CS294 HW1

Section 2.1

Below is the result of running behavioral cloning on the Ant task and the Hopper task. We see that BC performed about as well as the expert for the Ant task. However, for the Hopper task BC performed significantly worse than the Expert.

Table for Reward Comparisons

		Ant	Hopper
Expert	Mean	4781.36	3777.07
	Std	124.85	4.48
Behavioral Cloning	Mean	4777.91	1859.91
	Std	131.31	435.60

All results were tested on 20 rollouts. The BC model is a fully connected neural network with 2 Hidden Layers and 64 units in each layer. The models were trained on 20 rollouts of expert data for 200 epochs with the Adam optimizer.

Commands for Generating Data

Ant Expert Data

```
python run_expert.py experts/Ant-v2.pkl Ant-v2 --num_rollouts=20
```

Ant BC Data

```
python run_expert.py experts/Ant-v2.pkl Ant-v2 --num_rollouts=20 && python
3 train_bc_models.py Ant-v2 --epochs=200 --hidden_layers=2 && python3 run_
bc_policies.py bc_policies/Ant-v2.h5 Ant-v2
```

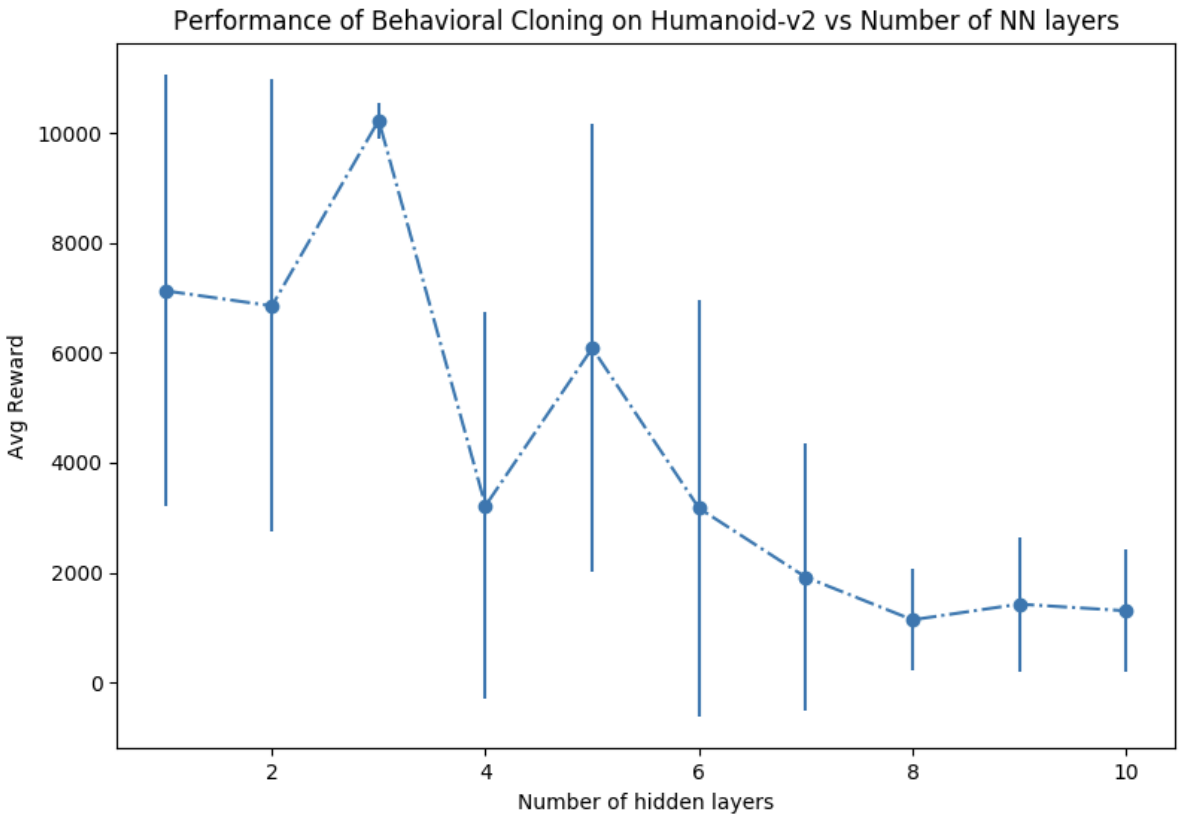
Hopper Expert Data

```
python run_expert.py experts/Hopper-v2.pkl Hopper-v2 --num_rollouts=20
```

Hopper BC Data

```
python3 train_bc_models.py Hopper-v2 --epochs=200 --hidden_layers=2 && pyt
hon3 run_bc_policies.py bc_policies/Hopper-v2.h5 Hopper-v2
```

Section 2.3



Effect of model complexity on Humanoid-v2 task

This experiment shows the effect of number of hidden layers on task performance. All models were trained for 500 epochs on 20 rollouts of expert data. Each hidden layer has 64 units. I choose this parameter because I hypothesized that a model with too many parameters would have poor performance due to overfitting/ insufficient amounts of data. There would be a good middle ground (which seems to be 3 hidden layers in this case) for optimal performance.

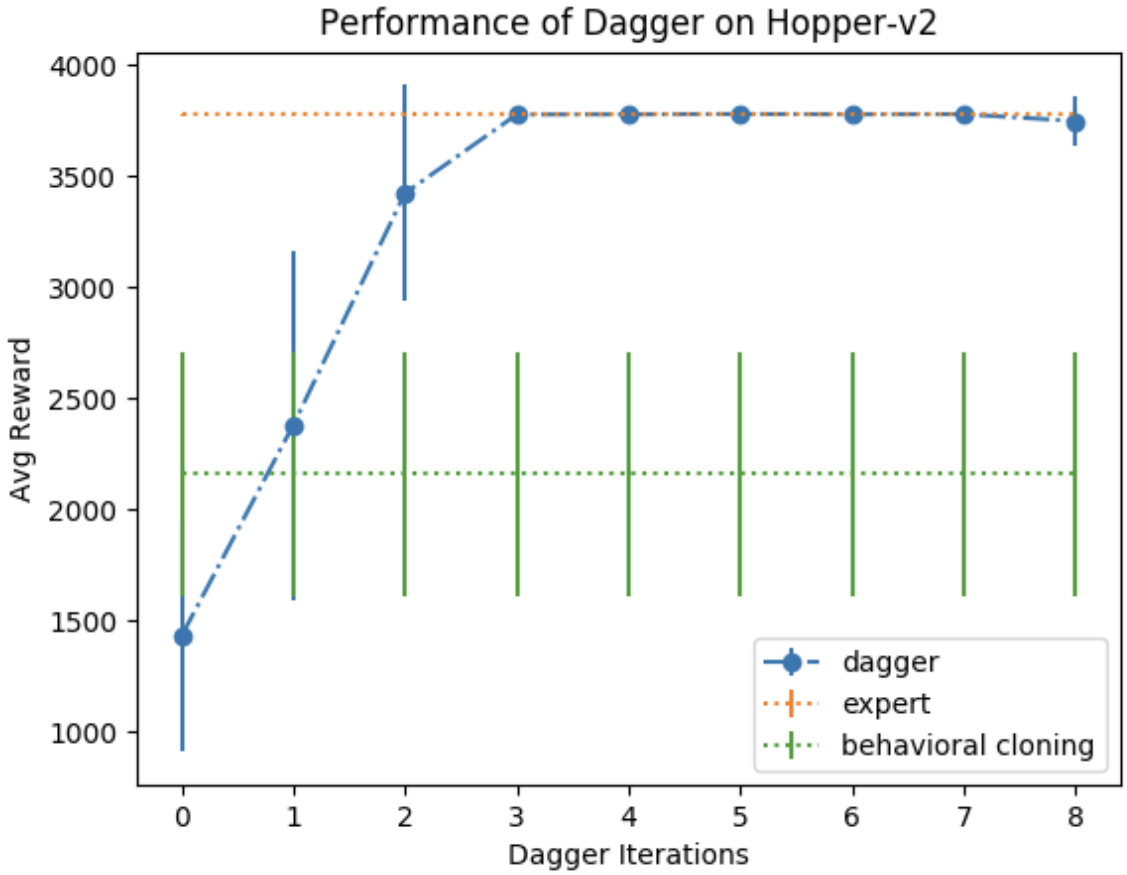
Command to train all models and then create graph.

```
python3 bc_layers_train.py && python3 bc_layers_execute.py Humanoid-v2
```

Can just run the below command to run agents rather retrain all the models

```
python3 bc_layers_execute.py Humanoid-v2
```

Section 3.2



The initial dagger agent was trained on 12 rollouts of expert data. The agent was then run for 1 rollout, the rollout data was labeled by the expert, and then the agent was retrained. This augmenting was done for 8 iterations. Thus the agent was trained on a total of 20 rollouts, 12 initial rollouts and 8 augmented rollouts.

Both the BC and Dagger models are a fully connected neural networks with 2 Hidden Layers and 64 units in each layer. The BC model was trained on 20 rollouts of expert data for 300 epochs with the Adam optimizer.

Command to train model and plot from saved model

```
python3 dagger.py && python3 graph_dagger.py
```