Problem Set 5: Behavior Cloning

6.4210/2 Robotic Manipulation — Fall 2026

Due: October 25, 2026

Out: October 8 **Due:** October 25

This problem set is about behavior cloning: learning a policy from demonstrations. The pencil-and-paper problems probe the core difficulties — multimodal demonstrations, the role of observation history, action chunking, and distribution shift — and the computational problem runs a diffusion policy on the Push-T task. Enter the answers in the **Gradescope** blocks in Gradescope; where a question requests a PDF, upload your written response to that question. The **Going further** prompts are for discussion and require no submission. All logarithms are natural.

1 Bookwork

1.1 Behavior cloning

Problem 1 Discrete and continuous action representations [5]

Model the robot as a point in a two-dimensional plane, with horizontal coordinate x and vertical coordinate y. At each discrete step n, it moves one unit to the right. Its scalar action $u[n]$ specifies the vertical coordinate of its next position (in the same units as x and y):

$$x[n+1] = x[n] + 1, \qquad y[n+1] = u[n].$$

There is a diamond-shaped obstacle,

$$|x - 1| + |y| \leq 0.75.$$

We have five demonstrations, each starting at $x[0] = 0$ and stepping through $x = 0, 1, 2$, that all go around the obstacle, some over the top and some under the bottom (Figure 1):

demo	$y[0]$	$y[1]$	$y[2]$	$u[0]$	$u[1]$
a	0	1	0	1	0
b	0.5	1	0	1	0
c	1	1	0	1	0
d	0	−1	0	−1	0
e	−1	−1	0	−1	0

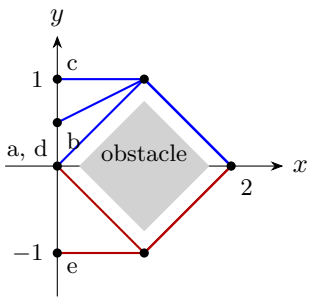


Figure 1: The five demonstrations, over the obstacle (a–c, blue) and under it (d, e, red). Demonstrations a and d both start at the origin.

Gradescope:

1.1 First, fit a simple linear policy, $u = \pi(x, y) = ky$, by least squares on the ten (state, action) pairs in the table. What is k? [1]

$k =$ 10/29.

Solution: $k = \sum yu / \sum y^2 = 2.5/7.25 = 10/29 \approx 0.345$.

1.2 Run the fitted policy from each of the five demonstrations' initial states. How many of the five rollouts hit the obstacle? [1]

Number of rollouts: **5**.

Solution: *All 5. The first step lands at $x = 1$ with $|y[1]| = k|y[0]| \le 0.345$, inside the obstacle, which at $x = 1$ covers $|y| \le 0.75$. At $(0, 0)$, $\pi = 0$ for any k: the demonstrations there say $u = 1$ and $u = -1$, and a least-squares fit of a single action can do no better than their mean.*

1.3 Now suppose we discretize the action, $u \in \{-1, 0, 1\}$, and learn a categorical distribution $p(u \mid x, y)$. Here *RT-style* refers to Robotics Transformer models that discretize continuous robot actions into tokens and predict a categorical distribution over those tokens. We train it by maximum likelihood (equivalently, minimizing the cross-entropy loss) on the five demonstrations. Assume the model is expressive enough to fit the training data as well as possible. What is $p(u \mid x = 0, y = 0)$? Enter $[p(u{=}-1), p(u{=}0), p(u{=}1)]$. [1]

Probabilities: [**0.5** , **0** , **0.5**].

Solution: $[0.5, 0, 0.5]$*: at $(0, 0)$ the data has one $u = -1$ (demo d) and one $u = 1$ (demo a), and maximum likelihood matches those frequencies. The averaged action, $u = 0$, has probability zero.*

1.4 Finally, consider a denoising diffusion policy, which keeps u continuous. At each state, it learns a denoiser $D(\tilde{u}, \sigma)$ that takes a noisy action $\tilde{u} = u + \sigma\epsilon$, with u from the demonstrations and $\epsilon \sim \mathcal{N}(0, 1)$, and predicts the clean u. It is trained by least squares, minimizing $\mathbb{E}\big[(D(u{+}\sigma\epsilon, \sigma){-}u)^2\big]$ over many noise levels σ. At $(x, y) = (0, 0)$, where the demonstrated actions are $u = \pm 1$ with equal probability, the minimizer is [1]

$$D(\tilde{u}, \sigma) = \mathbb{E}[u \mid \tilde{u}] = \tanh(\tilde{u}/\sigma^2).$$

You may use this expression without deriving it. What does $D(0.5, \sigma)$ approach as $\sigma \to \infty$, and as $\sigma \to 0$? Enter $[\lim_{\sigma\to\infty}, \lim_{\sigma\to 0}]$.

Limits: [**0** , **1**].

Solution: $[0, 1]$*. With a lot of noise, $\tilde{u}$ says almost nothing about which demonstration it came from, and the denoiser returns their mean — the same answer as the least-squares policy above. With little noise, it returns the nearer demonstration.*

1.5 To sample an action, start from a random $\tilde{u}$ and denoise it at a decreasing sequence of noise levels. Use the simplest such sampler, $\tilde{u} \leftarrow D(\tilde{u}, \sigma)$, with $\sigma = 1, 0.5, 0.25$ in turn, starting from $\tilde{u} = 0.2$. What is $\tilde{u}$ after each step? Enter three numbers, to two decimal places. [1]

Values after each step: [**0.20** , **0.66** , **1.00**].

Solution: $[0.20, 0.66, 1.00]$*. The early, noisy steps barely move it; the later ones commit it to the demonstration on its side of zero. Started from -0.2 it would reach -1, so with $\tilde{u}$ drawn symmetrically about zero, half the samples go over the obstacle and half under, and none go through it.*

Problem 2 Pauses in the data [6]

Human demonstrators pause: to think, to look, or to reposition their hand. Often times robot learning practitioners try to cut these pauses out of the training data, but that is a slippery slope (what is the sensor threshold for "zero velocity"? and sometimes the policy actually needs to pause... to wait for something to happen in the environment.) Let's see what a pause does to a behavior-cloned policy if we leave it in the training data; it's a good exercise for thinking through some of the implications of longer context lengths and action-sequence prediction.

Consider a robot on a line, $x[n+1] = x[n] + u[n]$, with discrete actions $u \in \{0, 1\}$ (stay or step forward). We have a single demonstration, starting from $x[0] = 0$, that pauses for three steps at $x = 2$:

n	0	1	2	3	4	5	6	7
$x[n]$	0	1	2	2	2	2	3	4
$u[n]$	1	1	0	0	0	1	1	

As in the previous problem, we learn a categorical distribution over actions by maximum likelihood, and assume the model is expressive enough to fit the training data as well as possible.

Gradescope:

2.1 First, a policy with no history, $p(u[n] \mid x[n])$. What is $p(u = 1 \mid x = 2)$? [1]

Probability: 1/4.

Solution: *1/4: the demonstration is at $x = 2$ four times, and steps forward once.*

2.2 An RT-style model often decodes greedily, taking the most likely action. OpenVLA, for instance, evaluates its policy with (`experiments/robot/openvla_utils.py`) [1]

```
action = vla.predict_action(**inputs, unnorm_key=unnorm_key, do_sample=
    False)
```

where `do_sample=False` tells Hugging Face's `generate()` to take the most likely token at each step. Run that greedy policy from $x[0] = 0$. What does $x[n]$ converge to as $n \to \infty$?

Limiting position: **2**.

Solution: *2. At $x = 2$, $u = 0$ is the most likely action, and taking it leaves the robot at $x = 2$, to make the same choice again: stuck forever.*

2.3 Now sample from $p(u \mid x)$ instead. The demonstration is at $x = 2$ for four time steps ($n = 2, \ldots, 5$). What is the expected number of time steps the sampled policy is at $x = 2$? [1]

Expected number of steps: **4**.

Solution: *4. Each step at $x = 2$ is an independent draw that leaves with probability $1/4$, so the time there is geometric with mean $1/(1/4) = 4$ — exactly the demonstration's. Sampling does not change the mean length of the pause; it makes the length random.*

2.4 What is the probability that the sampled policy is still at $x = 2$ at $n = 12$? [1]

Probability: $(3/4)^{10}$.

Solution: *$(3/4)^{10} \approx 0.056$: ten draws of $u = 0$ in a row, at $n = 2, \ldots, 11$. The mean is right, but the tail is long — about one rollout in eighteen pauses more than twice as long as the demonstrator did.*

2.5 Now condition on a history of the last h states, $p(u[n] \mid x[n-h+1], \ldots, x[n])$, so $h = 1$ is the policy above. (Take $x[n] = 0$ for $n < 0$.) What is the smallest h for which no history input in the demonstration appears with two different actions, so that even the greedy policy reproduces the demonstration exactly? [1]

Smallest h: **4**.

Solution: *4. With $h = 3$, the history $(2, 2, 2)$ appears at $n = 4$ with $u = 0$ and at $n = 5$ with $u = 1$. With $h = 4$, every history is distinct. In general the history must be longer than the pause, $h \geq P + 1$ for a pause of P steps.*

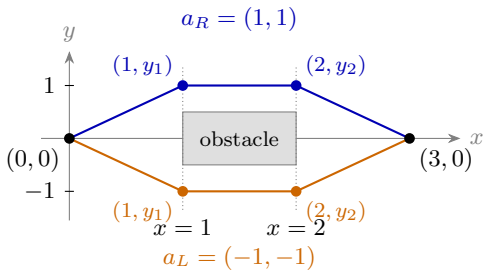
2.6 Instead, with no history, predict a chunk of K actions, $p(u[n], \ldots, u[n+K-1] \mid x[n])$, execute all K open loop, and then predict again. (Pad the end of the demonstration by repeating its last action, $u = 1$.) With $K = 2$, what is the most likely chunk at $x = 2$, and what is its probability? Enter $[u[n], u[n+1], p]$. [1]

Chunk and probability: [**0**, **0**, **0.5**].

Solution: $[0, 0, 0.5]$. *The chunks at $x = 2$ are $(0, 0)$, $(0, 0)$, $(0, 1)$ and $(1, 1)$, so the greedy policy is still stuck. With $K = 4$ they are $(0, 0, 0, 1)$, $(0, 0, 1, 1)$, $(0, 1, 1, 1)$ and $(1, 1, 1, 1)$: all distinct, each with probability $1/4$, and every one steps forward, so the robot leaves $x = 2$ within one chunk whichever it draws. The paused states all carried the same label, "stay", which outvoted the one "go"; history gives each a different input, and chunking gives each a different label.*

Problem 3 The average of two good motions [10]

A robot follows straight segments from $(0,0)$ through $(1,y_1)$ and $(2,y_2)$ to $(3,0)$. The x coordinates are fixed, so the policy only predicts $a=(y_1,y_2)$. At the same initial observation o, half the demonstrations use $a_L=(-1,-1)$ and half use $a_R=(1,1)$:



A motion succeeds exactly when both coordinates have the same sign and $|y_1|,|y_2|\geq 1/2$. This simplified rule encodes going around the obstacle on one side. A regression policy minimizes $\mathbb{E}[\|\hat{a}-A\|^2 \mid O=o]$, where A is the demonstrated motion.

Gradescope:

3.1 Give the optimal regression prediction and its expected squared error. $\hat{a}=$ (**0**, **0**); error: **2**. [2]

Solution: *The minimizer is the conditional mean $(0,0)$; either demonstration is squared distance 2 away.*

3.2 A distributional policy samples a_L or a_R, each with probability $1/2$. We score its sample $\hat{a}$ against a demonstration A from the same observation o, where A is a_L or a_R with probability $1/2$ each, independent of the policy's sample. What are the policy's success probability and its expected squared error $\mathbb{E}[\|\hat{a}-A\|^2]$? Success: **1**; error: **4**. [2]

Solution: *Every sampled motion succeeds. It matches the independent demonstration half the time; otherwise the squared distance is 8, giving mean error 4.*

3.3 Instead, independently sample each waypoint from $\{-1,1\}$ with equal probabilities. What is the success probability? **1/2**. [2]

Solution: *Only $(-1,-1)$ and $(1,1)$ succeed out of four equally likely pairs.*

3.4 A different policy always chooses the complete motion a_L. What is its success probability? **1**. More generally, suppose a policy chooses the complete motion a_L with probability p and a_R with probability $1-p$. Which values of p give 100% success? **$p\in[0,1]$**. Does 100% success in this task require reproducing the demonstrated 50/50 mix of routes? **No**. [2]

Solution: *Both routes succeed, so every mixture succeeds. Matching the demonstrations' frequencies is a different objective from completing the task.*

3.5 For the policy in the preceding part, minimizing the demonstration negative log likelihood gives $L(p)=-\frac{1}{2}\log p-\frac{1}{2}\log(1-p)$. Find the minimizing $p=$ **1/2**. [2]

Solution: *Likelihood is optimized only at the demonstrated mix, $p=1/2$.*

Going further: When would reproducing both routes matter, beyond completing this task? How would observing the demonstrator's intended route change the problem?

Solution: *A deterministic policy succeeds here but cannot reproduce the demonstrated diversity. Knowing the intended route removes the ambiguity for regression.*

Problem 4 DAgger: learn from where you actually go [8]

A robot has three nonterminal states: start S, corridor C, and recovery state R. State G is a terminal goal. The actions are F and L. The table gives deterministic transitions, expert labels, and the only two policies we will consider:

State	Next under F	Next under L	Expert	π_A	π_B
S	C	R	F	F	L
C	G	R	F	L	F
R	R	G	L	F	L

The initial dataset contains three labeled examples, $D_0 = [(S, F), (S, F), (C, F)]$: two examples from state S and one from state C. Repeated examples count separately. A rollout starts at state S and stops at state G or after four actions.

Gradescope:

4.1 How many mistakes does each policy make on D_0? π_A: **1**; π_B: **2**. Which policy makes fewer mistakes? π_A. [2]

Solution: *π_A misses the label at state C; π_B misses both examples at state S. Thus we select π_A without a tie.*

4.2 Execute π_A from state S for four actions. What are the learner's four actions? F, L, F, F. List the four states *before* actions and the final state. Visited: S, C, R, R; final: R. Give the expert labels at those four visited states: F, F, L, L. [2]

Solution: *The learner executes F, L, F, F. The expert is queried on the learner's states, so its labels differ from the executed actions.*

4.3 One DAgger iteration appends all four (visited state, expert label) pairs from Question 4.2 to D_0, including repeats. How many mistakes does each policy make on this seven-example dataset? π_A: **4**; π_B: **3**. Which is selected? π_B. [2]

Solution: *State S now occurs three times, and states C and R each occur twice. π_A misses both copies of C and R; π_B misses only the three copies of S.*

4.4 Execute the newly selected policy π_B from state S. List the states along its rollout, including the terminal state: S, R, G. How many actions does the robot take before first reaching goal state G? **2**. If instead we added two complete expert trajectories from state S to D_0, would fitting on that expert-only data select a different policy from the original fit? **No**. [2]

Solution: *π_B executes L, L and follows $S \to R \to G$. Each expert trajectory supplies $(S, F), (C, F)$, so the expert-only dataset would contain four examples at state S and three at state C. π_A makes three mistakes and π_B makes four, so we would still select π_A.*

Going further: How can a policy that disagrees with the expert at state S still complete the task? What would go wrong if we appended the learner's executed actions as labels? We used a pure learner rollout (expert mixing probability $\beta = 0$). Why might early DAgger iterations sometimes execute expert actions too?

Solution: *The class cannot reproduce the expert everywhere, but π_B recovers on its own route. Self-labeling reinforces mistakes; expert labels teach corrections. Expert mixing can limit poor early rollouts. One successful iteration here is not a general convergence guarantee.*

Problem 5 Position targets or velocity commands? [9]

A robot hand should stay at position $x = 0$ on a line. We compare predicting a velocity (how fast to move) with predicting a position (where to go), both when obtaining training labels and when executing commands. Use the movement rules below.

Gradescope:

5.1 The hand is stationary, but two measurements taken 0.02 seconds apart report -1 mm and $+1$ mm. Estimate its velocity by dividing the change in measured position by the elapsed time: **100** mm/s. If the same measurement errors occurred 0.01 seconds apart, what velocity would we estimate? **200** mm/s. [2]

Solution: *The estimates are* $[1-(-1)]/0.02 = 100$ *mm/s and* $[1-(-1)]/0.01 = 200$ *mm/s, although the true velocity is zero. Small position errors can give large errors in a velocity label, especially when measurements are close together in time.*

5.2 Now suppose position measurements are exact, but the policy makes an error in its prediction. It will not provide a new prediction for one second. During that interval, the robot updates its motion every 0.1 seconds. Compare these two cases, both starting at $x_0 = 0$: [5]

- The velocity command is $+5$ mm/s instead of zero. Each update moves the hand by $0.1(5) = 0.5$ mm: $x_{n+1} = x_n + 0.5$ mm.
- The position target is $b = 1$ mm instead of zero. At each update, the robot measures its current position and moves halfway from there to b: $x_{n+1} = x_n + (b - x_n)/2$.

The errors have been chosen so that both cases make the same first move. Fill in the hand positions, in mm; round to three decimal places.

Update n	Velocity command	Position target
1	**0.500**	**0.500**
2	**1.000**	**0.750**
3	**1.500**	**0.875**
10	**5.000**	**0.999**

If these commands were repeated indefinitely, whose distance from zero would grow without bound: the velocity-commanded hand, the position-commanded hand, or both? **Velocity**.

Solution: *The velocity command adds* 0.5 *mm each time, so* $x_n = 0.5n$ *mm. The position rule halves the remaining distance to* 1 *mm at each update, giving* $x_n = 1-2^{-n}$ *mm. Its error approaches* 1 *mm; it does not keep accumulating.*

5.3 Finally, suppose both policy predictions are correct: zero velocity and target position zero. A bump displaces the hand to 4 mm just after the prediction. Before the policy can predict again, the robot makes three updates using the same rules as above. Where is the hand after those updates? Velocity command: **4** mm; position target: **0.5** mm. [2]

Solution: *Zero velocity leaves the hand at* 4 *mm. Following the zero position target moves it to* 2*,* 1*, and then* 0.5 *mm. The robot corrects the position error even though the policy has not made a new prediction.*

Going further: When the velocity policy next observes the displaced hand, could it predict a command that brings the hand back? What advantage does the position target provide while the policy is still processing its next observation? Would the label-noise issue in the first part apply if we recorded velocity commands directly from the teleoperator?

Solution: *Yes: the next velocity prediction can move the hand back. Following a position target allows correction at every motion update, even between policy predictions. The distinction is how often position feedback is used. Recording velocity commands directly avoids the numerical differentiation in the first part.*

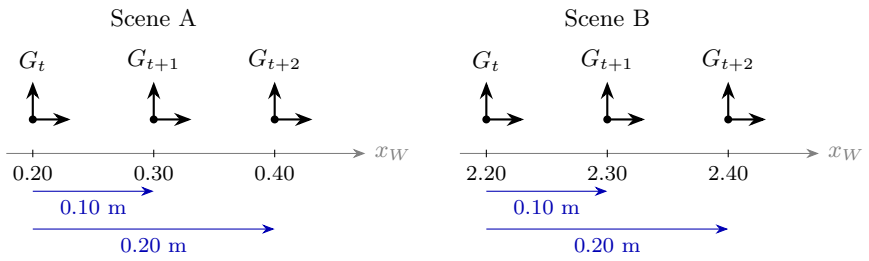
Problem 6 Relative to which frame? [11]

A policy predicts a chunk of two future gripper poses. Use Drake's notation: ${}^{W}X^{G_t}$ is the current gripper pose in the world frame, and ${}^{W}X^{G_{t+j}}$ is a future target pose. All frames in this example have the same orientation, and the hand moves only along the world x axis. Write

$$T(d) = \begin{bmatrix} I_3 & (d,0,0)^{\mathsf{T}} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix} ,$$

where d is in meters. Thus ${}^{W}X^{G_t} = T(h)$ places the current hand at world coordinate h.

The desired chunk moves the hand 0.10 m and then 0.20 m forward from its *current* pose. In scene A, $h = 0.20$ m; in scene B, $h = 2.20$ m. The local surroundings move with the hand, so a camera mounted on the hand produces the same image in both scenes.



Both displacements are measured from the same initial frame G_t.

Gradescope:

6.1 Fill in the translation parameters for the two ways of representing this chunk. Each entry below is a pair (d_1, d_2), in meters, representing transforms $(T(d_1), T(d_2))$. [2]

	Absolute: $({}^{W}X^{G_{t+1}}, {}^{W}X^{G_{t+2}})$	Relative: $({}^{G_t}X^{G_{t+1}}, {}^{G_t}X^{G_{t+2}})$
Scene A	$(0.30, 0.40)$	$(0.10, 0.20)$
Scene B	$(2.30, 2.40)$	$(0.10, 0.20)$

Solution: *Absolute targets include the starting world position. The relative transforms describe the same local motion in both scenes.*

6.2 First, the policy receives only the camera image; it does *not* receive ${}^{W}X^{G_t}$. Can a deterministic policy always predict the correct absolute chunk in both scenes? **No**. Can it always predict the correct relative chunk? **Yes**. The robot's execution code has access to the measured current pose, even when that pose is omitted from the policy's input. [2]

Solution: *The images are identical but the absolute targets differ, so a function of the image alone cannot choose the right absolute target in both cases. The relative targets are identical. The execution code combines them with the measured current pose to obtain the world targets.*

6.3 Now include ${}^{W}X^{G_t}$ in the policy's input. Can either representation express the correct mapping from input to target chunk? **Yes**. Complete the conversion from a relative target to a world target, using transform multiplication: [2]

$${}^{W}X^{G_{t+j}} = {}^{W}X^{G_t}\,{}^{G_t}X^{G_{t+j}} .$$

Solution: *With the current pose in the input, the policy has enough information for either representation. Multiply each relative target by the same current-pose transform ${}^{W}X^{G_t}$.*

6.4 Even when the current pose is an input, the required output ranges can differ. Suppose the starting position h ranges over $[-10, 10]$ m, and a target's forward displacement d independently ranges over $[0.05, 0.20]$ m. The absolute translation is $h + d$; the relative translation is d. Give their ranges: absolute ___[−9.95, 10.20]___ m; relative ___[0.05, 0.20]___ m. [5]

Suppose each scalar output is linearly scaled to $[0, 1]$ during training. An error of 0.01 in that scaled output corresponds to what translation error after undoing the scaling? Absolute: ___**201.5**___ mm; relative: ___**1.5**___ mm. Which representation needs a more accurate scaled prediction to meet a 1 mm position tolerance? ___**Absolute**___.

Solution: *The ranges have widths* 20.15 *m and* 0.15 *m. Multiplying by* 0.01 *gives* 0.2015 *m and* 0.0015 *m. For a* 1 *mm tolerance, the scaled errors must be at most* $0.001/20.15 \simeq 4.96\times10^{-5}$ *and* $0.001/0.15 \simeq 0.00667$*, respectively. A large output range can require much finer numerical accuracy for the same physical tolerance.*

Going further: Does supplying the current pose make the two representations equally easy to learn? Does the range calculation prove that absolute targets must be harder? How could a network predict a small displacement but still provide a world-frame target to the robot?

Solution: *Having enough information does not guarantee equally easy learning. Relative targets remove the large variation in starting world position, letting the network focus on the local motion. The calculation assumes the same scaled prediction error; it does not prove that trained networks will have that error. Architecture and scaling matter. A network can predict the relative transform and explicitly compose it with the measured current pose, rather than learn that composition from examples.*

Problem 7 Sharing and reweighting demonstrations [8]

We want to train a policy for a seven-degree-of-freedom robot, A. We have demonstrations from A, another type of seven-degree-of-freedom robot, B, and a six-degree-of-freedom robot, C. Their joint axes, link lengths, and joint limits differ. Assume the observations have already been put in a common format, and that gripper poses use consistent coordinate frames and the same definition of the gripper's tool frame.

Gradescope:

7.1 Suppose the policy predicts joint angles for robot A. Without converting the labels, can B's seven joint angles be used as target joint angles for A? **No**. Can C's six joint angles be used? **No**. [2]

Solution: *Having seven numbers does not make B's joint angles mean the same thing on A: the joints and kinematics differ. C's labels also have the wrong number of entries. Both require a mapping to A's joints, not direct reuse.*

7.2 Suppose instead that the policy predicts gripper poses ${}^{W}X^{G}$. Can demonstrations from all three robots use this same action representation? **Yes**. Does a gripper pose reached by B or C have to be reachable by A? **No**. [2]

Solution: *A pose describes a location and orientation without specifying how many joints produced it. It therefore provides common labels, given the frame conventions above. Whether A can reach a labeled pose is a separate question about A's kinematics and constraints.*

7.3 We use gripper-pose labels and decide to give more influence to data from A. Let $q(r)$ be the fraction of examples from robot r and $p(r)$ the desired fraction of total training weight: [2]

Robot	A (our robot, 7 DOF)	B (other 7 DOF)	C (6 DOF)
$q(r)$	0.2	0.3	0.5
$p(r)$	0.6	0.3	0.1

These desired fractions are our choice, not something determined by the number of joints. Compute the importance weights $w(r) = p(r)/q(r)$: $w(A) =$ **3**, $w(B) =$ **1**, $w(C) =$ **0.2**. In a batch of ten examples with two from A, three from B, and five from C, what fraction of the total weight belongs to A? **0.6**.

Solution: *The weights are 3, 1, and 0.2. The total weights of the three groups are 6, 3, and 1, so A accounts for 6/10 of the weight.*

7.4 A fixed policy has average prediction losses 0.1, 0.2, and 0.4 on examples from A, B, and C, respectively. Compute the unweighted average loss under q: **0.28**. Compute the weighted average $\sum_r q(r)w(r)\ell(r)$: **0.16**. [2]

Solution: *The unweighted loss is* $0.2(0.1)+0.3(0.2)+0.5(0.4) = 0.28$. *The weighted loss is* $0.6(0.1)+0.3(0.2)+0.1(0.4) = 0.16$. *The smaller number reflects the changed weighting; it is not evidence that the policy itself has improved.*

Going further: Even after expressing every action as a gripper pose, why might data from B and C be less useful than data from A? Think about the sets of hand poses and paths each robot can execute. Can importance weighting make an infeasible target feasible?

Solution: *The robots have different feasible regions in hand-pose space because of their link lengths, joint limits, and collision constraints. A pose or path demonstrated by B or C may be unreachable by A, or A may need a different approach to it. The seventh joint can also allow different ways to reach the same pose or avoid an obstacle; six joints do not in themselves prevent control of a full 3D pose. These differences can reduce the value of other-robot data even when the labels share a format. Weights change how much examples influence learning; they cannot change A's feasible motions.*

Problem 8 When should the robot ask for another demonstration? [5]

Two independently trained ensemble members predict a probability of taking the left route, $p_1(L \mid o)$ and $p_2(L \mid o)$. Define their average left-route probability $\bar{p}$ and their disagreement U as

$$\bar{p} = \frac{p_1 + p_2}{2}, \qquad U = \frac{(p_1 - \bar{p})^2 + (p_2 - \bar{p})^2}{2}.$$

We also compute $d(o)$, the distance to the nearest training observation in a fixed learned feature space. All distances below use the same scale.

Observation	$p_1(L \mid o)$	$p_2(L \mid o)$	$d(o)$
o_A	0.5	0.5	0.1
o_B	0.1	0.9	0.2
o_C	0.0	0.0	2.0

Gradescope:

8.1 Give $(\bar{p}, U)$ for each observation. [3]

o_A: (**0.5**, **0**); o_B: (**0.5**, **0.16**); o_C: (**0**, **0**).

Solution: *Only o_B produces disagreement: each prediction differs from 0.5 by 0.4, giving $U = 0.16$.*

8.2 Request an expert label whenever $U > 0.10$ *or* $d(o) > 1.0$. Which observations trigger a request? [2]
o_B, o_C. Which would be missed by using only the disagreement threshold? *o_C*.

Solution: *o_B triggers disagreement; o_C triggers distance despite consensus.*

Going further: At both o_A and o_B, averaging the two models' predictions gives probability 1/2 for each route. Why might asking for more data still be more useful at o_B? Can a unanimous ensemble still be wrong? Is either U or feature distance automatically a calibrated failure probability? How could queried labels feed back into the DAgger procedure?

Solution: *Agreement on a broad distribution may reflect two valid actions; disagreement suggests model uncertainty. Shared biases can make every member wrong. Neither heuristic is a probability guarantee, especially under distribution shift; risk estimates need validation. Append expert labels at queried learner states and retrain, while recognizing that selective queries change the data-collection scheme.*

Problem 9 Flow matching between noise and two data points [12]

Flow matching generates data by transporting samples from a simple noise distribution to the data distribution. Let $X_0 \sim \mathcal{N}(0,1)$, and independently choose X_1 with equal probability from the two data points $x^{(1)} = 2$ and $x^{(2)} = 4$. Define a straight path between them:

$$X_t = (1-t)X_0 + tX_1 \ , \qquad 0 \le t \le 1 \ .$$

Here noise is at $t = 0$ and clean data at $t = 1$. Use $t < 1$ when writing densities or velocities. You may write $\mathcal{N}(x; \mu, \sigma^2)$ for a Gaussian density evaluated at x; there is no need to expand its exponential.

Gradescope:

9.1 Fix $X_1 = x_1$ and t. What is the conditional distribution of X_t? Specify its mean and variance as functions of t and x_1. [2]

Mean: ___tx_1___; variance: ___$(1-t)^2$___.

Solution: *$X_t \mid X_1 = x_1$ is Gaussian with mean tx_1 and variance $(1-t)^2$.*

9.2 Now average over the two equally likely choices of X_1. Write the resulting density $p_t(x)$ of X_t. [2]

Solution:

$$p_t(x) = \tfrac{1}{2}\mathcal{N}(x; 2t, (1-t)^2) + \tfrac{1}{2}\mathcal{N}(x; 4t, (1-t)^2) \ .$$

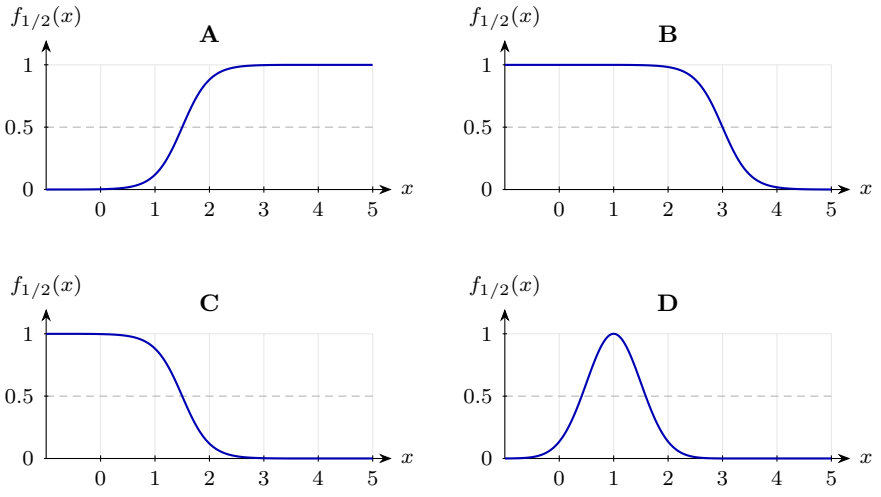
9.3 Observing $X_t = x$ gives information about which endpoint was chosen. Write the posterior probability $f_t(x) = \Pr(X_1 = 2 \mid X_t = x)$ in terms of the Gaussian densities. Upload your formula and a brief derivation as a PDF response to Question 9.3 in Gradescope. [2]

Solution: *Bayes' rule gives*

$$f_t(x) = \frac{\mathcal{N}(x; 2t, (1-t)^2)}{\mathcal{N}(x; 2t, (1-t)^2) + \mathcal{N}(x; 4t, (1-t)^2)} \ .$$

The equal prior probabilities cancel.

9.4 Which sketch shows $f_{1/2}(x)$, the probability that the clean endpoint is 2, given the noisy value x at time $t = 1/2$? Choose A, B, C, or D: ___**C**___. [2]



Solution: *At $t = 1/2$, the component means are 1 and 2, with variance 1/4. Thus $f_{1/2}(x) = 1/(1+\exp(4x-6))$: a decreasing sigmoid with limits 1 on the left and 0 on the right, crossing 1/2 at $x = 1.5$. The correct sketch is C. A reverses which endpoint's probability is plotted; B places the crossing between the clean endpoints rather than the noisy component means; D shows a bell-shaped curve rather than a posterior probability comparing the two endpoints.*

Setup for Questions 9.5 and 9.6. For a fixed endpoint x_1, the velocity field

$$v_t(x \mid x_1) = \frac{x_1 - x}{1 - t}$$

transports a point along its straight path to x_1 by time 1. To transport the noise distribution to a mixture of endpoints, use the conditional average

$$v_t(x) = \mathbb{E}\left[\frac{X_1 - x}{1 - t} \,\middle|\, X_t = x\right] \quad .$$

This velocity averages over the possible endpoints given the current noisy value, rather than choosing one endpoint in advance.

Gradescope:

9.5 Express $v_t(x)$ using $f_t(x)$ and the two endpoint velocities. Upload your formula and a brief explanation of the posterior weights as a PDF response to Question 9.5 in Gradescope. [2]

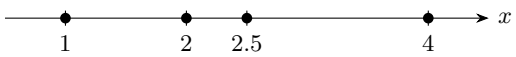
Solution:

$$v_t(x) = f_t(x)\frac{2 - x}{1 - t} + (1 - f_t(x))\frac{4 - x}{1 - t} \quad .$$

9.6 At $t = 1/2$, indicate the direction of the velocity at the four marked positions in the figure. Give left or right at $x = 1, 2, 2.5, 4$, respectively: [2]

right, **right**, **right**, **left**.

Evaluate $v_{1/2}(x)$ at the four marked positions.



Solution: *Here $v_{1/2}(x) = 8 - 2x - 4f_{1/2}(x)$. At $x = 1$, both endpoint velocities point right. At $x = 2$, the first vanishes and the second points right. At $x = 2.5$, $f_{1/2}(2.5) = 1/(1 + e^4)$, so the rightward term dominates. At $x = 4$, only the small leftward term remains: $v_{1/2}(4) = -4/(1 + e^{10}) < 0$.*

Going further: For a fixed x away from 2, 3, and 4, describe the direction and magnitude of $v_t(x)$ as t approaches 1.

Solution: *Away from the midpoint 3, the posterior concentrates on the closer endpoint: 2 for $x < 3$ and 4 for $x > 3$. The velocity therefore points toward that endpoint. For a fixed nonzero distance from it, the magnitude grows like that distance divided by $1 - t$. This concerns a fixed test location; points moving along the flow also change their distance from the data.*

Problem 10 How much evidence is enough? [8]

Two robot policies, A and B, are evaluated on a task with a binary outcome: success or failure. Each trial starts from an independently sampled configuration from the same distribution, and the testing conditions are the same for both policies. Let p_A and p_B be their true success probabilities.

Consider three separate experiments, each with N trials per policy:

N	Policy A: successes/trials	Policy B: successes/trials
5	4/5	3/5
10	8/10	6/10
100	80/100	60/100

In each case, the sample size and a *two-sided* test of $H_0 : p_A = p_B$ against $H_1 : p_A \neq p_B$ were chosen in advance. Use significance level $\alpha = 0.05$ for each experiment considered on its own. Round numerical answers to three significant digits.

Gradescope:

10.1 Use Fisher's exact test to compute the p-value for each experiment. Is the difference statistically significant at the specified level? You may use statistical software: the input table has rows for A and B and columns for successes and failures. [3]

N	Exact p-value	Significant? (yes/no)
5	**1.00**	**No**
10	**0.628**	**No**
100	**0.00319**	**Yes**

Solution: *The success/failure tables are* $\left[\begin{smallmatrix}4&1\\3&2\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}8&2\\6&4\end{smallmatrix}\right]$, *and* $\left[\begin{smallmatrix}80&20\\60&40\end{smallmatrix}\right]$. *The two-sided Fisher p-values are* 1, 0.62848297, *and* 0.00319182. *Only the last is below* 0.05*; the observed difference favors A.*

10.2 Now use the normal approximation. With observed rates $\hat{p}_A = s_A/N$, $\hat{p}_B = s_B/N$, and pooled rate $\hat{p} = (s_A + s_B)/(2N)$, compute [3]

$$z_{\text{obs}} = \frac{\hat{p}_A - \hat{p}_B}{\sqrt{2\hat{p}(1-\hat{p})/N}} \quad , \qquad p_{\text{approx}} = 2\big[1 - \Phi(|z_{\text{obs}}|)\big] \quad ,$$

where Φ is the standard normal cumulative distribution function. Also check whether each group has at least 5 expected successes and 5 expected failures under the null: $N\hat{p} \geq 5$ and $N(1-\hat{p}) \geq 5$.

N	z_{obs}	Approximate p-value	Passes count check?
5	**0.690**	**0.490**	**No**
10	**0.976**	**0.329**	**No**
100	**3.09**	**0.00203**	**Yes**

Solution: *The pooled rate is* 0.7 *in all three experiments, so* $z_{\text{obs}} = 0.2\sqrt{N/0.42}$. *Expected successes per group are* 3.5, 7, *and* 70*; expected failures are* 1.5, 3, *and* 30. *Only* $N = 100$ *passes the count check. The normal approximation and exact test give the same significance decisions here, but their p-values differ substantially for the small experiments.*

10.3 What is A's observed advantage over B in all three experiments? **20** percentage points. Do the first two experiments establish that $p_A = p_B$? **No**. [2]

Solution: *All three compare* 80% *with* 60%*, a difference of* 20 *percentage points. Failure to reject equality does not establish equality; the small experiments leave substantial uncertainty.*

Going further: Why does increasing N change the strength of the evidence even though the observed success rates stay the same? Does statistical significance alone tell us whether the improvement matters for the robot's application?

Solution: *The measured effect stays fixed, while the uncertainty in the difference decreases roughly as* $1/\sqrt{N}$*. The larger experiment therefore provides stronger evidence of a real difference. Whether a* 20*-percentage-point improvement matters depends on the task, the consequences of failure, and other costs of using the policy.*

Problem 11 Representing 3D rotations [12]

A robot policy might receive the hand's orientation as an input or predict a desired orientation as an output. Keep the coordinate frames fixed, and write R for the rotation matrix ${}^{W}R^{G}$. We will compare several ways to encode this rotation as numbers. Recall that unit quaternions q and $-q$ represent the same rotation.

Consider three possible difficulties:

J: nearby rotations, distant numbers.
Rotations arbitrarily close to one another can have numeric representations far apart in ordinary Euclidean distance. This concerns the inputs or training labels, not whether the formula converting numbers to a rotation is continuous.

U: non-unique.
Two different valid numeric representations can describe exactly the same rotation.

C: constrained.
Not every real-valued tuple is already a valid instance of the stated representation; conditions such as non-negativity or orthogonality must be enforced.

Classify the representations *as stated*, before any repair of network outputs.

Gradescope:

11.1 For each row, list all applicable letters from $\{J, U, C\}$, or write "none." No angle wrapping or quaternion sign alignment is assumed unless specified in the row. [5]

Representation	Difficulties
Roll–pitch–yaw angles $(\phi, \theta, \psi) \in \mathbb{R}^3$, with $R = R_z(\psi)R_y(\theta)R_x(\phi)$ and no restrictions on angle ranges.	*J, U*
Unit quaternion $q \in \mathbb{R}^4$, allowing both signs.	*J, U, C*
Unit quaternion with its sign chosen so that its first nonzero component is positive (components ordered scalar first, then x, y, z).	*J, C*
All nine entries of R.	*C*
The second and third columns of R, stored as six numbers. These are actual matrix columns, not arbitrary vectors that will later be converted into columns.	*C*

Solution: *Angles differing by 2π give the same rotation, and nearby rotations can have very different angle values. Every real angle triple still defines a rotation, so no validity constraint is needed.*

The two unit quaternions q and $-q$ are distance 2 apart but represent the same rotation; inconsistent signs can therefore give distant labels for nearby rotations. Unit length is a constraint. Choosing the first nonzero component positive removes the duplication, but the chosen quaternion jumps in sign as a rotation passes through some half-turns.

A rotation matrix is unique and changes continuously with orientation, but requires $R^{\mathsf{T}}R = I$ and $\det R = 1$. Its second and third columns are also unique and continuous. They must be orthogonal unit vectors; their cross product uniquely determines the first column.

11.2 Imagine that we trained a network to predict the second and third columns of R. At execution time, it outputs $u, v \in \mathbb{R}^3$, which need not be unit length or orthogonal. How can we turn them into a valid rotation? Use u to fix the direction of the second column r_2, then remove from v its component along r_2. Assume $u \neq 0$ and v is not parallel to u. Fill in the normalization, projection, and cross-product steps: [4]

$r_2 =$ ___$u/\|u\|$___; $w =$ ___$v - (r_2^\mathsf{T} v) r_2$___.

$r_3 =$ ___$w/\|w\|$___; $r_1 =$ ___$r_2 \times r_3$___.

Assemble the result as $R = [\, r_1 \; r_2 \; r_3 \,]$.

Solution: *This is Gram–Schmidt orthonormalization starting with column two. The projection leaves w perpendicular to r_2, and normalization makes r_3 a unit vector. The order $r_2 \times r_3$ completes a right-handed orthonormal basis, so $R^\mathsf{T} R = I$ and $\det R = 1$. Normalizing u and v separately would not generally make them orthogonal.*

11.3 Apply your construction to $u = (2, 0, 0)^\mathsf{T}$ and $v = (1, 0, 3)^\mathsf{T}$. Give the resulting first column: $r_1 =$ ___$(0, -1, 0)^\mathsf{T}$___. What is $\det R$? ___**1**___. If we reversed the cross-product order in the last step, what would the determinant be? ___-1___. [3]

Solution: *We get $r_2 = (1, 0, 0)^\mathsf{T}$, $w = (0, 0, 3)^\mathsf{T}$, and $r_3 = (0, 0, 1)^\mathsf{T}$. Thus*

$$R = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} .$$

Reversing the cross product negates the first column and changes the determinant to -1, giving an orthogonal matrix that is not a rotation.

Going further: What goes wrong when $u = 0$ or when u, v are parallel? After the conversion, can different raw network outputs produce the same rotation? Which difficulties matter when these representations are used as inputs, and which require action when they are predicted as outputs?

Solution: *The first or second normalization divides by zero. Near these cases the conversion is sensitive to small errors; an implementation needs to detect them and use a valid fallback or request another prediction. Adding an epsilon to a denominator alone does not guarantee a rotation.*

Different raw outputs can produce the same result: (u, v) and $(2u, v + u)$ give identical columns under this construction. This does not contradict the table, which classifies the actual orthonormal columns. Discontinuities and duplicate encodings can complicate learning from inputs or labels. Valid input rotations already satisfy their constraints; unconstrained network outputs need a conversion that enforces validity.

Background reading. Ross, Gordon, and Bagnell's DAgger paper; and Zhou et al., *On the Continuity of Rotation Representations in Neural Networks*. Problem 9 is adapted from Question 6 of the 6.7900 Fall 2024 alternative final exam.

1.2 Additional problems for 6.4212

Problem 12 Diffusion: adding noise and learning to remove it [8]

Consider a diffusion model in one dimension. Its forward process is

$$X_t = \sqrt{1 - \beta_t}\, X_{t-1} + \sqrt{\beta_t}\, Z_t \;, \qquad Z_t \sim \mathcal{N}(0, 1) \;,$$

where each Z_t is fresh independent noise. Define $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. At a chosen noise level, we can sample directly from the same conditional distribution using

$$X_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, Z \;, \qquad Z \sim \mathcal{N}(0, 1) \;.$$

Here x_0 is a fixed clean example. The new Z represents the combined noise in this direct construction, rather than a particular step's Z_t. Write $q_t(x_t \mid x_0)$ for the resulting conditional density.

Gradescope:

12.1 Give the mean and variance of $q_t(x_t \mid x_0)$. [2]

Mean: $\sqrt{\bar{\alpha}_t}\, x_0$; variance: $1 - \bar{\alpha}_t$.

Solution: *The clean-example term is constant. The Gaussian noise has mean zero and variance one, so scaling it gives variance $1 - \bar{\alpha}_t$.*

12.2 Suppose $\bar{\alpha}_T = 0.8$, and we start the reverse sampler from a standard normal. Does this choice of T ensure that the forward process has become close to standard Gaussian noise? **No**. [1]

Solution: *The conditional mean is still $\sqrt{0.8}\, x_0$ and the conditional variance is only 0.2. Substantial information about the data remains, so the marginal distribution at T need not match the standard normal used to initialize sampling. The terminal $\bar{\alpha}_T$ should normally be close to zero for that initialization.*

12.3 One training update uses a sampled noise value z in [5]

$$\nabla_\theta \left| z - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, z, t\right)\right|^2 \quad .$$

For each quantity below, identify its expression in this update, or write "not present." You may abbreviate the network's first input as x_t.

(i) A sample from $q_t(x_t \mid x_0)$: $\sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, z$.

(ii) The mean of $q_t(x_t \mid x_0)$: $\sqrt{\bar{\alpha}_t}\, x_0$.

(iii) The standard deviation of $q_t(x_t \mid x_0)$: $\sqrt{1 - \bar{\alpha}_t}$.

(iv) A prediction of the individual forward-step noise Z_t: **not present**.

(v) A prediction of the standardized combined noise Z used in the direct construction from x_0 to x_t: $\epsilon_\theta(x_t, t)$.

Solution: *The first network input is the noisy sample itself. Its deterministic term is the conditional mean, and the coefficient of z is the conditional standard deviation. The network is trained against the combined noise value z; it is not trained against the last individual step's noise.*

Going further: Given the current value x_t, a reverse sampler may draw fresh noise $Z \sim \mathcal{N}(0, 1)$ and set

$$X_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t Z \quad .$$

What changes if we simply set $\sigma_t = 0$ at every step? Must every sample then be identical? Must the output distribution remain correct?

Solution: *Each update becomes deterministic given its input, but the initial X_T is still random, so samples need not be identical. Deleting the noise term generally changes the output distribution; it does not preserve a general stochastic sampler's distribution automatically. A deterministic sampler such as DDIM uses a different update designed for that purpose.*

Suppose we instead try to generate a clean example in one step using $\hat{x}_0 = x_T - \epsilon_\theta(x_T, T)$. What is wrong with this expression? Even after correcting its scaling, why can a single denoising prediction have difficulty producing a good sample when $\bar{\alpha}_T$ is very small?

Solution: *Solving the forward equation for a clean-example estimate gives*

$$\hat{x}_0 = \frac{x_T - \sqrt{1-\bar{\alpha}_T}\,\epsilon_\theta(x_T, T)}{\sqrt{\bar{\alpha}_T}} \quad .$$

The proposed expression omits these scale factors. With an ideal squared-loss noise predictor, the corrected expression is the conditional mean of the clean example given x_T. Nearly pure noise gives little information about which clean example generated it, so this mean can average incompatible modes. The small denominator also amplifies errors in the noise prediction. Iterative sampling refines a candidate through intermediate noise levels instead.

2 Computational questions

2.1 Hands on with Diffusion Policy

Problem 13 Diffusion Policy on Push-T [8]

For this problem, you'll experiment with a slightly modified "Push-T" example from the original Diffusion Policy paper, develop your intuition, and answer some basic questions.

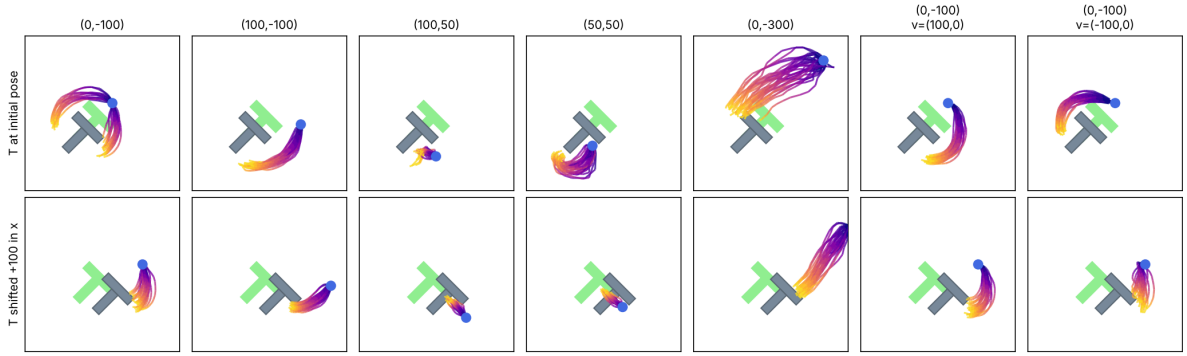
You'll work in `diffusion_policy_pusht_state.ipynb`. It includes the code for the Push-T environment and diffusion policy, so no separate Python files are needed. The notebook downloads a pretrained policy, and should run locally without GPU resources.

Setup. Open the notebook in Colab or Jupyter and run its setup cells in order. They install the dependencies and load the bundled helpers; the helper implementation is hidden in a collapsed setup cell. First, just step through the notebook and make sure you understand each of the steps.

The `show_action_samples` method is useful for visualizing the learned distribution. For each of the initial conditions below, indicate whether the distribution is uni-modal or bi-modal and whether the motions look appropriate.

- Keeping the T at its initial position.
  - Compare the action distribution for pusher positions (relative to T) $0, -100$ vs $100, -100$.
  - Compare the action distribution for pusher positions (relative to T) $100, 50$ vs $50, 50$.
  - Compare the action distribution for pusher positions (relative to T) $0, -100$ vs $0, -300$.
  - Compare the action distribution for pusher positions (relative to T) $0, -100$ but now initialize the velocity to $100, 0$ and then $-100, 0$.
- Displace the T 100 pixels in the +x direction. Repeat the pusher displacement tests.

Solution: *Pusher positions are relative to the T, and velocities are in the world frame. From* `show_action_samples` *with its defaults (32 samples, seed 0):*



pusher	*T at initial pose*	*T shifted +100 in x*
$0, -100$	***bi-modal**: around either end of the crossbar*	*uni-modal: around the right end*
$100, -100$	*uni-modal: around the near (right) end*	*uni-modal: under the crossbar's right end*
$100, 50$	*uni-modal: short move in beside the stem*	*uni-modal: short move to the stem*
$50, 50$	*uni-modal: loops around to the stem's end*	*uni-modal: pushes on the side of the stem*
$0, -300$	*uni-modal: all go to the left end*	*uni-modal: comes in to the right end*
$0, -100$, $v = (100, 0)$	*uni-modal: right*	*uni-modal: right*
$0, -100$, $v = (-100, 0)$	*uni-modal: left*	*uni-modal: right*

The motions all look appropriate: each moves the pusher to a side of the T from which pushing brings it toward the goal. The distribution is bi-modal only where the choice is symmetric, with the pusher centered behind the crossbar of the T at its initial pose; there the two modes go around opposite ends. Moving the pusher off center, or farther away, settles on one side. The velocity in the observation history picks the mode the pusher is already moving toward. With the T displaced, the T's position decides the side, and the leftward velocity no longer changes it.