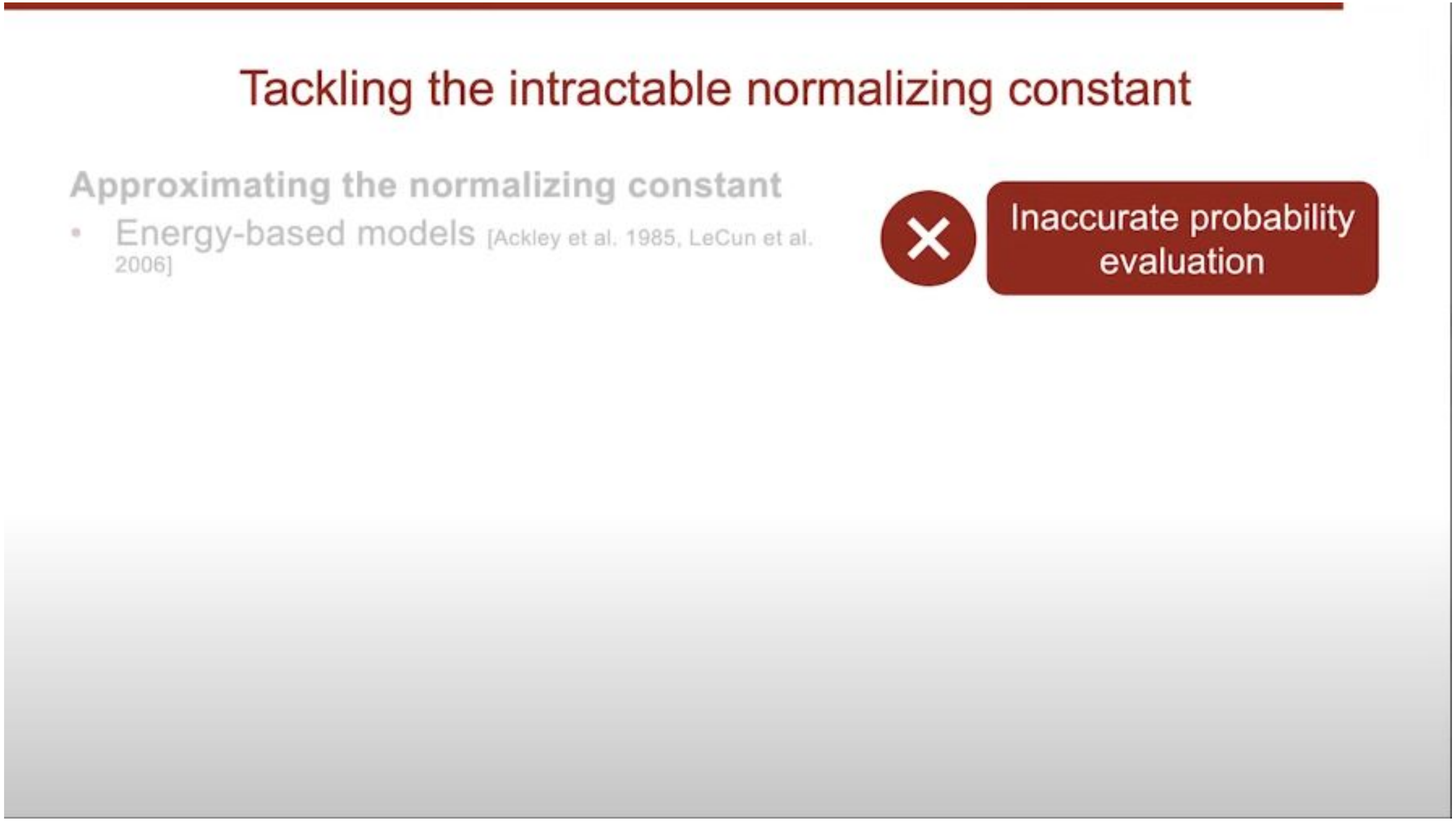
Tackling the intractable normalizing constant

Approximating the normalizing constant

- Energy-based models [Ackley et al. 1985, LeCun et al. 2006]

Slides by Prof. Yang Song: https://www.youtube.com/watch?v=wMmqCMwuM2Q&t=4141s

Small Maths Detour



Slides by Prof. Yang Song: https://www.youtube.com/watch?v=wMmqCMwuM2Q&t=4141s

Small Maths Detour

Tackling the intractable normalizing constant

Approximating the normalizing constant

- Energy-based models [Ackley et al. 1985, LeCun et al. 2006]

✕ Inaccurate probability evaluation

Using restricted neural network models

- Autoregressive models [Bengio & Bengio 2000, van den Oord et al. 2016]
- Normalizing flow models [Dinh et al. 2014, Rezende & Mohamed 2015]
- Variational auto-encoders [Kingma & Welling 2014, Rezende et al. 2014]

Slides by Prof. Yang Song: https://www.youtube.com/watch?v=wMmqCMwuM2Q&t=4141s

Small Maths Detour

Tackling the intractable normalizing constant

Approximating the normalizing constant

- Energy-based models [Ackley et al. 1985, LeCun et al. 2006]

Using restricted neural network models

- Autoregressive models [Bengio & Bengio 2000, van den Oord et al. 2016]
- Normalizing flow models [Dinh et al. 2014, Rezende & Mohamed 2015]
- Variational auto-encoders [Kingma & Welling 2014, Rezende et al. 2014]

✕ Inaccurate probability evaluation

✕ Restricted model family

Slides by Prof. Yang Song: https://www.youtube.com/watch?v=wMmqCMwuM2Q&t=4141s

Small Maths Detour

Tackling the intractable normalizing constant

Approximating the normalizing constant

- Energy-based models [Ackley et al. 1985, LeCun et al. 2006]

✕ Inaccurate probability evaluation

Using restricted neural network models

- Autoregressive models [Bengio & Bengio 2000, van den Oord et al. 2016]
- Normalizing flow models [Dinh et al. 2014, Rezende & Mohamed 2015]
- Variational auto-encoders [Kingma & Welling 2014, Rezende et al. 2014]

✕ Restricted model family

Modeling the Generation Process Only

- Generative Adversarial Networks (GANS) [Goodfellow et al. 2014]

Slides by Prof. Yang Song: https://www.youtube.com/watch?v=wMmqCMwuM2Q&t=4141s

Small Maths Detour

Tackling the intractable normalizing constant

Approximating the normalizing constant

- Energy-based models [Ackley et al. 1985, LeCun et al. 2006]

✕ Inaccurate probability evaluation

Using restricted neural network models

- Autoregressive models [Bengio & Bengio 2000, van den Oord et al. 2016]
- Normalizing flow models [Dinh et al. 2014, Rezende & Mohamed 2015]
- Variational auto-encoders [Kingma & Welling 2014, Rezende et al. 2014]

✕ Restricted model family

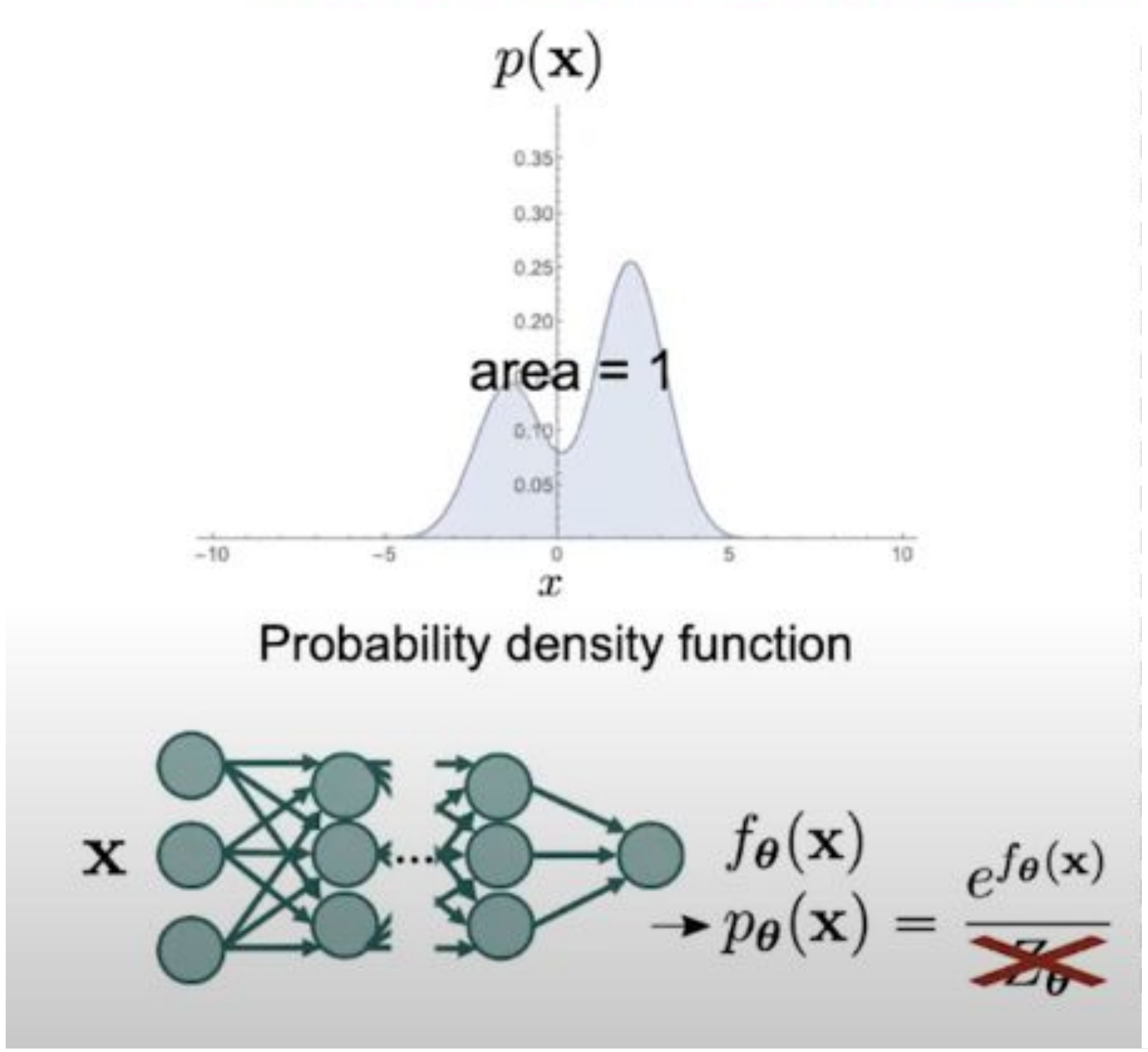
Modeling the Generation Process Only

- Generative Adversarial Networks (GANS) [Goodfellow et al. 2014]

✕ Cannot evaluate probabilities

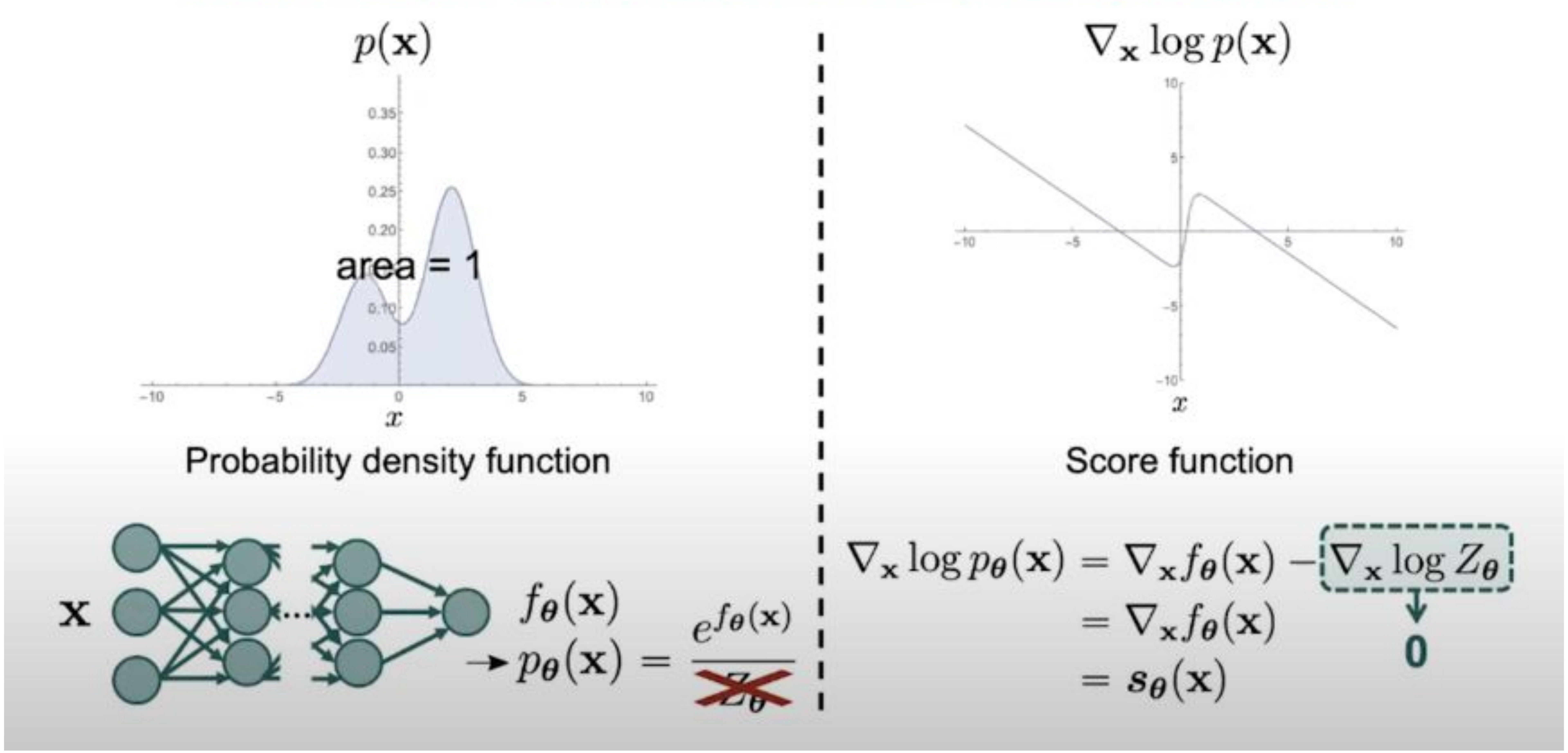
Slides by Prof. Yang Song: https://www.youtube.com/watch?v=wMmqCMwuM2Q&t=4141s

Small Maths Detour



Slides by Prof. Yang Song: https://www.youtube.com/watch?v=wMmqCMwuM2Q&t=4141s

Small Maths Detour



Slides by Prof. Yang Song: https://www.youtube.com/watch?v=wMmqCMwuM2Q&t=4141s

3. Stable Training

Because diffusion model directly predict the gradient of energy function

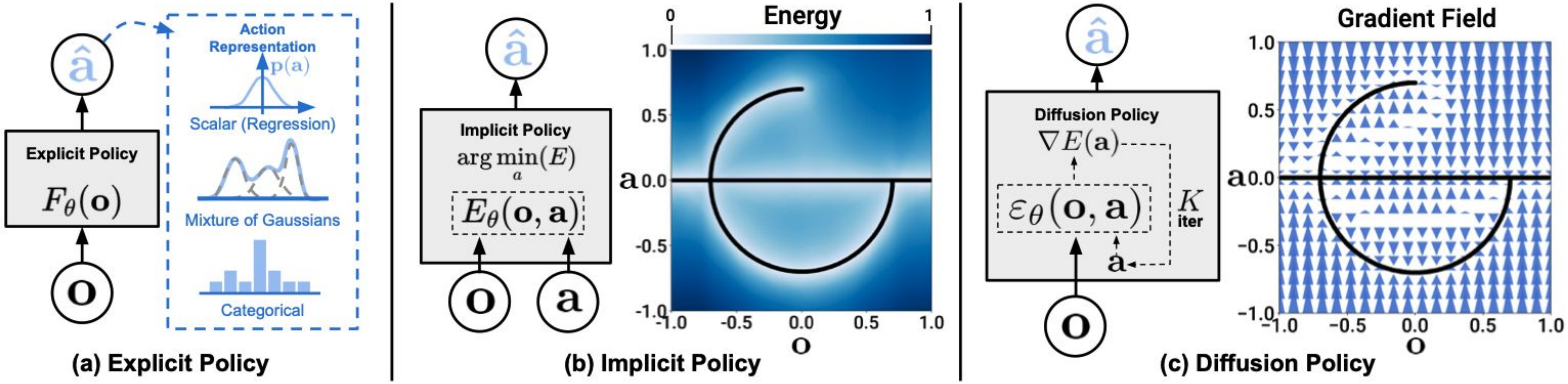


Figure 1. Policy Representations. a) Explicit policy with different types of action representations. b) Implicit policy learns an energy function conditioned on both action and observation and optimizes for actions that minimize the energy landscape c) Diffusion policy refines noise into actions via a learned gradient field. This formulation provides stable training, allows the learned policy to accurately model multimodal action distributions, and accommodates high-dimensional action sequences.

C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [O]. Available: https://arxiv.org/abs/2303.04137

3. Stable Training

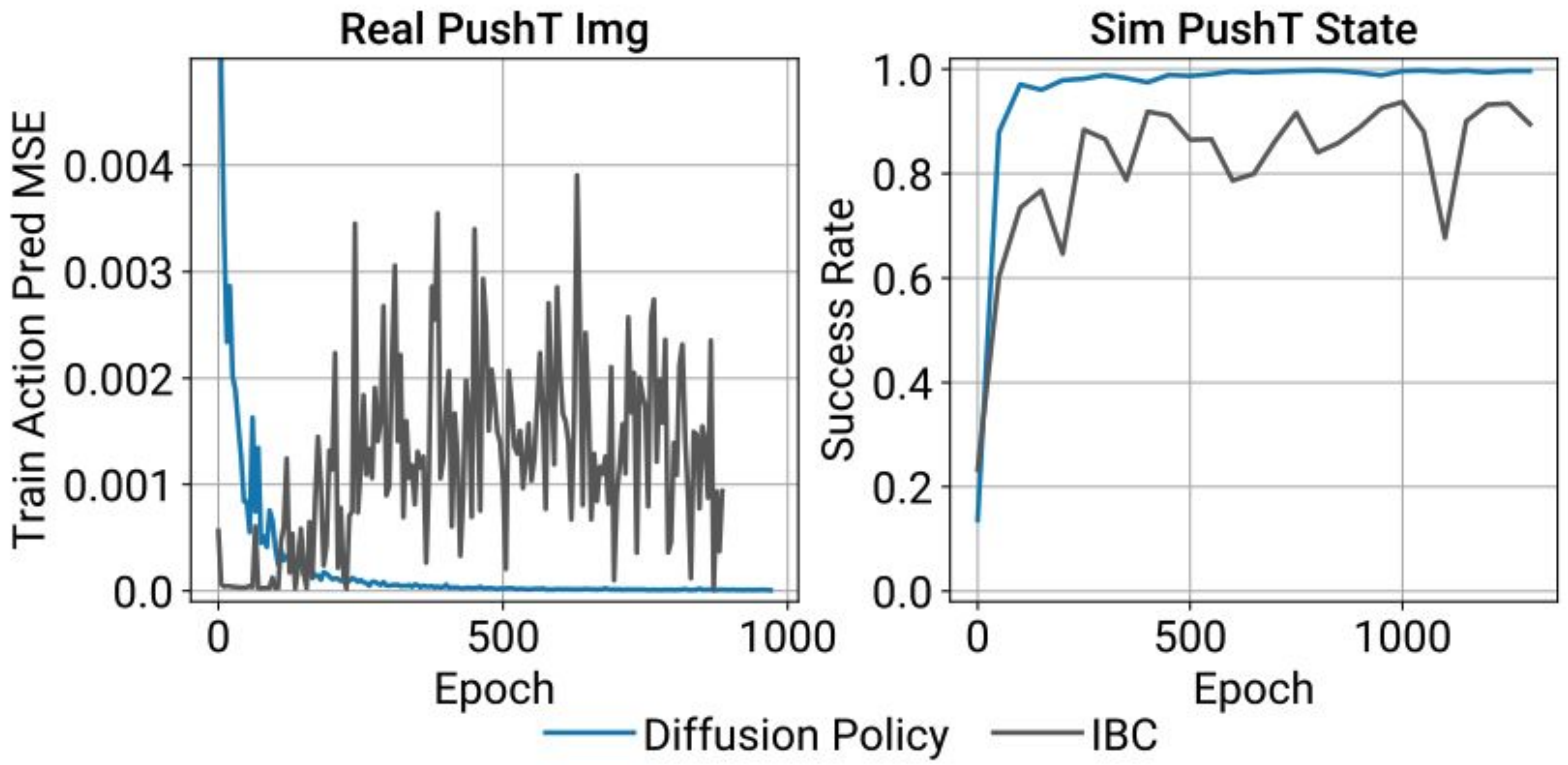


Figure 6. Training Stability. Left: IBC fails to infer training actions with increasing accuracy despite smoothly decreasing training loss for energy function. Right: IBC's evaluation success rate oscillates, making checkpoint selection difficult (evaluated using policy rollouts in simulation).

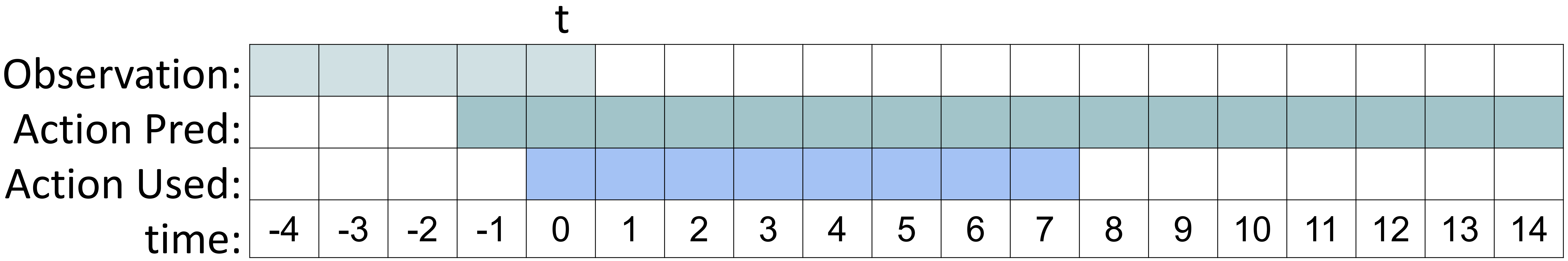
C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

Overview

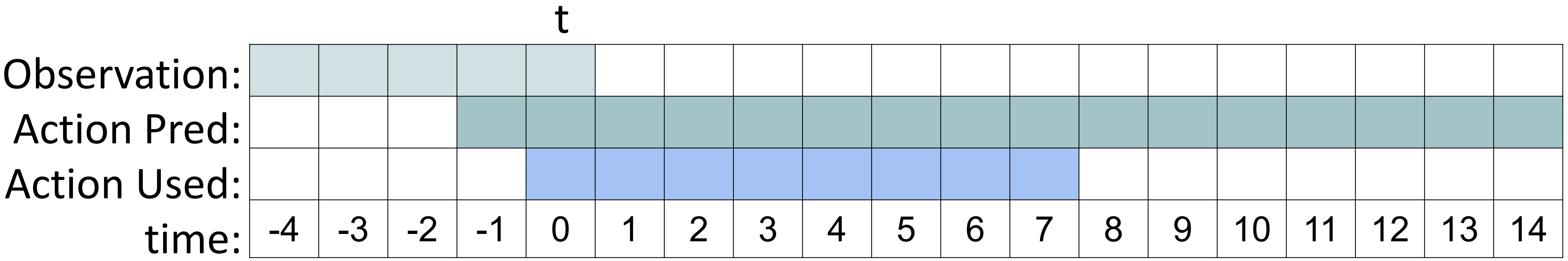
- Motivation behind diffusion models ✓
- Denoising Diffusion Probabilistic Models ✓
  - Diffusion Process and Reverse Diffusion Process ✓
  - Training and Sampling a Diffusion Model ✓
  - Results from a Diffusion Model that we trained ✓
- Conditioned v/s Unconditioned Diffusion ✓
- Policy Learning ✓
- Toy Problem ✓
- Diffusion Policy on Real Robots
  - Advantages ✓
  - Network Architect
  - Evaluation & Results

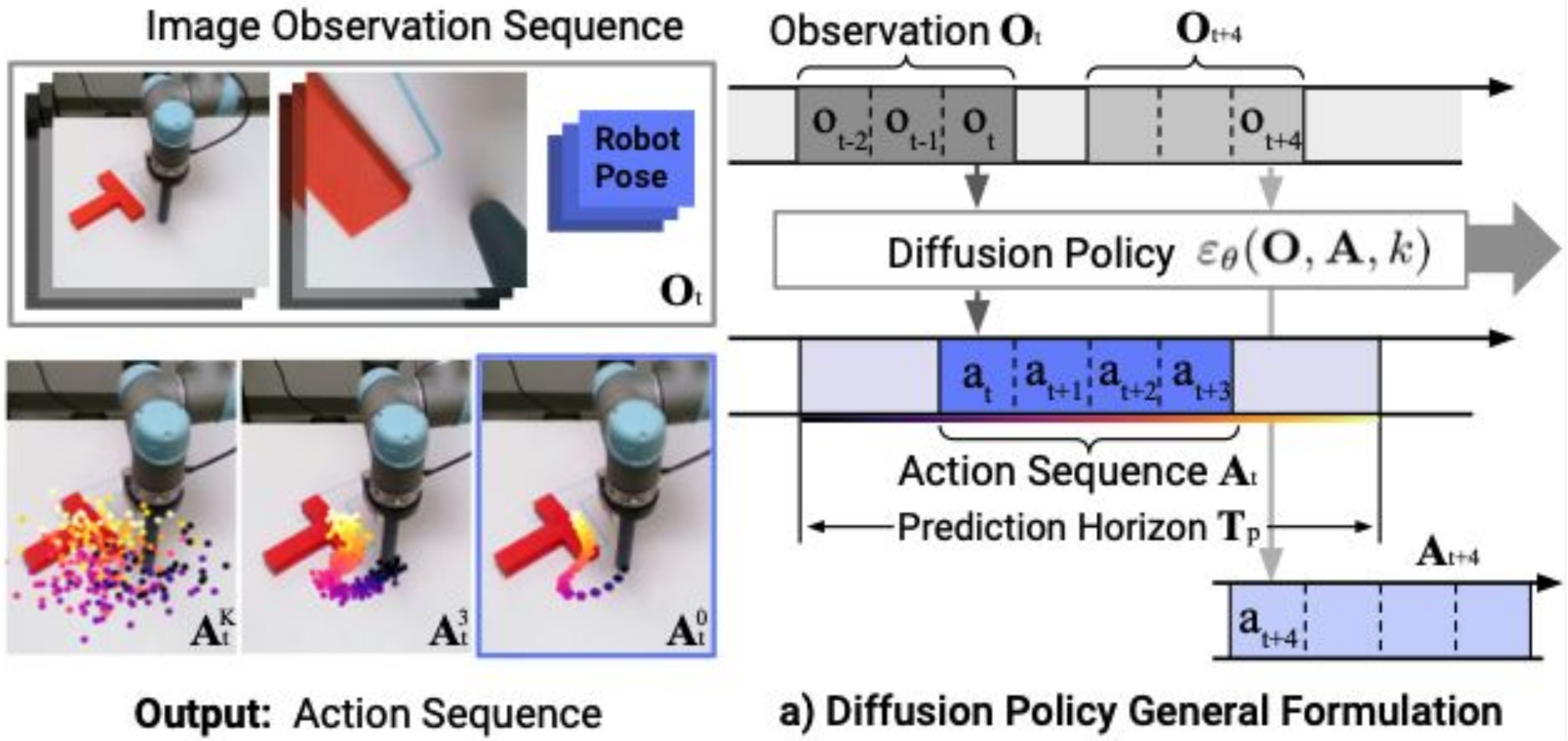
Closed Loop Action-Sequence Prediction

Closed Loop Action-Sequence Prediction



Closed Loop Action-Sequence Prediction

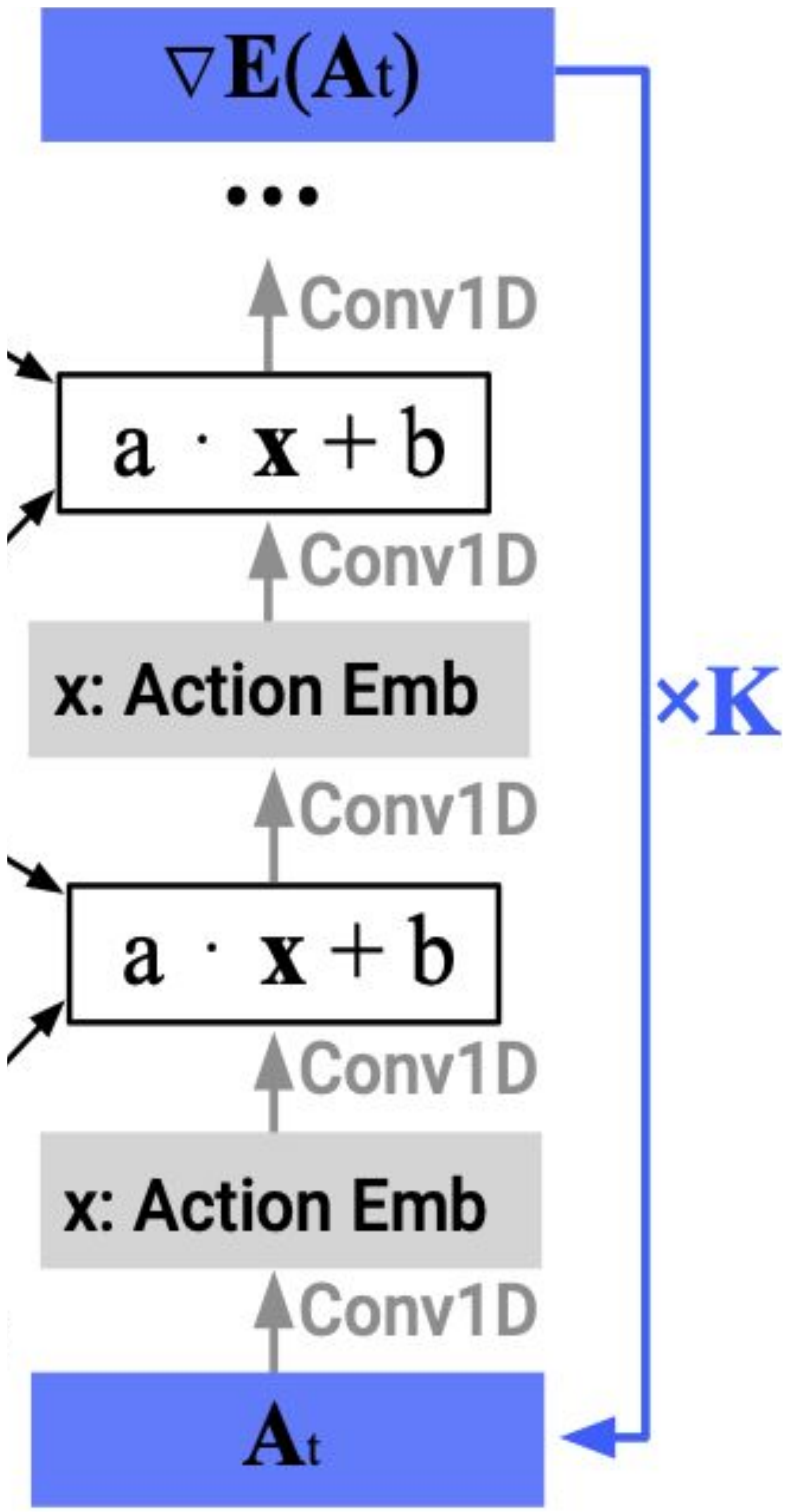




C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

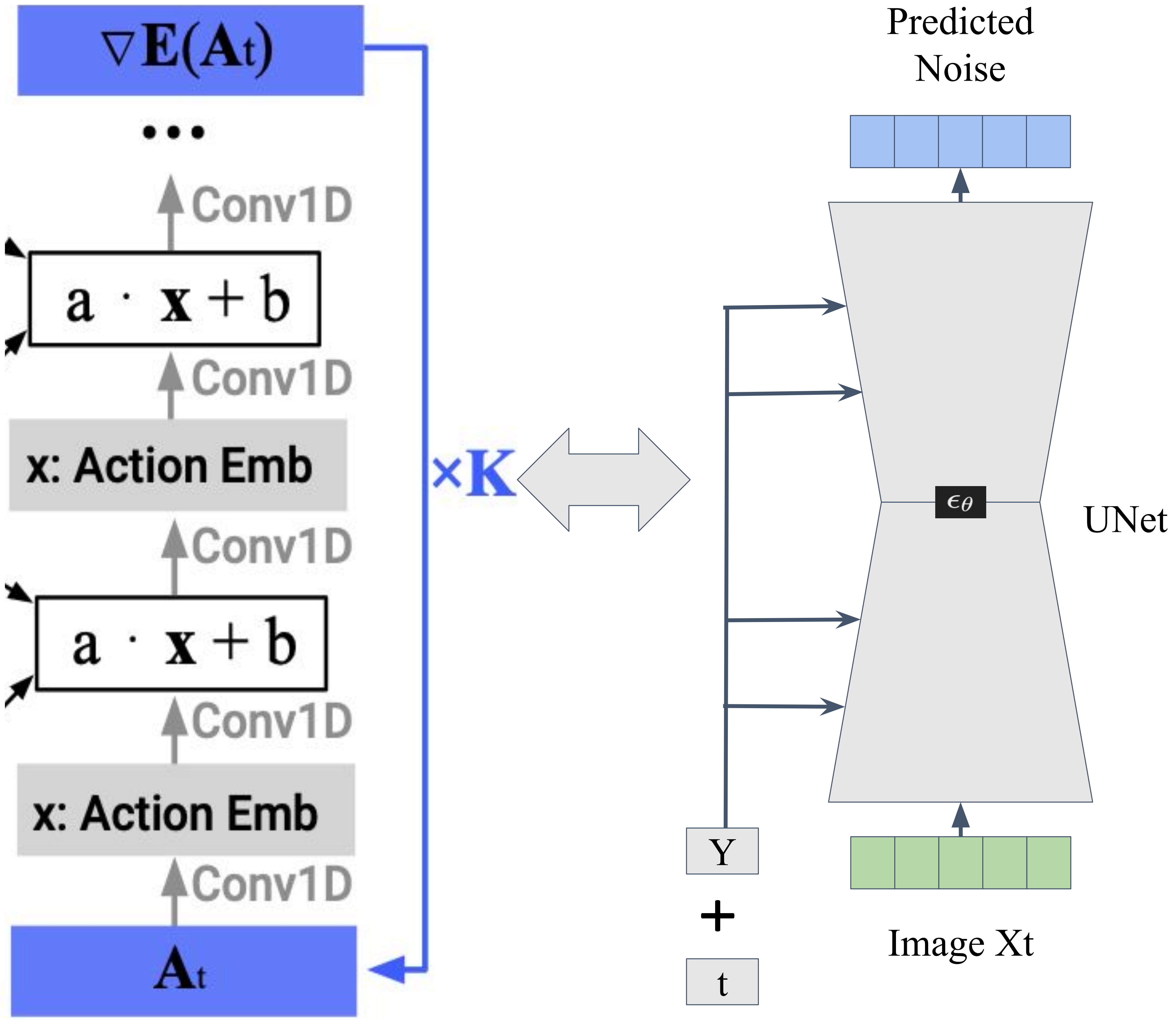
Network Architecture for ε_θ

Network Architecture for ε_θ



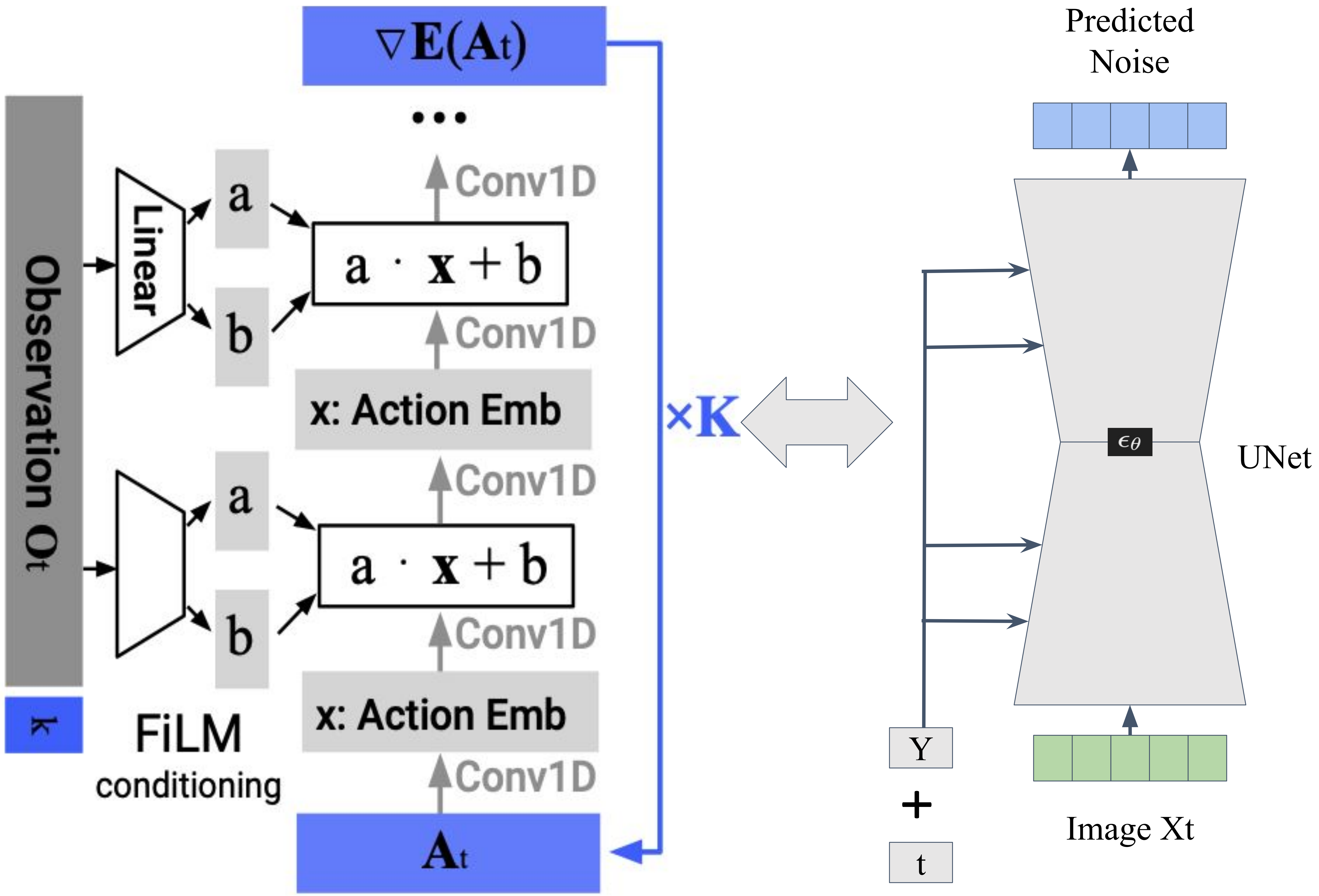
C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137
Image Source: https://www.assemblyai.com/blog/how-imagen-actually-works/

Network Architecture for ε_θ



C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137
Image Source: https://www.assemblyai.com/blog/how-imagen-actually-works/

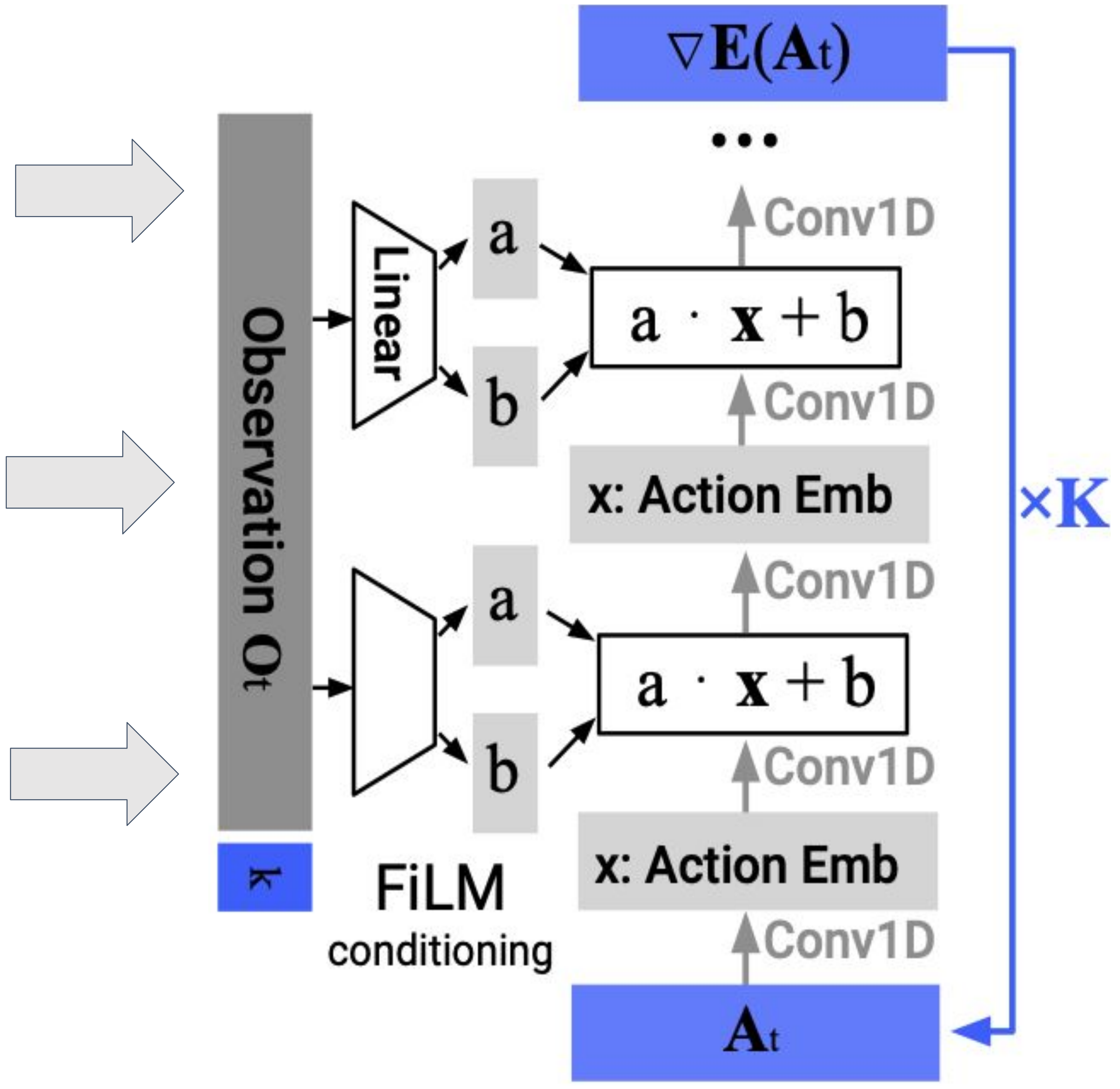
Network Architecture for ϵ_θ



C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

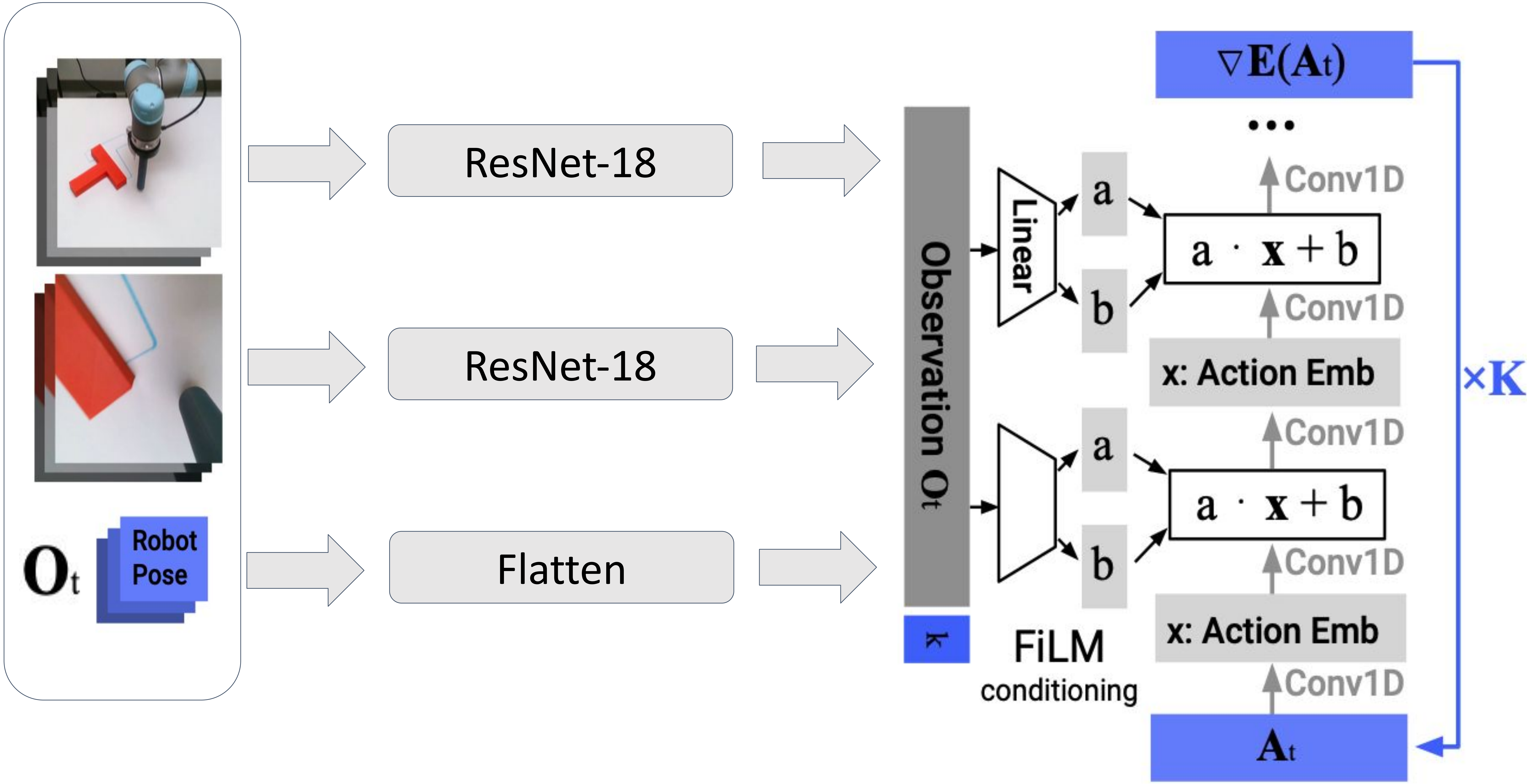
Image Source: https://www.assemblyai.com/blog/how-imagen-actually-works/

Network Architecture for ε_θ



C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

Network Architecture for ε_θ



C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

Overview

- Motivation behind diffusion models ✓
- Denoising Diffusion Probabilistic Models ✓
  - Diffusion Process and Reverse Diffusion Process ✓
  - Training and Sampling a Diffusion Model ✓
  - Results from a Diffusion Model that we trained ✓
- Conditioned v/s Unconditioned Diffusion ✓
- Policy Learning ✓
- Toy Problem ✓
- Diffusion Policy on Real Robots
  - Advantages ✓
  - Network Architect ✓
  - Evaluation & Results

Evaluation

Real World Push-T task:

- trained with 136 Proficient-Human demonstrations

	Human	IBC		LSTM-GMM		Diffusion Policy			
	Demo	pos	vel	pos	vel	T-E2E	ImgNet	R3M	E2E
IoU	0.84	0.14	0.19	0.24	0.25	0.53	0.24	0.66	**0.80**
Succ%	1.00	0.00	0.00	0.20	0.10	0.65	0.15	0.80	**0.95**
Dur.	20.3	56.3	41.6	47.3	51.7	57.5	55.8	31.7	**22.9**

- Results on 20 rollouts
- Rollout is Successful if $\text{IoU}_{\text{Last Step}}$ > min(IoU from human demonstrations)
- Max. steps 600 in a rollout

C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

Evaluation

Real World Push-T task:
- trained with 136 Proficient-Human demonstrations

C. Chi *et al.*, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," *arXiv preprint arXiv:2303.04137*, 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

Visual Results

Let’s go to the website to see the policies performance in action:
https://diffusion-policy.cs.columbia.edu

Overview

- Motivation behind diffusion models ✓
- Denoising Diffusion Probabilistic Models ✓
  - Diffusion Process and Reverse Diffusion Process ✓
  - Training and Sampling a Diffusion Model ✓
  - Results from a Diffusion Model that we trained ✓
- Conditioned v/s Unconditioned Diffusion ✓
- Policy Learning ✓
- Toy Problem ✓
- Diffusion Policy on Real Robots ✓
  - Advantages ✓
  - Network Architect ✓
  - Evaluation & Results ✓

Next Lecture:
Student Lecture 7
Dual-arm Manipulation

Happy Thanksgiving!

No class on 11/27

DeepRob

[Group 8] Lecture 6
Diffusion Models and Policy Learning
by Nirshal Chandra Sekar and Mohit Yadav
University of Minnesota

DeepRob

[Group 8] Lecture 6

Diffusion Models and Policy Learning

by Nirshal Chandra Sekar and Mohit Yadav

University of Minnesota