Zhengpu Zhao CS285 HW1 Report

TLDR:

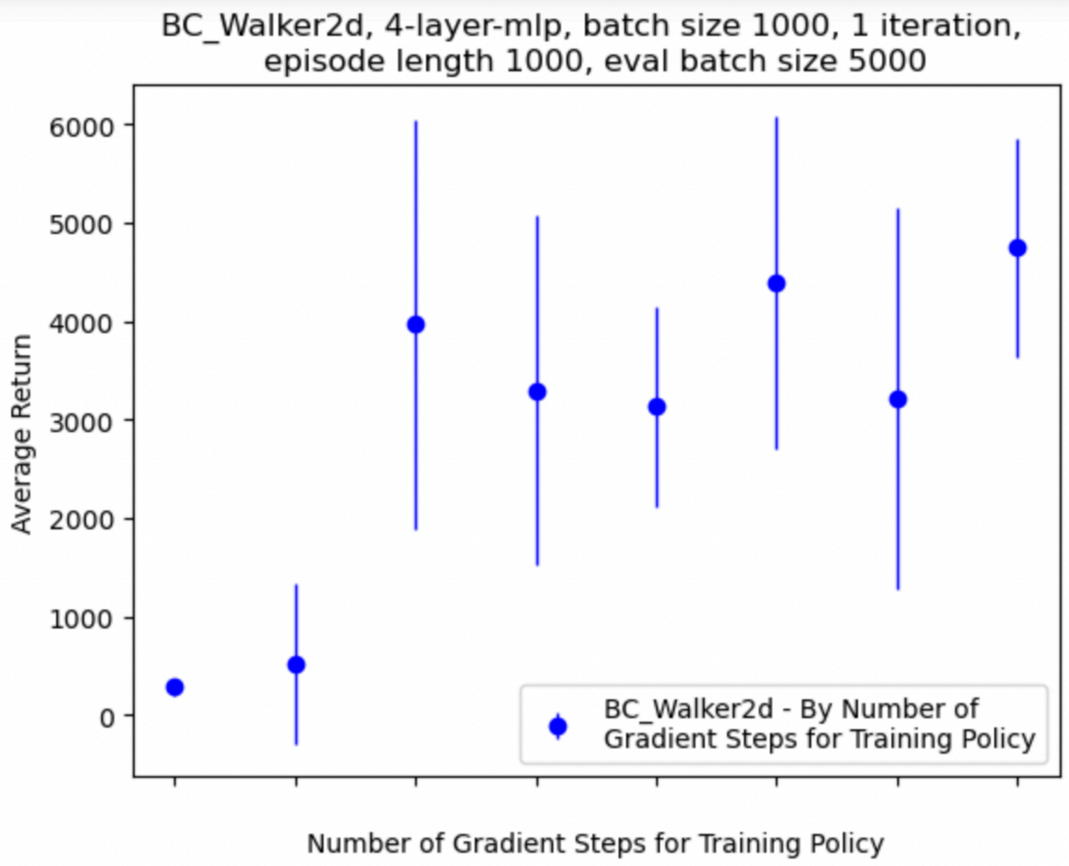
Submitting the PDF. Make a PDF report containing:
Table 1 for a table of results from Question 1.2,
Figure 1 for Question 1.3. and
Figure 2 with results from question 2.2.

Table 1 for a table of results from Question 1.2:

	bc_ant	expert_ant	bc_walker2d	expert_walker2d
Average Return	4669.265625	4713.6533203125	289.3726501464844	5566.845703125
Standard Deviation	81.988525390625	12.196533203125	86.68399810791016	9.237548828125
Min Return	4585.6728515625	4701.45654296875	6.183746337890625	5557.6083984375
Max Return	4798.67333984375	4725.849609375	440.492919921875	5576.08349609375

BC_Ant Achievement as percentage of Expert_Ant: 99%
BC_Walker2d Achievement as percentage of Expert_Walker2d: 5.2%

Figure 1 for Question 1.3:

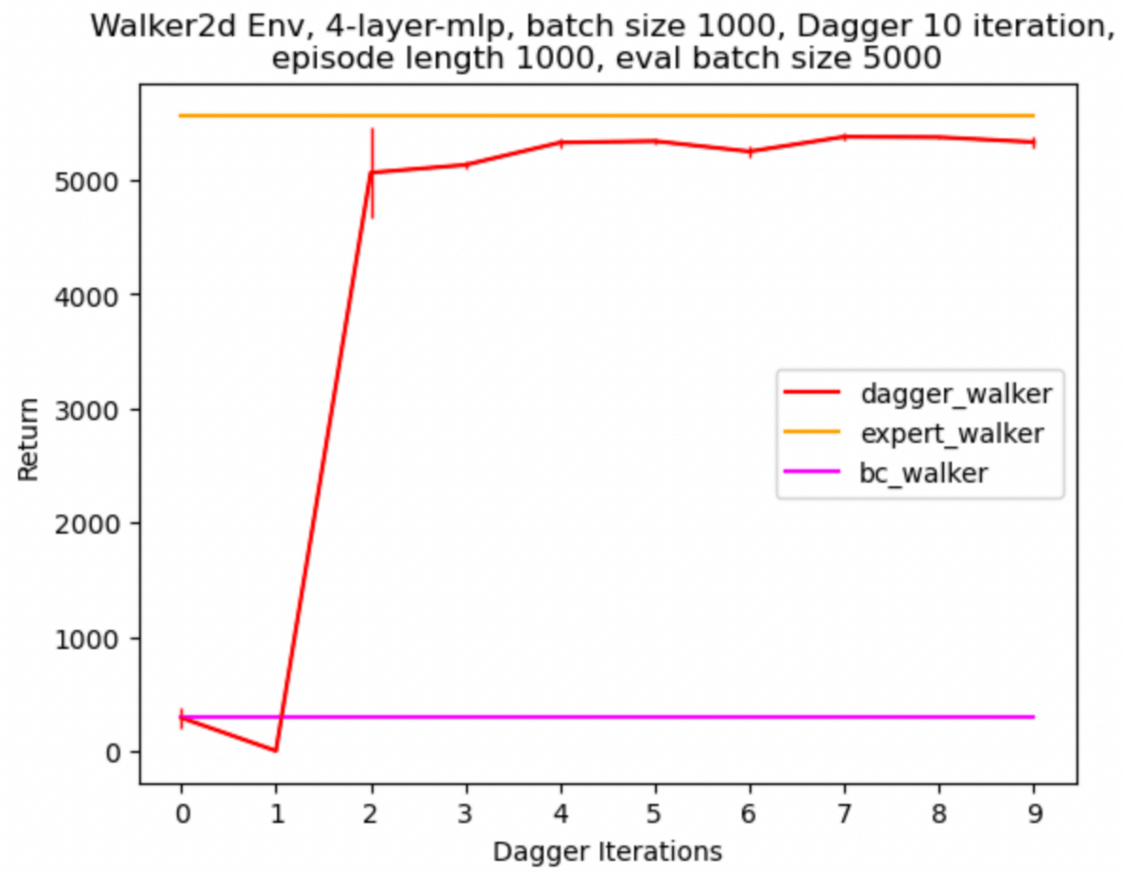


Here I choose - Number of Gradient Steps for Training Policy - to be tuned. The MLP worked fine for the Ant evnvironment leading me to believe that the model itself should be sufficient since the environments aren't too different.

The reason for imitation learning to not work well in the begining could be due to divergence of training track too fast into the training. Thus, by providing more training data and more gradient steps to be performed on the agent we are likely to get closer trajectories to the training trajectory and hence better, longer, and more training down the line.

It seems the hyperparameter tuning is working well as average return is going up and eventually approached close performance to expert level.

Figure 2 with results from question 2.2:



In [1]:

```
import os
import numpy as np
import pandas as pd

from collections import defaultdict
from tensorboard.backend.event_processing.event_accumulator import EventAccumulator
```

In [2]:

```
def tabulate_events(dpath):
    summary_iterators = [EventAccumulator(os.path.join(dpath, dname)).Reload() for dname in os.listd

    tags = summary_iterators[0].Tags()['scalars']

    for it in summary_iterators:
        assert it.Tags()['scalars'] == tags

    out = defaultdict(list)
    steps = []

    for tag in tags:
        steps = [e.step for e in summary_iterators[0].Scalars(tag)]

        for events in zip(*[acc.Scalars(tag) for acc in summary_iterators]):
            assert len(set(e.step for e in events)) == 1

            out[tag].append([e.value for e in events])

    return out, steps


def to_df(dpath):
    dirs = os.listdir(dpath)

    d, steps = tabulate_events(dpath)
    tags, values = zip(*d.items())
    np_values = np.array(values)

    whole_df = pd.DataFrame()

    for index, tag in enumerate(tags):
        values_col = np_values[index]
        sq_val = values_col.squeeze()
        if sq_val.shape == ():
            a_col = sq_val.reshape((1,))
        else:
            a_col = sq_val
        whole_df[tag] = a_col

    return whole_df

def get_file_path(dpath, tag):
    file_name = tag.replace("/", "_") + '.csv'
    folder_path = os.path.join(dpath, 'csv')
    if not os.path.exists(folder_path):
        os.makedirs(folder_path)
    return os.path.join(folder_path, file_name)
```

Use:

path = "Path"

to_csv(path)

Q1.2

python cs285/scripts/run_hw1.py
--expert_policy_file cs285/policies/experts/Ant.pkl
--env_name Ant-v4 --exp_name bc_ant --n_iter 1
--expert_data cs285/expert_data/expert_data_Ant-v4.pkl
--video_log_freq -1 --eval_batch_size 5000
--ep_len 1000

python cs285/scripts/run_hw1.py
--expert_policy_file cs285/policies/experts/Walker2d.pkl
--env_name Walker2d-v4 --exp_name bc_walker2d --n_iter 1
--expert_data cs285/expert_data/expert_data_Walker2d-v4.pkl
--video_log_freq -1 --eval_batch_size 5000
--ep_len 1000

BC_Ant

q1_bc_ant_Ant-v4_12-09-2022_00-35-43

In [3]:
```
bc_ant_exp_path = "q1_bc_ant_Ant-v4_12-09-2022_00-35-43"
bc_ant_exp_df = to_df(bc_ant_exp_path)
# ignore error
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
File /opt/anaconda3/envs/cs285/lib/python3.8/site-packages/tensorboard/compat/__init__.py:42, in tf()
     41 try:
---> 42     from tensorboard.compat import notf  # noqa: F401
     43 except ImportError:

ImportError: cannot import name 'notf' from 'tensorboard.compat' (/opt/anaconda3/envs/cs285/lib/python3
.8/site-packages/tensorboard/compat/__init__.py)

During handling of the above exception, another exception occurred:

RuntimeError                              Traceback (most recent call last)
RuntimeError: module compiled against API version 0xf but this version of numpy is 0xe
```

In [4]:
```
bc_ant_exp_df.head()
```

Out[4]:

	Eval_AverageReturn	Eval_StdReturn	Eval_MaxReturn	Eval_MinReturn	Eval_AverageEpLen	Train_AverageReturn	Train_StdReturn	Train_Ma:
0	4669.265625	81.988525	4798.67334	4585.672852	1000.0	4713.65332	12.196533	4725

In [5]:
```
bc_ant_exp_Eval_AverageReturn = bc_ant_exp_df["Eval_AverageReturn"][0]
bc_ant_exp_Train_AverageReturn = bc_ant_exp_df["Train_AverageReturn"][0]
achieve_percentage = bc_ant_exp_Eval_AverageReturn / bc_ant_exp_Train_AverageReturn
print("{} %".format(achieve_percentage*100))
```

```
99.05831650534797 %
```

BC_Walker2d

q1_bc_walker2d_Walker2d-v4_12-09-2022_00-36-44

In [6]:
```
bc_walker2d_exp_path = "q1_bc_walker2d_Walker2d-v4_12-09-2022_00-36-44"
bc_walker2d_exp_df = to_df(bc_walker2d_exp_path)
bc_walker2d_exp_df.head()
```

Out[6]:

	Eval_AverageReturn	Eval_StdReturn	Eval_MaxReturn	Eval_MinReturn	Eval_AverageEpLen	Train_AverageReturn	Train_StdReturn	Train_Ma:
0	289.37265	86.683998	440.49292	6.183746	126.650002	5566.845703	9.237549	5576

In [7]:
```
bc_walker2d_exp_Eval_AverageReturn = bc_walker2d_exp_df["Eval_AverageReturn"][0]
bc_walker2d_exp_Train_AverageReturn = bc_walker2d_exp_df["Train_AverageReturn"][0]
achieve_percentage = bc_walker2d_exp_Eval_AverageReturn / bc_walker2d_exp_Train_AverageReturn
print("{} %".format(achieve_percentage*100))
```

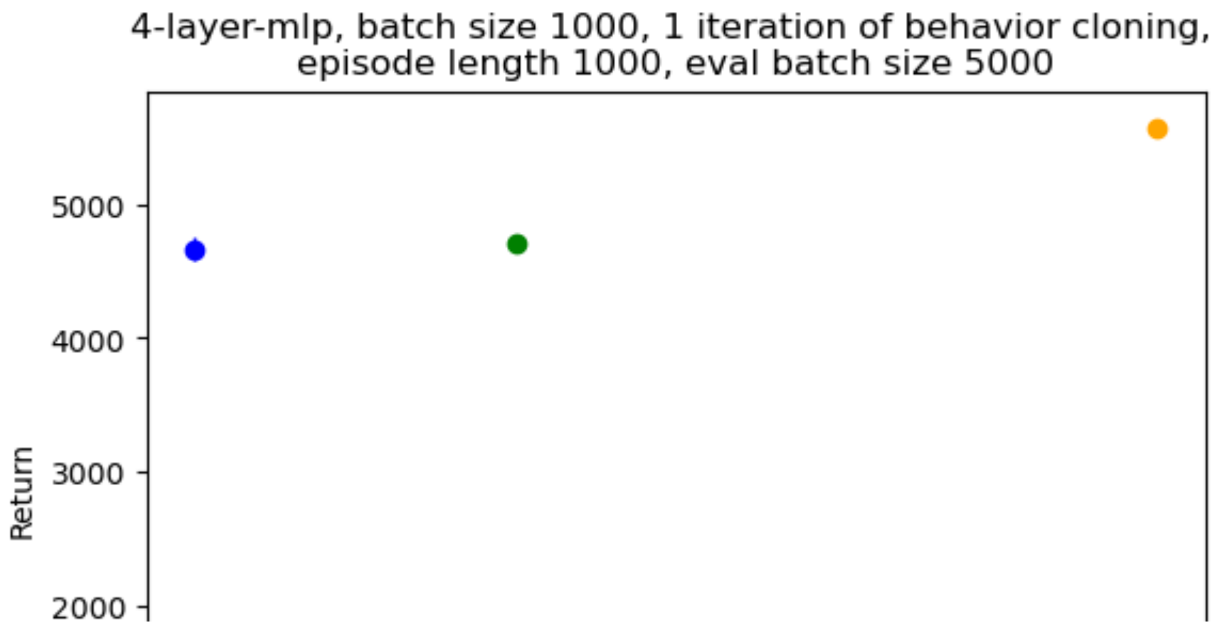
```
5.198143896534484 %
```

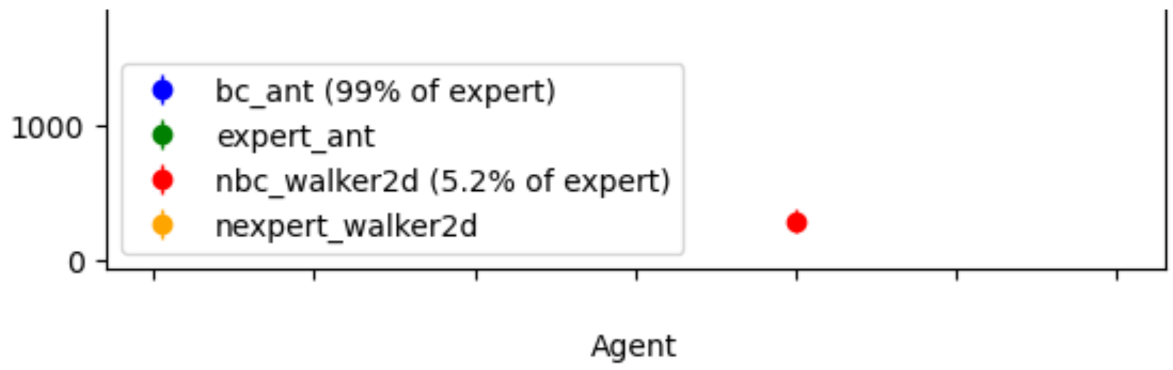
In [8]:

In [8]:

```python
from tabulate import tabulate
import matplotlib.pyplot as plt


means = ['Average Return',
         bc_ant_exp_Eval_AverageReturn,
         bc_ant_exp_Train_AverageReturn,
         bc_walker2d_exp_Eval_AverageReturn,
         bc_walker2d_exp_Train_AverageReturn]

std = ['Standard Deviation',
         bc_ant_exp_df["Eval_StdReturn"][0],
         bc_ant_exp_df["Train_StdReturn"][0],
         bc_walker2d_exp_df["Eval_StdReturn"][0],
         bc_walker2d_exp_df["Train_StdReturn"][0]]

mins = ['Min Return',
         bc_ant_exp_df["Eval_MinReturn"][0],
         bc_ant_exp_df["Train_MinReturn"][0],
         bc_walker2d_exp_df["Eval_MinReturn"][0],
         bc_walker2d_exp_df["Train_MinReturn"][0]]

maxes = ['Max Return',
         bc_ant_exp_df["Eval_MaxReturn"][0],
         bc_ant_exp_df["Train_MaxReturn"][0],
         bc_walker2d_exp_df["Eval_MaxReturn"][0],
         bc_walker2d_exp_df["Train_MaxReturn"][0]]


#Create stacked errorbars:
plt.errorbar(np.array(1), np.array(means[1]), np.array(std[1]), color='blue', fmt='ok', lw=1)
plt.errorbar(np.array(2), np.array(means[2]), np.array(std[2]), color='green', fmt='ok', lw=1)
plt.errorbar(np.array(3), np.array(means[3]), np.array(std[3]), color='red', fmt='ok', lw=1)
plt.errorbar(np.array(4), np.array(means[4]), np.array(std[4]), color='orange', fmt='ok', lw=1)
plt.legend(['bc_ant (99% of expert)','expert_ant',
         'nbc_walker2d (5.2% of expert)','nexpert_walker2d'])
plt.xlabel("Agent")
plt.ylabel("Return")
plt.xticks(color='w')
plt.title("4-layer-mlp, batch size 1000, 1 iteration of behavior cloning, \nepisode length 1000, eval bat
plt.show()

Q1_2_table = [['  ','bc_ant','expert_ant','bc_walker2d','expert_walker2d'],
           means, std, mins, maxes]
print(tabulate(Q1_2_table, tablefmt='fancy_grid'))

print('BC_Ant Achievement as percentage of Expert_Ant: 99%')
print('BC_Walker2d Achievement as percentage of Expert_Walker2d: 5.2%')
```

```
/var/folders/80/mq_tqq2929b94txcm70_rbyr0000gn/T/ipykernel_93228/4274029093.py:31: UserWarning: color is redundantly defined by the 'color' keyword argument and the fmt string "ok" (-> color='k'). The keyword argument will take precedence.
  plt.errorbar(np.array(1), np.array(means[1]), np.array(std[1]), color='blue', fmt='ok', lw=1)
/var/folders/80/mq_tqq2929b94txcm70_rbyr0000gn/T/ipykernel_93228/4274029093.py:32: UserWarning: color is redundantly defined by the 'color' keyword argument and the fmt string "ok" (-> color='k'). The keyword argument will take precedence.
  plt.errorbar(np.array(2), np.array(means[2]), np.array(std[2]), color='green', fmt='ok', lw=1)
/var/folders/80/mq_tqq2929b94txcm70_rbyr0000gn/T/ipykernel_93228/4274029093.py:33: UserWarning: color is redundantly defined by the 'color' keyword argument and the fmt string "ok" (-> color='k'). The keyword argument will take precedence.
  plt.errorbar(np.array(3), np.array(means[3]), np.array(std[3]), color='red', fmt='ok', lw=1)
/var/folders/80/mq_tqq2929b94txcm70_rbyr0000gn/T/ipykernel_93228/4274029093.py:34: UserWarning: color is redundantly defined by the 'color' keyword argument and the fmt string "ok" (-> color='k'). The keyword argument will take precedence.
  plt.errorbar(np.array(4), np.array(means[4]), np.array(std[4]), color='orange', fmt='ok', lw=1)
```





	bc_ant	expert_ant	bc_walker2d	expert_walker2d
Average Return	4669.265625	4713.6533203125	289.3726501464844	5566.845703125
Standard Deviation	81.988525390625	12.196533203125	86.68399810791016	9.237548828125
Min Return	4585.6728515625	4701.45654296875	6.183746337890625	5557.6083984375
Max Return	4798.67333984375	4725.849609375	440.492919921875	5576.08349609375

```
BC_Ant Achievement as percentage of Expert_Ant: 99%
BC_Walker2d Achievement as percentage of Expert_Walker2d: 5.2%
```

Plotting BA_Ant and BC_Walker2d Achievements

Q1.3

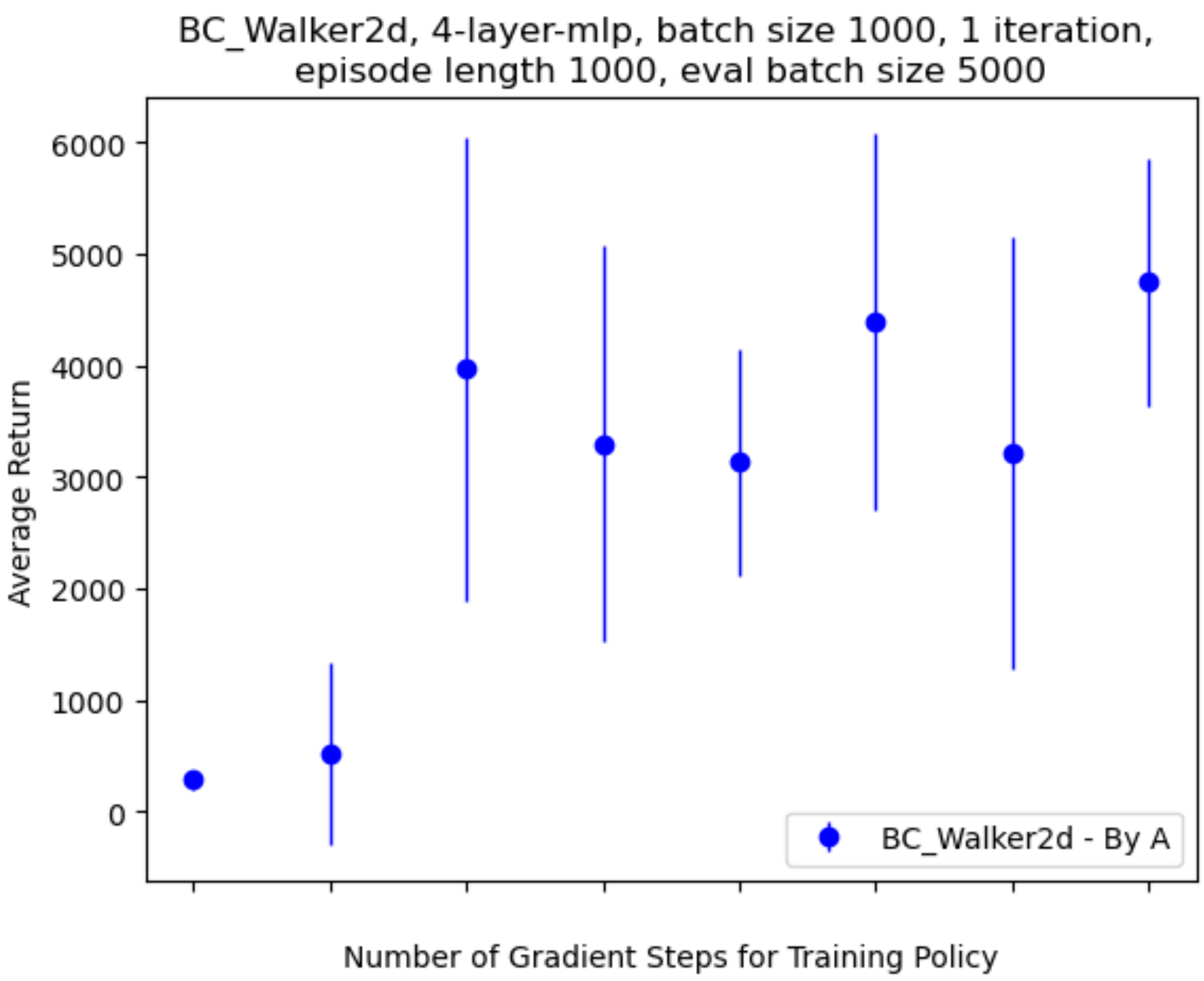
for i in 2000 3000 4000 5000 6000 7000 8000
do
python cs285/scripts/run_hw1.py
--expert_policy_file cs285/policies/experts/Walker2d.pkl
--env_name Walker2d-v4 --exp_name bc_walker2d --n_iter 1
--expert_data cs285/expert_data/expert_data_Walker2d-v4.pkl
--video_log_freq -1 --eval_batch_size 5000
--ep_len 1000 --num_agent_train_steps_per_iter $i
done

q1_bc_walker2d_Walker2d-v4_12-09-2022_00-51-27
q1_bc_walker2d_Walker2d-v4_12-09-2022_00-51-29
q1_bc_walker2d_Walker2d-v4_12-09-2022_00-51-32
q1_bc_walker2d_Walker2d-v4_12-09-2022_00-51-34
q1_bc_walker2d_Walker2d-v4_12-09-2022_00-51-37
q1_bc_walker2d_Walker2d-v4_12-09-2022_00-51-39

In [9]:

```
q_1_3_list_of_df = []
dir_name_prefix = 'q1_bc_walker2d_Walker2d-v4_12-09-2022_00-51-'
for i in [27,29,30,32,34,37,39]:
    dir_name = dir_name_prefix + '{}'.format(i)
    exp_df = to_df(dir_name)
    q_1_3_list_of_df.append(exp_df)

means = [bc_walker2d_exp_Eval_AverageReturn]
stds = [bc_walker2d_exp_df["Eval_StdReturn"][0]]
for df in q_1_3_list_of_df:
    Eval_AverageReturn = df["Eval_AverageReturn"][0]
    Eval_StdReturn = df["Eval_StdReturn"][0]
    means.append(Eval_AverageReturn)
    stds.append(Eval_StdReturn)

plt.errorbar(np.array([1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000 ]),
             np.array(means), np.array(stds), color='blue', fmt='ok', lw=1)

plt.legend(['BC_Walker2d - By A'],loc='lower right')
plt.xlabel("Number of Gradient Steps for Training Policy")
plt.ylabel("Average Return")
plt.xticks(color='w')
plt.title("BC_Walker2d, 4-layer-mlp, batch size 1000, 1 iteration, \nepisode length 1000, eval batch
plt.show()

```

```
/var/folders/80/mq_tqq2929b94txcm70_rbyr0000gn/T/ipykernel_93228/3643757393.py:16: UserWarning: color i
s redundantly defined by the 'color' keyword argument and the fmt string "ok" (-> color='k'). The keywo
rd argument will take precedence.
  plt.errorbar(np.array([1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000 ]),
```



Here I choose - Number of Gradient Steps for Training Policy - to be tuned. The MLP worked fine for the Ant evnvironment leading me to believe that the model itself should be sufficient since the environments aren't too different.

The reason for imitation learning to not work well in the begining could be due to divergence of training track too fast into the training. Thus, by providing more training data and more gradient steps to be performed on the agent we are likely to get closer trajectories to the training trajectory and hence better, longer, and more training down the line.

It seems the hyperparameter tuning is working well as average return is going up and eventually approached close performance to expert level.

Q2.2

q2_dagger_ant_Ant-v4_12-09-2022_01-29-46

q2_dagger_walker2d_Walker2d-v4_12-09-2022_01-31-24

python cs285/scripts/run_hw1.py
--expert_policy_file cs285/policies/experts/Ant.pkl
--env_name Ant-v4 --exp_name dagger_ant --n_iter 10
--do_dagger --expert_data cs285/expert_data/expert_data_Ant-v4.pkl
--video_log_freq -1 --eval_batch_size 5000 --ep_len 1000

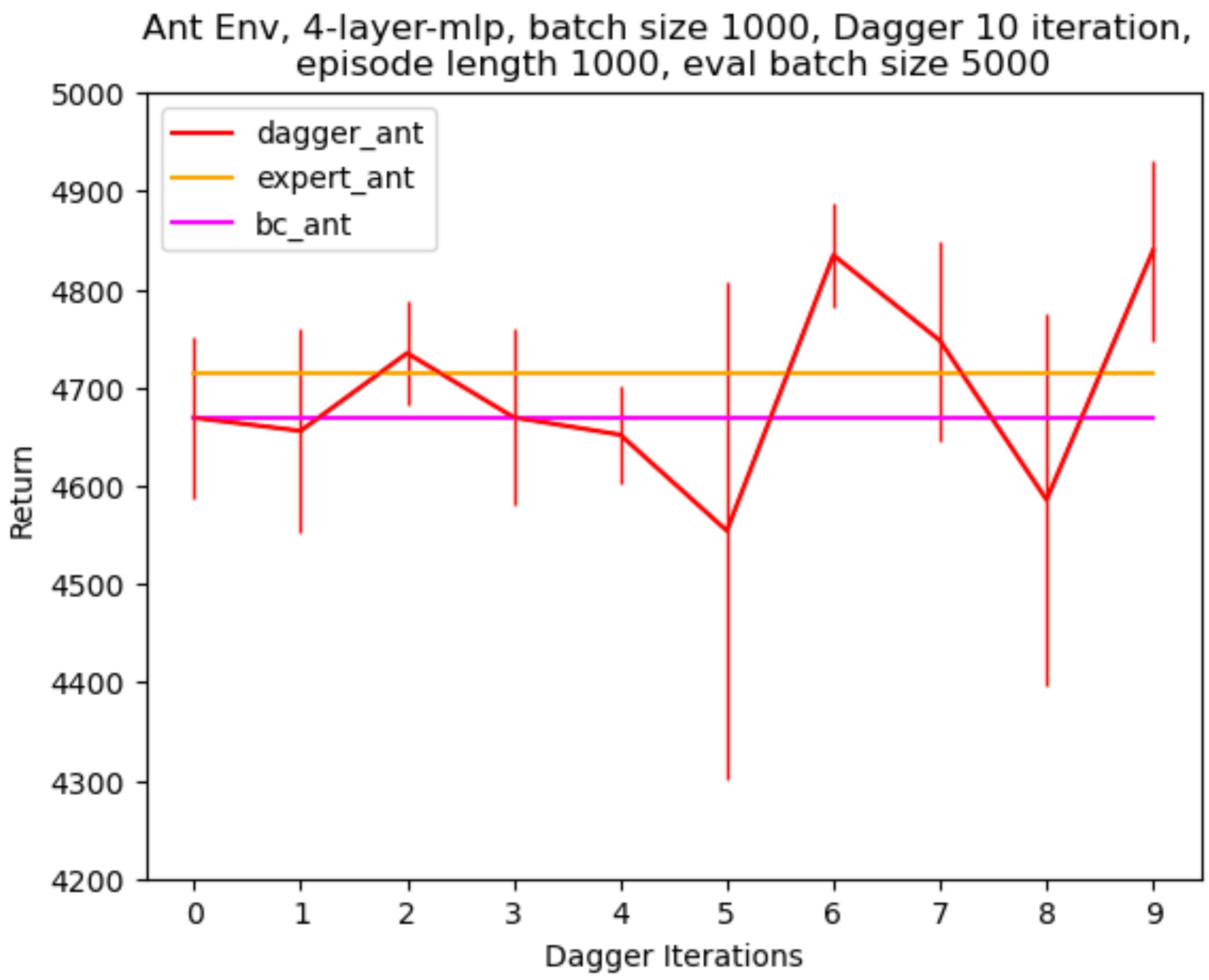
python cs285/scripts/run_hw1.py
--expert_policy_file cs285/policies/experts/Walker2d.pkl
--env_name Walker2d-v4 --exp_name dagger_walker2d --n_iter 10
--do_dagger --expert_data cs285/expert_data/expert_data_Walker2d-v4.pkl
--video_log_freq -1 --eval_batch_size 5000 --ep_len 1000

In [10]:

```
dagger_ant_exp_path = "q2_dagger_ant_Ant-v4_12-09-2022_01-29-46"
dagger_ant_exp_df = to_df(dagger_ant_exp_path)

dagger_walker2d_exp_path = "q2_dagger_walker2d_Walker2d-v4_12-09-2022_01-31-24"
dagger_walker2d_exp_df = to_df(dagger_walker2d_exp_path)
```

In [11]:

```
dagger_ant_Eval_AverageReturn = dagger_ant_exp_df["Eval_AverageReturn"]
dagger_ant_Eval_StdReturn = dagger_ant_exp_df["Eval_StdReturn"]
dagger_ant_eval = dagger_ant_Eval_AverageReturn
expert_ant = np.ones(len(dagger_ant_Eval_AverageReturn)) * dagger_ant_exp_df["Train_AverageReturn"][
bc_ant = np.ones(len(dagger_ant_Eval_AverageReturn)) * bc_ant_exp_Eval_AverageReturn
plt.errorbar(np.array([0,1,2,3,4,5,6,7,8,9]), np.array(dagger_ant_Eval_AverageReturn),
             np.array(dagger_ant_Eval_StdReturn), color='red', lw=1)

plt.plot(dagger_ant_eval, color = 'r')
plt.plot(expert_ant, color = 'orange')
plt.plot(bc_ant, color = 'magenta')
plt.ylim([4200,5000])
plt.xticks([0,1,2,3,4,5,6,7,8,9])
plt.legend(['dagger_ant','expert_ant','bc_ant'])
plt.xlabel("Dagger Iterations")
plt.ylabel("Return")
plt.title("Ant Env, 4-layer-mlp, batch size 1000, Dagger 10 iteration, \nepisode length 1000, eval b
plt.show()
```



In [12]:
```
dagger_walker_Eval_AverageReturn = dagger_walker2d_exp_df["Eval_AverageReturn"]
dagger_walker_Eval_StdReturn = dagger_walker2d_exp_df["Eval_StdReturn"]
dagger_walker_eval = dagger_walker_Eval_AverageReturn
expert_walker = np.ones(len(dagger_walker_Eval_AverageReturn)) * dagger_walker2d_exp_df["Train_Average
bc_walker = np.ones(len(dagger_ant_Eval_AverageReturn)) * bc_walker2d_exp_Eval_AverageReturn
plt.errorbar(np.array([0,1,2,3,4,5,6,7,8,9]), np.array(dagger_walker_Eval_AverageReturn),
             np.array(dagger_walker_Eval_StdReturn), color='red', lw=1)

plt.plot(dagger_walker_eval, color = 'r')
plt.plot(expert_walker, color = 'orange')
plt.plot(bc_walker, color = 'magenta')
#plt.ylim([3000,5000])
plt.xticks([0,1,2,3,4,5,6,7,8,9])
plt.legend(['dagger_walker','expert_walker', 'bc_walker'])
plt.xlabel("Dagger Iterations")
plt.ylabel("Return")
plt.title("Walker2d Env, 4-layer-mlp, batch size 1000, Dagger 10 iteration, \nepisode length 1000, eva
plt.show()
```