Assignment 2: Imitation Learning

In this assignment, you will implement the basic components of an Imitation Learning system, behavior cloning, and DAgger.

Instructions

- This is an individual assignment. You are not allowed to discuss the problems with other students.
- Part of this assignment will be autograded by gradescope. You can use it as immediate feedback to improve your answers. You can resubmit as many times as you want.
- All your solution, code, analysis, graphs, explanations should be done in this same notebook.
- Please make sure to execute all the cells before you submit the notebook to the gradescope. - You will not get points for the plots if they are not generated already.
- If you have questions regarding the assignment, you can ask for clarifications in Piazza. You should use the corresponding tag for this assignment.
- Start Early! Some of the cells can take about an hour to run on CPU. You will need time to generate the results.

When Submitting to GradeScope: Be sure to

1. Submit a .ipynb notebook to the Assignment 2 - Code section on Gradescope.
2. Submit a pdf version of the notebook to the Assignment 2 - Report entry.

Note: You can choose to submit responses in either English or French.

Before starting the assignment, make sure that you have downloaded all the tests related for the assignment and put them in the appropriate locations. If you run the next cell, we will set this all up automatically for you in a dataset called public, which will contain both the data and tests you use.

This assignment has 4 questions. You will learn to:

1. Implement basic components in an Imitation Learning/RL setup.
2. Implement behavior cloning.
3. Implement DAgger.
4. Analyze different aspects of the DAgger algorithm.

In [ ]:
```
!apt update
!apt install -y --no-install-recommends \
        build-essential \
        curl \
        git \
        gnupg2 \
```

```
        make \
        cmake \
        ffmpeg \
        swig \
        libz-dev \
        unzip \
        zlib1g-dev \
        libglfw3 \
        libglfw3-dev \
        libxrandr2 \
        libxinerama-dev \
        libxi6 \
        libxcursor-dev \
        libgl1-mesa-dev \
        libgl1-mesa-glx \
        libglew-dev \
        libosmesa6-dev \
        lsb-release \
        ack-grep \
        patchelf \
        wget \
        xpra \
        xserver-xorg-dev \
        ffmpeg
!apt-get install python-opengl -y
!apt install xvfb -y
```

In [ ]:

```
!pip install gymnasium[mujoco]
!pip install torch
!pip install tqdm
!pip install matplotlib
!pip install pyvirtualdisplay
```

In [ ]:

```
!pip install otter-grader
!git clone https://github.com/chandar-lab/INF8250ae-assignments-2023.git public
```

In [6]:

```
#@title set up virtual display

from pyvirtualdisplay import Display

display = Display(visible=0, size=(1400, 900))
display.start()
```

Out[6]:

```
<pyvirtualdisplay.display.Display at 0x7edd33b81b10>
```

In [7]:

```
import gymnasium as gym
from gymnasium import wrappers
import torch
import numpy as np
from tqdm import tqdm
import matplotlib.pyplot as plt
import pickle
import os
import glob
import io
import base64
from IPython.display import HTML
from IPython import display as ipythondisplay
```

In [8]:
```
import otter
grader = otter.Notebook(colab=True, tests_dir='./public/a2/tests')
```

In [9]:
```
def plot(
    xs_list,
    means_list,
    stds_list,
    losses_list,
    labels_list=None,
    min=None,
    running_average=5,
):
    fig, ax = plt.subplots(1, 2, figsize=(10, 5))
    if labels_list is None:
        labels_list = [f"Agent {idx}" for idx in range(len(means_list))]
    for xs, means, stds, losses, label in zip(
        xs_list, means_list, stds_list, losses_list, labels_list
    ):
        kernel = np.ones(running_average) / running_average
        means_convolved = np.convolve(means, kernel, mode="same")
        stds_convolved = np.convolve(stds, kernel, mode="same")
        ax[0].plot(xs, means_convolved, label=label)
        ax[0].fill_between(
            xs,
            np.array(means_convolved) - np.array(stds_convolved),
            np.array(means_convolved) + np.array(stds_convolved),
            alpha=0.5,
        )
        ax[1].plot(xs, losses, label=label)
    if min is not None:
        ax[0].set_ylim(min, None)
    ax[0].legend()
    ax[0].set_ylabel("Reward")
    ax[1].set_ylabel("Loss")

    return fig, ax
```

In [37]:
```
class ExpertAgent(torch.nn.Module):
    def __init__(self, filename):
        super().__init__()
        self._network = torch.load(filename)
        self._network.eval()

    def get_action(self, obs: np.array):
        """
        Get action from the expert agent.

        Args:
            obs: np.array of shape (state_dim,)
        Returns:
            action: np.array of shape (action_dim,)
        """
        obs = torch.tensor(obs, dtype=torch.float32)
        return self._network(obs).cpu().detach().numpy()
```

Q1 Getting started with RL (10 pts)

For this assignment, we will be using the Ant-v4 environment. The goal in this environment is to have the "Ant" run as far as it can for 1000 timesteps, with the reward being a linear combination of how far it ran, how long it was in a "healthy" state, and a penalty for taking actions that are too large. The actions control the torque for the motors at each of the 8 joints of the agent.

This environment is part of the gymnasium package, a library which provides a standard interface for environments used across many different RL research projects. For this assignment, you will need to familiar with the interface provided by the Env class. Specifically, `env.reset()` and `env.step()`. `env.reset()` resets the environment and agent to the start of the episode. It does not have any required arguments, and it returns `(obs, info)`, where `obs` is the first observation of the episode, and `info` is a dictionary containing additional information (you will not need to interact with `info`). To take actions in the environment, call `env.step`, which takes in an action, and returns `(obs, reward, terminated, truncated, info)`, where `obs` is the next state, `reward` is the reward for step just taken, `terminated` refers to whether the episode entered a terminal state, `truncated` refers to whether the episode was ended before entering a terminal state, and `info` contains any extra info the environment wants to provide.

Q1.a: Agent Evaluation (4 pts)

As a warmup and introduction to interactive environments, implement the `evaluate_agent` function below. It should collect `num_episodes` trajectories in the environment, and return the mean and standard deviation of the episode returns.

In [11]:
```
def evaluate_agent(agent, env, num_episodes):
    """ Collect num_episodes trajectories for the agent and compute mean and st
    rewards. Remember to reset the environment before each episode.
    Args:
        agent: Agent, agent to evaluate
        env: gym.Env, environment to evaluate agent on
        num_episodes: int, number of episodes to evaluate the agent for
    Returns:
        mean_return: float, mean return over the episodes
        std_return: float, standard deviation of the return over the episodes
    """
    returns = []
    # TODO
    for episode in range(num_episodes):
      obs, _ = env.reset()
      ret = 0
      while (True):
        action = agent.get_action(obs)
        obs, reward, terminated, truncated, _ = env.step(action)
        ret += reward
        if terminated or truncated:
          break
      returns.append(ret)
```

```
    returns = np.array(returns)
    mean_returns = np.mean(returns)
    std_returns = np.std(returns)
    return mean_returns, std_returns
```

In [12]:
```
grader.check("q1a")
```

Out[12]:

q1a passed! 🚀

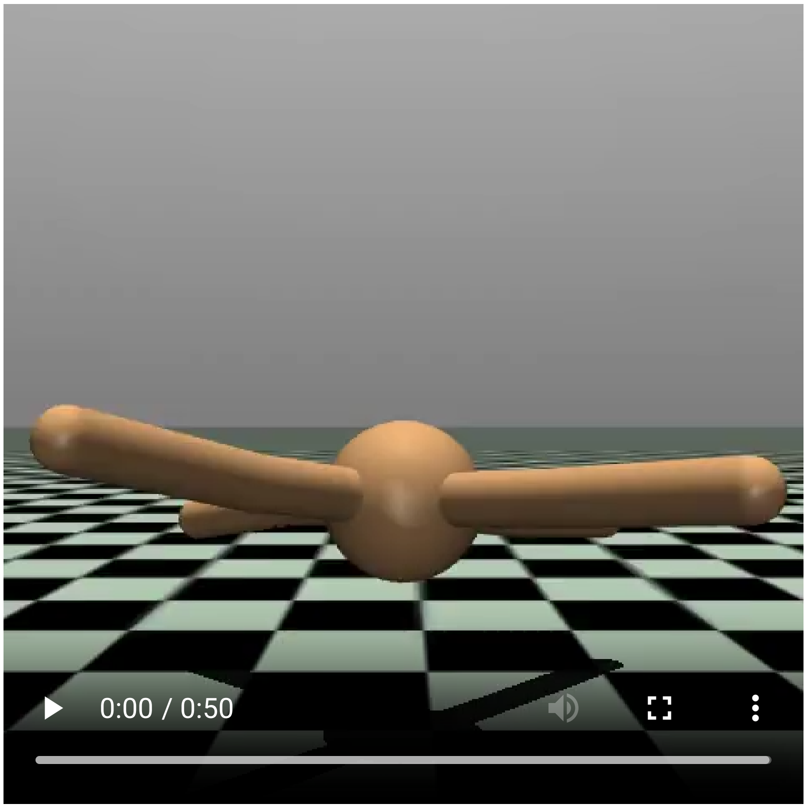
In [13]:
```
VIDEO_LOCATION = "./content/video"


def show_video():
    mp4list = glob.glob(f"{VIDEO_LOCATION}/*.mp4")
    if len(mp4list) > 0:
        mp4 = mp4list[0]
        video = io.open(mp4, "r+b").read()
        encoded = base64.b64encode(video)
        ipythondisplay.display(
            HTML(
                data="""<video alt="test" autoplay
                loop controls style="height: 400px;">
                <source src="data:video/mp4;base64,{0}" type="video/mp4" />
             </video>""".format(
                    encoded.decode("ascii")
                )
            )
        )
    else:
        print("Could not find video")


def create_video(vis_env, agent, name_prefix="imitation_learning"):
    vis_env = wrappers.RecordVideo(vis_env, VIDEO_LOCATION, name_prefix=name_pr
    evaluate_agent(agent, vis_env, 1)
    vis_env.close_video_recorder()
    show_video()
```

Let's now visualize what this looks like.

In [14]:
```
env = gym.make("Ant-v4")
vis_env = gym.make("Ant-v4", render_mode="rgb_array")
a = env.action_space
expert_1mil = ExpertAgent("./public/a2/experts/network_1mil.pt")
mean, std = evaluate_agent(expert_1mil, env, 10)
print(f"Expert mean return: {mean} +/- {std}")
create_video(vis_env, expert_1mil, "expert_1mil")
```

```
Expert mean return: 4326.538382429369 +/- 1398.7059618573292
Moviepy - Building video /content/content/video/expert_1mil-episode-0.mp4.
Moviepy - Writing video /content/content/video/expert_1mil-episode-0.mp4
```

```
Moviepy - Done !
Moviepy - video ready /content/content/video/expert_1mil-episode-0.mp4
```



Q1.b: Replay Buffer (3 pts)

Next, we will implement a replay buffer. In RL, we typically store states, actions, rewards, next states, and termination for each transition, but for this assignment, because we are only doing imitation learning (not learning from rewards!), we only need to store states and actions for each transition. Fill in the missing sample function below.

In [25]:
```
class ReplayBuffer:
    def __init__(self, max_size=100_000):
        self._max_size = max_size
        self._states = None
        self._actions = None

    def add_rollouts(self, rollouts):
        """
        Add rollouts to the buffer

        Args:
            rollouts: dict, with keys "states" and "actions", with shapes
                (rollout_length, state_dim) and (rollout_length, action_dim)
                respectively.
        """
        if self._states is None:
            self._states = rollouts["states"][-self._max_size :]
            self._actions = rollouts["actions"][-self._max_size :]
        else:
            self._states = np.concatenate([self._states, rollouts["states"]])[
                -self._max_size :
            ]
            self._actions = np.concatenate([self._actions, rollouts["actions"]]
```

```
                -self._max_size :
            ]

    def sample(self, batch_size):
        """
        Sample batch_size elements from the buffer without replacement.

        Args:
            batch_size: int, number of elements to sample
        Returns:
            states: np.array of shape (batch_size, state_dim)
            actions: np.array of shape (batch_size, action_dim)
        """
        if self._states is None or self._actions is None:
            raise ValueError("No data in buffer")

        # TODO: Sample batch_size random elements from self.states and self.act
        rollout_size = len(self._states)
        rand_idx = np.random.randint(low=0, high=rollout_size, size=batch_size
        states = self._states[rand_idx]
        actions = self._actions[rand_idx]
        return states, actions

    def __len__(self):
        return len(self._states) if self._states is not None else 0
```

In [26]:
```
grader.check("q1b")
```

Out[26]:

q1b passed! 🌟

Q1.c: Agent (3 pts)

Finally, we come to the agent, which is the entity that selects actions to perform in the environment. We've provided the network architecture below. It's up to you to fill in the agent's `forward` and `get_action` functions. They do similar things, but keep in mind the expected function signature!

In [38]:
```
class Agent(torch.nn.Module):
    def __init__(self, obs_dim, action_dim):
        super().__init__()
        self._network = torch.nn.Sequential(
            torch.nn.Linear(obs_dim, 256),
            torch.nn.ReLU(),
            torch.nn.Linear(256, 256),
            torch.nn.ReLU(),
            torch.nn.Linear(256, action_dim),
        )

    def forward(self, obs_tensor: torch.Tensor):
        """
        Returns the actions for a batch of observations.

        Args:
            obs_tensor: torch.Tensor of shape (batch_size, obs_dim)
        Returns:
```

```python
            action_tensor: torch.Tensor of shape (batch_size, action_dim)
        """
        # TODO
        return self._network(obs_tensor)

    def get_action(self, obs: np.ndarray) -> np.ndarray:
        """
        Get action from the agent for a single observation.

        Args:
            obs: np.ndarray of shape (obs_dim,)
        Returns:
            action: np.ndarray of shape (action_dim,)
        """

        # TODO Predict the action given the observation
        tensor_obs = torch.tensor(obs, dtype=torch.float32)
        return self._network(tensor_obs).cpu().detach().numpy()
        # pass
```

In [39]:
```python
grader.check("q1c")
```

Out[39]:

q1c passed! 🙌

Q2: Behavior cloning (20 pts)

Q2.a Implement Behavior Cloning (15 pts)

We now come to our first Imitation Learning algorithm: behavior cloning. Run `steps` steps of gradient descent using the `optimizer` with the predictions coming from the `agent` and input and targets coming from the `buffer` in batch sizes of `batch_size`. Since this is a continuous action space, we will be using a regression loss, specifically average mean squared error: $l(\mathbf{x}, \mathbf{y}) = \frac{\sum_{m=1}^{M} \sum_{n=1}^{N} (x_n^m - y_n^m)^2}{N \times M}$, where M is the batch size, N is the dimension of each sample, and x_n^m refers to the n-th dimension of the m-th sample.

In [29]:
```python
def behavior_cloning(agent, optimizer, buffer, batch_size=128, steps=1000):
    """
    Args:
        agent: Agent, agent to train
        optimizer: torch.optim.Optimizer, optimizer to use
        buffer: ReplayBuffer, buffer to sample from
        batch_size: int, batch size
        steps: int, number of steps to train
    Returns:
        loss: float, Average loss over the last 5 steps
    """

    losses = []
    # TODO: Implement the behavior cloning training loop
    # Hint: Store the loss values in losses list to compute the final average o
    # last 5 steps
```

```
    # Hint: Take a look at torch.nn.functional for useful functions for comput
    # loss

    for step in range(steps):
      states, actions = buffer.sample(batch_size)

      states = torch.tensor(states, dtype=torch.float32)
      actions = torch.tensor(actions, dtype=torch.float32)
      preds = agent(states)

      loss = torch.nn.functional.mse_loss(input=actions, target=preds, reductio

      optimizer.zero_grad()
      loss.backward()
      optimizer.step()

      losses.append(loss.cpu().detach().numpy())

    return np.mean(losses[-5:])
```

In [30]:

```
grader.check("q2.a")
```

Out[30]:

q2.a passed! 🌈

Q2.b Run Behavior Cloning (5 pts)

Run behavior cloning on the curated data given above for 1000 steps. Then evaluate the agent for 10 episodes, reporting the mean and standard deviation. You should get at least 50% of the average expert return.

In [31]:

```
with open("./public/a2/expert_data/expert_data_Ant-v4.pkl", "rb") as f:
    data = pickle.load(f)
    states = np.concatenate([trajectory["observation"][:, :27] for trajectory i
    actions = np.concatenate([trajectory["action"] for trajectory in data])
    data_average_reward = np.mean([np.sum(trajectory["reward"]) for trajectory
print(f"Average expert return: {data_average_reward}")
```

```
Average expert return: 4713.6533203125
```

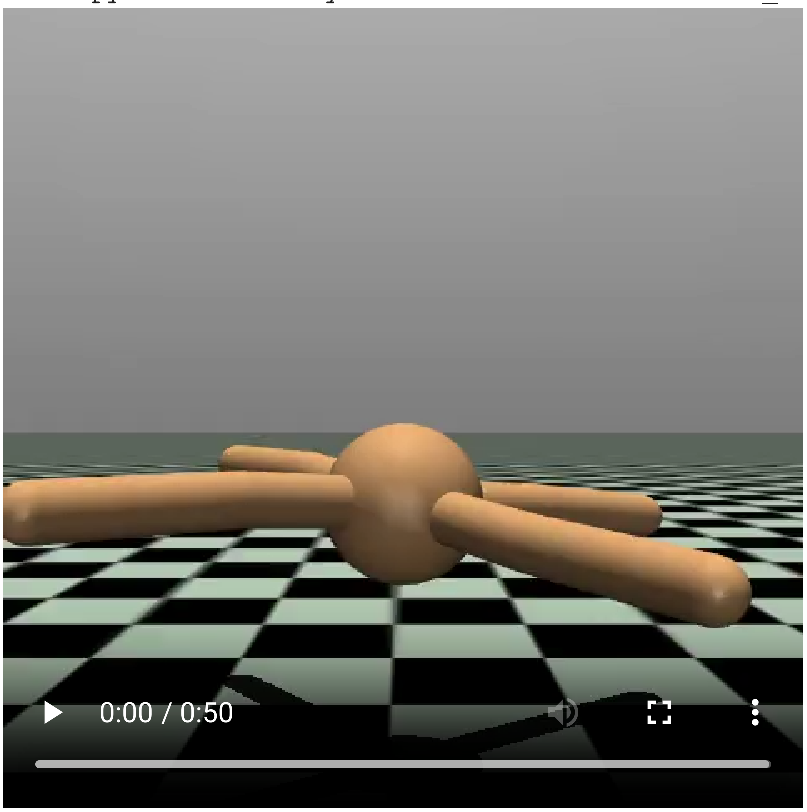
In [32]:

```
BATCH_SIZE = 128
STEPS = 1000
bc_agent = Agent(env.observation_space.shape[0], env.action_space.shape[0])
optimizer = torch.optim.Adam(bc_agent.parameters(), lr=5e-3)
bc_buffer = ReplayBuffer()

# TODO: Add the states and actions from the expert curated data to the buffer.
# Then run behavior cloning and evaluate the agent.

bc_buffer.add_rollouts(rollouts={'states':states, 'actions':actions})
behavior_cloning(bc_agent, optimizer, bc_buffer, batch_size=BATCH_SIZE, steps=S

mean, std = evaluate_agent(bc_agent, env, num_episodes=10)


print(
    f"The agent trained on the curated dataset has an average reward of {mean}
```

```
)
create_video(vis_env, bc_agent, name_prefix="ant_curated")
```

```
The agent trained on the curated dataset has an average reward of 4026.0866326
696073 +/- 1085.630951574564
```

```
/usr/local/lib/python3.10/dist-packages/gymnasium/wrappers/record_video.py:94:
UserWarning: WARN: Overwriting existing videos at /content/content/video folde
r (try specifying a different `video_folder` for the `RecordVideo` wrapper if
this is not desired)
  logger.warn(
Moviepy - Building video /content/content/video/ant_curated-episode-0.mp4.
Moviepy - Writing video /content/content/video/ant_curated-episode-0.mp4

Moviepy - Done !
Moviepy - video ready /content/content/video/ant_curated-episode-0.mp4
```



Q3 DAgger Implementation (30 pts)

Finally, we look at the Dataset Aggregation (DAgger) algorithm. Each iteration of this algorithm involves dataset collection, data relabeling with an expert policy, and behavior cloning.

In [36]:
```
def relabel_with_expert(states, expert_agent):
    """
    Args:
        states: np.array of shape (batch_size, state_dim)
        expert_agent: ExpertAgent
    Returns:
        actions: np.array of shape (batch_size, action_dim)
    """
```

```
    actions = []

    # TODO: Loop through the states, and get the expert action
    # for each state
    # Hint: Use expert_agent.get_action
    actions = expert_agent.get_action(states)
    return np.array(actions)
```

In [33]:
```
def collect_rollouts(env, agent, n_to_collect=1000):
    """
    Args:
        env: gym.Env
        agent: Agent
        n_to_collect: int, number of states to collect
    Returns:
        states: np.array of shape (n_to_collect, state_dim)
        actions: np.array of shape (n_to_collect, action_dim)
    """
    states = []
    actions = []
    state, _ = env.reset()
    done = False

    ### TODO: Collect rollouts until we have n_to_collect states
    # Hint: Remember to reset the environment when a rollout is finished

    for i in range(n_to_collect):
      old_state = state
      action = agent.get_action(old_state)
      state, reward, terminated, truncated, _ = env.step(action)

      states.append(old_state)
      actions.append(action)

      done = terminated or truncated
      if done:
        env.reset()
        continue
    env.reset()
    return np.array(states), np.array(actions)
```

In [40]:
```
def seed_data(env, expert_agent, buffer, n_to_collect=1000):
    """
    Collects rollouts using the expert agent and adds them to the buffer.

    Args:
        env: gym.Env
        expert_agent: ExpertAgent
        buffer: ReplayBuffer
        n_to_collect: int, number of samples to collect
    """

    ### TODO: Implement this function
    states, actions = collect_rollouts(env, expert_agent, n_to_collect)
    buffer.add_rollouts({'states':states, 'actions':actions})
```

In [41]:
```
grader.check("q3")
```

Out[41]:

q3 passed! 💯

In [42]:
```
def dagger_iteration(
    agent,
    optimizer,
    expert_agent,
    env,
    buffer,
    n_to_collect,
    steps=1000,
    batch_size=128,
):
    """
    Implements one iteration of the DAgger algorithm. Collects the rollouts usi
    agent, relabels them using the expert, and trains the agent for `steps` ste
    behavior cloning.

    Args:
        agent: Agent
        optimizer: torch.optim.Optimizer
        expert_agent: ExpertAgent
        env: gym.Env
        buffer: ReplayBuffer
        n_to_collect: int, number of samples to collect
        steps: int, number of steps to train
        batch_size: int, batch size
    Returns:
        loss: float, Average loss over the last 5 steps of behavior
            cloning
    """

    ### TODO: Implement one iteration of the DAgger algorithm
    states, actions = collect_rollouts(env, agent, n_to_collect)
    expert_actions = relabel_with_expert(states, expert_agent)
    buffer.add_rollouts({'states':states, 'actions':expert_actions})
    loss = behavior_cloning(agent, optimizer, buffer, batch_size=batch_size, st

    return loss
```

In [43]:
```
def dagger(
    agent,
    optimizer,
    expert_agent,
    env,
    buffer,
    collect_per_iteration=2000,
    n_iterations=10,
    gradient_steps=1000,
    batch_size=128,
    n_episodes_eval=10,
):
    """
    Runs the DAgger algorithm for `n_iterations` iterations. The loss from each
    iteration is stored and returned. After each iteration, the agent is evalua
    `n_episodes_eval` episodes. The mean and std of the rewards are stored and
```

```
    Args:
        agent: Agent
        optimizer: torch.optim.Optimizer
        expert_agent: ExpertAgent
        env: gym.Env
        buffer: ReplayBuffer
        collect_per_iteration: int, number of samples to collect per iteration
        n_iterations: int, number of DAgger iterations
        gradient_steps: int, number of steps to train the agent for per iterati
        batch_size: int, batch size
        n_episodes_eval: int, number of episodes to evaluate the agent for
    Returns:
        losses: list of floats, losses from each DAgger iteration
        means: list of floats, mean rewards from each DAgger iteration
        stds: list of floats, std of rewards from each DAgger iteration
    """
    losses, means, stds = [], [], []

    ### TODO: Implement the DAgger algorithm
    # Hint: It might be helpful when running stuff later on to also print
    # which iteration of DAgger you are on

    for iteration in range(n_iterations):
      iteration_loss = dagger_iteration(agent, optimizer, expert_agent, env, bu
                                        , collect_per_iteration, gradient_steps,
                                        batch_size)
      losses.append(iteration_loss)

      mean, std = evaluate_agent(agent, env, n_episodes_eval)
      means.append(mean)
      stds.append(std)

      print(f'Daggar iter {iteration}: loss {iteration_loss} mean rewards {mean

    return losses, means, stds
```

Q4 Analyzing DAgger

Now, you will perform various experiments to test and analyze the performance of behavior cloning and DAgger.

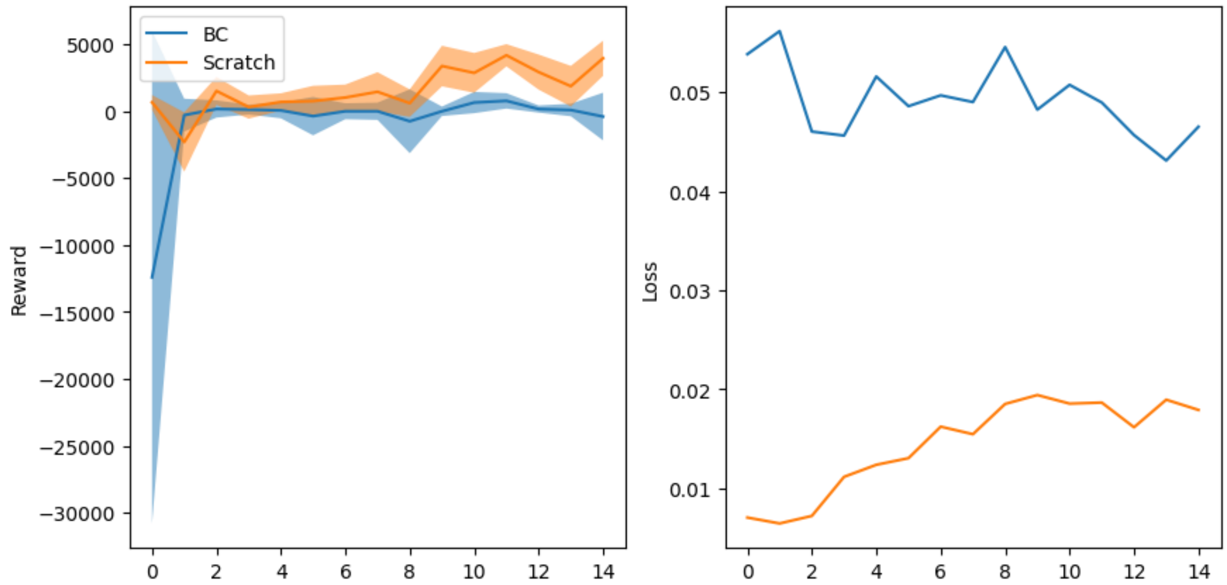
Q4.a: DAgger with policy drift

You currently have access to two agents: the `expert_1mil` policy that we provided you, and the `bc_agent` learned through behavior cloning the curated expert data. Starting from the same agent and replay buffer as the behavior cloning experiment above, run 15 iterations of DAgger with the `expert_1mil` policy. Then, reset the agent and buffer, do 15 iterations of DAgger with the `expert_1mil` policy starting from a random agent and empty replay buffer. Plot the loss and average mean with standard deviation using the plotting function above.

In [44]:
```
# Run DAgger starting from agent pretrained on curated data data
expert = ExpertAgent("./public/a2/experts/network_1mil.pt")
agent = bc_agent
buffer = bc_buffer
optimizer = torch.optim.Adam(agent.parameters(), lr=5e-3)
losses_bc, means_bc, stds_bc = dagger(
    agent, optimizer, expert, env, buffer, 2000, 15, 2000, 128, 10
)

# Run DAgger starting from scratch, using the same expert
agent = Agent(env.observation_space.shape[0], env.action_space.shape[0])
buffer = ReplayBuffer()
optimizer = torch.optim.Adam(agent.parameters(), lr=5e-3)
seed_data(env, expert, buffer, 2000)
losses_scratch, means_scratch, stds_scratch = dagger(
    agent, optimizer, expert, env, buffer, 2000, 15, 2000, 128, 10
)
plot(
    [np.arange(len(losses_bc)), np.arange(len(losses_scratch))],
    [means_bc, means_scratch],
    [stds_bc, stds_scratch],
    [losses_bc, losses_scratch],
    ["BC", "Scratch"],
    running_average=1,
)
```

```
Daggar iter 0: loss 0.053851883858442307 mean rewards -12411.510373442721, std reward 18377.547876455923
Daggar iter 1: loss 0.0561748743057251 mean rewards -302.44948700384174, std reward 1223.2113184991513
Daggar iter 2: loss 0.04602999612689018 mean rewards 154.7085571204719, std reward 642.2024880389368
Daggar iter 3: loss 0.04562633857131004 mean rewards 111.34061203331194, std reward 366.79585491417237
Daggar iter 4: loss 0.05159445479512215 mean rewards 48.54166397768819, std reward 589.0156277252217
Daggar iter 5: loss 0.04859461262822151 mean rewards -377.2165468946029, std reward 1434.2156121600044
Daggar iter 6: loss 0.04968901723623276 mean rewards -14.009150564215469, std reward 594.0887767552216
Daggar iter 7: loss 0.049012456089258194 mean rewards -14.83271775051801, std reward 643.5941596028308
Daggar iter 8: loss 0.05456589534878731 mean rewards -760.4434477708637, std reward 2382.6753813981177
Daggar iter 9: loss 0.048246659338474274 mean rewards -15.683727840870684, std reward 351.75723810117796
Daggar iter 10: loss 0.050749778747558594 mean rewards 635.6452357029266, std reward 794.0721694093387
Daggar iter 11: loss 0.04896288737654686 mean rewards 762.0131198002298, std reward 573.9454010756453
Daggar iter 12: loss 0.045673951506614685 mean rewards 149.72669026460818, std reward 267.0259642500106
Daggar iter 13: loss 0.043101660907268524 mean rewards 78.47892316444272, std reward 466.6359181172754
Daggar iter 14: loss 0.04651679843664169 mean rewards -409.2174261632862, std reward 1778.6928190066171
Daggar iter 0: loss 0.00702967494726181 mean rewards 652.7813209402722, std reward 542.4177982850364
Daggar iter 1: loss 0.0064337230287492275 mean rewards -2311.2732213170134, std reward 2208.9697817069964
Daggar iter 2: loss 0.007193677127361298 mean rewards 1501.6971986144858, std reward 1038.791453594428
Daggar iter 3: loss 0.011142143979668617 mean rewards 297.457499068086, std reward 866.3864590499553
Daggar iter 4: loss 0.012360415421426296 mean rewards 678.9963072486862, std reward 645.5416374860877
Daggar iter 5: loss 0.013030603528022766 mean rewards 745.9252219959818, std reward 1140.3361614671633
Daggar iter 6: loss 0.01621313951909542 mean rewards 1010.3242029202702, std reward 965.5413571465923
Daggar iter 7: loss 0.015451696701347828 mean rewards 1439.3596165223587, std reward 1462.0762312291974
Daggar iter 8: loss 0.018508316949009895 mean rewards 590.0189314891071, std reward 1026.2836854355141
Daggar iter 9: loss 0.019407670944929123 mean rewards 3362.973373648093, std reward 1514.5480232677114
Daggar iter 10: loss 0.01854241080582142 mean rewards 2851.7705977324995, std reward 1484.7835809790092
Daggar iter 11: loss 0.018645621836185455 mean rewards 4166.151237276654, std reward 829.6744380799139
Daggar iter 12: loss 0.01615149714052677 mean rewards 2911.066994995135, std reward 1308.7886585900478
Daggar iter 13: loss 0.01893678680062294 mean rewards 1856.380937973177, std reward 1516.1690428740253
Daggar iter 14: loss 0.017903026193380356 mean rewards 3941.2530558258245, std reward 1308.8015983738499
```

Out[44]:
```
(<Figure size 1000x500 with 2 Axes>,
 array([<Axes: ylabel='Reward'>, <Axes: ylabel='Loss'>], dtype=object))
```

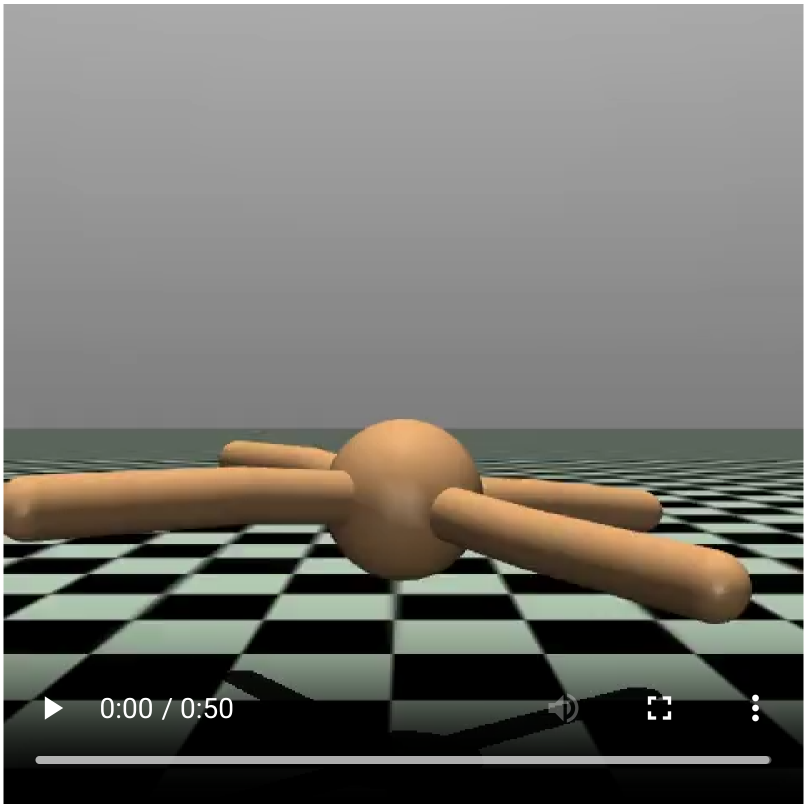


From the results above, it is clear that an agent started from scratch then taking data through Daggar from the expert does better than starting with behavior cloning then running daggar.

For the rest of this assignment, we will be using a new expert agent. Evaluate and visualize it below.

In [45]:
```
expert_2mil = ExpertAgent("./public/a2/experts/network_2mil.pt")
mean, std = evaluate_agent(expert_2mil, env, 10)
print(f"Expert mean return: {mean} +/- {std}")
create_video(vis_env, expert_2mil, "expert_2mil")
```

```
Expert mean return: 5540.091606309945 +/- 1015.624886294598
Moviepy - Building video /content/content/video/expert_2mil-episode-0.mp4.
Moviepy - Writing video /content/content/video/expert_2mil-episode-0.mp4

Moviepy - Done !
Moviepy - video ready /content/content/video/expert_2mil-episode-0.mp4
```



Q4.b Exploring the effect of the effect of the strength of the expert on DAgger

We now look at how the strength of the expert affects our imitation learned algorithm. The `expert_1mil` and `expert_2mil` are both policies from the same training run, except the `expert_1mil` was trained for 1 million steps and `expert_2mil` was trained for 2 million steps.

from the results below, the agent that was trained for 1 million steps starts better, but soon the agent trained for 2 million steps exceeds it in terms of return.

In [46]:
```
N_ITERS = 50
N_DATA_PER_ITER = 2000
N_GRADIENT_STEPS = 2000
expert_strength_data = {
    "all_means": [],
    "all_stds": [],
    "all_losses": [],
    "all_xs": [],
}
for expert in [expert_1mil, expert_2mil]:
    agent = Agent(env.observation_space.shape[0], env.action_space.shape[0])
    optimizer = torch.optim.Adam(agent.parameters(), lr=5e-3)
    buffer = ReplayBuffer()
    seed_data(env, expert, buffer, 2000)

    # TODO: Run DAgger for the given expert
    losses, means, stds = dagger(agent, optimizer, expert, env, buffer, N_DATA_
```

```
xs = np.arange(N_ITERS) + 1
expert_strength_data["all_xs"].append(xs)
expert_strength_data["all_means"].append(means)
expert_strength_data["all_stds"].append(stds)
expert_strength_data["all_losses"].append(losses)
```

```
Daggar iter 0: loss 0.005640469491481781 mean rewards -891.2385282584971, std
reward 2120.460316361217
Daggar iter 1: loss 0.006118227727711201 mean rewards 692.3879219161608, std r
eward 516.6166023524502
Daggar iter 2: loss 0.011012531816959381 mean rewards 182.550507829654, std re
ward 433.05506795830524
Daggar iter 3: loss 0.01379055343568325 mean rewards 1234.4787228021817, std r
eward 804.9395265016319
Daggar iter 4: loss 0.01856410875916481 mean rewards 419.74415574246007, std r
eward 643.2415389430918
Daggar iter 5: loss 0.019754478707909584 mean rewards 1924.4487853162289, std
reward 1329.0864479339223
Daggar iter 6: loss 0.020957063883543015 mean rewards 1304.2142977613796, std
reward 1290.130387018249
Daggar iter 7: loss 0.0191842969506979 mean rewards 1253.528568924086, std rew
ard 1264.8398736646582
Daggar iter 8: loss 0.022587869316339493 mean rewards 961.0425173376237, std r
eward 1212.0671020185423
Daggar iter 9: loss 0.022563226521015167 mean rewards 2636.5707210928986, std
reward 1211.2128069367031
Daggar iter 10: loss 0.020928001031279564 mean rewards 2965.6525222217233, std
reward 1497.106490188569
Daggar iter 11: loss 0.02475527860224247 mean rewards 3746.3889273319714, std
reward 1231.2089184159088
Daggar iter 12: loss 0.024702038615942 mean rewards 972.0334940053087, std rew
ard 1292.0244802506832
Daggar iter 13: loss 0.027123814448714256 mean rewards 4098.595529376212, std
reward 1198.415281729211
Daggar iter 14: loss 0.023219645023345947 mean rewards 3983.1655071233276, std
reward 1157.0169194712423
Daggar iter 15: loss 0.023891976103186607 mean rewards 3886.4521006168966, std
reward 1082.161326611298
Daggar iter 16: loss 0.02569752000272274 mean rewards 3333.630438975095, std r
eward 1806.9398053902137
Daggar iter 17: loss 0.024920683354139328 mean rewards 4126.7789064000135, std
reward 1377.813909744562
Daggar iter 18: loss 0.024908775463700294 mean rewards 4101.482235833633, std
reward 1347.0834785269644
Daggar iter 19: loss 0.022397827357053757 mean rewards 3261.3233207712988, std
reward 1318.7347424831776
Daggar iter 20: loss 0.024118348956108093 mean rewards 3563.038923300789, std
reward 1241.9394506521191
Daggar iter 21: loss 0.024448268115520477 mean rewards 2854.9012325357035, std
reward 1621.7507408496456
Daggar iter 22: loss 0.023420561105012894 mean rewards 3828.59045438654, std r
eward 1137.7701038136852
Daggar iter 23: loss 0.024259280413389206 mean rewards 2677.8975766961107, std
reward 1998.2000401132177
Daggar iter 24: loss 0.02168317325413227 mean rewards 4077.8026772749035, std
reward 1328.8389689435348
Daggar iter 25: loss 0.024797901511192322 mean rewards 4596.167670346857, std
reward 47.84644016041856
Daggar iter 26: loss 0.02206668071448803 mean rewards 3799.1120656571666, std
reward 1458.5593124241452
Daggar iter 27: loss 0.023472633212804794 mean rewards 4053.183191500192, std
reward 1356.548247250767
Daggar iter 28: loss 0.02264994941651821 mean rewards 4366.578719799198, std r
eward 706.6912505308586
Daggar iter 29: loss 0.020363379269838333 mean rewards 3737.59305327759, std r
eward 1557.728663822125
```

```
Daggar iter 30: loss 0.02167263813316822 mean rewards 3930.986152796944, std r
eward 1279.3890820432898
Daggar iter 31: loss 0.022363563999533653 mean rewards 4021.4936266738537, std
reward 1397.5812238865356
Daggar iter 32: loss 0.0234963558614254 mean rewards 4095.9964220286397, std r
eward 1246.2817954510026
Daggar iter 33: loss 0.02225363627076149 mean rewards 3402.0999635534436, std
reward 1601.9501232792245
Daggar iter 34: loss 0.023491747677326202 mean rewards 3861.885584218241, std
reward 1476.7355465705273
Daggar iter 35: loss 0.02151157520711422 mean rewards 4323.513051397572, std r
eward 530.6682997036055
Daggar iter 36: loss 0.022884462028741837 mean rewards 3918.961538570915, std
reward 1022.9576424278326
Daggar iter 37: loss 0.021897464990615845 mean rewards 4092.369003693226, std
reward 768.4506991942153
Daggar iter 38: loss 0.020398445427417755 mean rewards 3585.0798961011787, std
reward 1525.1859402112873
Daggar iter 39: loss 0.025509754195809364 mean rewards 3055.9969026848676, std
reward 1671.4940811148158
Daggar iter 40: loss 0.023746702820062637 mean rewards 3423.2016982282503, std
reward 1749.414531974463
Daggar iter 41: loss 0.02298416942358017 mean rewards 2931.8941653480683, std
reward 2089.2452768369158
Daggar iter 42: loss 0.021753709763288498 mean rewards 3574.497869927434, std
reward 1640.9883252541943
Daggar iter 43: loss 0.021872544661164284 mean rewards 3486.5009637721364, std
reward 1408.6857523089575
Daggar iter 44: loss 0.02270788513123989 mean rewards 3627.532680277612, std r
eward 1218.6074317024218
Daggar iter 45: loss 0.020859502255916595 mean rewards 3715.5903901791135, std
reward 1220.9714221279557
Daggar iter 46: loss 0.023558948189020157 mean rewards 4163.795791003341, std
reward 893.7392886529548
Daggar iter 47: loss 0.022047577425837517 mean rewards 4098.219643636335, std
reward 943.7356778538261
Daggar iter 48: loss 0.02141118422150612 mean rewards 3585.9067079429597, std
reward 1195.828293884772
Daggar iter 49: loss 0.02192659303545952 mean rewards 2837.0715397456347, std
reward 1479.914131004714
Daggar iter 0: loss 0.005803184118121862 mean rewards 73.6637808141817, std re
ward 1699.3029089832717
Daggar iter 1: loss 0.003920483402907848 mean rewards -2144.511504691577, std
reward 3650.8807326254396
Daggar iter 2: loss 0.009922606870532036 mean rewards -419.73029027461945, std
reward 1312.1557730257716
Daggar iter 3: loss 0.011331619694828987 mean rewards -606.4456427927204, std
reward 699.5700322862292
Daggar iter 4: loss 0.012785017490386963 mean rewards 5.189703223965644, std r
eward 609.0954369604425
Daggar iter 5: loss 0.018123364076018333 mean rewards 78.09755272735275, std r
eward 450.21438520856975
Daggar iter 6: loss 0.02012821100652218 mean rewards 513.288464010988, std rew
ard 976.8372184131026
Daggar iter 7: loss 0.02414259873330593 mean rewards 1073.7637115091197, std r
eward 1088.8042459216138
Daggar iter 8: loss 0.019057825207710266 mean rewards 1242.6609583009945, std
reward 1319.4746008616353
Daggar iter 9: loss 0.022217875346541405 mean rewards 378.29194595745196, std
reward 800.8231645039853
```

```
Daggar iter 10: loss 0.020559554919600487 mean rewards 445.796181037478, std r
eward 806.8459242582101
Daggar iter 11: loss 0.023888397961854935 mean rewards 3196.7944911611244, std
reward 2051.008170111115
Daggar iter 12: loss 0.024959737434983253 mean rewards 1109.8522106984758, std
reward 1196.50099575177
Daggar iter 13: loss 0.0275846179574728 mean rewards 3123.417252438796, std re
ward 2061.5512378864832
Daggar iter 14: loss 0.026446884498000145 mean rewards 2452.888507855518, std
reward 1984.6087206531759
Daggar iter 15: loss 0.028966030105948448 mean rewards 4895.1160757612415, std
reward 589.8543773039216
Daggar iter 16: loss 0.025905132293701172 mean rewards 1897.9817263794769, std
reward 1779.999177300487
Daggar iter 17: loss 0.027665670961141586 mean rewards 3950.1255689157797, std
reward 1884.704090226156
Daggar iter 18: loss 0.026595909148454666 mean rewards 5269.245410281747, std
reward 123.92051852320265
Daggar iter 19: loss 0.02628614380955696 mean rewards 3730.470540048534, std r
eward 1905.4482530225528
Daggar iter 20: loss 0.02850746549665928 mean rewards 3585.036440633439, std r
eward 1902.3313758472646
Daggar iter 21: loss 0.032224010676145554 mean rewards 4534.947315701722, std
reward 986.4651113872619
Daggar iter 22: loss 0.029574517160654068 mean rewards 4672.888878119974, std
reward 1521.8614860041528
Daggar iter 23: loss 0.02956385537981987 mean rewards 5373.08464489438, std re
ward 181.785005551004
Daggar iter 24: loss 0.02575113996863365 mean rewards 3713.445181507489, std r
eward 1870.5402224436814
Daggar iter 25: loss 0.02677900530397892 mean rewards 4326.310004529385, std r
eward 1817.3450097066823
Daggar iter 26: loss 0.02789776399731636 mean rewards 1389.7830415966896, std
reward 1070.5456148101773
Daggar iter 27: loss 0.02932826615869999 mean rewards 2672.706609963682, std r
eward 2157.319569676169
Daggar iter 28: loss 0.027939921244978905 mean rewards 4755.125138440756, std
reward 1048.2208034271384
Daggar iter 29: loss 0.029603878036141396 mean rewards 4928.799531020399, std
reward 1624.372229908566
Daggar iter 30: loss 0.028027033433318138 mean rewards 4609.845897204357, std
reward 1631.6437869904157
Daggar iter 31: loss 0.02881128154695034 mean rewards 5442.676868465347, std r
eward 118.66007096455014
Daggar iter 32: loss 0.028711756691336632 mean rewards 3975.3710193441707, std
reward 2197.6814873831563
Daggar iter 33: loss 0.02657311223447323 mean rewards 5275.05801225944, std re
ward 96.07408791587926
Daggar iter 34: loss 0.02668122388422489 mean rewards 3119.53157812907, std re
ward 2280.7988106880066
Daggar iter 35: loss 0.025041639804840088 mean rewards 5289.508202667643, std
reward 119.28425332338529
Daggar iter 36: loss 0.025941897183656693 mean rewards 5566.238875546445, std
reward 80.2278738473552
Daggar iter 37: loss 0.028371300548315048 mean rewards 4934.229104578208, std
reward 2198.630444354788
Daggar iter 38: loss 0.0249335877597332 mean rewards 3226.898931638551, std re
ward 2285.68917386535
Daggar iter 39: loss 0.026734029874205 59 mean rewards 5241.7618642059015, std
reward 1578.4154273483032
```

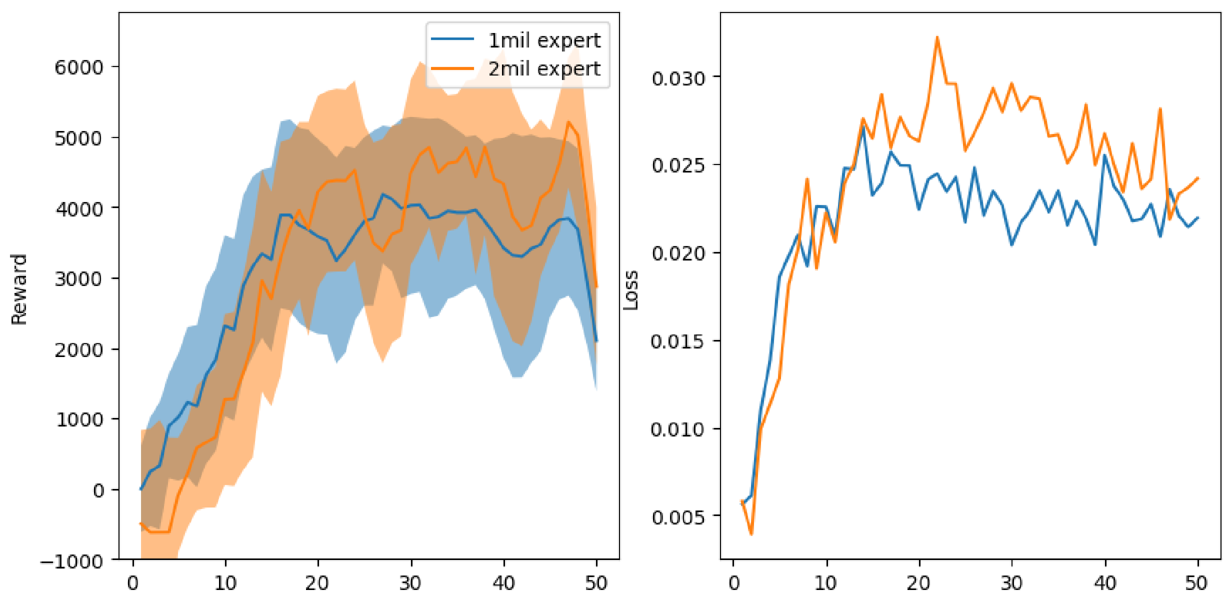
```
Daggar iter 40: loss 0.024984123185276985 mean rewards 3030.951187235468, std reward 2204.701993162545
Daggar iter 41: loss 0.023393727838993073 mean rewards 5217.155329277843, std reward 1312.0967388413148
Daggar iter 42: loss 0.02617359533905983 mean rewards 2598.452985763132, std reward 1455.7813472391924
Daggar iter 43: loss 0.0235968679189682 mean rewards 2268.298208033265, std reward 1703.1227527255091
Daggar iter 44: loss 0.024145979434251785 mean rewards 5564.618649334285, std reward 146.67038217798202
Daggar iter 45: loss 0.028139162808656693 mean rewards 4994.379494080943, std reward 933.2085608790251
Daggar iter 46: loss 0.021835511550307274 mean rewards 5748.619950216896, std reward 46.971423805221725
Daggar iter 47: loss 0.023301411420106888 mean rewards 4473.900029812744, std reward 1983.3425042853446
Daggar iter 48: loss 0.023662108927965164 mean rewards 5237.02926226283, std reward 1527.8947993068912
Daggar iter 49: loss 0.02417425438761711 mean rewards 4647.734516797263, std reward 2096.7584932277227
```

In [47]:

```python
plot(
    expert_strength_data["all_xs"],
    expert_strength_data["all_means"],
    expert_strength_data["all_stds"],
    expert_strength_data["all_losses"],
    [f"{expert} expert" for expert in ["1mil", "2mil"]],
    min=-1000,
)
```

Out[47]:

```
(<Figure size 1000x500 with 2 Axes>,
 array([<Axes: ylabel='Reward'>, <Axes: ylabel='Loss'>], dtype=object))
```



Q4.c Exploring the effect of the number of iterations on DAgger

We will now look at how the frequency of the number of DAgger iterations affects the performance. To make it fair, make sure to control for the total amount of data and gradient steps that will be taken by the algorithm.

from the results below it seems having very few iterations (N=5) yields bad results, using large number of iterations (100, 200) gives good initial performance (agents gets higher reward intially compared to lower values), but eventually (due to the high varaince) it is hard to conclude, but it seems any value in [25, 50, 100, 200] yields good resuls with small differences between them.

In [48]:
```
TOTAL_DATA = 100_000
TOTAL_GRADIENT_STEPS = 100_000
n_iters_data = {
    "all_means": [],
    "all_stds": [],
    "all_losses": [],
    "all_xs": [],
}
expert = ExpertAgent("./public/a2/experts/network_2mil.pt")
for n_iters in [5, 25, 50, 100, 200]:
    agent = Agent(env.observation_space.shape[0], env.action_space.shape[0])
    optimizer = torch.optim.Adam(agent.parameters(), lr=5e-3)
    buffer = ReplayBuffer()
    seed_data(env, expert, buffer, 2000)

    # TODO: Run DAgger for n_iters iterations
    grad_steps = TOTAL_GRADIENT_STEPS // n_iters
    n_data_per_iter = TOTAL_DATA // n_iters

    losses, means, stds = dagger(agent, optimizer, expert, env, buffer, n_data_

    xs = 100_000 / n_iters * (np.arange(n_iters) + 1)
    n_iters_data["all_xs"].append(xs)
    n_iters_data["all_means"].append(means)
    n_iters_data["all_stds"].append(stds)
    n_iters_data["all_losses"].append(losses)
```

```
Daggar iter 0: loss 0.002099753823131323 mean rewards -945.5467681043598, std
reward 3185.856472600903
Daggar iter 1: loss 0.006205093115568161 mean rewards -143.5593691255735, std
reward 1318.110960460655
Daggar iter 2: loss 0.011689399369060993 mean rewards -368.8221121078119, std
reward 1533.4925933370664
Daggar iter 3: loss 0.019047411158680916 mean rewards 2076.777707491599, std r
eward 1608.4684875610533
Daggar iter 4: loss 0.021763348951935768 mean rewards 2956.4680668892697, std
reward 2186.2168033521575
Daggar iter 0: loss 0.004194870125502348 mean rewards 177.70770052354655, std
reward 956.1307301631809
Daggar iter 1: loss 0.0034548796247690916 mean rewards -831.8220076320533, std
reward 1680.9298560962498
Daggar iter 2: loss 0.010017191991209984 mean rewards -425.52749268919445, std
reward 439.42317488077384
Daggar iter 3: loss 0.011912635527551174 mean rewards -50.16616811731367, std
reward 710.5748612059174
Daggar iter 4: loss 0.013775855302810669 mean rewards 95.20466458843039, std r
eward 531.1267569420514
Daggar iter 5: loss 0.017239918932318687 mean rewards 311.3014863488574, std r
eward 1546.9892239931553
Daggar iter 6: loss 0.021987810730934143 mean rewards 437.5793699493095, std r
eward 936.4549778143362
Daggar iter 7: loss 0.020242545753717422 mean rewards 2038.4381794052777, std
reward 1889.707192735697
Daggar iter 8: loss 0.023939600214362144 mean rewards 2547.5833943597004, std
reward 2048.4130032168655
Daggar iter 9: loss 0.022526990622282028 mean rewards 1037.3313279373137, std
reward 1470.5279230902163
Daggar iter 10: loss 0.026140619069337845 mean rewards 1812.6401826068607, std
reward 3358.5246682293473
Daggar iter 11: loss 0.028379222378134727 mean rewards 2622.6422975975593, std
reward 1944.2236416547312
Daggar iter 12: loss 0.032015856355428696 mean rewards 4189.209462176337, std
reward 1746.4302038921423
Daggar iter 13: loss 0.027410829439759254 mean rewards 5386.750387827813, std
reward 95.43926015877368
Daggar iter 14: loss 0.025935286656022072 mean rewards 4013.2693573943384, std
reward 1933.3553929314494
Daggar iter 15: loss 0.028990497812628746 mean rewards 3550.784950955111, std
reward 2529.5455884407015
Daggar iter 16: loss 0.02728043496608734 mean rewards 4255.172790221108, std r
eward 1712.9695580629061
Daggar iter 17: loss 0.02612905204296112 mean rewards 5411.543767195734, std r
eward 129.8798578577501
Daggar iter 18: loss 0.022490736097097397 mean rewards 5209.552631721438, std
reward 362.89297284162546
Daggar iter 19: loss 0.02338312938809395 mean rewards 5337.619585139717, std r
eward 672.6196185023813
Daggar iter 20: loss 0.026945913210511208 mean rewards 5357.058642045221, std
reward 440.8803225165206
Daggar iter 21: loss 0.025373464450240135 mean rewards 5143.053050646763, std
reward 978.4404881131351
Daggar iter 22: loss 0.025334039703011513 mean rewards 5431.090893267314, std
reward 70.76129308102223
Daggar iter 23: loss 0.024129483848810196 mean rewards 5645.552405077411, std
reward 96.07372163408293
Daggar iter 24: loss 0.025008995085954666 mean rewards 5310.781251953217, std
reward 855.6009079308085
```

```
Daggar iter 0: loss 0.004340504761785269 mean rewards -95.01779416341806, std
reward 1506.5689036952645
Daggar iter 1: loss 0.004941399209201336 mean rewards 17.044205184388183, std
reward 875.4265951080221
Daggar iter 2: loss 0.009586247615516186 mean rewards -428.7569497758507, std
reward 998.0177136725813
Daggar iter 3: loss 0.010499769821763039 mean rewards 1.5515828462853278, std
reward 607.3298643073974
Daggar iter 4: loss 0.010836566798388958 mean rewards 147.31342743962676, std
reward 168.16566742167342
Daggar iter 5: loss 0.013379720039665699 mean rewards 87.23127382165057, std r
eward 609.0712454076042
Daggar iter 6: loss 0.012084489688277245 mean rewards -1.3657375688240223, std
reward 545.2398492675917
Daggar iter 7: loss 0.01607154682278633 mean rewards 510.68686068458146, std r
eward 470.8605305325448
Daggar iter 8: loss 0.017604690045118332 mean rewards 364.0763984638732, std r
eward 257.6209177214234
Daggar iter 9: loss 0.024312064051628113 mean rewards 328.2220222884066, std r
eward 1050.9492994216141
Daggar iter 10: loss 0.021806444972753525 mean rewards 366.8104014429633, std
reward 480.0436866404656
Daggar iter 11: loss 0.02128789573907852 mean rewards 731.5590865105355, std r
eward 912.3235070692405
Daggar iter 12: loss 0.02320072241127491 mean rewards 1308.4738592339945, std
reward 1377.4344555991756
Daggar iter 13: loss 0.026955371722579002 mean rewards 1107.5880265419507, std
reward 1556.2650074130881
Daggar iter 14: loss 0.02449142560362816 mean rewards 2050.405242471431, std r
eward 1517.2167990825014
Daggar iter 15: loss 0.025306645780801773 mean rewards 2946.450967946278, std
reward 1905.4346920549506
Daggar iter 16: loss 0.030059874057769775 mean rewards 2403.1070067897194, std
reward 2134.5075602880515
Daggar iter 17: loss 0.02683030627667904 mean rewards 2928.132362042087, std r
eward 1881.6976236440041
Daggar iter 18: loss 0.028475742787122726 mean rewards 1780.5828403757303, std
reward 1839.8669994977333
Daggar iter 19: loss 0.027696704491972923 mean rewards 2898.278634091287, std
reward 2175.587741386836
Daggar iter 20: loss 0.025874042883515358 mean rewards 3473.5881882894428, std
reward 2259.9994886388795
Daggar iter 21: loss 0.027574345469474792 mean rewards 3868.9394384363027, std
reward 1628.1560255711884
Daggar iter 22: loss 0.025733422487974167 mean rewards 4473.385839306431, std
reward 1658.1925251813702
Daggar iter 23: loss 0.030871331691741943 mean rewards 3987.7888041423003, std
reward 2045.274759186979
Daggar iter 24: loss 0.025713836774230003 mean rewards 4979.781474426993, std
reward 1478.9856487777192
Daggar iter 25: loss 0.02654879167675972 mean rewards 4663.356887623979, std r
eward 1184.1740562093526
Daggar iter 26: loss 0.028440605849027634 mean rewards 4064.432094211062, std
reward 1880.5998283351066
Daggar iter 27: loss 0.02735775150358677 mean rewards 1317.880978789382, std r
eward 1972.1745965527468
Daggar iter 28: loss 0.02752070501446724 mean rewards 1813.1008225892515, std
reward 1429.5568406526636
Daggar iter 29: loss 0.02613154612481594 mean rewards 2810.260335017521, std r
eward 1773.079979785651
```

```
Daggar iter 30: loss 0.024194030091166496 mean rewards 4876.7854856521935, std
reward 1556.1949785919505
Daggar iter 31: loss 0.026533111929893494 mean rewards 4791.971189025606, std
reward 1497.8666841870072
Daggar iter 32: loss 0.027473514899611473 mean rewards 5559.627079846329, std
reward 320.1273983432901
Daggar iter 33: loss 0.028716478496789932 mean rewards 4327.963091421909, std
reward 1626.9120009984565
Daggar iter 34: loss 0.024904843419790268 mean rewards 4892.731565136568, std
reward 1088.0445089301481
Daggar iter 35: loss 0.023951495066285133 mean rewards 4968.4479183768235, std
reward 1224.0158888903593
Daggar iter 36: loss 0.02514715865254402 mean rewards 4758.022184497588, std r
eward 1595.2768544535336
Daggar iter 37: loss 0.02830500900745392 mean rewards 5297.32685586809, std re
ward 1234.2294467503702
Daggar iter 38: loss 0.027063583955168724 mean rewards 4640.112269559731, std
reward 1824.033614281709
Daggar iter 39: loss 0.02554512955248356 mean rewards 5207.468945168352, std r
eward 101.15287627939433
Daggar iter 40: loss 0.024172168225049973 mean rewards 2426.864759829719, std
reward 1704.557977693907
Daggar iter 41: loss 0.0260450541973114 mean rewards 4525.19351352744, std rew
ard 1564.8873852684715
Daggar iter 42: loss 0.023325327783823013 mean rewards 5183.032507848149, std
reward 1282.7063463221316
Daggar iter 43: loss 0.02658199891448021 mean rewards 5237.9440342388325, std
reward 1489.8534857567136
Daggar iter 44: loss 0.02347707934677601 mean rewards 5329.265552064402, std r
eward 594.7615270955549
Daggar iter 45: loss 0.0234313253313303 mean rewards 4905.854366316287, std re
ward 1363.6465263709808
Daggar iter 46: loss 0.02486512064933777 mean rewards 3266.046811789699, std r
eward 2052.7470449818193
Daggar iter 47: loss 0.02476268820464611 mean rewards 5614.500587301883, std r
eward 117.84401256657051
Daggar iter 48: loss 0.025026684626936913 mean rewards 5633.3612298409535, std
reward 81.9534420400211
Daggar iter 49: loss 0.024681244045495987 mean rewards 4498.972280435235, std
reward 1830.4678215364736
Daggar iter 0: loss 0.007571830414235592 mean rewards 786.7635467809015, std r
eward 93.52565254405638
Daggar iter 1: loss 0.004959133453667164 mean rewards -990.5335275642417, std
reward 1948.6378700811497
Daggar iter 2: loss 0.007664863951504230 5 mean rewards -3020.748817647128, std
reward 2543.4641051209164
Daggar iter 3: loss 0.008821602910757065 mean rewards 84.3717680650929, std re
ward 738.928775159933
Daggar iter 4: loss 0.014719346538186073 mean rewards 420.9366541477365, std r
eward 285.08212483148804
Daggar iter 5: loss 0.013690145686268806 mean rewards -33.93571939693783, std
reward 814.4629304802161
Daggar iter 6: loss 0.01148195844143629 mean rewards 43.98578844674798, std re
ward 207.9627833414621
Daggar iter 7: loss 0.013303965330123901 mean rewards 247.39628065486392, std
reward 301.29089085216634
Daggar iter 8: loss 0.012400055304169655 mean rewards -385.14252404320166, std
reward 1760.0877230535016
Daggar iter 9: loss 0.018395066261291504 mean rewards -129.2917785486703, std
reward 485.0567866066696
```

```
Daggar iter 10: loss 0.024225138127803802 mean rewards 98.8238438433621, std r
eward 312.3932831633755
Daggar iter 11: loss 0.02153930813074112 mean rewards 289.8454644534371, std r
eward 514.6116362420338
Daggar iter 12: loss 0.028360839933156967 mean rewards 263.8950592047534, std
reward 403.65630509920777
Daggar iter 13: loss 0.021235842257738113 mean rewards 576.0262081008099, std
reward 946.1100082935085
Daggar iter 14: loss 0.026062022894620895 mean rewards 279.99400411110184, std
reward 560.6217384191239
Daggar iter 15: loss 0.019846606999635696 mean rewards 1403.1541305569503, std
reward 1359.8104762320859
Daggar iter 16: loss 0.0247170589864254 mean rewards 3565.189061800376, std re
ward 1001.4930644456051
Daggar iter 17: loss 0.02435525692999363 mean rewards 2088.67428692677, std re
ward 1758.2481176641247
Daggar iter 18: loss 0.026794567704200745 mean rewards 1815.5621109734916, std
reward 1313.5192952455195
Daggar iter 19: loss 0.02705596387386322 mean rewards 725.6746631233278, std r
eward 732.8170119245682
Daggar iter 20: loss 0.02744011022150516 mean rewards 2656.0481163825957, std
reward 1589.9531647162532
Daggar iter 21: loss 0.02922377921640873 mean rewards 2190.4002043601713, std
reward 2171.238170322145
Daggar iter 22: loss 0.028566380962729454 mean rewards 3376.678323072486, std
reward 1482.7561507414182
Daggar iter 23: loss 0.031128251925110817 mean rewards 1812.6558908638508, std
reward 1641.002191847489
Daggar iter 24: loss 0.027659479528665543 mean rewards 1414.385472109109, std
reward 1557.3264861603752
Daggar iter 25: loss 0.028471508994698524 mean rewards 3848.5995198486753, std
reward 1595.2611129051745
Daggar iter 26: loss 0.02853384055197239 mean rewards 2234.3407781900364, std
reward 1830.0383038345983
Daggar iter 27: loss 0.02867039479315281 mean rewards 3429.5874326295807, std
reward 1413.402672499423
Daggar iter 28: loss 0.02966121956706047 mean rewards 3619.5340692966965, std
reward 1919.6198030337837
Daggar iter 29: loss 0.03202506899833679 mean rewards 2637.2010008685406, std
reward 2284.1303648234175
Daggar iter 30: loss 0.029834335669875145 mean rewards 4991.207555490626, std
reward 1073.9221832252708
Daggar iter 31: loss 0.027953267097473145 mean rewards 1245.9971439292244, std
reward 756.3368166829591
Daggar iter 32: loss 0.030212009325623512 mean rewards 4667.651735423354, std
reward 85.8147259703015
Daggar iter 33: loss 0.02669808827340603 mean rewards 2628.7390959442732, std
reward 1827.6085804250174
Daggar iter 34: loss 0.029385024681687355 mean rewards 4451.641055880008, std
reward 1622.7724828821918
Daggar iter 35: loss 0.02790871262550354 mean rewards 5484.524065043936, std r
eward 161.08954969255717
Daggar iter 36: loss 0.028192583471536636 mean rewards 4442.46569694299, std r
eward 1155.720090125586
Daggar iter 37: loss 0.0295554306358099 mean rewards 1719.403511494452, std re
ward 1668.86062149702
Daggar iter 38: loss 0.030580079182982445 mean rewards 3705.263890522753, std
reward 2103.1122778337194
Daggar iter 39: loss 0.02786206640303135 mean rewards 2060.2628117135027, std
reward 1893.0911748415433
```

```
Daggar iter 40: loss 0.028497014194726944 mean rewards 2406.8130946156693, std
reward 2108.142256808393
Daggar iter 41: loss 0.03232405334711075 mean rewards 4590.47361133023, std re
ward 1828.5794163607516
Daggar iter 42: loss 0.030042558908462524 mean rewards 4929.0367086276365, std
reward 1642.338651080666
Daggar iter 43: loss 0.028583809733390808 mean rewards 2788.0128558052656, std
reward 2214.7251285910916
Daggar iter 44: loss 0.028356436640024185 mean rewards 4109.4699884876345, std
reward 2071.560784190841
Daggar iter 45: loss 0.030410949140787125 mean rewards 4498.430336741051, std
reward 1343.3859839178303
Daggar iter 46: loss 0.03051217831671238 mean rewards 5273.414966613743, std r
eward 510.58206561641066
Daggar iter 47: loss 0.02942628040909767 mean rewards 4457.801479178496, std r
eward 1731.3930513736068
Daggar iter 48: loss 0.02945486083626747 mean rewards 5350.043728163774, std r
eward 639.9125391700392
Daggar iter 49: loss 0.025725826621055603 mean rewards 4756.468350512934, std
reward 1773.3054650250558
Daggar iter 50: loss 0.025864994153380394 mean rewards 5318.495721846583, std
reward 605.2233643142335
Daggar iter 51: loss 0.028052005916833878 mean rewards 3722.0910531854324, std
reward 2234.1889817431243
Daggar iter 52: loss 0.02941727638244629 mean rewards 4859.505915247469, std r
eward 1606.2481792934486
Daggar iter 53: loss 0.031552888453006744 mean rewards 4975.735947962991, std
reward 1253.415654263733
Daggar iter 54: loss 0.02856374718248844 mean rewards 4642.916406662799, std r
eward 2133.9476946842965
Daggar iter 55: loss 0.026192557066679 mean rewards 5238.313826212984, std rew
ard 873.4122387138044
Daggar iter 56: loss 0.029988115653395653 mean rewards 5170.506471670922, std
reward 1126.1714324311793
Daggar iter 57: loss 0.0251697339117527 mean rewards 5029.041643954206, std re
ward 1282.2325649657737
Daggar iter 58: loss 0.02651488408446312 mean rewards 3298.3328646701457, std
reward 1836.5998197949177
Daggar iter 59: loss 0.025721630081534386 mean rewards 5621.548867796532, std
reward 67.27859509436045
Daggar iter 60: loss 0.026498083025217056 mean rewards 4030.5919067597943, std
reward 2306.4246651621406
Daggar iter 61: loss 0.027113113552331924 mean rewards 5423.062861132429, std
reward 80.43405423906334
Daggar iter 62: loss 0.02786991000175476 mean rewards 3430.7330166992697, std
reward 1976.408619373103
Daggar iter 63: loss 0.02720389887690544 mean rewards 4696.520311317956, std r
eward 1621.3612070176434
Daggar iter 64: loss 0.026211649179458618 mean rewards 5201.295051334241, std
reward 655.4706409148538
Daggar iter 65: loss 0.02496684156358242 mean rewards 5474.400484256787, std r
eward 77.3958376818472
Daggar iter 66: loss 0.026969676837325096 mean rewards 3560.620271219169, std
reward 1538.0789734909106
Daggar iter 67: loss 0.02634492516517639 mean rewards 5223.011320697979, std r
eward 199.15324846324543
Daggar iter 68: loss 0.02954237163066864 mean rewards 4233.1300206241085, std
reward 1798.8067634234508
Daggar iter 69: loss 0.026916777715086937 mean rewards 5144.162208797461, std
reward 1131.5516823368653
```

```
Daggar iter 70: loss 0.028026631101965904 mean rewards 5353.744718585966, std
reward 41.8828325287489
Daggar iter 71: loss 0.026897016912698746 mean rewards 4863.624934387651, std
reward 1112.770381987721
Daggar iter 72: loss 0.026119261980056763 mean rewards 5652.130163506474, std
reward 72.96760637050724
Daggar iter 73: loss 0.022677814587950706 mean rewards 5794.581272326158, std
reward 102.65007731465906
Daggar iter 74: loss 0.026208648458123207 mean rewards 4898.442148367938, std
reward 1517.5990858378461
Daggar iter 75: loss 0.026223599910736084 mean rewards 3905.014501076648, std
reward 2360.964621553347
Daggar iter 76: loss 0.027718210592865944 mean rewards 4984.269506045932, std
reward 1520.2675509591463
Daggar iter 77: loss 0.024881167337298393 mean rewards 3828.7423653951955, std
reward 1760.236808023673
Daggar iter 78: loss 0.026548538357019424 mean rewards 5541.970478888315, std
reward 192.49642678954996
Daggar iter 79: loss 0.028668951243162155 mean rewards 5798.255285096206, std
reward 161.93471747579926
Daggar iter 80: loss 0.02373998984694481 mean rewards 5299.775017111148, std r
eward 942.6063772434164
Daggar iter 81: loss 0.02555093728005886 mean rewards 5074.421579085032, std r
eward 85.78703966663596
Daggar iter 82: loss 0.02376752719283104 mean rewards 5675.247312976412, std r
eward 85.71482256209168
Daggar iter 83: loss 0.02395172044634819 mean rewards 5463.863682119192, std r
eward 801.916731044771
Daggar iter 84: loss 0.026534471660852432 mean rewards 4554.198617578021, std
reward 1828.6777830988026
Daggar iter 85: loss 0.022496977820999247 mean rewards 4600.4059377405265, std
reward 1937.0267852291468
Daggar iter 86: loss 0.022443093359470367 mean rewards 5508.758490427683, std
reward 116.63693436671073
Daggar iter 87: loss 0.026451528072357178 mean rewards 5666.99161590781, std r
eward 135.2657338682932
Daggar iter 88: loss 0.02522340789437294 mean rewards 4858.6462338802485, std
reward 1559.0412510229164
Daggar iter 89: loss 0.025265362113714218 mean rewards 5080.971423721061, std
reward 1269.035483210199
Daggar iter 90: loss 0.02639835700392723 mean rewards 3750.3711973403488, std
reward 1705.3464735935145
Daggar iter 91: loss 0.026426345109939575 mean rewards 5400.43284656165, std r
eward 772.992483740186
Daggar iter 92: loss 0.022095490247011185 mean rewards 4512.560857457051, std
reward 1993.864891845253
Daggar iter 93: loss 0.02579565905034542 mean rewards 4626.9588085702835, std
reward 1997.2250032916243
Daggar iter 94: loss 0.026531722396612167 mean rewards 5256.603718213024, std
reward 1399.1092392805647
Daggar iter 95: loss 0.026703689247369766 mean rewards 5202.087549699463, std
reward 1768.340474207093
Daggar iter 96: loss 0.024356482550501823 mean rewards 5536.795087452762, std
reward 91.33276778165184
Daggar iter 97: loss 0.024499494582414627 mean rewards 5745.2888311874885, std
reward 100.21108593857637
Daggar iter 98: loss 0.024753486737608891 mean rewards 4817.594221865844, std r
eward 1724.3478707524757
Daggar iter 99: loss 0.02280179224908352 mean rewards 2805.1108169612835, std
reward 2279.1637468959602
```

```
Daggar iter 0: loss 0.013875273987650871 mean rewards -256.31599999784095, std
reward 110.23512267002155
Daggar iter 1: loss 0.007361716590821743 mean rewards 742.7332121239835, std r
eward 227.9536688514709
Daggar iter 2: loss 0.007055684924125671 mean rewards 84.45837771764546, std r
eward 560.8405994330208
Daggar iter 3: loss 0.009362992830574512 mean rewards -245.41566738814564, std
reward 1643.1922305538792
Daggar iter 4: loss 0.008072992786765099 mean rewards 143.20594266941964, std
reward 670.9631881614001
Daggar iter 5: loss 0.012403760105371475 mean rewards -134.7465062377116, std
reward 522.2349587578876
Daggar iter 6: loss 0.01626964472234249 mean rewards -71.93906662693561, std r
eward 459.4415424911271
Daggar iter 7: loss 0.02008369192481041 mean rewards 175.0979863615833, std re
ward 707.6679705535354
Daggar iter 8: loss 0.016592448577284813 mean rewards 213.50119811429823, std
reward 281.2326368970621
Daggar iter 9: loss 0.015013453550636768 mean rewards 310.12541958702286, std
reward 218.7363582827332
Daggar iter 10: loss 0.017711687833070755 mean rewards 568.3654891754522, std
reward 357.7300726841577
Daggar iter 11: loss 0.01978706754744053 mean rewards 158.54599798177915, std
reward 365.9709594764076
Daggar iter 12: loss 0.02325984463095665 mean rewards 474.71181385357323, std
reward 763.1516413767317
Daggar iter 13: loss 0.02071470394730568 mean rewards 387.5843166512063, std r
eward 409.13829828697993
Daggar iter 14: loss 0.02559654973447323 mean rewards 500.74640930969434, std
reward 485.00263307440076
Daggar iter 15: loss 0.023507773876190186 mean rewards 172.32066837851445, std
reward 514.9515420456256
Daggar iter 16: loss 0.022474251687526703 mean rewards 796.305085596348, std r
eward 568.2345449565308
Daggar iter 17: loss 0.027789616957306862 mean rewards 393.41291120570577, std
reward 989.2824124363046
Daggar iter 18: loss 0.025861283764243126 mean rewards 206.009794973933, std r
eward 2230.753919800928
Daggar iter 19: loss 0.027178162708878517 mean rewards 801.2039379213373, std
reward 1463.444072894904
Daggar iter 20: loss 0.02501908876001835 mean rewards 291.4636333235838, std r
eward 999.1555093685042
Daggar iter 21: loss 0.026974955573678017 mean rewards 132.2119878449772, std
reward 505.4877021431896
Daggar iter 22: loss 0.02931458316743374 mean rewards 254.3745372862103, std r
eward 759.2842259489773
Daggar iter 23: loss 0.030888631939888 mean rewards -578.1695152122694, std re
ward 1107.7562634760566
Daggar iter 24: loss 0.028428807854652405 mean rewards 458.724544600982, std r
eward 582.7084290618865
Daggar iter 25: loss 0.035541560500860214 mean rewards 1254.1314486497263, std
reward 1437.78097240275
Daggar iter 26: loss 0.03460564464330673 mean rewards 901.6467215030254, std r
eward 791.9740982089697
Daggar iter 27: loss 0.028885072097182274 mean rewards 953.8578208175747, std
reward 2098.2319244632727
Daggar iter 28: loss 0.03153281658887863 mean rewards 1585.5150627729915, std
reward 1484.7419593786635
Daggar iter 29: loss 0.03143114596605301 mean rewards 266.82675330116814, std
reward 991.8589206344556
```

```
Daggar iter 30: loss 0.03173146769404411 mean rewards 2301.4981349037835, std
reward 2243.4738677965515
Daggar iter 31: loss 0.03293696790933609 mean rewards 630.862244376209, std re
ward 1014.911782971678
Daggar iter 32: loss 0.037803132086992264 mean rewards 4142.319048139813, std
reward 1010.5141565200674
Daggar iter 33: loss 0.03678792342543602 mean rewards 3254.4051211436226, std
reward 2206.072083035917
Daggar iter 34: loss 0.033286530524492264 mean rewards 3567.7736713613294, std
reward 1396.7357107834703
Daggar iter 35: loss 0.033215828239917755 mean rewards 1096.7876466806476, std
reward 1852.0617873469077
Daggar iter 36: loss 0.03422728553414345 mean rewards 1960.536582444875, std r
eward 2118.2713112419183
Daggar iter 37: loss 0.028926650062203407 mean rewards 3802.314597636575, std
reward 1479.229680936906
Daggar iter 38: loss 0.033560119569301605 mean rewards 1521.195764030627, std
reward 1065.5202353784393
Daggar iter 39: loss 0.029507901519536972 mean rewards 1590.3665570958563, std
reward 1426.1427970736622
Daggar iter 40: loss 0.034694086760282516 mean rewards 810.6112976370077, std
reward 792.6013054852324
Daggar iter 41: loss 0.032361485064029694 mean rewards 3466.99939674291, std r
eward 1583.423629676748
Daggar iter 42: loss 0.030852198600769043 mean rewards 2106.2067812894734, std
reward 1976.8940182524705
Daggar iter 43: loss 0.03326883912086487 mean rewards 3805.4696591278216, std
reward 1476.7567102680578
Daggar iter 44: loss 0.03463200479745865 mean rewards 4461.63953359617, std re
ward 1488.1740827209603
Daggar iter 45: loss 0.03355249762535095 mean rewards 2892.7741922179644, std
reward 1761.3552925552754
Daggar iter 46: loss 0.03381911665201187 mean rewards 3710.135939202082, std r
eward 1600.69818550409
Daggar iter 47: loss 0.029614806175231934 mean rewards 3807.3758102713523, std
reward 1720.1853695057835
Daggar iter 48: loss 0.03186749294400215 mean rewards 3773.5847580722, std rew
ard 1662.7843182330848
Daggar iter 49: loss 0.03131304681301117 mean rewards 2980.0944023307698, std
reward 2175.1135903382387
Daggar iter 50: loss 0.032127510756254196 mean rewards 3106.2457388358357, std
reward 1893.7020289889279
Daggar iter 51: loss 0.03341943770647049 mean rewards 4042.468140330967, std r
eward 2098.8479978973505
Daggar iter 52: loss 0.03282805532217026 mean rewards 4317.504722300129, std r
eward 1555.2075662884859
Daggar iter 53: loss 0.030041079968214035 mean rewards 3347.665198403526, std
reward 2285.5965275680433
Daggar iter 54: loss 0.030260220170021057 mean rewards 4872.830313905817, std
reward 608.6875321635947
Daggar iter 55: loss 0.03174682706594467 mean rewards 3279.5254705375382, std
reward 2005.5016895287201
Daggar iter 56: loss 0.03378256782889366 mean rewards 2531.6391399341655, std
reward 1537.6960700873628
Daggar iter 57: loss 0.03184332698583603 mean rewards 4398.703359656782, std r
eward 1452.2754215720531
Daggar iter 58: loss 0.036462921649217606 mean rewards 4170.251143599571, std
reward 1657.301234374608
Daggar iter 59: loss 0.03147781267762184 mean rewards 3423.6244213109376, std
reward 2423.3857957173627
```

```
Daggar iter 60: loss 0.030529562383890152 mean rewards 3309.62374036029, std r
eward 2037.6937618789282
Daggar iter 61: loss 0.03095894679427147 mean rewards 4828.9963866012895, std
reward 1527.4375663000696
Daggar iter 62: loss 0.030348505824804306 mean rewards 4548.611705930905, std
reward 1411.5332771999906
Daggar iter 63: loss 0.03113844431936741 mean rewards 4335.742894264757, std r
eward 2009.1872784675506
Daggar iter 64: loss 0.031140482053160667 mean rewards 3399.438925424095, std
reward 1765.7960091241384
Daggar iter 65: loss 0.03116197884082794 2 mean rewards 3537.764105684489, std
reward 1771.314886225389
Daggar iter 66: loss 0.03283857926726341 mean rewards 4968.357949046148, std r
eward 852.8408365824395
Daggar iter 67: loss 0.032043300569057465 mean rewards 4208.7162777554095, std
reward 2201.674910222721
Daggar iter 68: loss 0.028574427589774132 mean rewards 5426.237142076868, std
reward 150.56539196783132
Daggar iter 69: loss 0.03116079792380333 mean rewards 4677.325047481444, std r
eward 1206.0439587030246
Daggar iter 70: loss 0.02956230379641056 mean rewards 4108.896650304084, std r
eward 1479.718434753418
Daggar iter 71: loss 0.029523298144340515 mean rewards 5281.1136381655415, std
reward 94.6050062318246
Daggar iter 72: loss 0.02983616292476654 mean rewards 4560.80886006318, std re
ward 1973.5818852361444
Daggar iter 73: loss 0.02943498268723488 mean rewards 4376.430425816839, std r
eward 1341.6943535279506
Daggar iter 74: loss 0.030384790152311325 mean rewards 4324.618573769485, std
reward 2175.2699200822026
Daggar iter 75: loss 0.031327709555625916 mean rewards 5109.996305372865, std
reward 1288.7238477530882
Daggar iter 76: loss 0.030117500573396683 mean rewards 4825.01871121159, std r
eward 1521.238489587178
Daggar iter 77: loss 0.027673985809087753 mean rewards 4321.076887464353, std
reward 1686.6223862080078
Daggar iter 78: loss 0.03052043542265892 mean rewards 5095.596817751021, std r
eward 689.403093321017
Daggar iter 79: loss 0.028111431747674942 mean rewards 5438.438732777767, std
reward 181.62031036776668
Daggar iter 80: loss 0.029740775004029274 mean rewards 2668.200387467879, std
reward 2212.3386539320136
Daggar iter 81: loss 0.030266959220170975 mean rewards 5244.828965795074, std
reward 1271.0905183490636
Daggar iter 82: loss 0.0280330590903759 mean rewards 4539.342134223954, std re
ward 1737.5726649467704
Daggar iter 83: loss 0.029668111354112625 mean rewards 4400.078909788898, std
reward 1737.5230703070508
Daggar iter 84: loss 0.028467336669564247 mean rewards 5320.568682113153, std
reward 707.98399982154
Daggar iter 85: loss 0.028601715341210365 mean rewards 4807.597246272973, std
reward 1541.5946309172768
Daggar iter 86: loss 0.03111284412443638 mean rewards 4483.821193721731, std r
eward 1956.514984168075
Daggar iter 87: loss 0.030295412987470627 mean rewards 4887.88080320701, std r
eward 1299.8590109634258
Daggar iter 88: loss 0.029807988554239273 mean rewards 4699.062338702893, std
reward 1684.764975304674
Daggar iter 89: loss 0.028751423582434654 mean rewards 4500.785695253514, std
reward 1481.685817983293
```

```
Daggar iter 90: loss 0.029840102419257164 mean rewards 5101.558351485317, std
reward 1098.5571308139936
Daggar iter 91: loss 0.028064584359526634 mean rewards 5460.910216929637, std
reward 370.30919840233435
Daggar iter 92: loss 0.026835694909095764 mean rewards 3058.014922854777, std
reward 2055.5186864538523
Daggar iter 93: loss 0.028207814320921898 mean rewards 4941.7881347680395, std
reward 1648.9778049183833
Daggar iter 94: loss 0.02874942123889923 mean rewards 3594.89229567234, std re
ward 2114.9907783821845
Daggar iter 95: loss 0.02802031673491001 mean rewards 5305.472430375512, std r
eward 143.69226682236538
Daggar iter 96: loss 0.03082745149731636 mean rewards 5583.978038383835, std r
eward 169.99283244345995
Daggar iter 97: loss 0.027373695746064186 mean rewards 3771.5190805487546, std
reward 2326.3084782756755
Daggar iter 98: loss 0.032633163034915924 mean rewards 5318.561187870969, std
reward 1018.0946628255075
Daggar iter 99: loss 0.02825121209025383 mean rewards 5829.8015168194515, std
reward 203.04203522660558
Daggar iter 100: loss 0.031106572598218918 mean rewards 5324.308417602388, std
reward 424.8631270348861
Daggar iter 101: loss 0.02748667635023594 mean rewards 5136.691628796752, std
reward 1725.8668057735713
Daggar iter 102: loss 0.026433199644088745 mean rewards 5171.720367635071, std
reward 691.8217230332365
Daggar iter 103: loss 0.0314817801117897 mean rewards 5516.216385913941, std r
eward 122.01738302309907
Daggar iter 104: loss 0.02968817949295044 mean rewards 5522.450851131327, std
reward 140.8417998115263
Daggar iter 105: loss 0.03153606131672859 mean rewards 4061.2324591350093, std
reward 2604.9719318289863
Daggar iter 106: loss 0.025760764256119728 mean rewards 5338.074355743544, std
reward 554.096608870378
Daggar iter 107: loss 0.026087414473295212 mean rewards 4427.164342325149, std
reward 1604.8698068853175
Daggar iter 108: loss 0.0270947627723217 mean rewards 4561.512840947262, std r
eward 1975.8590942181497
Daggar iter 109: loss 0.029517028480768204 mean rewards 5379.484110635003, std
reward 160.58501201115382
Daggar iter 110: loss 0.031303633004426956 mean rewards 5063.6886369303365, st
d reward 1258.9008129812066
Daggar iter 111: loss 0.02802690491080284 mean rewards 5135.784331291394, std
reward 80.64662168822302
Daggar iter 112: loss 0.02870999276638031 mean rewards 5739.594925993034, std
reward 78.87212583964693
Daggar iter 113: loss 0.026308288797736168 mean rewards 5383.639036550904, std
reward 821.6931637499002
Daggar iter 114: loss 0.026678362861275673 mean rewards 5609.877546125843, std
reward 95.33682620388734
Daggar iter 115: loss 0.024519044905900955 mean rewards 5305.841192691345, std
reward 862.2591402722954
Daggar iter 116: loss 0.029411816969513893 mean rewards 5497.706998383685, std
reward 473.5079049942677
Daggar iter 117: loss 0.026898542419075966 mean rewards 5551.7752810445645, st
d reward 142.15501997478336
Daggar iter 118: loss 0.0285748690366745 mean rewards 3196.690482183661, std r
eward 1771.9988469443067
Daggar iter 119: loss 0.030723923817276955 mean rewards 5393.50884474773, std
reward 683.5722637987885
```

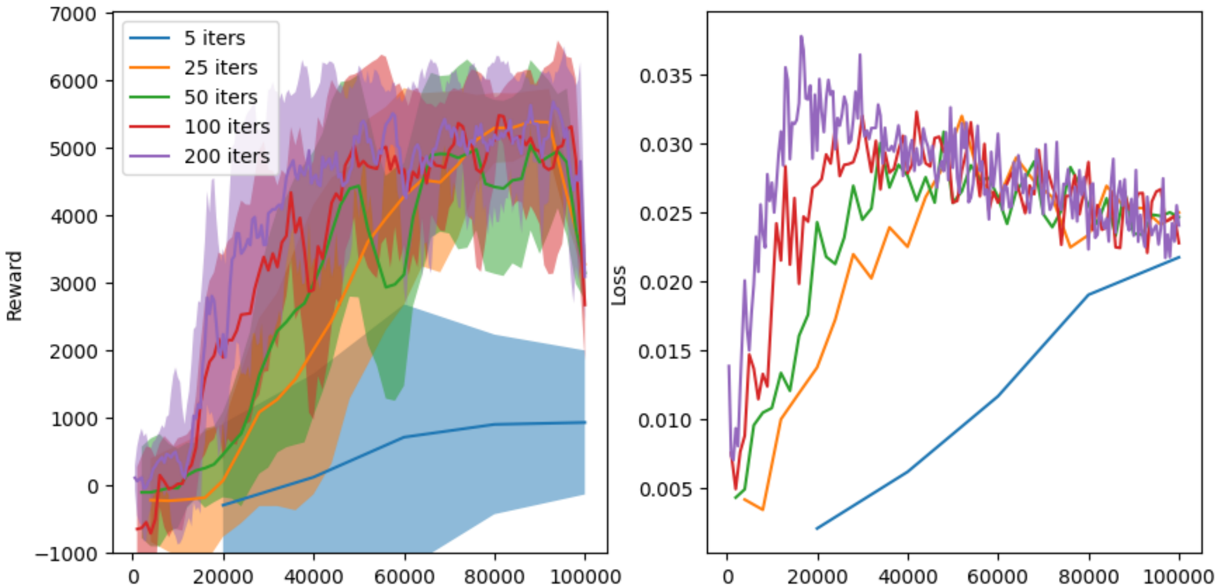
```
Daggar iter 120: loss 0.02780064381659031 mean rewards 3639.8898045145056, std
reward 2003.2217896213408
Daggar iter 121: loss 0.025218581780791283 mean rewards 3785.620852577946, std
reward 2446.8195361522558
Daggar iter 122: loss 0.026978086680173874 mean rewards 5248.716888816163, std
reward 1069.2435395082732
Daggar iter 123: loss 0.02681097947061062 mean rewards 5123.548281607688, std
reward 120.12154610350659
Daggar iter 124: loss 0.02537912130355835 mean rewards 5656.010668222112, std
reward 122.10932185114018
Daggar iter 125: loss 0.025453973561525345 mean rewards 4573.655908515544, std
reward 1797.060400975581
Daggar iter 126: loss 0.028666924685239792 mean rewards 4389.852125465682, std
reward 1502.2744309216318
Daggar iter 127: loss 0.02664482221007347 mean rewards 5478.938458397743, std
reward 543.243109257189
Daggar iter 128: loss 0.029164433479309082 mean rewards 4641.836696115233, std
reward 1666.4208985518426
Daggar iter 129: loss 0.028178876265883446 mean rewards 4010.1165000488954, st
d reward 2288.8900025850844
Daggar iter 130: loss 0.02641768380999565 mean rewards 5449.246961673775, std
reward 80.95166576099305
Daggar iter 131: loss 0.027807017788290977 mean rewards 4781.30073452816, std
reward 1772.7977359768458
Daggar iter 132: loss 0.025786280632019043 mean rewards 4890.65860542488, std
reward 1646.2789248263316
Daggar iter 133: loss 0.024997124448418617 mean rewards 4510.053272355204, std
reward 2149.3274005889507
Daggar iter 134: loss 0.026754405349493027 mean rewards 5252.211716360036, std
reward 732.5355237039968
Daggar iter 135: loss 0.026752552017569542 mean rewards 5471.090057797087, std
reward 89.88949345784224
Daggar iter 136: loss 0.029605945572257042 mean rewards 5450.31867892536, std
reward 695.1119468541146
Daggar iter 137: loss 0.028690293431282043 mean rewards 5609.310396716375, std
reward 80.8295038148554
Daggar iter 138: loss 0.026658857241272926 mean rewards 4922.0677761349925, st
d reward 893.3558786801979
Daggar iter 139: loss 0.024891462177038193 mean rewards 5663.777996265739, std
reward 167.12857728206282
Daggar iter 140: loss 0.029957711696624756 mean rewards 4737.987488621998, std
reward 1754.9007019493918
Daggar iter 141: loss 0.026446420699357986 mean rewards 5707.83918799141, std
reward 145.43815258057649
Daggar iter 142: loss 0.025212442502379417 mean rewards 4764.497216117947, std
reward 1849.6539597869523
Daggar iter 143: loss 0.025587115436792374 mean rewards 5602.669983747468, std
reward 114.92441348872791
Daggar iter 144: loss 0.02652689814567566 mean rewards 5461.158832014956, std
reward 672.5678311805232
Daggar iter 145: loss 0.030124599114060402 mean rewards 3724.872028314932, std
reward 1887.5788943011464
Daggar iter 146: loss 0.025696909055113792 mean rewards 5570.943995479707, std
reward 100.87221271389815
Daggar iter 147: loss 0.02520790323615074 mean rewards 5840.188479175269, std
reward 54.81218701573352
Daggar iter 148: loss 0.027091259136795998 mean rewards 4355.009571094359, std
reward 2125.360992357938
Daggar iter 149: loss 0.027034346014261246 mean rewards 5006.065726717682, std
reward 1473.2192999743636
```

```
Daggar iter 150: loss 0.02816125378012657 mean rewards 4651.3037849146685, std
reward 1849.8490370146233
Daggar iter 151: loss 0.026113038882613182 mean rewards 5683.996917136438, std
reward 67.14825675376837
Daggar iter 152: loss 0.024593958631157875 mean rewards 5526.3679642801035, st
d reward 521.7417971670882
Daggar iter 153: loss 0.026863807812333107 mean rewards 5714.216046035445, std
reward 107.5902378841128
Daggar iter 154: loss 0.024369746446609497 mean rewards 4342.797097510672, std
reward 1817.53979088251
Daggar iter 155: loss 0.02807800844311714 mean rewards 5473.339132632416, std
reward 619.1881451229033
Daggar iter 156: loss 0.026283452287316322 mean rewards 4317.890944600893, std
reward 1968.9255063565658
Daggar iter 157: loss 0.027566218748688698 mean rewards 5536.920228027184, std
reward 97.87003972681377
Daggar iter 158: loss 0.025737490504980087 mean rewards 5066.548609239086, std
reward 1690.6068134360144
Daggar iter 159: loss 0.022343460470438004 mean rewards 5209.466332036725, std
reward 1057.4506258002316
Daggar iter 160: loss 0.026972925290465355 mean rewards 5048.885697105461, std
reward 1291.931246361514
Daggar iter 161: loss 0.02712300419807434 mean rewards 5091.100491557403, std
reward 1190.127139185523
Daggar iter 162: loss 0.023183109238743782 mean rewards 5229.476262345565, std
reward 922.4026493660411
Daggar iter 163: loss 0.023412439972162247 mean rewards 5060.540794445373, std
reward 1672.9506560642374
Daggar iter 164: loss 0.025250717997550964 mean rewards 5661.805214348134, std
reward 117.63263544031966
Daggar iter 165: loss 0.024510199204087257 mean rewards 4728.990970001667, std
reward 1750.9109647193686
Daggar iter 166: loss 0.02384783700108528 mean rewards 5518.20934933494, std r
eward 449.9383087295552
Daggar iter 167: loss 0.02458704076707363 mean rewards 5343.690213781696, std
reward 1117.9323439922368
Daggar iter 168: loss 0.022916628047823906 mean rewards 5777.417364221187, std
reward 103.76299494957426
Daggar iter 169: loss 0.027165371924638748 mean rewards 3926.5537969030324, st
d reward 1713.5994868386022
Daggar iter 170: loss 0.028900880366563797 mean rewards 5108.527834418035, std
reward 1125.9633670871417
Daggar iter 171: loss 0.023566821590065956 mean rewards 5654.321384358115, std
reward 56.550229556692926
Daggar iter 172: loss 0.026308918371796608 mean rewards 4528.8155549066105, st
d reward 1913.9325770114854
Daggar iter 173: loss 0.027280639857053757 mean rewards 5332.744033086158, std
reward 1073.6013504059426
Daggar iter 174: loss 0.025495026260614395 mean rewards 5740.020472011233, std
reward 123.00658256418252
Daggar iter 175: loss 0.027417708188295364 mean rewards 5652.936333407106, std
reward 72.43533491372928
Daggar iter 176: loss 0.024294979870319366 mean rewards 5618.880888664154, std
reward 119.550859620925
Daggar iter 177: loss 0.023712527006864548 mean rewards 5791.4703995720465, st
d reward 102.87852785093563
Daggar iter 178: loss 0.025996629148721695 mean rewards 4025.6053594567916, st
d reward 2467.9433587263484
Daggar iter 179: loss 0.023332204669713974 mean rewards 5663.164835582616, std
reward 85.50426641977015
```

```
Daggar iter 180: loss 0.028088847175240517 mean rewards 5663.924006914179, std
reward 90.90078710719834
Daggar iter 181: loss 0.0228975061327219 mean rewards 4280.67113635348, std re
ward 2118.5044728368225
Daggar iter 182: loss 0.024835603311657906 mean rewards 4612.487310717824, std
reward 1693.5301526998726
Daggar iter 183: loss 0.0232972614467144 mean rewards 5424.530807574715, std r
eward 869.7885537591584
Daggar iter 184: loss 0.022519279271364212 mean rewards 5667.601764174504, std
reward 70.10625309068075
Daggar iter 185: loss 0.026152800768613815 mean rewards 5878.847604155862, std
reward 73.55248180062398
Daggar iter 186: loss 0.024348460137844086 mean rewards 5813.7177762398405, st
d reward 235.0452810758499
Daggar iter 187: loss 0.023373235017061234 mean rewards 5610.387224066764, std
reward 67.57875301731896
Daggar iter 188: loss 0.02383442409336567 mean rewards 4932.070590610267, std
reward 2247.4175563085028
Daggar iter 189: loss 0.024089477956295013 mean rewards 5641.133772218128, std
reward 79.30172769339038
Daggar iter 190: loss 0.022704390808939934 mean rewards 5424.934327342069, std
reward 1073.272442687521
Daggar iter 191: loss 0.023843932896852493 mean rewards 5261.540972042564, std
reward 1302.6101566926304
Daggar iter 192: loss 0.027483830228447914 mean rewards 4477.2920434154075, st
d reward 2078.413018809689
Daggar iter 193: loss 0.021726591512560844 mean rewards 5635.942017781636, std
reward 65.92143784776077
Daggar iter 194: loss 0.02341526374220848 mean rewards 3215.4551819722137, std
reward 2187.0103981236603
Daggar iter 195: loss 0.02178328111767769 mean rewards 4178.511370324561, std
reward 2064.311060839915
Daggar iter 196: loss 0.024240605533123016 mean rewards 4349.401697434112, std
reward 1970.659300457403
Daggar iter 197: loss 0.022915489971637726 mean rewards 5522.99769692575, std
reward 395.0533104737912
Daggar iter 198: loss 0.025531619787216187 mean rewards 5294.157068153942, std
reward 803.1175606770216
Daggar iter 199: loss 0.024081788957118988 mean rewards 4647.629207111504, std
reward 2083.5712121771594
```

In [49]:
```
plot(
    n_iters_data["all_xs"],
    n_iters_data["all_means"],
    n_iters_data["all_stds"],
    n_iters_data["all_losses"],
    [f"{n_iters} iters" for n_iters in [5, 25, 50, 100, 200]],
    min=-1000,
)
```

Out[49]:
```
(<Figure size 1000x500 with 2 Axes>,
 array([<Axes: ylabel='Reward'>, <Axes: ylabel='Loss'>], dtype=object))
```


5 iters
25 iters
50 iters
100 iters
200 iters
Reward
Loss
7000
6000
5000
4000
3000
2000
1000
0
−1000
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0
20000
40000
60000
80000
100000