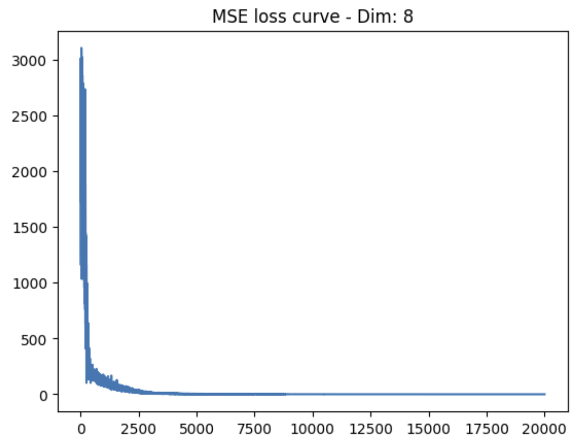
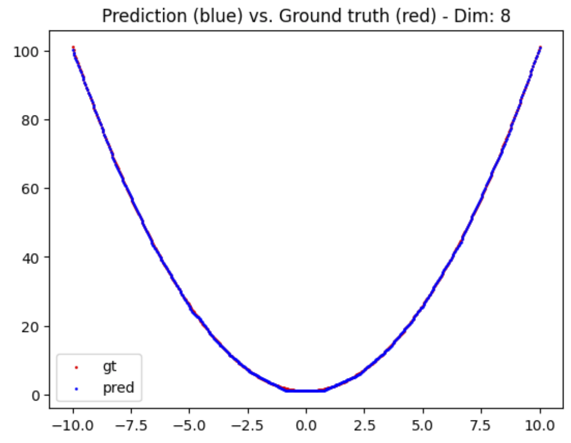
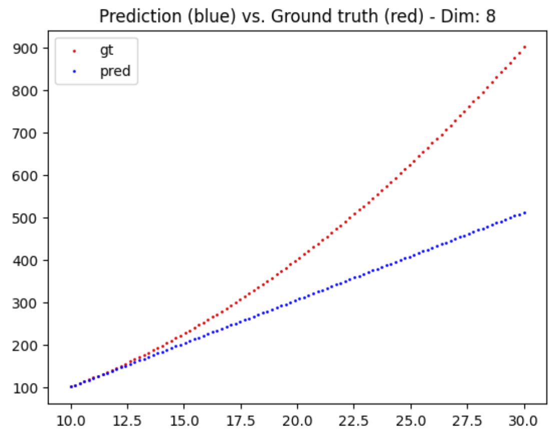
Luke Davidson
CS 5180
Ex8

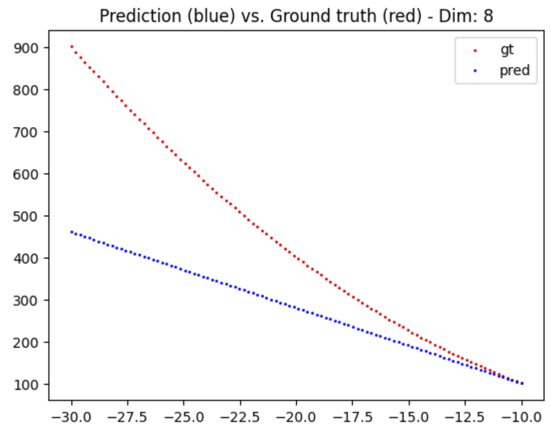
1.) d.) Plots of my results for hidden layer dimensions of 8, 16, 64, and 128 are shown below:

<u>Hidden Layer Dim: 8</u>

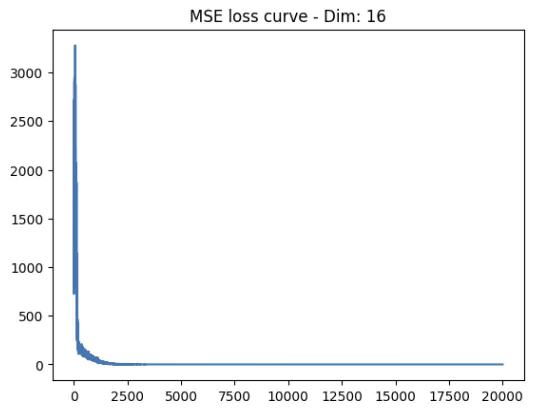


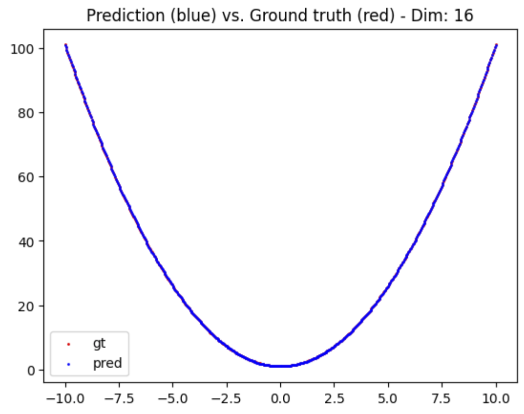


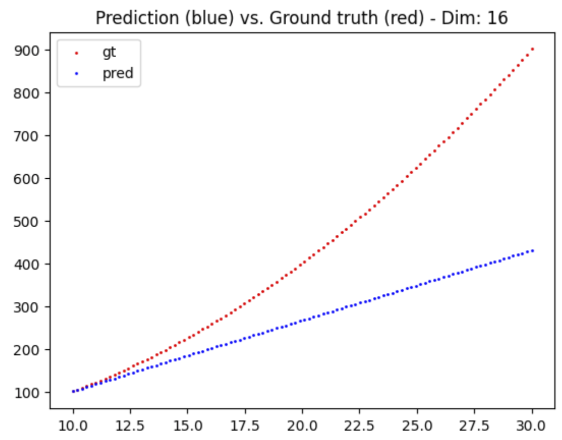


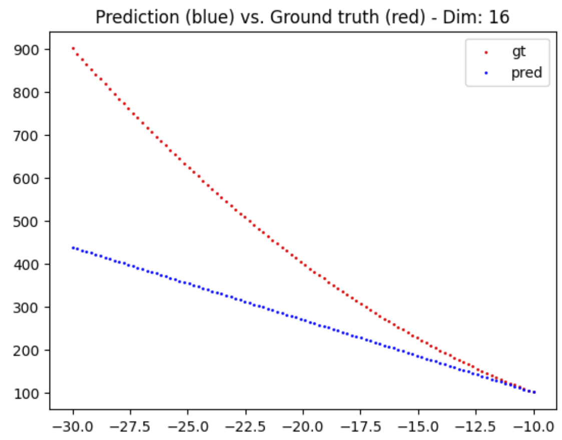


<u>Hidden Layer Dim: 16</u>

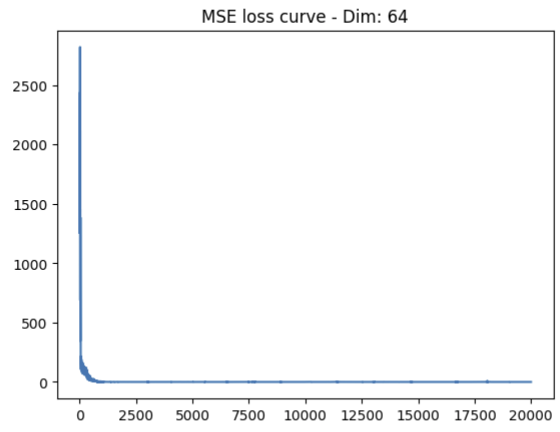


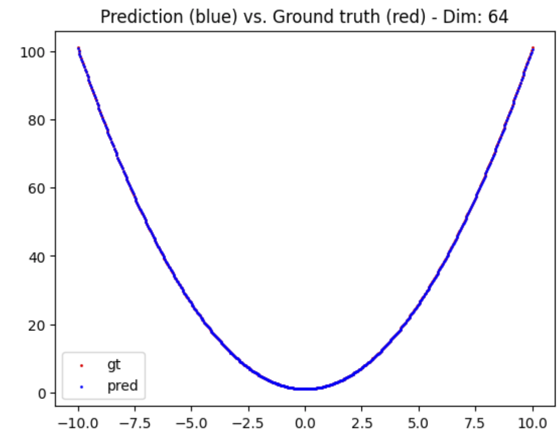


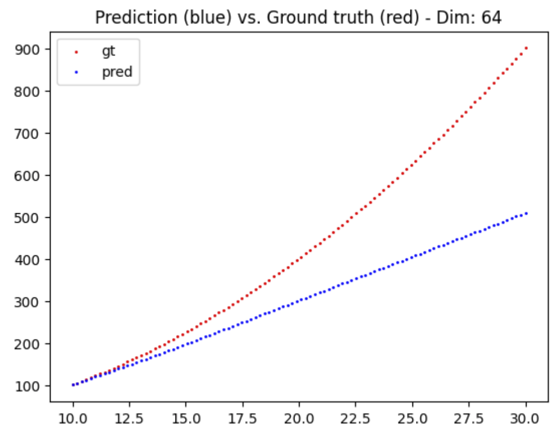


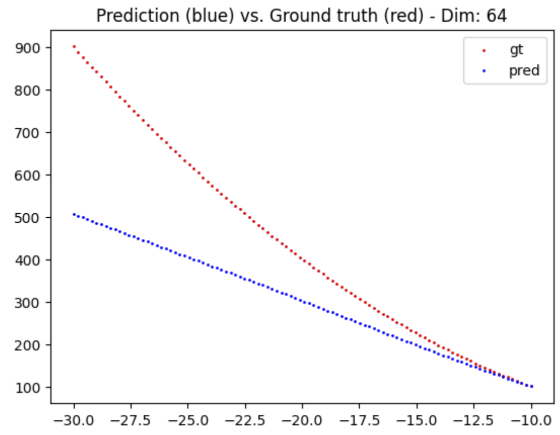


Hidden Layer Dim: 64

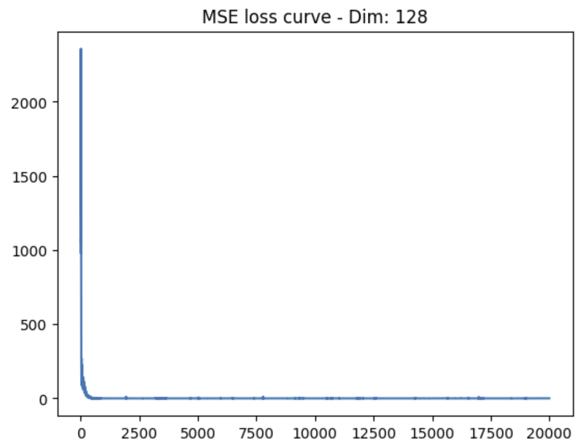


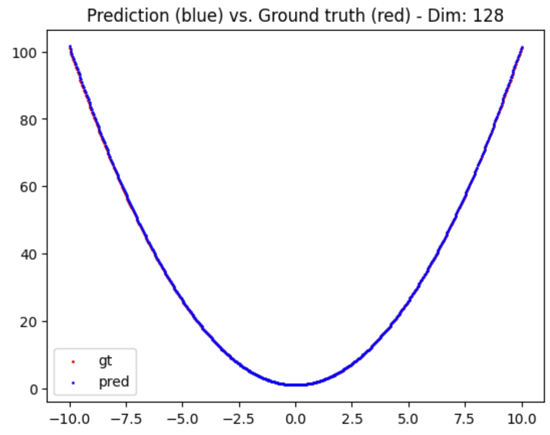


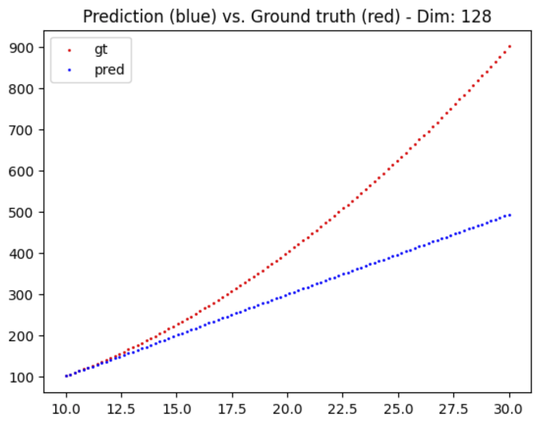


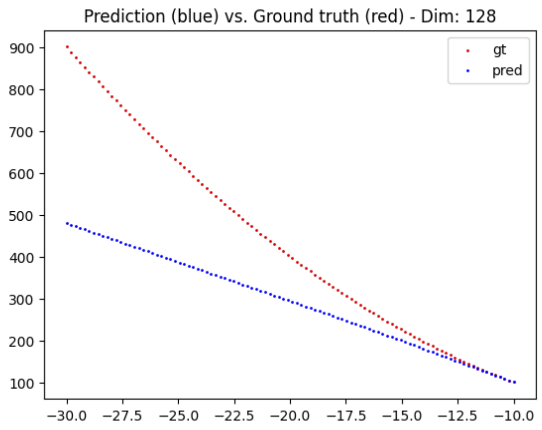


Hidden Layer Dim: 128



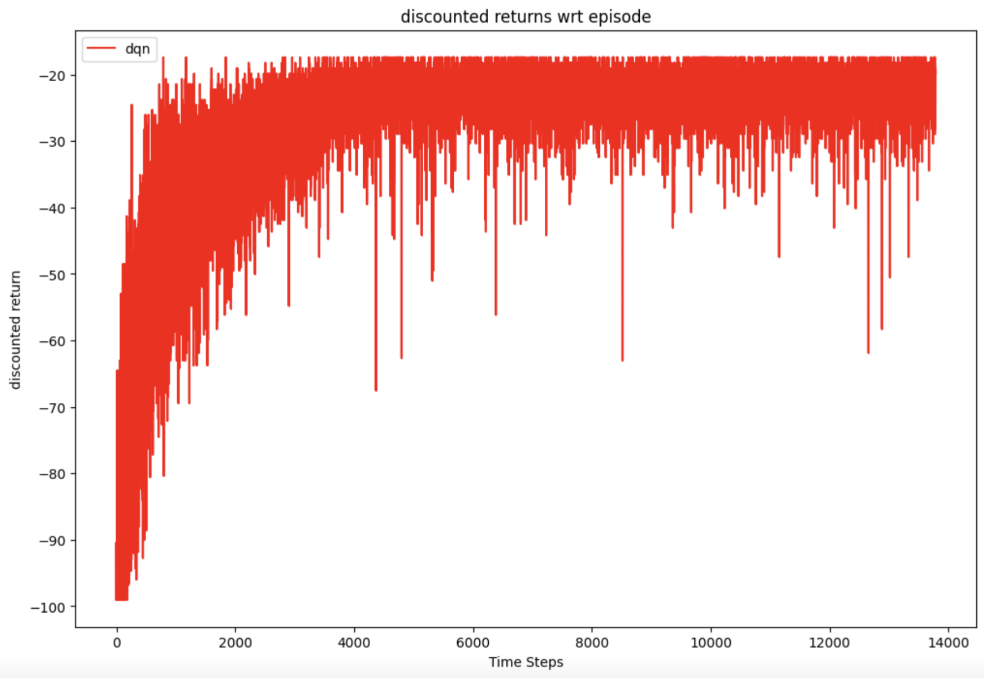


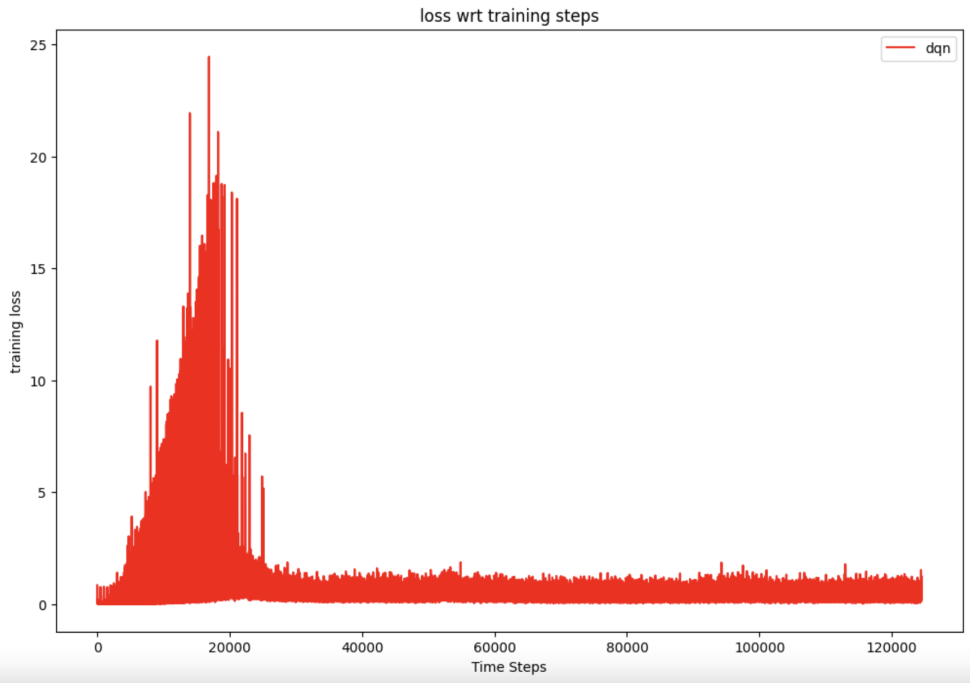




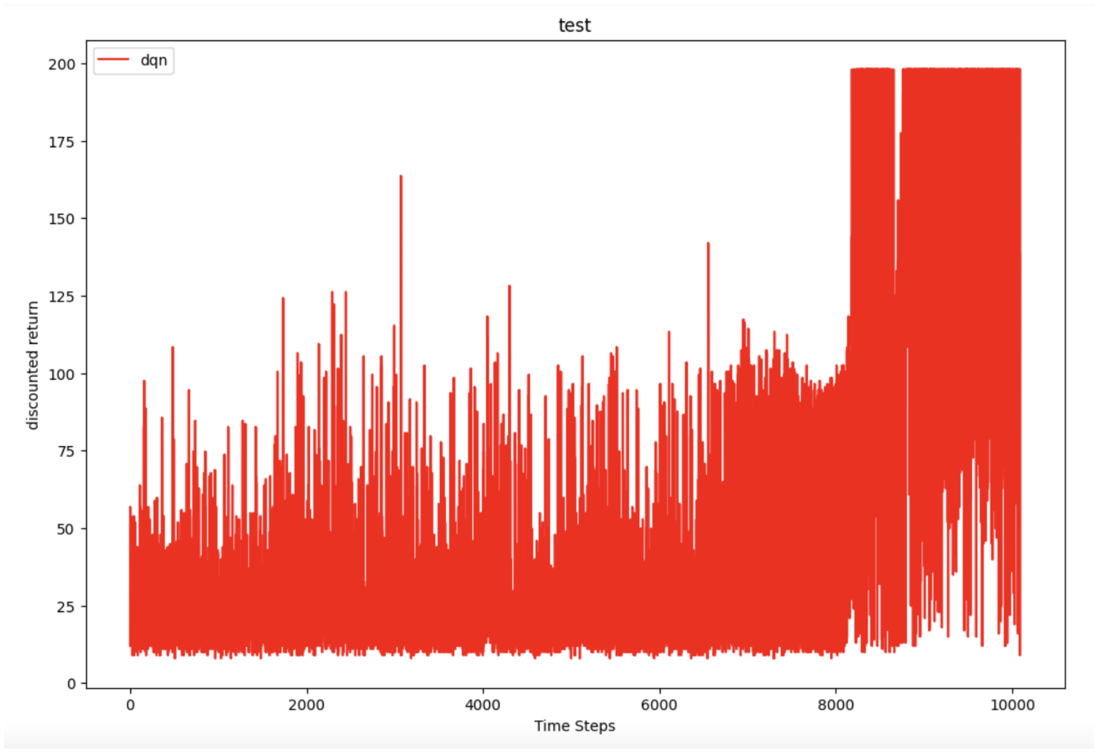
1e.) As seen in the above plots, the learned model from the NN is extremely accurate for the range [-10, 10], even for the lowest dimension hidden layer (8). The top right graph of each dimension, the ground truth vs. predicted y-values for the [-10, 10] range, shows an almost identical correlation, so much so that you cannot see the red ground truth scatter. However, in all dimensions, the performance clearly greatly worsens as the absolute value of the range increases above 10. This is due to the same issue we saw in ex7: the weight parameters of a neural network will become very large if the features are also large. The large weight values will cause further learning to be difficult at a certain point, seen by the linearity of our predictions as the input *x* increases. For this reason, normalization of datasets is often a key step in obtaining accurate NN results. Finally, as the dimension of the hidden layers increases, we can see the results are getting more accurate (within the range [-10, 10]). This is mainly seen in the level of variance in the MLE Loss plot as the epoch number increases. Comparing the graphs of dims 8 and 128, it is clear that the graph for a dimension of 128 shows that the network was able to converge to a minimum level of loss faster and without as much variability as the network with dimension 8 hidden layers. This is due to the fact that with an increased number of neurons, there is essentially more of a chance that a more exact answer will be found due to the increased number of connections and computation. However that is also a con to increasing the dimension of the hidden layers, as a high number will greatly increase the computation needed in each forward and backward iteration of the network and may not be necessary. This is seen by running the above network and monitoring run time and loss. The network with hidden layer dimensions of 8 runs much faster than the network with hidden layer dimensions of 128 and ultimately meets a very similar level of loss (almost 0). A typical rule of thumb is the number of neurons in a hidden layer, or multiple layers, should be around the average of the number of inputs and number of outputs.

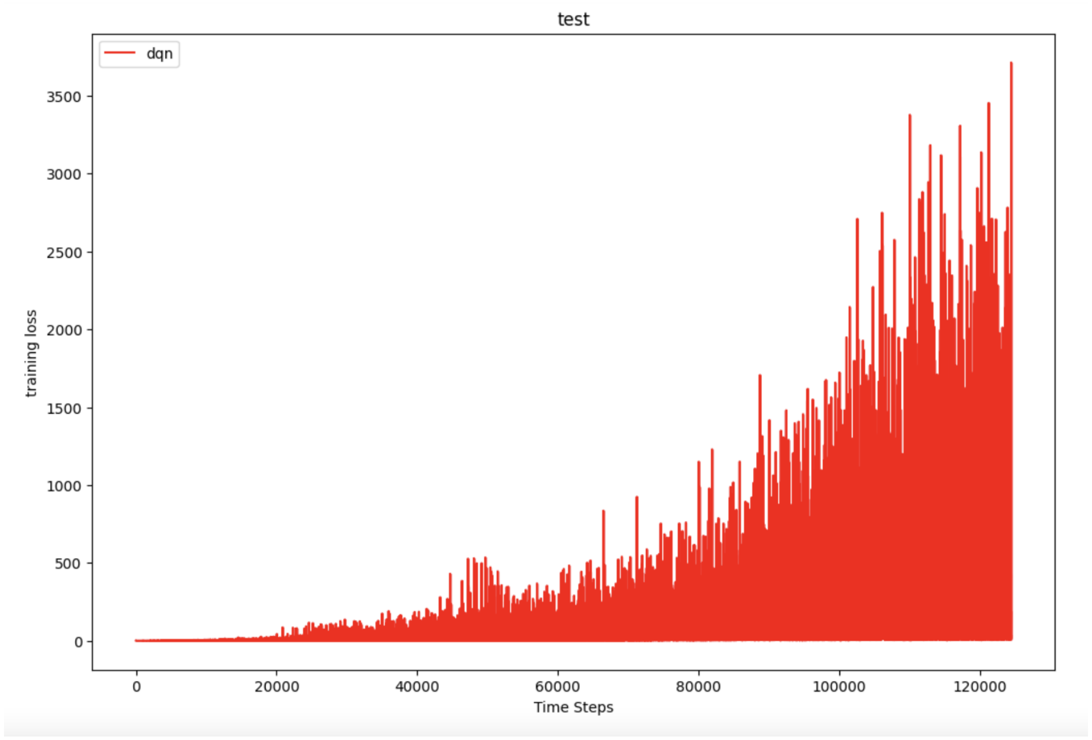
2.) d.) Plot for Four Rooms DQN Discounted Return and Training Loss





3.) Return and loss plots for the cart pole system are seen below:

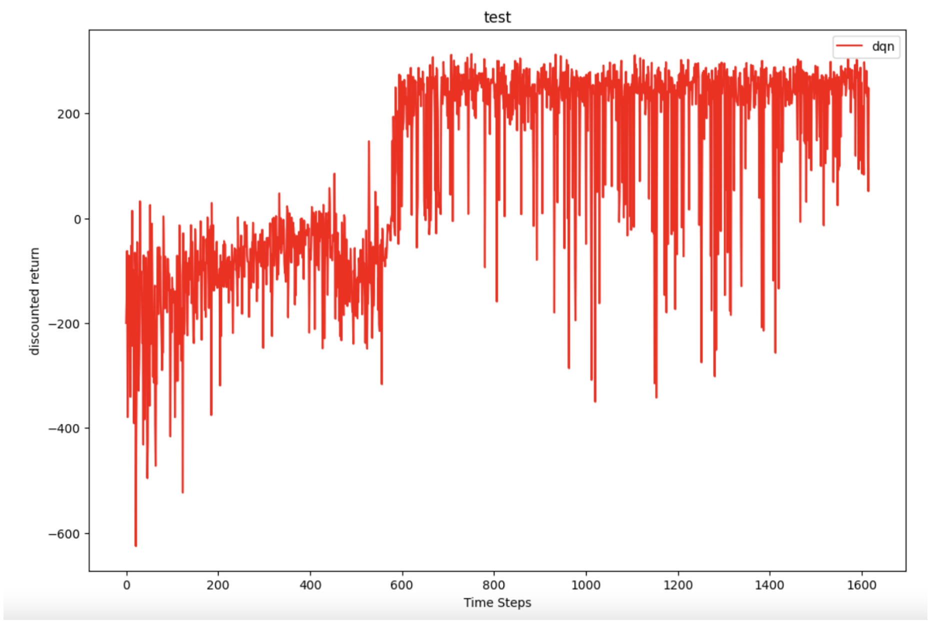


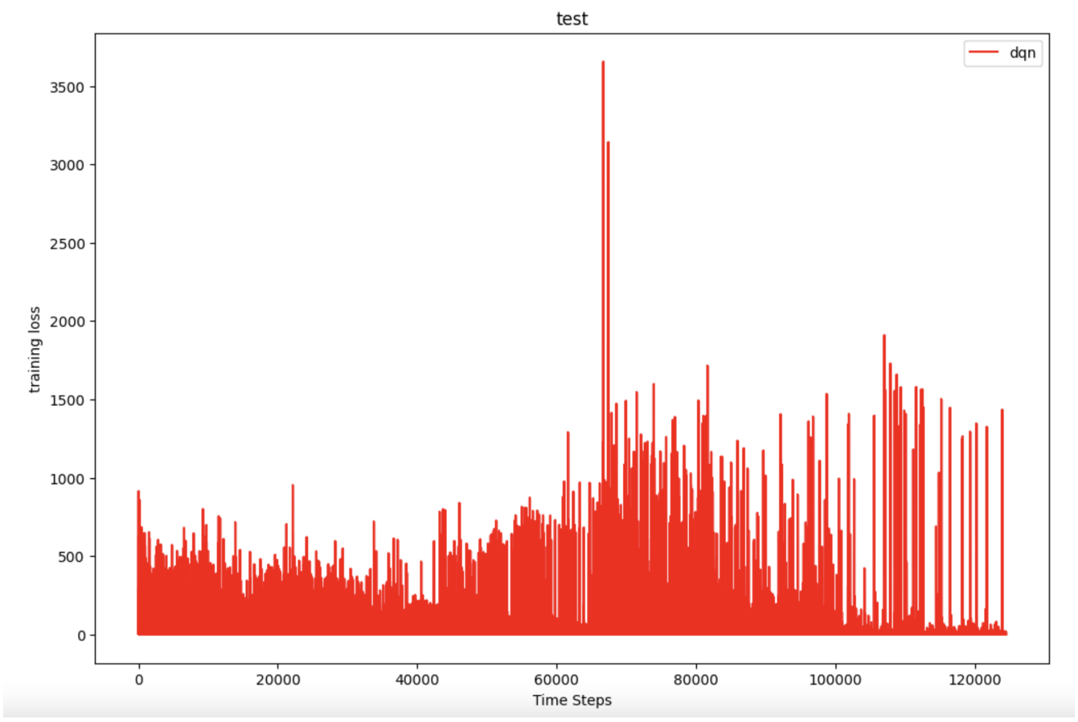


Unfortunately I did not have too much luck tuning my network. I did get the return to increase near the end of my training, although never to a stable value/loss like the four rooms implementation. Parameters can be seen below:

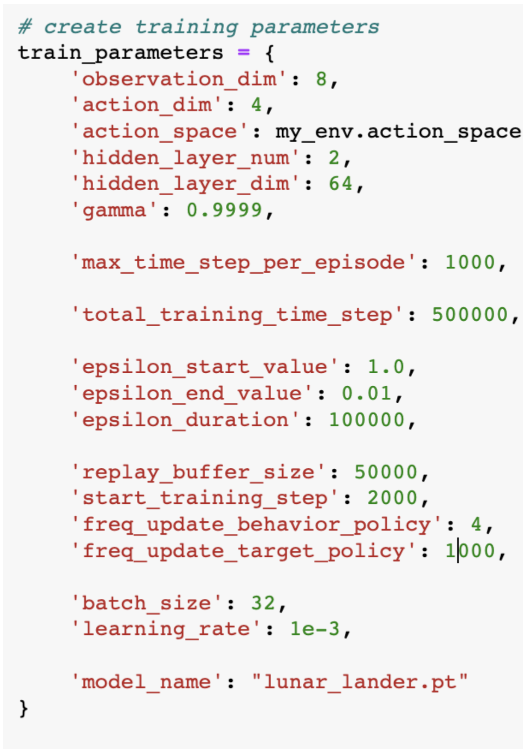
```python
# create training parameters
train_parameters = {
    'observation_dim': 4,
    'action_dim': 2,
    'action_space': my_env.action_space,
    'hidden_layer_num': 2,
    'hidden_layer_dim': 32,
    'gamma': 0.9995,

    'max_time_step_per_episode': 200,

    'total_training_time_step': 500_000,

    'epsilon_start_value': 1.0,
    'epsilon_end_value': 0.1,
    'epsilon_duration': 250000,

    'replay_buffer_size': 500_000,
    'start_training_step': 2000,
    'freq_update_behavior_policy': 4,
    'freq_update_target_policy': 2000,

    'batch_size': 64,
    'learning_rate': 1e-3,

    'model_name': "cartpole_v1.pt"
}
```

Return and loss plots for the Lunar Lander system are seen below:





After tuning, I was able to see an increase and convergence in discounted return, and a slow decrease in training loss after ~ 80000 timesteps. Parameters can be seen below:



```
# create training parameters
train_parameters = {
    'observation_dim': 8,
    'action_dim': 4,
    'action_space': my_env.action_space
    'hidden_layer_num': 2,
    'hidden_layer_dim': 64,
    'gamma': 0.9999,

    'max_time_step_per_episode': 1000,

    'total_training_time_step': 500000,

    'epsilon_start_value': 1.0,
    'epsilon_end_value': 0.01,
    'epsilon_duration': 100000,

    'replay_buffer_size': 50000,
    'start_training_step': 2000,
    'freq_update_behavior_policy': 4,
    'freq_update_target_policy': 1000,

    'batch_size': 32,
    'learning_rate': 1e-3,

    'model_name': "lunar_lander.pt"
}
```