HW2 Report - Mike Liu

Train & Test

Ablation Study

This code uses `ablation.json` file to config the parameters, and experiments to run are specified by network strings. e.g.:

```
{
  "networks": [
    "DQN_h:t64t64_lr:0.001_e:0.3_ed:1_em:0_tu:100_bs:32_rs:10000",
    "DQN_h:t64t64_lr:0.001_e:0.3_ed:1_em:0_tu:100_bs:32_rs:32",
    "DQN_h:t64t64_lr:0.001_e:0.3_ed:1_em:0_tu:1_bs:32_rs:10000",
    "DQN_h:t64t64_lr:0.001_e:0.3_ed:1_em:0_tu:1_bs:32_rs:32",
    "DQN_h:t64t64_lr:0.001_e:0.3_ed:1_em:0_tu:100_bs:32_rs:10000",
    "DQN_h:t64t64_lr:0.001_e:0.1_ed:1_em:0_tu:100_bs:32_rs:10000",
    "DQN_h:t64t64_lr:0.001_e:0.5_ed:1_em:0_tu:100_bs:32_rs:10000",
    "DQN_h:t32t32t32_lr:0.001_e:1_ed:0.99_em:0.3_tu:100_bs:32_rs:10000"
  ]
}
```

By adding different network string here, the strings will be parsed into network structure and parameters for each experiment. Take the first line as an example,

```
"DQN_h:t64t64_lr:0.001_e:0.3_ed:1_em:0_tu:100_bs:32_rs:10000"
```

`DQN` is the network name, `h` is the hidden layer, `t64t64` means two hidden layers with 64 neurons each with tanh activation. It supports `t` for tanh activation, `s` for sigmoid activation, `r` for relu activation, `l` for leaky relu activation, `e` for elu activation, etc. `lr` is the learning rate, `e` is the epsilon, `ed` is the epsilon decay, `em` is the epsilon minimum, `tu` is the number of steps between target updates, `bs` is the batch size, `rs` is the replay size.

By setting these parameters, one can test different network structures and parameters, as required on the assignment:

- change `h` to test different network structures
- change `e, ed, em` to test different epsilon values and epsilon decay
- change `tu` to 1 to reset the target network every step
- change `rs` equal to 32 to cancel the replay buffer

One can also set training episodes and runs in `ablation.json`, i.e.

```
{
  "networks": [...],
```

```
  "episodes": 2000,
  "runs": 10
}
```

which tells the code to run 10 training runs of 2000 episodes each.

Therefore, this config above will run 8 experiments with different network structures and parameters, 10 runs each, and 2000 episodes each run.

There is **no hardcoded** experiments in the code, but done by **parsing the json file**. In which way, students can add new experiments with minimal effort.

Commandline Arguments

A few commandline arguments controls the behavior of the experiment process:

- `--config`: specify the config file to use, default is `ablation.json`
- `--load`: specify if to load the model from the checkpoint, default is `False`
- `--train`: specify if to train the model, default is `False`

Learning Curve Plotting

The training and test process will save all figures but the learning curve plots in `figures/`. The rewards data will be saved after the training, and the plotting of learning curve is done in `plot_learning_curves.py` script.

The ablation plotting is configured by the `learning_curve_comp` entry in `ablation.json`, for example,

```
{
 "learning_curve_comp": [
  [
   "DQN_h:t64t64_lr:0.001_e:0.3_ed:1_em:0_tu:100_bs:32_rs:10000",
   "DQN_h:t64t64_lr:0.001_e:0.3_ed:1_em:0_tu:100_bs:32_rs:32",
   "DQN_h:t64t64_lr:0.001_e:0.3_ed:1_em:0_tu:1_bs:32_rs:10000",
   "DQN_h:t64t64_lr:0.001_e:0.3_ed:1_em:0_tu:1_bs:32_rs:32"
  ],
  [
   "DQN_h:t64t64_lr:0.001_e:1_ed:0.99_em:0.3_tu:100_bs:32_rs:10000",
   "DQN_h:t64t64_lr:0.001_e:0.3_ed:1_em:0_tu:100_bs:32_rs:10000",
   "DQN_h:t32t32t32_lr:0.001_e:1_ed:0.99_em:0.3_tu:100_bs:32_rs:10000",
   "DQN_h:r64r64_lr:0.001_e:0.3_ed:1_em:0_tu:100_bs:32_rs:10000"
  ]
 ]
}
```

This configuration will plot 2 figures, each figure will have 4 learning curves, w.r.t. the 4 network strings in each list. Legends will be automatically generated from the network strings.

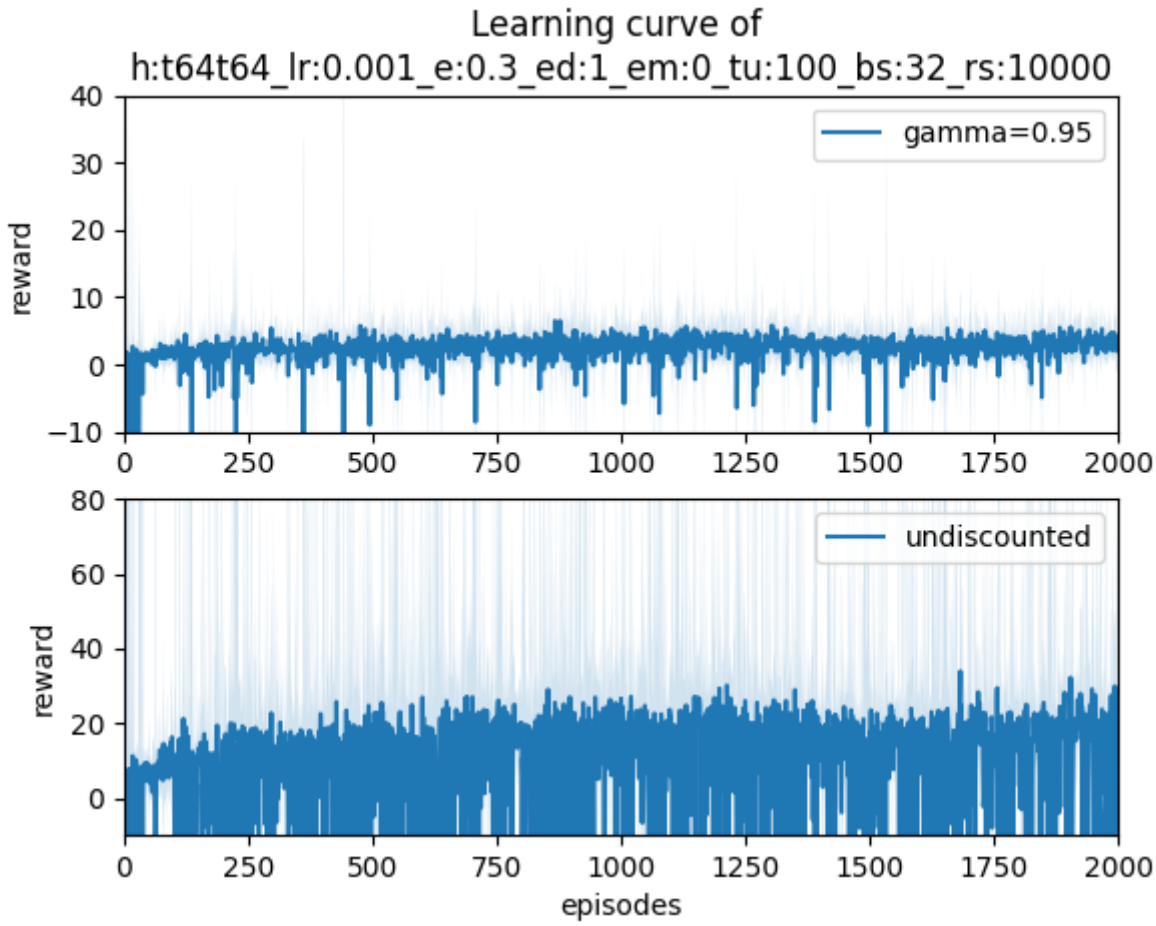
Run

```
python3 train_discreteaction_pendulum.py --train   # plotting included
```

Results

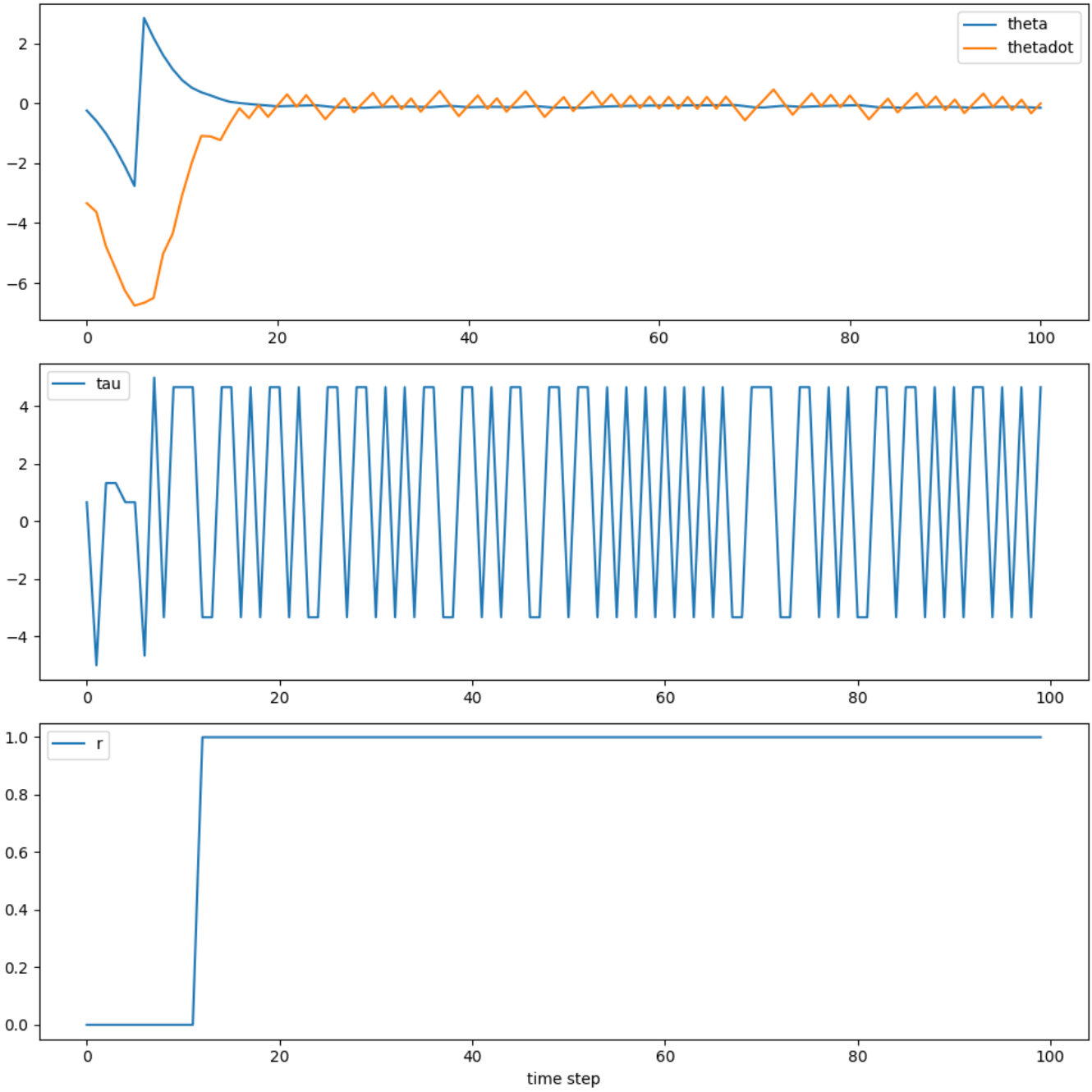
1. Learning Curve

Each experiment are run only 10 times, for the limit of training time. Example results are from the first training run of the agent setting: `DQN_h:t64t64_lr:0.001_e:0.3_ed:1_em:0_tu:100_bs:32_rs:10000`, i.e. 2 hidden layers with 64 neurons each, tanh activation, learning rate 0.001, epsilon 0.3, no epsilon decay, target update every 100 steps, batch size 32, replay size 10000.



2. Example Trajectory

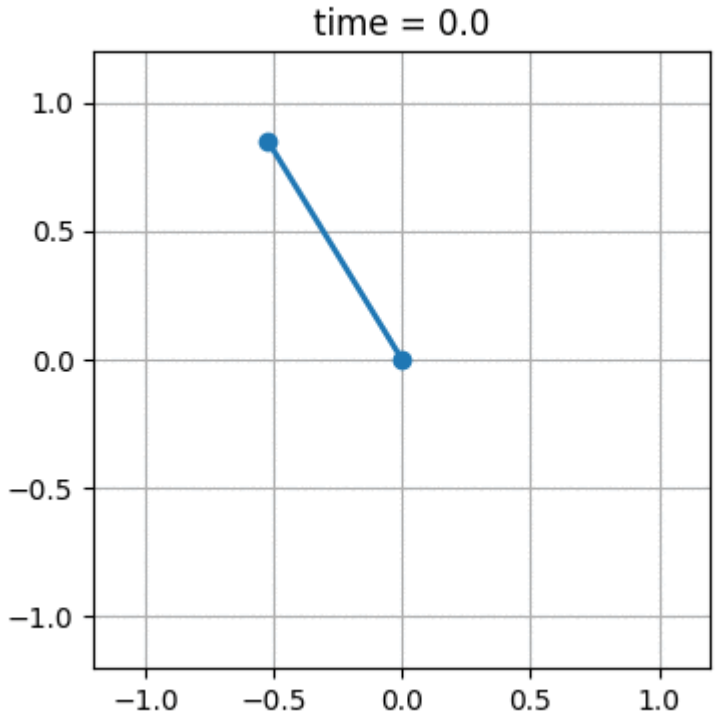
Example agent same as above.



3. Animation of Example Trajectory

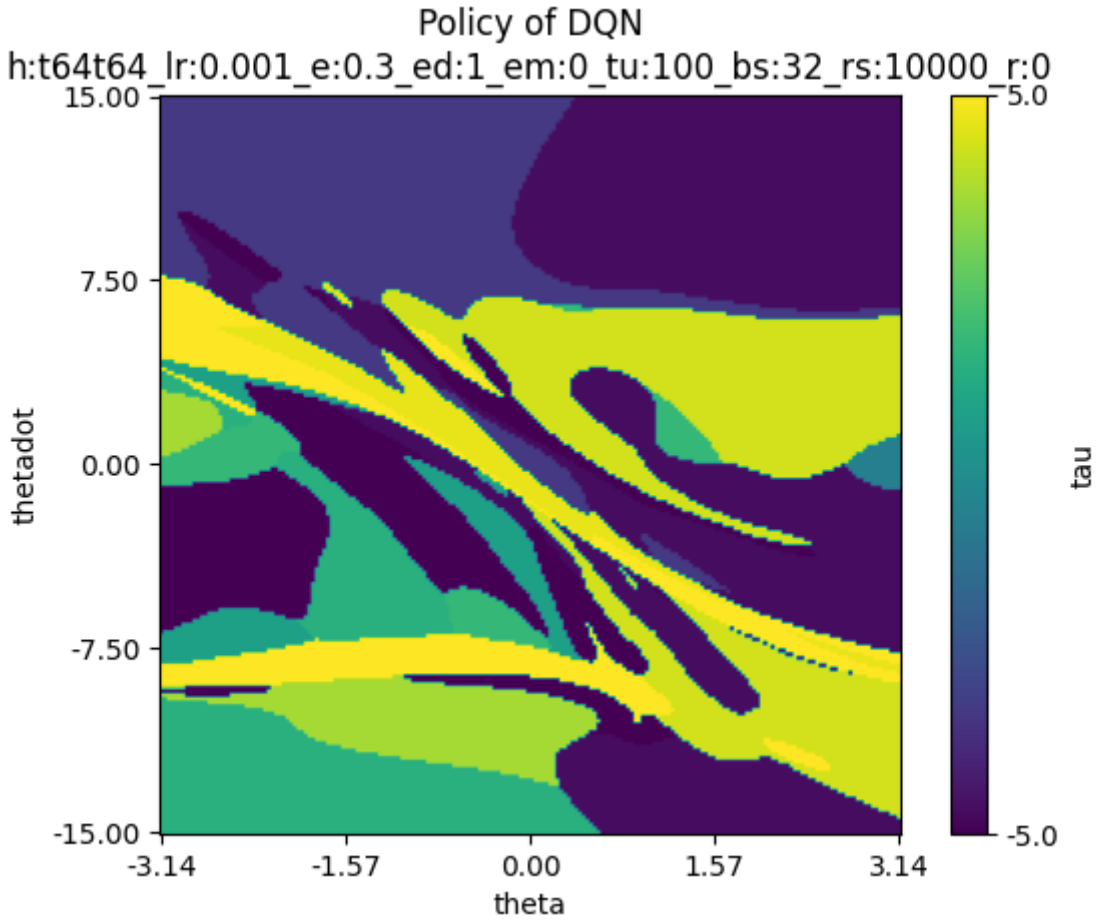
Example agent same as above.

Link to gif



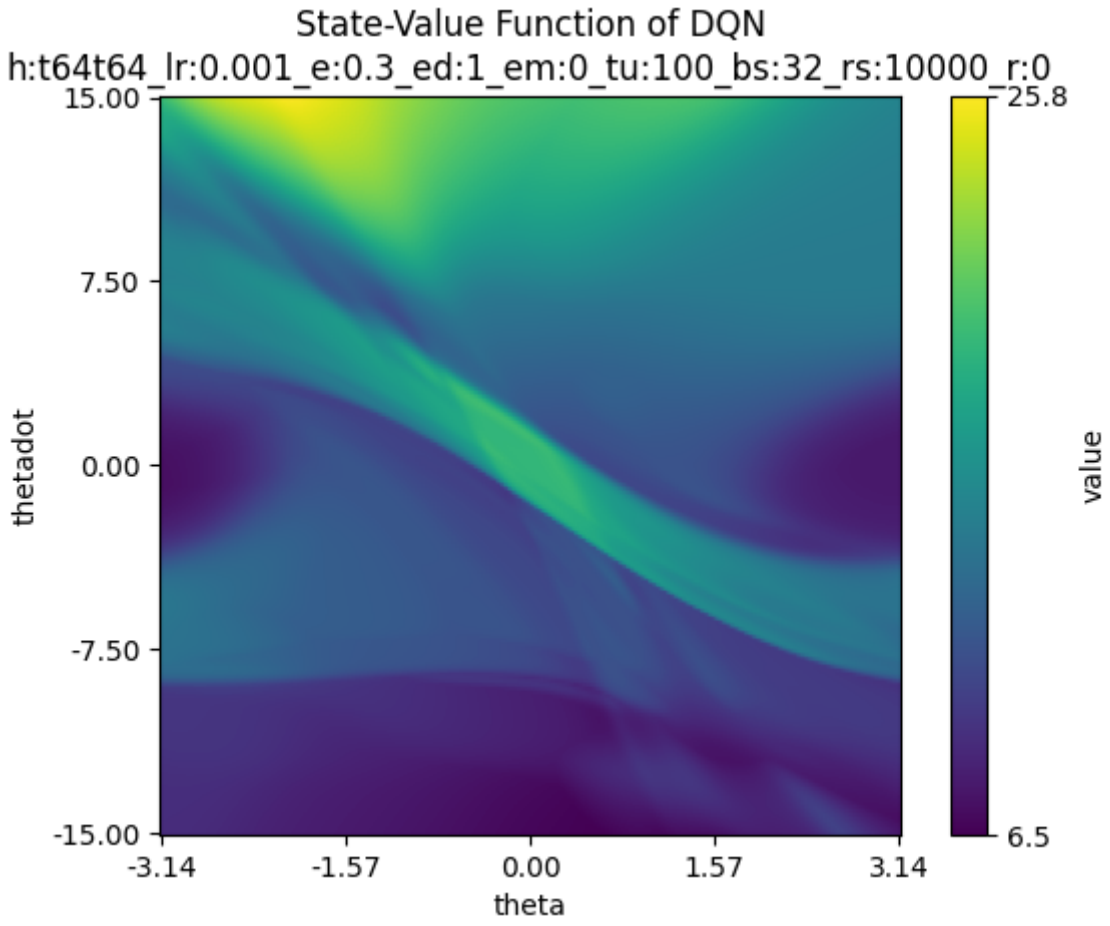
4. Policy Plotting

Example agent same as above.

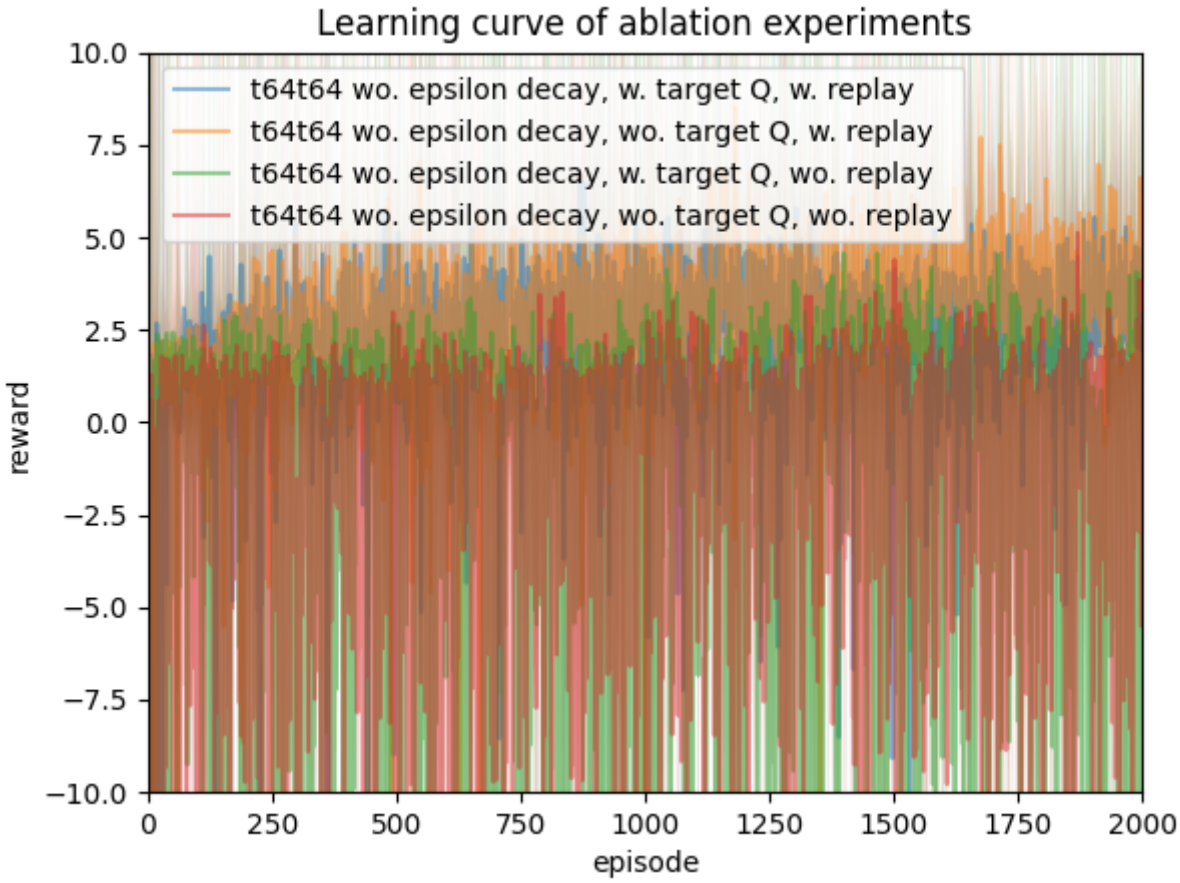


5. State-Value Function Plotting

Example agent same as above.



6. Ablation Study



- Lacking of experience replay results in unstable training process, and reduced performance.
- Lacking of target Q network results slow down the convergence, and less stable training, also results in a worse policy.

More Ablation

Result

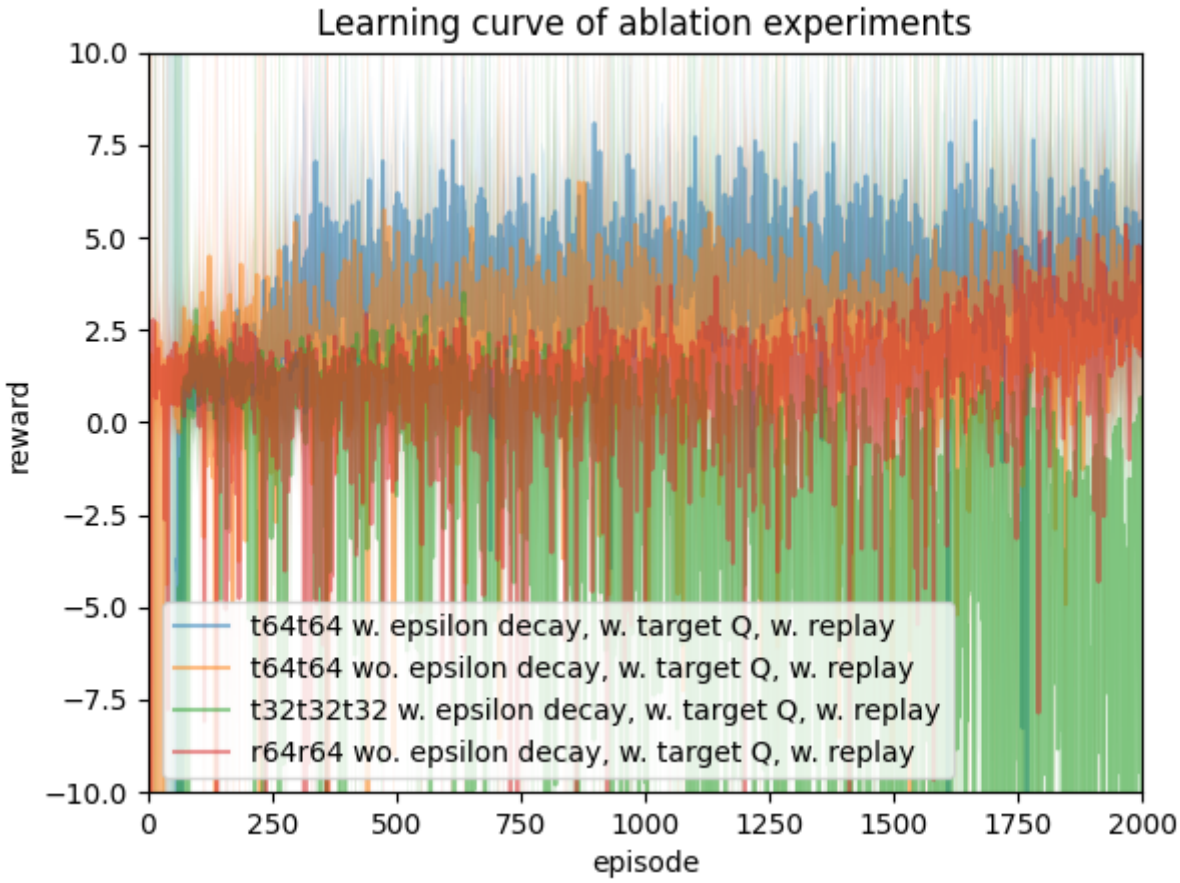
More ablation study is experimented on the following settings:

```
"DQN_h:t64t64_lr:0.001_e:1_ed:0.99_em:0.3_tu:100_bs:32_rs:10000",
"DQN_h:t64t64_lr:0.001_e:0.3_ed:1_em:0_tu:100_bs:32_rs:10000",
"DQN_h:t32t32t32_lr:0.001_e:1_ed:0.99_em:0.3_tu:100_bs:32_rs:10000",
"DQN_h:r64r64_lr:0.001_e:0.3_ed:1_em:0_tu:100_bs:32_rs:10000"
```

i.e.,

- epsilon 1 decay to 0.3, vs. epsilon fixed to 0.3, i.e. setting 1 vs 2
- 2x64 hidden layers, vs. 3x32 hidden layers, i.e. setting 1 vs 3
- tanh activation, vs relu activation, i.e. setting 1 vs 4

The results are shown below:



Discussion

1. Under the parameters of 2000 max episodes and `lr` is 0.001, 3x32 structure fails to converge, while 2x64 structure converges to a good policy.
2. Epsilon delay can improve the convergence speed, and also provides a better policy than fixed epsilon.
3. ReLU activation has a slower convergence speed than tanh activation, but provides an equally good policy, after 2000 episodes.