INFSCI 2935: Human-Robot Interaction
Reinforcement Learning Project

Vishruth Pradeep Reddy
vpr8@pitt.edu

Before we dive into the project, I would like to provide a brief description as to what each hyperparameter is and how they affect the model with change in values.

Hyper Parameter Description

1. Buffer Size:
- Description: The size of the memory for past experiences.
- Purpose: Stores diverse experiences for learning.
- Increase: Captures more diverse experiences but increases memory usage.
- Decrease: Limits experience variety, potentially affecting learning quality.

2. Learning Starts:
- Description: Delay before the model begins learning.
- Purpose: Gathers initial data for informed learning.
- Increase: Better data collection but delays learning.
- Decrease: Starts learning sooner but might begin with noisy or sparse data.

3. Batch Size:
- Description: Number of experiences used in each learning update.
- Purpose: Affects learning stability and speed.
- Increase: More stable learning but slower training due to larger computations.
- Decrease: Faster training but might result in less stable learning due to noisier updates.

4. Gamma:
- Description: Balances immediate vs. future rewards.
- Purpose: Determines the importance of future rewards.
- Increase: Prioritizes future rewards but might overlook immediate gains.
- Decrease: Focuses more on immediate rewards, potentially disregarding long-term gains.

5. Train Frequency:
- Description: Frequency of updating the model's parameters.
- Purpose: Controls learning speed and parameter updates.
- Increase: Faster learning but might introduce more instability due to frequent updates.
- Decrease: Slower learning but could lead to more stable updates.

6. Gradient Steps:
- Description: Number of optimization steps after each batch.
- Purpose: Affects convergence speed and stability.
- Increase: Faster convergence but might introduce instability due to larger updates.
- Decrease: Slower convergence but potentially more stable updates.

7. Target Update Interval:

- Description: Frequency of updating a secondary network.
- Purpose: Stabilizes learning by controlling update frequency.
- Increase: More stable learning but slower adaptation to changes.
- Decrease: Faster adaptation to changes but might introduce instability.

8. Exploration Fraction:

- Description: Determines exploration vs. exploitation.
- Purpose: Balances new strategy discovery and exploiting known ones.
- Increase: More exploration but might delay exploiting known strategies.
- Decrease: Exploits known strategies faster but might miss better ones.

9. Verbose:

- Description: Controls the level of output during training.
- Purpose: Provides information about the training process.
- Increase: More detailed information displayed during training.
- Decrease: Less information or no output during training.

10. Training Steps:

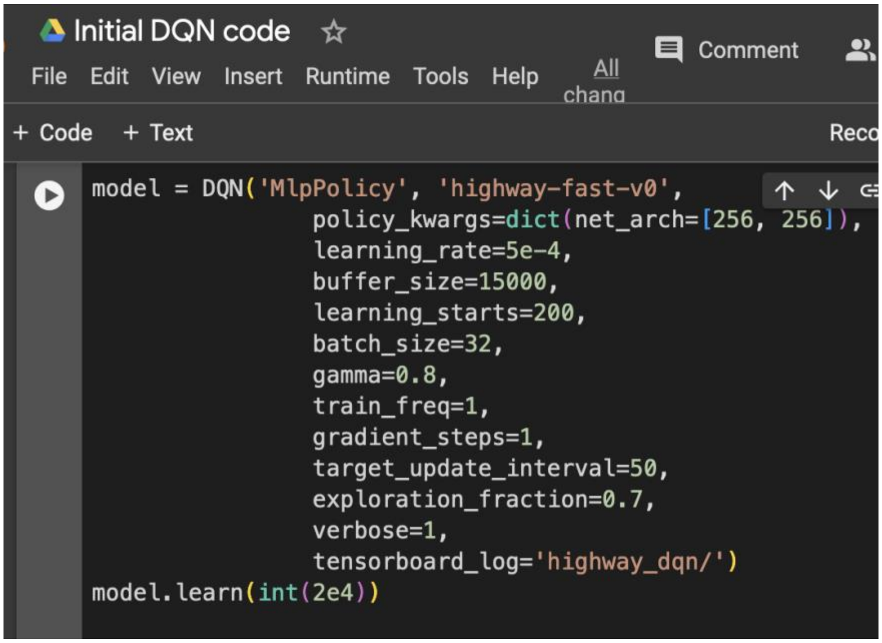
- Description: The total number of steps for training the model.
- Purpose: Sets the duration or extent of the training process.
- Increase: More training steps might lead to further learning and refinement.
- Decrease: Fewer steps might result in quicker but potentially less optimal learning.

Algorithm 1: Initial DQN Model (as given in the boiler-plate code)

Survived 20+ seconds? NO

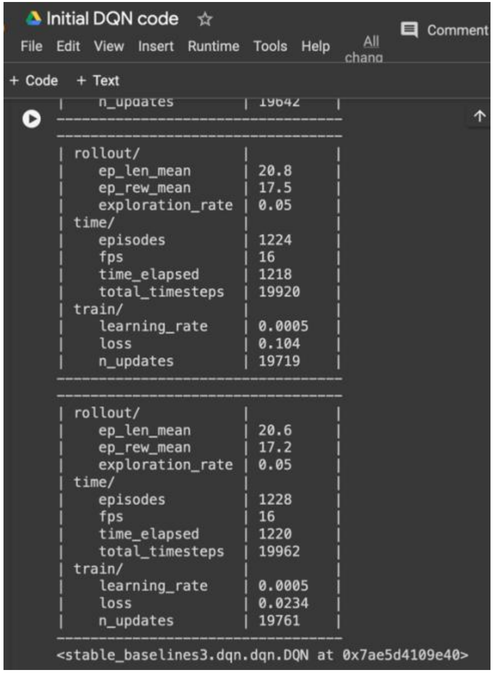
Hyperparameter	Description	Value
Network Architecture	Number of hidden layers	**2**
Neuron Count	Number of neurons in each hidden layer	**[256, 256]**
Learning rate	Rate of parameter updates	**5e-4**
Buffer size	Size of experience replay buffer	**15000**
Learning starts	Number of steps before learning begins	**200**
Batch size	Number of experiences per training iteration	**32**
Gamma	Discount factor for future rewards	**0.8**
Train frequency	Frequency of model training	**1**
Gradient steps	Number of optimization steps after each batch	**1**
Target update interval	Frequency of updating target network	**50**
Exploration fraction	Fraction of exploration during training	**0.7**
Verbose	Verbosity level during training	**1**
Training steps	Total steps for training	**20,000**

Model:



```
model = DQN('MlpPolicy', 'highway-fast-v0',
                policy_kwargs=dict(net_arch=[256, 256]),
                learning_rate=5e-4,
                buffer_size=15000,
                learning_starts=200,
                batch_size=32,
                gamma=0.8,
                train_freq=1,
                gradient_steps=1,
                target_update_interval=50,
                exploration_fraction=0.7,
                verbose=1,
                tensorboard_log='highway_dqn/')
model.learn(int(2e4))
```

Final Training Stats:

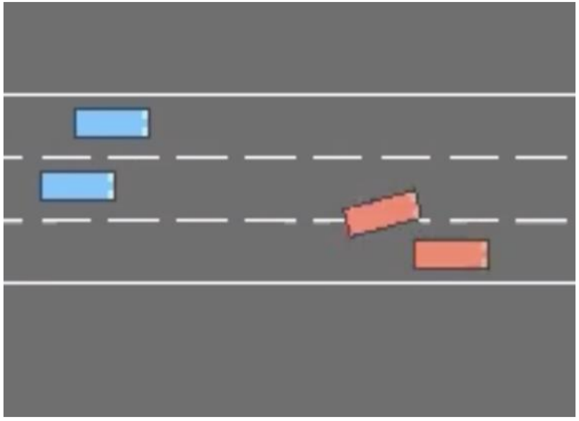


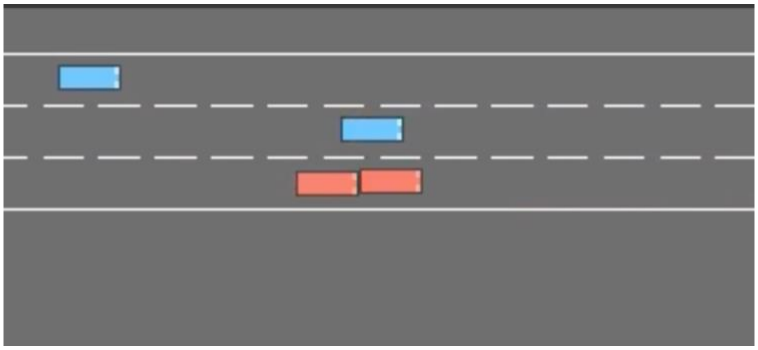
```
|    n_updates        | 19642    |
-----------------------------------
-----------------------------------
| rollout/            |          |
|    ep_len_mean      | 20.8     |
|    ep_rew_mean      | 17.5     |
|    exploration_rate | 0.05     |
| time/               |          |
|    episodes         | 1224     |
|    fps              | 16       |
|    time_elapsed     | 1218     |
|    total_timesteps  | 19920    |
| train/              |          |
|    learning_rate    | 0.0005   |
|    loss             | 0.104    |
|    n_updates        | 19719    |
-----------------------------------
-----------------------------------
| rollout/            |          |
|    ep_len_mean      | 20.6     |
|    ep_rew_mean      | 17.2     |
|    exploration_rate | 0.05     |
| time/               |          |
|    episodes         | 1228     |
|    fps              | 16       |
|    time_elapsed     | 1220     |
|    total_timesteps  | 19962    |
| train/              |          |
|    learning_rate    | 0.0005   |
|    loss             | 0.0234   |
|    n_updates        | 19761    |
-----------------------------------
<stable_baselines3.dqn.dqn.DQN at 0x7ae5d4109e40>
```

Images:

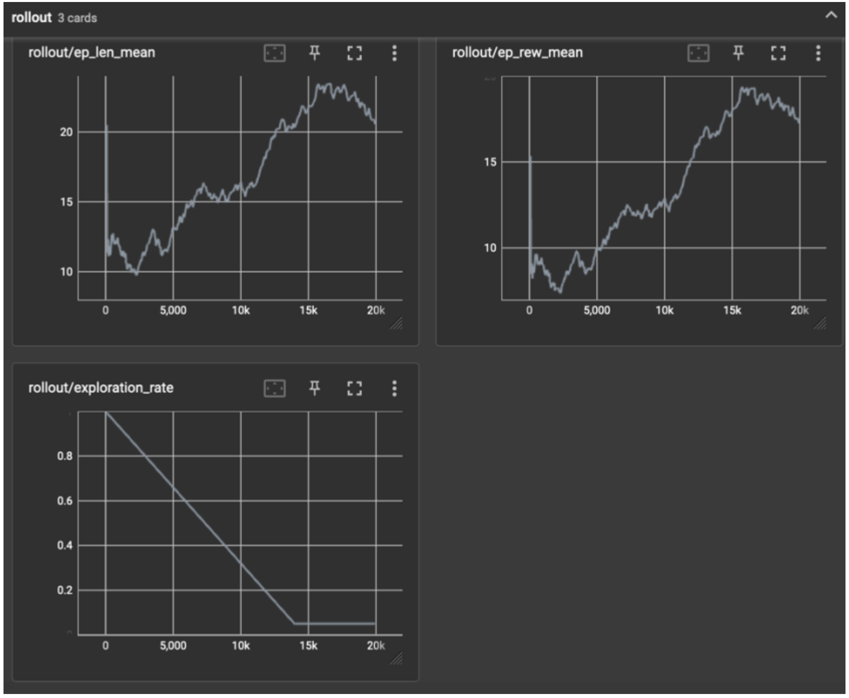
The model could run without crashing for a maximum period of 13 seconds.

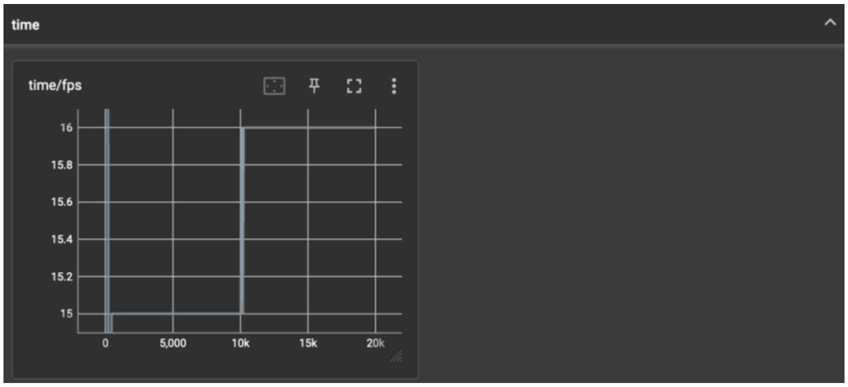
Image of the crash in 2 episodes, ran successfully in 1 episode:

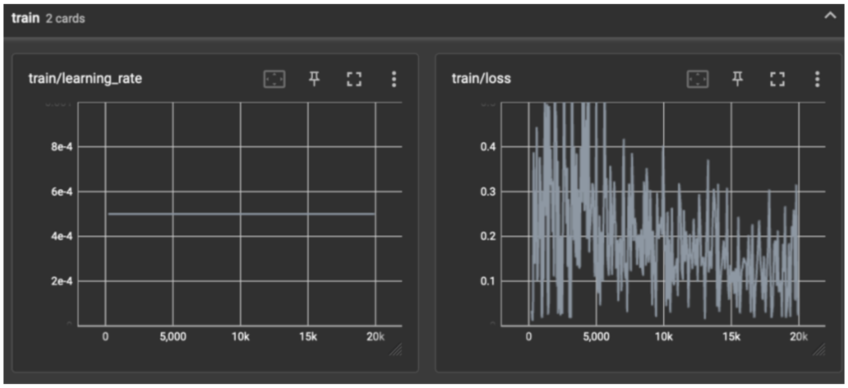




Graphs:


rollout 3 cards
rollout/ep_len_mean
20
15
10
0
5,000
10k
15k
20k
rollout/ep_rew_mean
15
10
0
5,000
10k
15k
20k
rollout/exploration_rate
0.8
0.6
0.4
0.2
0
5,000
10k
15k
20k


time
time/fps
16
15.8
15.6
15.4
15.2
15
0
5,000
10k
15k
20k

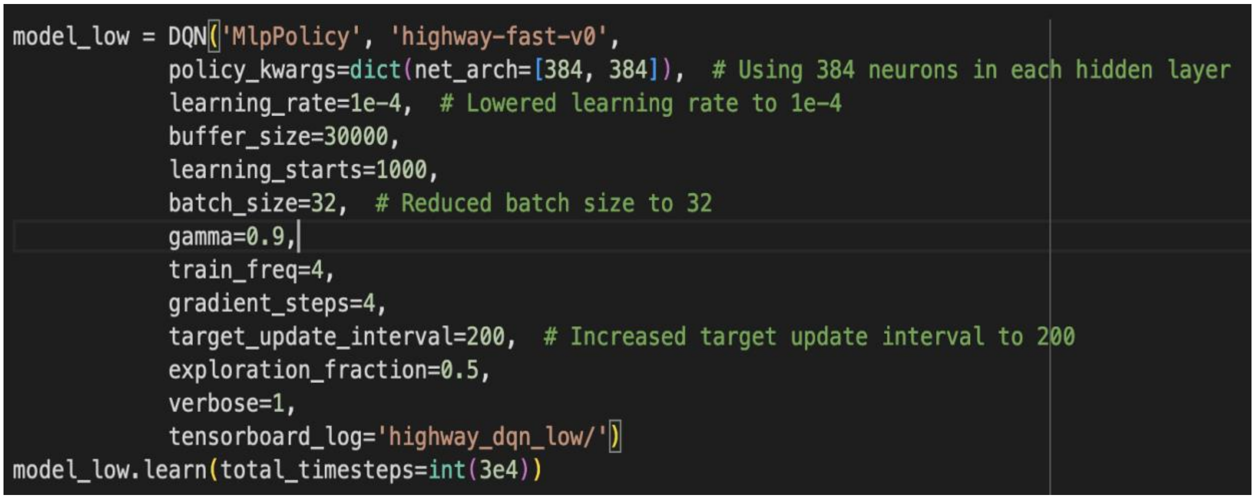

train 2 cards
train/learning_rate
8e-4
6e-4
4e-4
2e-4
0
5,000
10k
15k
20k
train/loss
0.4
0.3
0.2
0.1
0
5,000
10k
15k
20k

Algorithm 2: DQN Model (with low hyperparameter values for Learning Rate, Batch Size and Target Update Interval)

Survived 20+ seconds? YES

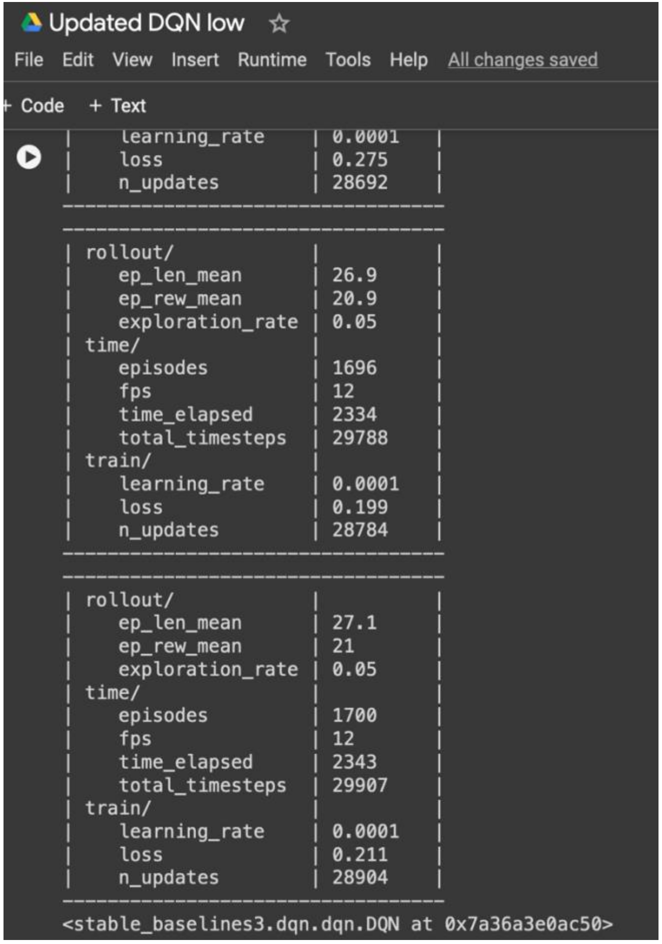
Hyperparameter	Description	Value
Network Architecture	Number of hidden layers	**2**
Neuron Count	Number of neurons in each hidden layer	**[384, 384]**
Learning rate	Rate of parameter updates	**1e-4**
Buffer size	Size of experience replay buffer	**30000**
Learning starts	Number of steps before learning starts	**1000**
Batch size	Number of experiences per training iteration	**32**
Gamma	Discount factor for future rewards	**0.9**
Train frequency	Frequency of model training	**4**
Gradient steps	Number of optimization steps after each batch	**4**
Target update interval	Frequency of updating target network	**200**
Exploration fraction	Fraction of exploration during training	**0.5**
Verbose	Level of output during training	**1**
Training steps	Total steps for training	**30000**

Model:



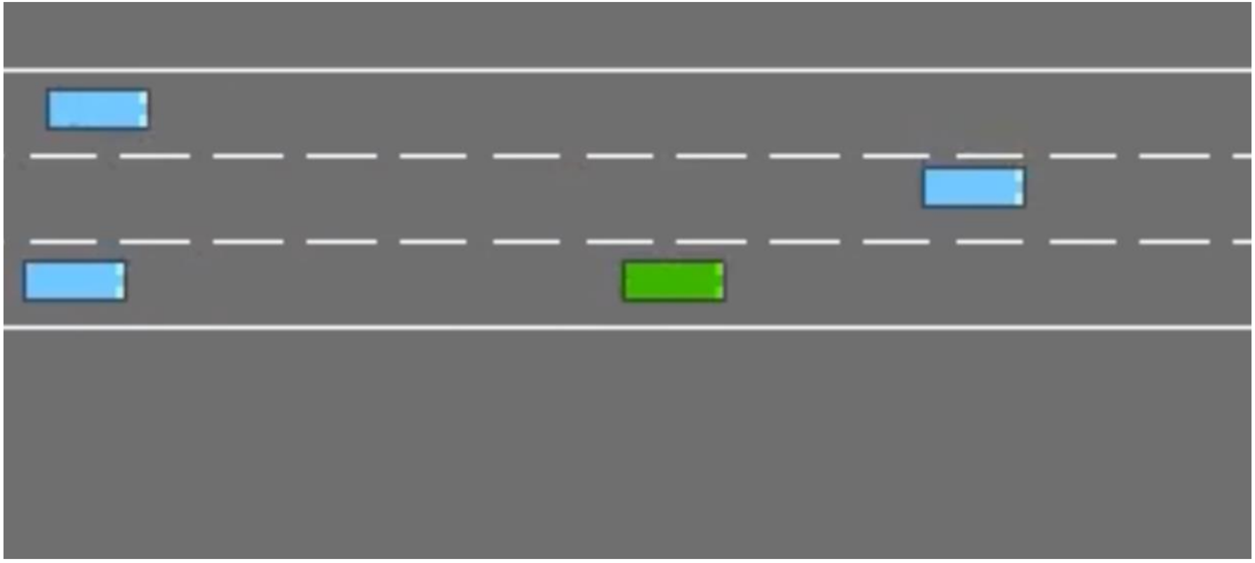
```
model_low = DQN('MlpPolicy', 'highway-fast-v0',
            policy_kwargs=dict(net_arch=[384, 384]),  # Using 384 neurons in each hidden layer
            learning_rate=1e-4,  # Lowered learning rate to 1e-4
            buffer_size=30000,
            learning_starts=1000,
            batch_size=32,  # Reduced batch size to 32
            gamma=0.9,
            train_freq=4,
            gradient_steps=4,
            target_update_interval=200,  # Increased target update interval to 200
            exploration_fraction=0.5,
            verbose=1,
            tensorboard_log='highway_dqn_low/')
model_low.learn(total_timesteps=int(3e4))
```

Final Training Stats:

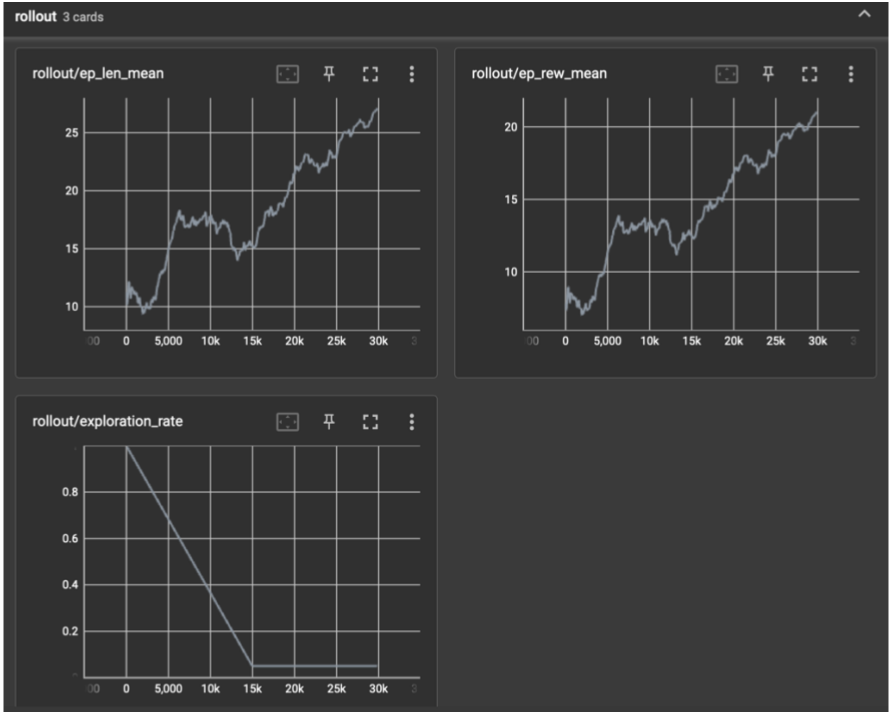


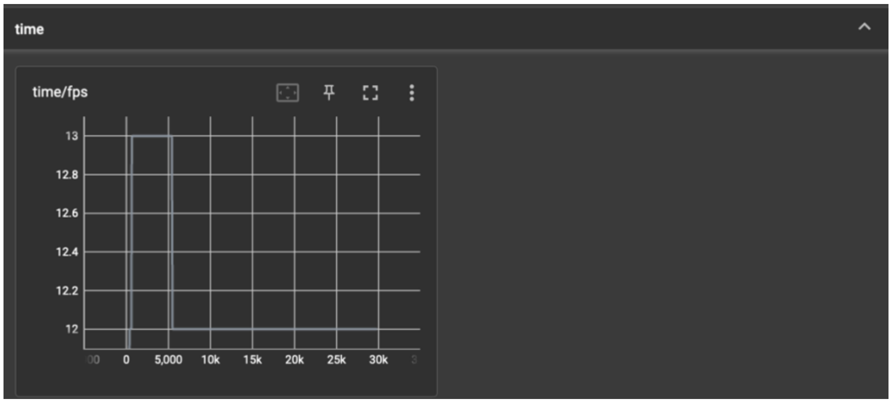
```
|    learning_rate     | 0.0001   |
|    loss              | 0.275    |
|    n_updates         | 28692    |
----------------------------------
----------------------------------
| rollout/             |          |
|    ep_len_mean       | 26.9     |
|    ep_rew_mean       | 20.9     |
|    exploration_rate  | 0.05     |
| time/                |          |
|    episodes          | 1696     |
|    fps               | 12       |
|    time_elapsed      | 2334     |
|    total_timesteps   | 29788    |
| train/               |          |
|    learning_rate     | 0.0001   |
|    loss              | 0.199    |
|    n_updates         | 28784    |
----------------------------------
----------------------------------
| rollout/             |          |
|    ep_len_mean       | 27.1     |
|    ep_rew_mean       | 21       |
|    exploration_rate  | 0.05     |
| time/                |          |
|    episodes          | 1700     |
|    fps               | 12       |
|    time_elapsed      | 2343     |
|    total_timesteps   | 29907    |
| train/               |          |
|    learning_rate     | 0.0001   |
|    loss              | 0.211    |
|    n_updates         | 28904    |
----------------------------------
<stable_baselines3.dqn.dqn.DQN at 0x7a36a3e0ac50>
```

Image (No signs for crashing in any episode, ran successfully in all 3 episodes):
Travel Speed for car was relatively slow.



Graphs:


rollout 3 cards
rollout/ep_len_mean
25
20
15
10
0
5,000
10k
15k
20k
25k
30k
rollout/ep_rew_mean
20
15
10
0
5,000
10k
15k
20k
25k
30k
rollout/exploration_rate
0.8
0.6
0.4
0.2
0
5,000
10k
15k
20k
25k
30k

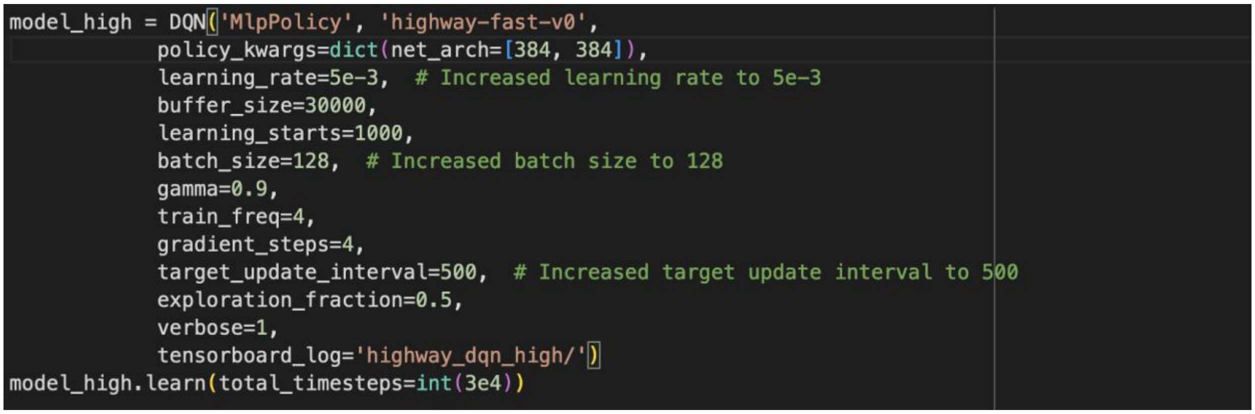

time
time/fps
13
12.8
12.6
12.4
12.2
12
0
5,000
10k
15k
20k
25k
30k

Algorithm 3: (DQN with high hyperparameter values for Learning Rate, Batch Size and Target Update Interval)

Survived 20+ seconds? NO

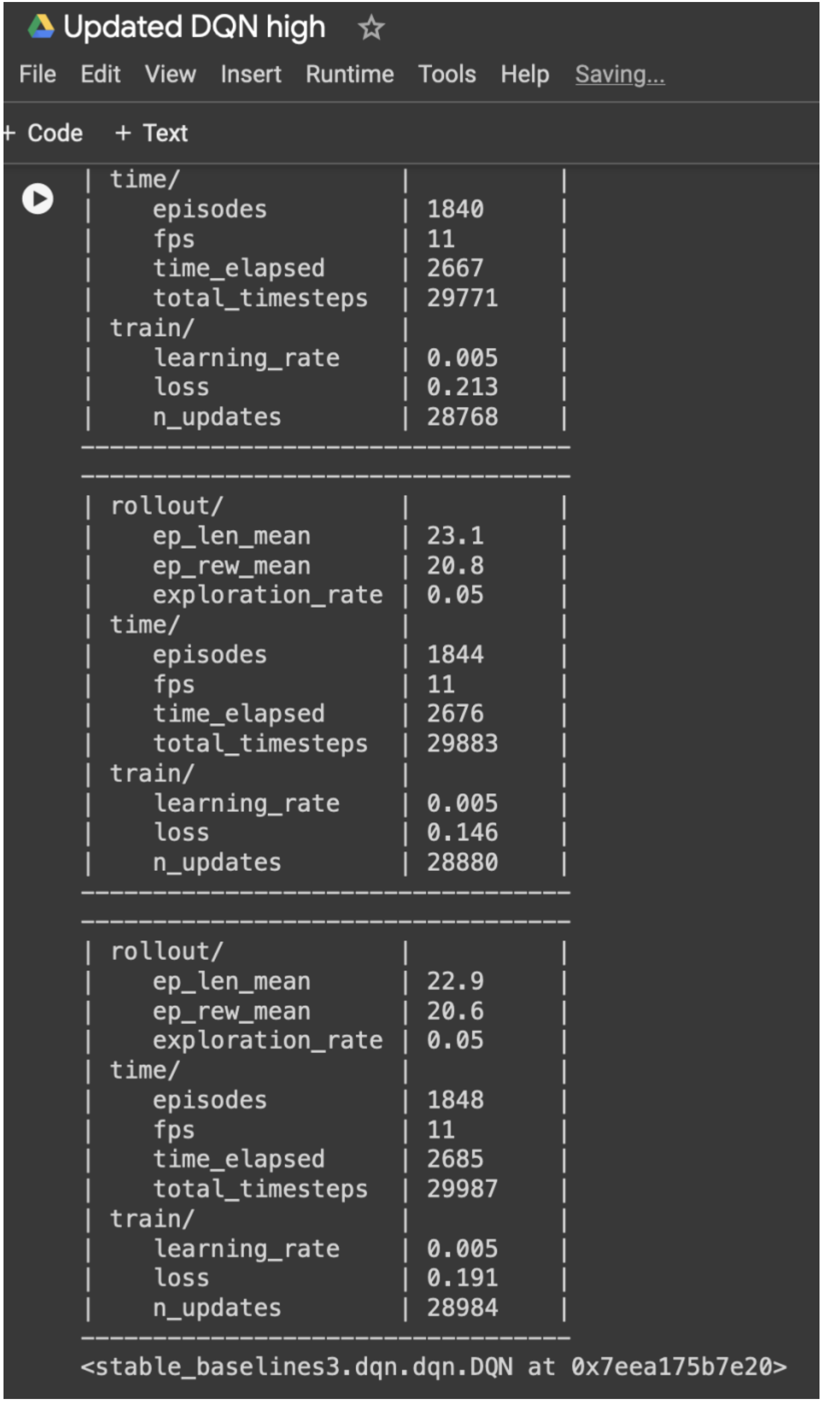
Hyperparameter	Description	Value
Network Architecture	Number of hidden layers	2
Neuron Count	Number of neurons in each hidden layer	[384, 384]
Learning rate	Rate of parameter updates	5e-3
Buffer size	Size of experience replay buffer	30000
Learning starts	Number of steps before learning starts	1000
Batch size	Number of experiences per training iteration	128
Gamma	Discount factor for future rewards	0.9
Train frequency	Frequency of model training	4
Gradient steps	Number of optimization steps after each batch	4
Target update interval	Frequency of updating target network	500
Exploration fraction	Fraction of exploration during training	0.5
Verbose	Level of output during training	1
Training steps	Total steps for training	30000

Model:



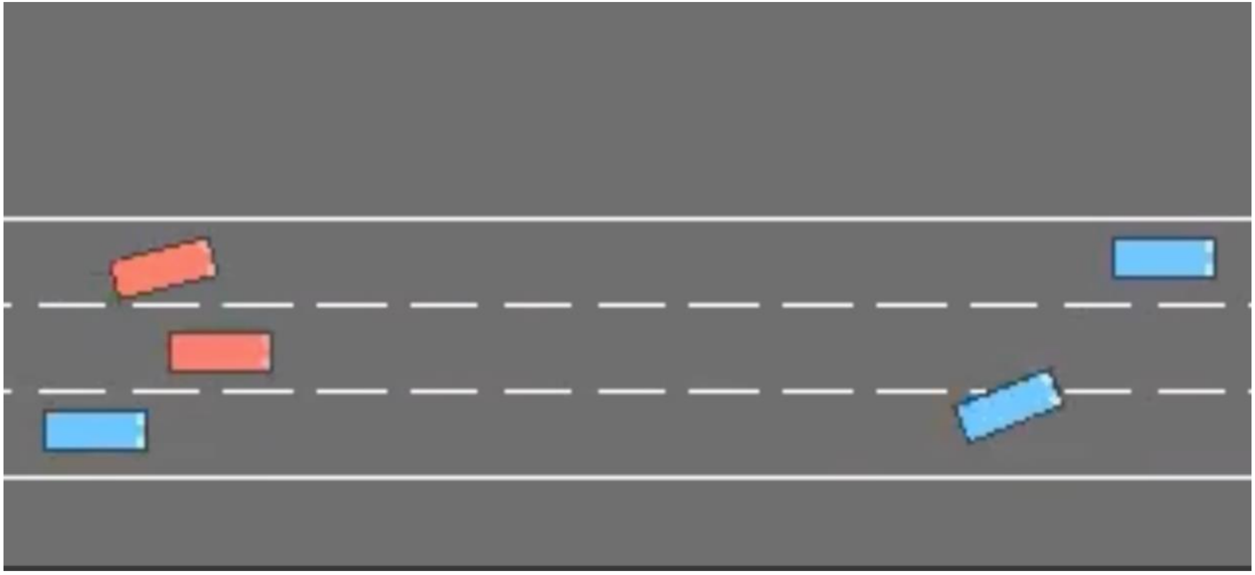
```
model_high = DQN('MlpPolicy', 'highway-fast-v0',
            policy_kwargs=dict(net_arch=[384, 384]),
            learning_rate=5e-3,  # Increased learning rate to 5e-3
            buffer_size=30000,
            learning_starts=1000,
            batch_size=128,  # Increased batch size to 128
            gamma=0.9,
            train_freq=4,
            gradient_steps=4,
            target_update_interval=500,  # Increased target update interval to 500
            exploration_fraction=0.5,
            verbose=1,
            tensorboard_log='highway_dqn_high/')
model_high.learn(total_timesteps=int(3e4))
```

Final Training Stats:

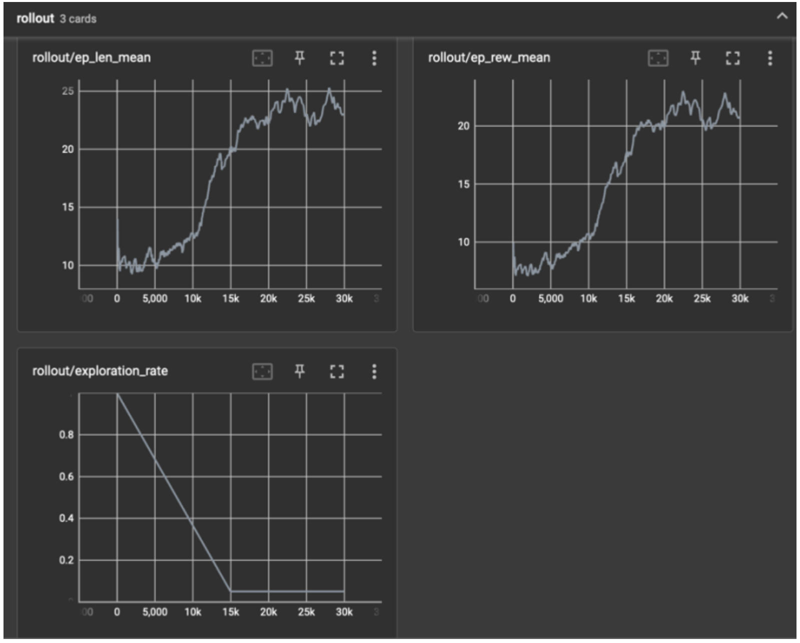


```
| time/                 |          |
|    episodes           | 1840     |
|    fps                | 11       |
|    time_elapsed       | 2667     |
|    total_timesteps    | 29771    |
| train/                |          |
|    learning_rate      | 0.005    |
|    loss               | 0.213    |
|    n_updates          | 28768    |
----------------------------------
----------------------------------
| rollout/              |          |
|    ep_len_mean        | 23.1     |
|    ep_rew_mean        | 20.8     |
|    exploration_rate   | 0.05     |
| time/                 |          |
|    episodes           | 1844     |
|    fps                | 11       |
|    time_elapsed       | 2676     |
|    total_timesteps    | 29883    |
| train/                |          |
|    learning_rate      | 0.005    |
|    loss               | 0.146    |
|    n_updates          | 28880    |
----------------------------------
----------------------------------
| rollout/              |          |
|    ep_len_mean        | 22.9     |
|    ep_rew_mean        | 20.6     |
|    exploration_rate   | 0.05     |
| time/                 |          |
|    episodes           | 1848     |
|    fps                | 11       |
|    time_elapsed       | 2685     |
|    total_timesteps    | 29987    |
| train/                |          |
|    learning_rate      | 0.005    |
|    loss               | 0.191    |
|    n_updates          | 28984    |
----------------------------------
<stable_baselines3.dqn.dqn.DQN at 0x7eea175b7e20>
```

Image (Car crashed in 1 episode and ran successfully in 2 episodes):
Travel Speed of cars was relatively fast.



Graphs:

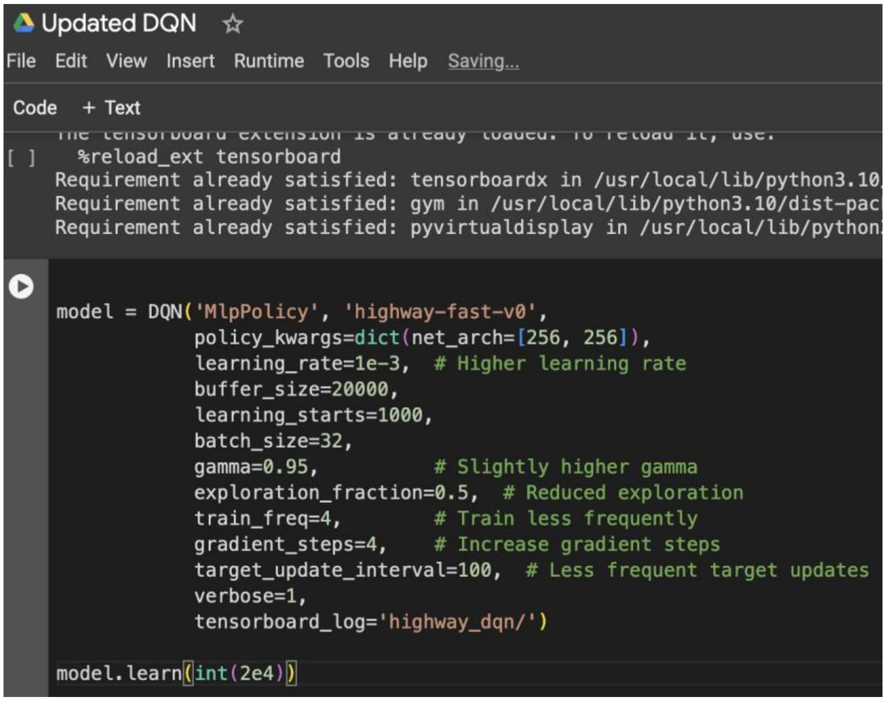


Algorithm 4: (DQN with my own hyperparameter values for optimal run period of 20+ seconds)

Survived 20+ seconds? YES

Hyperparameter	Description	Value
Network Architecture	Number of hidden layers	2
Neuron Count	Number of neurons in each hidden layer	[256, 256]
Learning rate	Rate of parameter updates	1e-3
Buffer size	Size of experience replay buffer	20000
Learning starts	Number of steps before learning starts	1000
Batch size	Number of experiences per training iteration	32
Gamma	Discount factor for future rewards	0.95
Train frequency	Frequency of model training	4
Gradient steps	Number of optimization steps after each batch	4
Target update interval	Frequency of updating target network	100
Exploration fraction	Fraction of exploration during training	0.5
Verbose	Level of output during training	1
Training steps	Total steps for training	20000

Model:

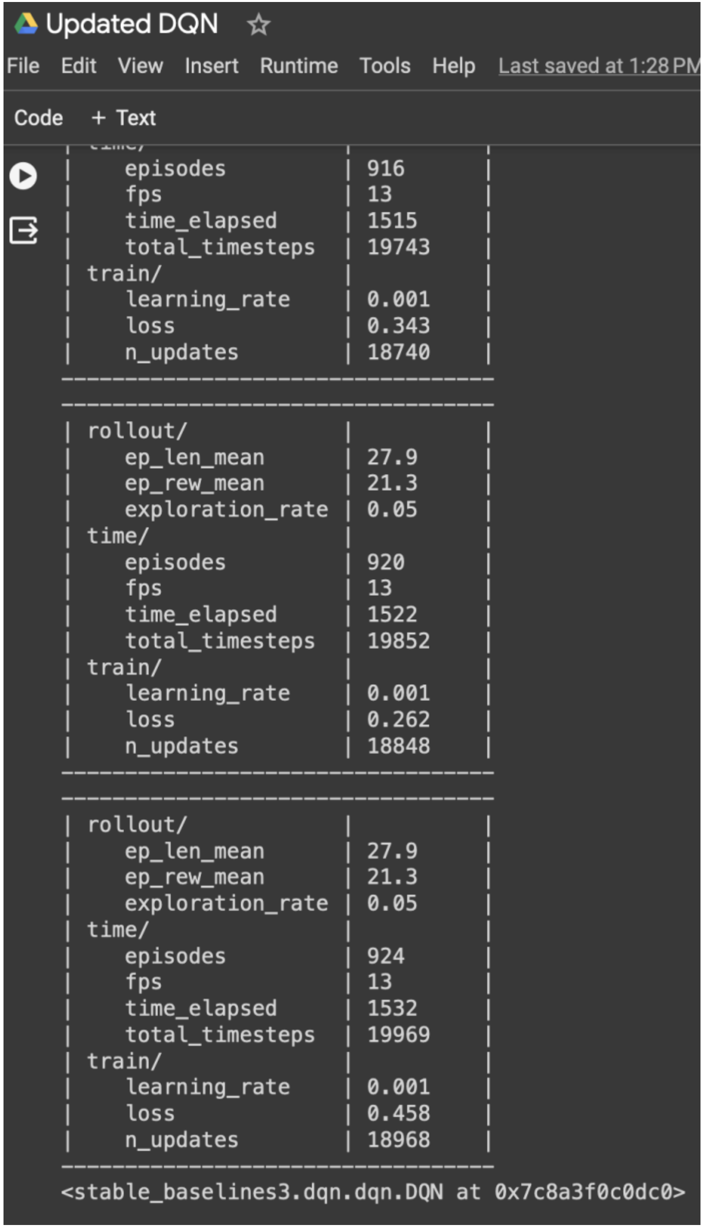


```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Requirement already satisfied: tensorboardx in /usr/local/lib/python3.10
Requirement already satisfied: gym in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: pyvirtualdisplay in /usr/local/lib/python
```

```
model = DQN('MlpPolicy', 'highway-fast-v0',
            policy_kwargs=dict(net_arch=[256, 256]),
            learning_rate=1e-3,  # Higher learning rate
            buffer_size=20000,
            learning_starts=1000,
            batch_size=32,
            gamma=0.95,          # Slightly higher gamma
            exploration_fraction=0.5,  # Reduced exploration
            train_freq=4,        # Train less frequently
            gradient_steps=4,    # Increase gradient steps
            target_update_interval=100,  # Less frequent target updates
            verbose=1,
            tensorboard_log='highway_dqn/')

model.learn(int(2e4))
```

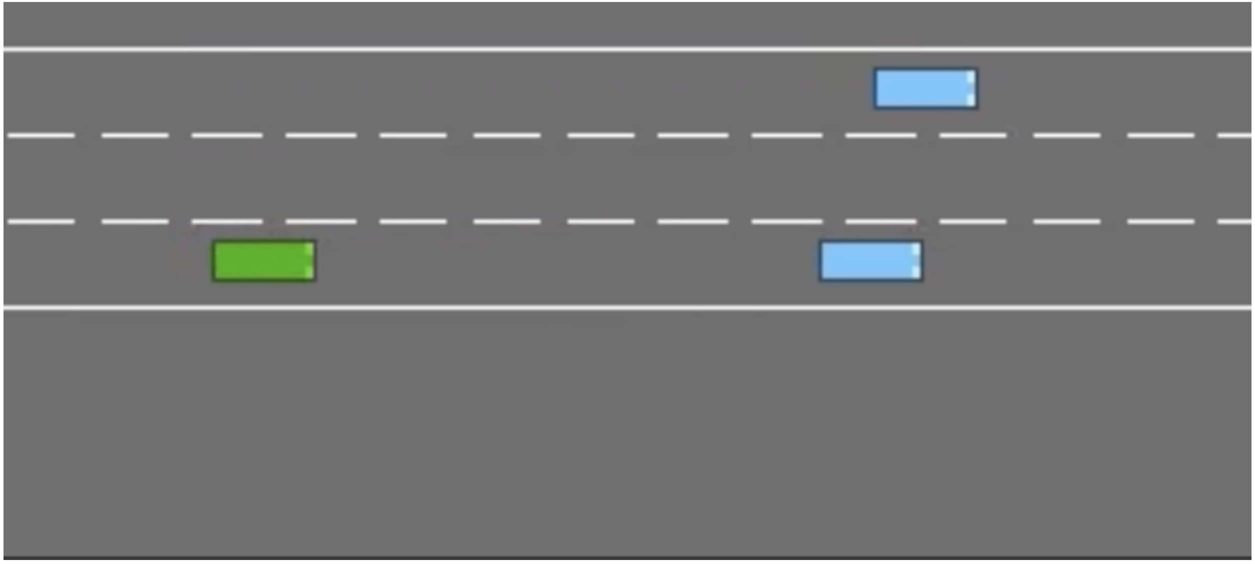
Final Stats of Training:


Updated DQN
File Edit View Insert Runtime Tools Help Last saved at 1:28 PM
Code + Text

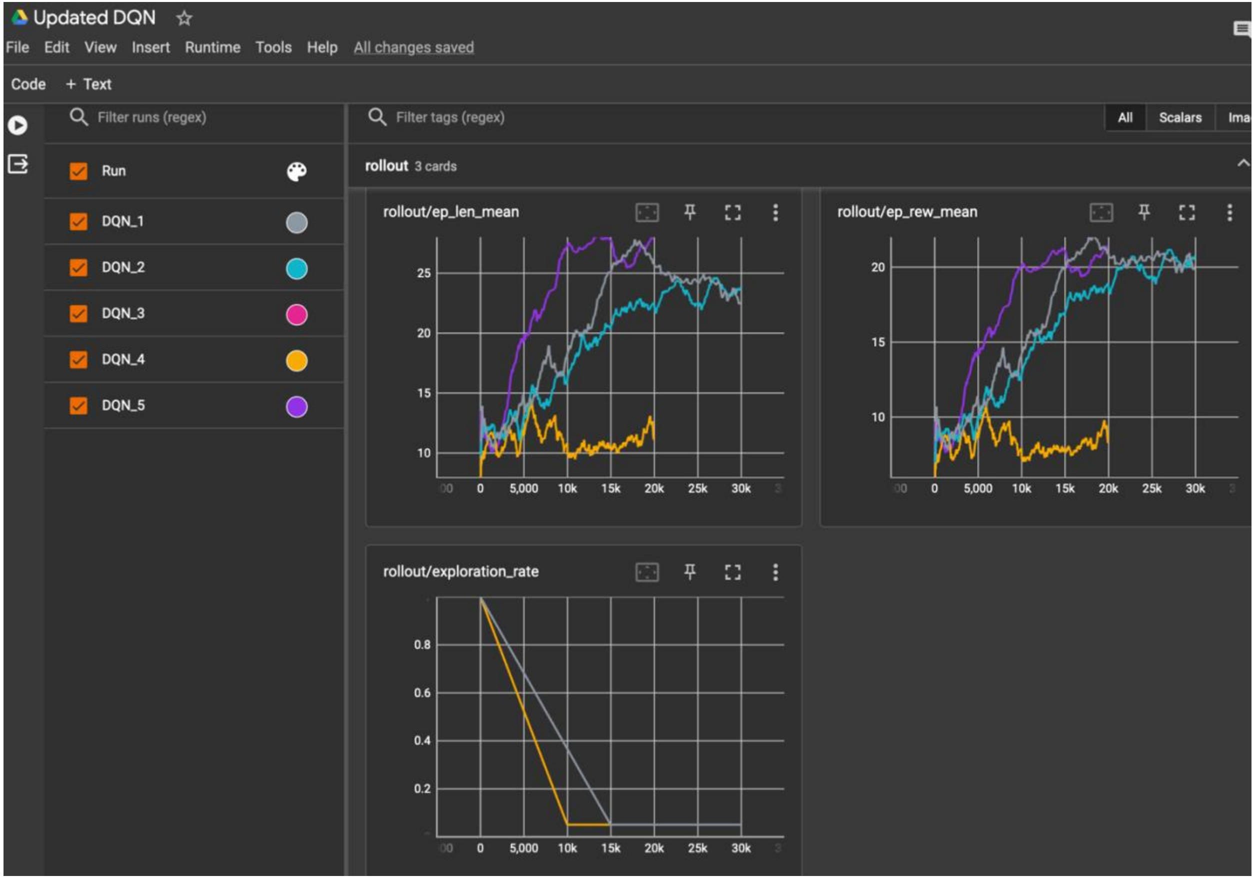
```
| time/                |           |
|    episodes          | 916       |
|    fps               | 13        |
|    time_elapsed      | 1515      |
|    total_timesteps   | 19743     |
| train/               |           |
|    learning_rate     | 0.001     |
|    loss              | 0.343     |
|    n_updates         | 18740     |
----------------------------------
----------------------------------
| rollout/             |           |
|    ep_len_mean       | 27.9      |
|    ep_rew_mean       | 21.3      |
|    exploration_rate  | 0.05      |
| time/                |           |
|    episodes          | 920       |
|    fps               | 13        |
|    time_elapsed      | 1522      |
|    total_timesteps   | 19852     |
| train/               |           |
|    learning_rate     | 0.001     |
|    loss              | 0.262     |
|    n_updates         | 18848     |
----------------------------------
----------------------------------
| rollout/             |           |
|    ep_len_mean       | 27.9      |
|    ep_rew_mean       | 21.3      |
|    exploration_rate  | 0.05      |
| time/                |           |
|    episodes          | 924       |
|    fps               | 13        |
|    time_elapsed      | 1532      |
|    total_timesteps   | 19969     |
| train/               |           |
|    learning_rate     | 0.001     |
|    loss              | 0.458     |
|    n_updates         | 18968     |
----------------------------------
<stable_baselines3.dqn.dqn.DQN at 0x7c8a3f0c0dc0>
```

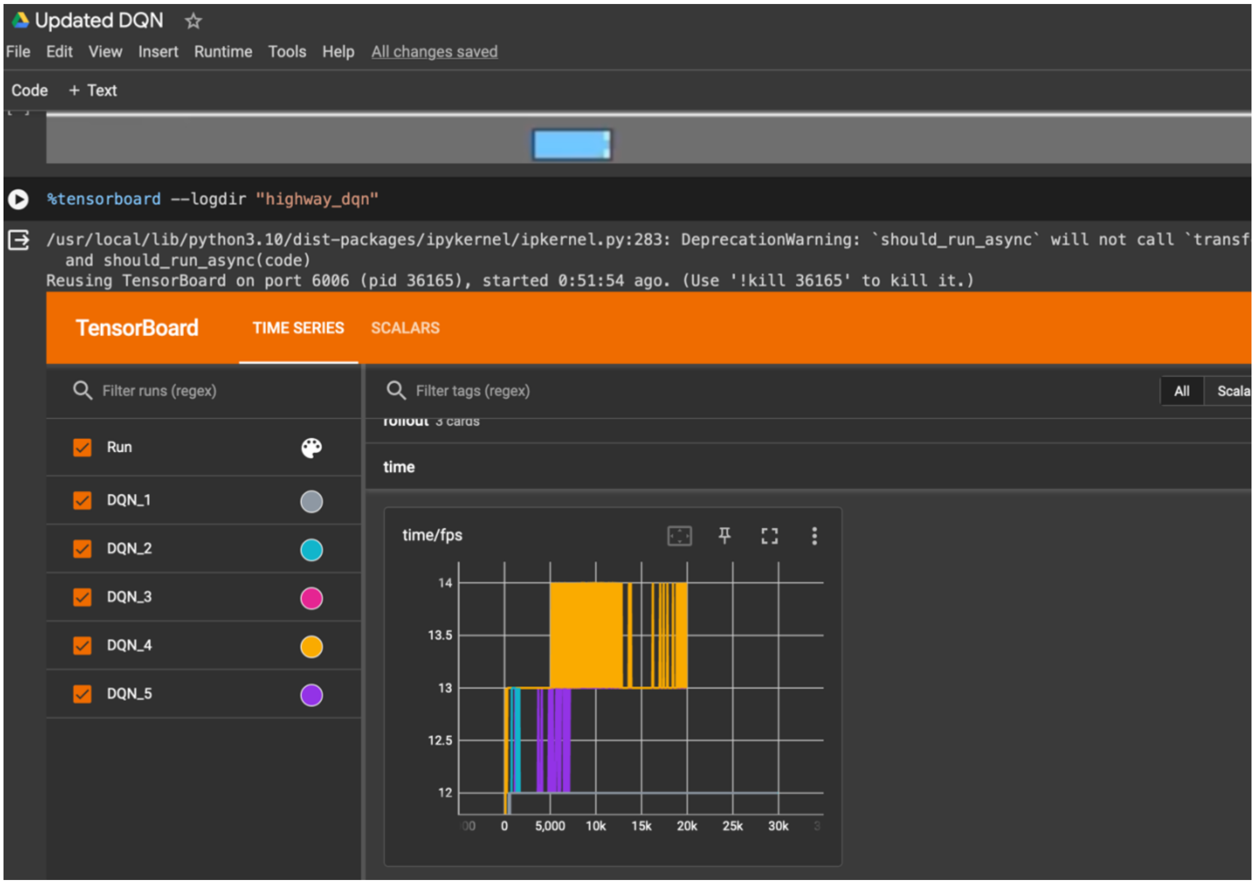
Image:

No signs of crashing in all episodes.



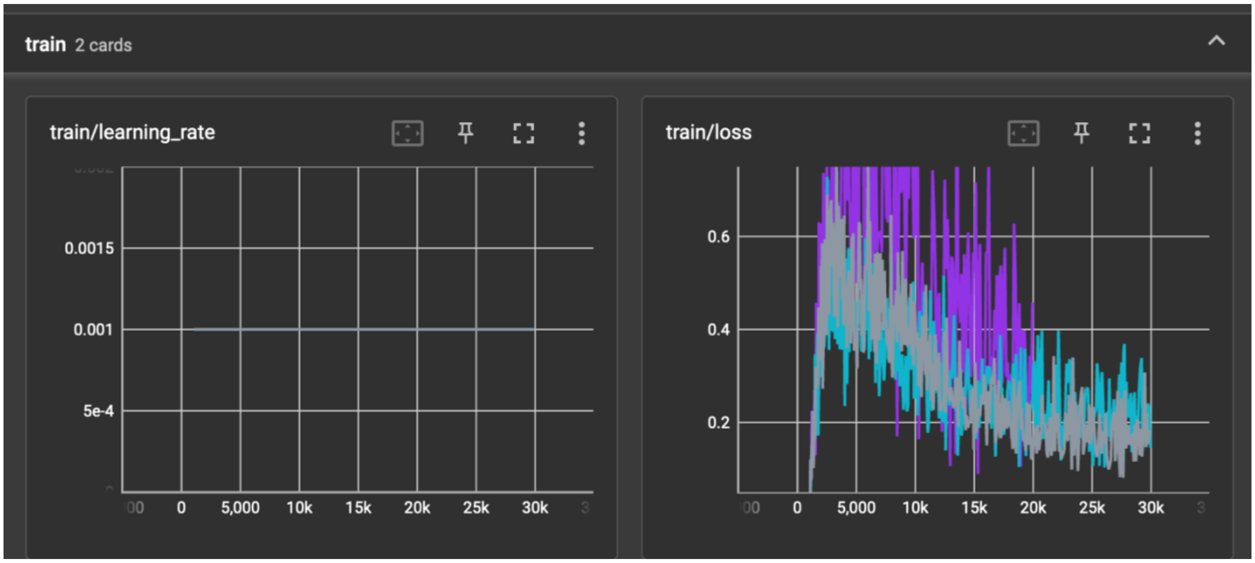
Graphs:





```
%tensorboard --logdir "highway_dqn"
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transf
  and should_run_async(code)
Reusing TensorBoard on port 6006 (pid 36165), started 0:51:54 ago. (Use '!kill 36165' to kill it.)
```

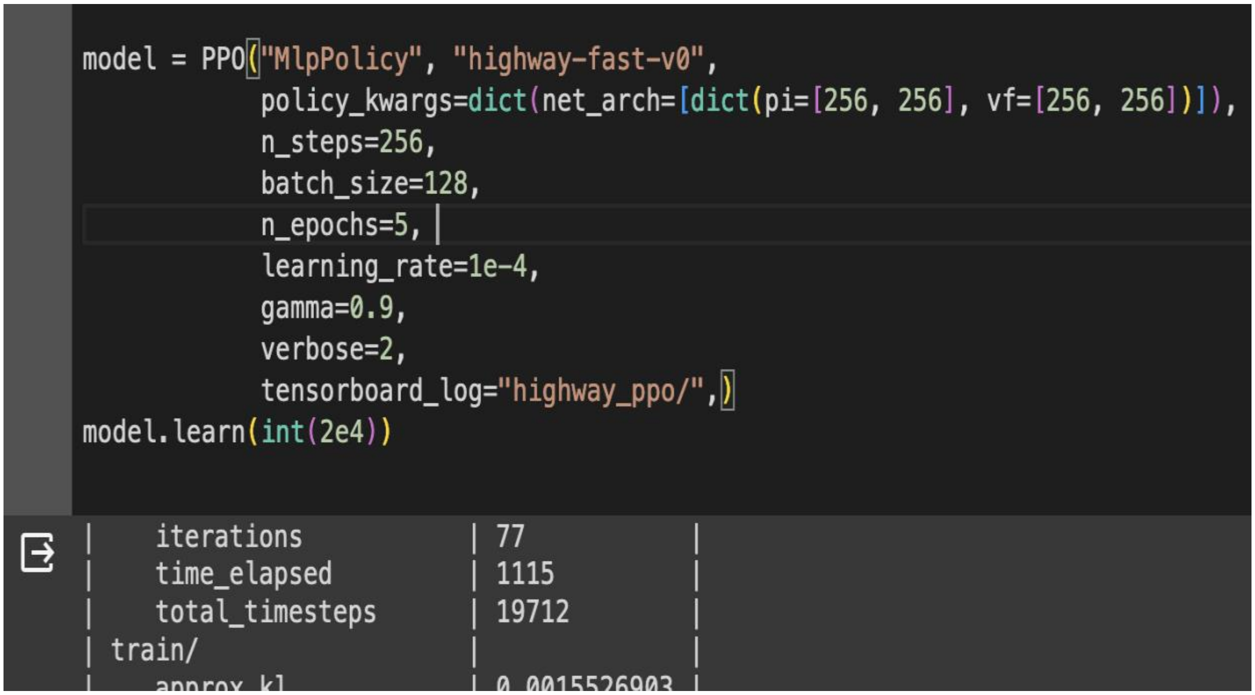


Algorithm 5: PPO (Proximal Policy Optimization Algorithm)

Survived 20+ seconds? YES

Hyperparameter	Description	Value
Network Architecture	Number of hidden layers	4
Neuron Count	Number of neurons in each hidden layer	[256, 256, 256, 256]
Batch size	Number of steps to optimize at each iteration	128
n_epochs	Number of optimization epochs	5
Learning rate	Rate of learning for the optimizer	1e-4
Gamma	Discount factor for future rewards	0.9
Verbose	Level of output during training	2
Training steps	Total steps for training	20000

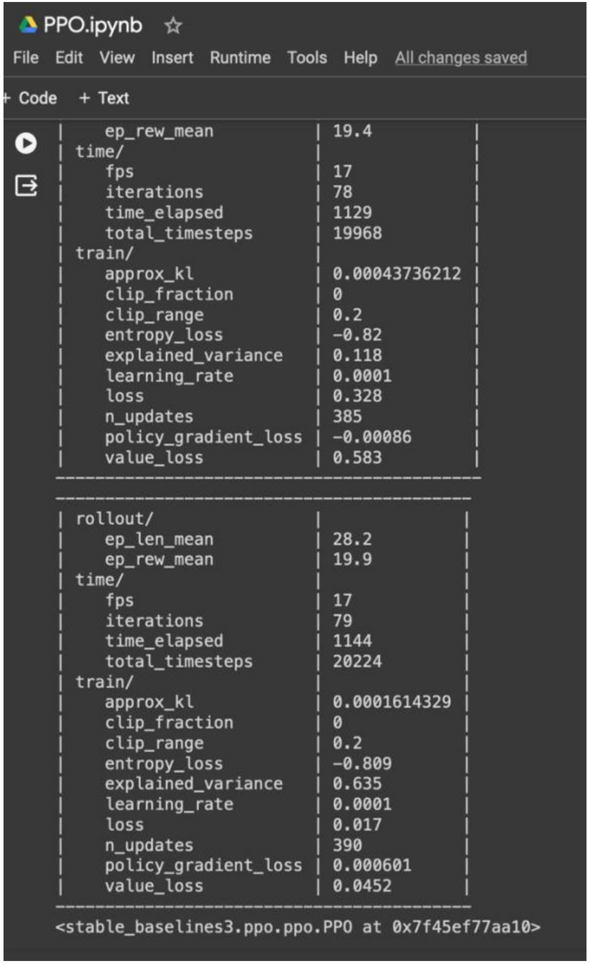
Model:



```
model = PPO("MlpPolicy", "highway-fast-v0",
             policy_kwargs=dict(net_arch=[dict(pi=[256, 256], vf=[256, 256])]),
             n_steps=256,
             batch_size=128,
             n_epochs=5,
             learning_rate=1e-4,
             gamma=0.9,
             verbose=2,
             tensorboard_log="highway_ppo/",)
model.learn(int(2e4))
```

```
|    iterations           | 77           |
|    time_elapsed         | 1115         |
|    total_timesteps      | 19712        |
| train/                  |              |
|    approx_kl            | 0.0015526903 |
```

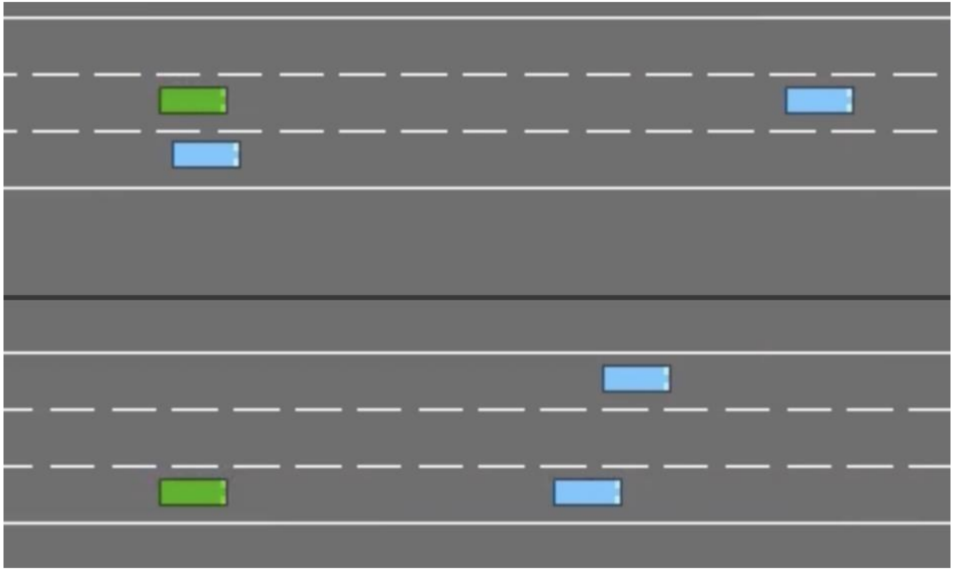
Final Training Stats:



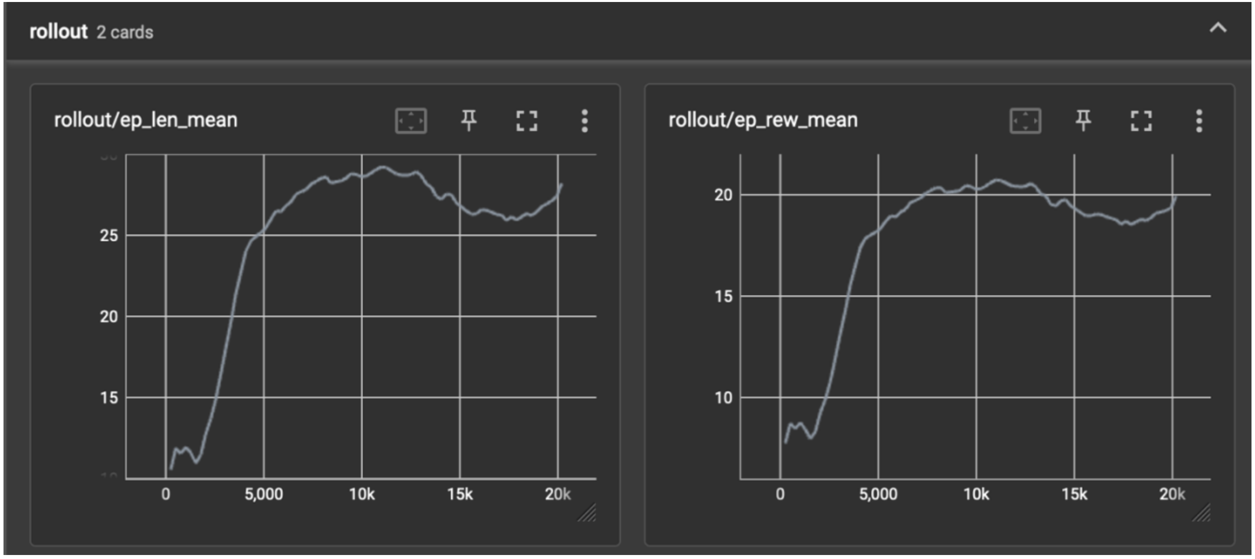
```
|    ep_rew_mean          | 19.4          |
| time/                   |               |
|    fps                  | 17            |
|    iterations           | 78            |
|    time_elapsed         | 1129          |
|    total_timesteps      | 19968         |
| train/                  |               |
|    approx_kl            | 0.00043736212 |
|    clip_fraction        | 0             |
|    clip_range           | 0.2           |
|    entropy_loss         | -0.82         |
|    explained_variance   | 0.118         |
|    learning_rate        | 0.0001        |
|    loss                 | 0.328         |
|    n_updates            | 385           |
|    policy_gradient_loss | -0.00086      |
|    value_loss           | 0.583         |
-------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 28.2         |
|    ep_rew_mean          | 19.9         |
| time/                   |              |
|    fps                  | 17           |
|    iterations           | 79           |
|    time_elapsed         | 1144         |
|    total_timesteps      | 20224        |
| train/                  |              |
|    approx_kl            | 0.0001614329 |
|    clip_fraction        | 0            |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.809       |
|    explained_variance   | 0.635        |
|    learning_rate        | 0.0001       |
|    loss                 | 0.017        |
|    n_updates            | 390          |
|    policy_gradient_loss | 0.000601     |
|    value_loss           | 0.0452       |
------------------------------------------
<stable_baselines3.ppo.ppo.PPO at 0x7f45ef77aa10>
```

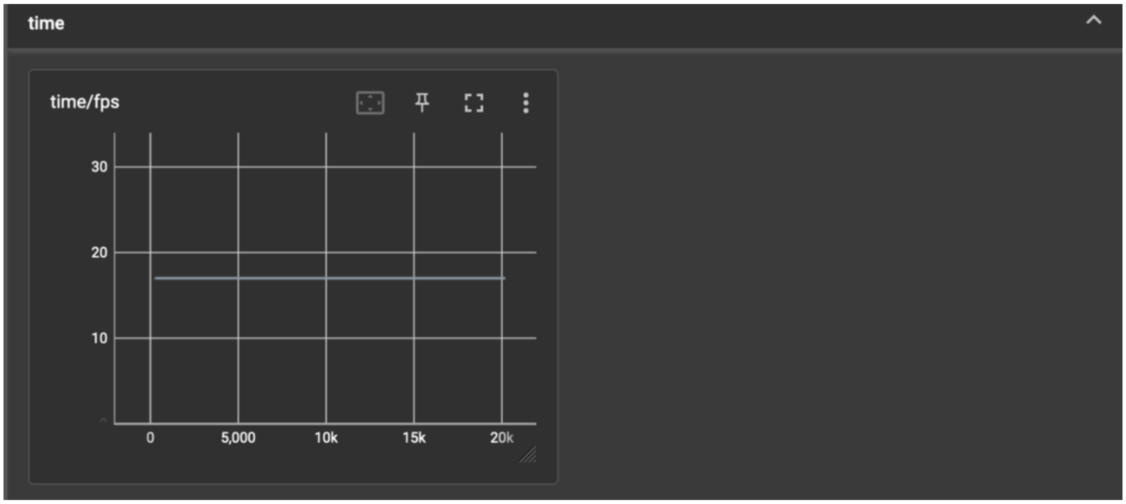
Images:

No signs of crashing in all 4 episodes



Graphs:


rollout 2 cards
rollout/ep_len_mean
25
20
15
0
5,000
10k
15k
20k
rollout/ep_rew_mean
20
15
10
0
5,000
10k
15k
20k


time
time/fps
30
20
10
0
5,000
10k
15k
20k