HW3

Problem 1

i. ***Proof***

$$
\begin{aligned}
L_{\pi_{\theta_1}}(\pi_{\theta_1}) &= \eta(\pi_{\theta_1}) + \sum_{s\in\mathcal{S}} d_\mu^{\pi_{\theta_1}}(s) \sum_{a\in\mathcal{A}} \pi_{\theta_1}(s|a) A^{\pi_{\theta_1}}(s,a) \\
&= \eta(\pi_{\theta_1}) + \sum_{s\in\mathcal{S}} d_\mu^{\pi_{\theta_1}}(s) \sum_{a\in\mathcal{A}} \pi_{\theta_1}(s|a)(Q^{\pi_{\theta_1}}(s,a) - V^{\pi_{\theta_1}}(s)) \\
&= \eta(\pi_{\theta_1}) + \sum_{s\in\mathcal{S}} d_\mu^{\pi_{\theta_1}}(s) \Big(\sum_{a\in\mathcal{A}} \pi_{\theta_1}(s|a) Q^{\pi_{\theta_1}}(s,a) - V^{\pi_{\theta_1}}(s) \Big) \\
&= \eta(\pi_{\theta_1}) + \sum_{s\in\mathcal{S}} d_\mu^{\pi_{\theta_1}}(s) \cdot 0
\end{aligned}
$$

ii. To calculate the gradient of $L_{\pi_{\theta_1}}(\pi_\theta)$, differentiate the above expression with respect to θ:

$$
\nabla_\theta L_{\pi_{\theta_1}}(\pi_\theta) = \nabla_\theta \left(\eta(\pi_{\theta_1})\right) + \nabla_\theta \left(\sum_{s\in S} d^{\pi_{\theta_1}}(s) \sum_{a\in A} \pi_\theta(a|s) A^{\pi_{\theta_1}}(s,a) \right)
$$

Notice that:

- The term $\eta(\pi_{\theta_1})$ does not depend on θ since it is evaluated at θ_1, so $\nabla_\theta \eta(\pi_{\theta_1}) = 0$.
- The gradient of the second term needs to be evaluated. Here, we differentiate $\pi_\theta(a|s)$ with respect to θ.

Using the policy gradient theorem:

$$
\nabla_\theta \left(\sum_{s\in S} d^{\pi_{\theta_1}}(s) \sum_{a\in A} \pi_\theta(a|s) A^{\pi_{\theta_1}}(s,a) \right) = \sum_{s\in S} d^{\pi_{\theta_1}}(s) \sum_{a\in A} \nabla_\theta \pi_\theta(a|s) A^{\pi_{\theta_1}}(s,a)
$$

When $\theta = \theta_1$, the expression simplifies because the expectation over the stationary distribution $d^{\pi_{\theta_1}}$ of the advantage function $A^{\pi_{\theta_1}}$ weighted by $\nabla_\theta \pi_\theta(a|s)$ matches the policy gradient formula for the expected return $\eta(\pi_\theta)$:

$$
\nabla_\theta \eta(\pi_\theta)\big|\theta = \theta_1 = \sum_{s\in S} d^{\pi_{\theta_1}}(s) \sum_{a\in A} \nabla_\theta \pi_\theta(a|s) A^{\pi_{\theta_1}}(s,a)\big|_{\theta=\theta_1}
$$

Therefore:

$$
\nabla_\theta L_{\pi_{\theta_1}}(\pi_\theta)\big|\theta = \theta_1 = \nabla\theta \eta(\pi_\theta)\big|_{\theta=\theta_1}
$$

Problem 2

a. The Lagrangian $\mathcal{L}(\theta, \lambda)$ for the constrained optimization problem is given by:

$$
\mathcal{L}(\theta,\lambda) = -(\nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k})^T(\theta - \theta_k) + \lambda \left(\frac{1}{2}(\theta-\theta_k)^T H_{\theta_k}(\theta-\theta_k) - \delta \right)
$$

Setting $\frac{\partial \mathcal{L}}{\partial \theta} = 0$ leads to:

$$
-\nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k} + \lambda H_{\theta_k}(\theta - \theta_k) = 0
$$

Then we can get θ:

$$
\theta = \theta_k + \frac{1}{\lambda} H_{\theta_k}^{-1} \nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k}
$$

Plug θ back into $\mathcal{L}(\theta,\lambda)$:

$$
\begin{aligned}
\mathcal{L}(\theta,\lambda) &= -(\nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k})^T(\frac{1}{\lambda} H_{\theta_k}^{-1} \nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k}) + \lambda \left(\frac{1}{2}(\frac{1}{\lambda} H_{\theta_k}^{-1} \nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k})^T H_{\theta_k} (\frac{1}{\lambda} H_{\theta_k}^{-1} \nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k}) - \delta \right) \\
&= -\frac{1}{\lambda} \left(\nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k}\right)^T H_{\theta_k}^{-1} \left(\nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k}\right) + \frac{1}{2\lambda} \left(\nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k}\right)^T H_{\theta_k}^{-1} \left(\nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k}\right) - \lambda\delta \\
&= -\frac{1}{2\lambda} \left(\nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k}\right)^T H_{\theta_k}^{-1} \left(\nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k}\right) - \lambda\delta
\end{aligned}
$$

Hence, $D(\lambda)$ is as provided. As for λ^*:

$$
\frac{d}{d\lambda} D(\lambda) = \frac{1}{2\lambda^2} \left(\nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k}\right)^T H_{\theta_k}^{-1} \left(\nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k}\right) - \delta = 0
$$

$$
\lambda^* = \sqrt{\frac{\left(\nabla\theta L_{\theta_k}(\theta)|_{\theta=\theta_k}\right)^T H_{\theta_k}^{-1} \left(\nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k}\right)}{2\delta}}
$$

b. In part a, we already prove that $\theta^* = \theta_k + \frac{1}{\lambda} H_{\theta_k}^{-1} \nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k}$ if $\theta^* = \arg\min_\theta \mathcal{L}(\theta, \lambda)$, then we can get α:

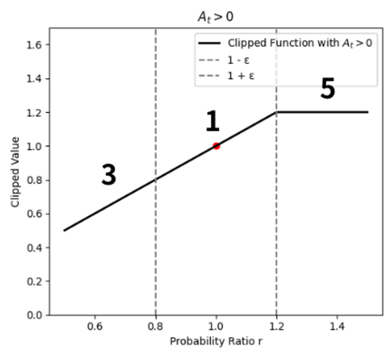
$$\alpha = \frac{1}{\lambda^*} = \sqrt{\frac{2\delta}{(\nabla\theta L_{\theta_k}(\theta)|_{\theta=\theta_k})^T H_{\theta_k}^{-1} (\nabla_\theta L_{\theta_k}(\theta)|_{\theta=\theta_k})}}$$

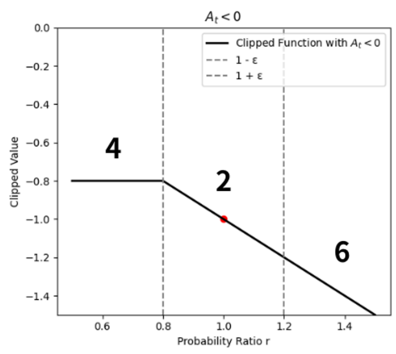
Problem 3

Original PPO-Clip

$$L_{s,a}^{\text{clip}}(\theta;\theta_k) := \min\left\{\frac{\pi_\theta(a\mid s)}{\pi_{\theta_k}(a\mid s)} A^{\theta_k}(s,a), \text{clip}\left(\frac{\pi_\theta(a\mid s)}{\pi_{\theta_k}(a\mid s)}, 1-\epsilon, 1+\epsilon\right) A^{\theta_k}(s,a)\right\}$$

$p_t(\theta) > 0$	A_t	Return Value of *min*	Objective is Clipped	Sign of Objective	Gradient
$p_t(\theta) \in [1-\epsilon, 1+\epsilon]$	$+$	$p_t(\theta)A_t$	no	$+$	✓
$p_t(\theta) \in [1-\epsilon, 1+\epsilon]$	$-$	$p_t(\theta)A_t$	no	$-$	✓
$p_t(\theta) < 1-\epsilon$	$+$	$p_t(\theta)A_t$	**no**	$+$	✓
$p_t(\theta) < 1-\epsilon$	$-$	$(1-\epsilon)A_t$	yes	$-$	0
$p_t(\theta) > 1+\epsilon$	$+$	$(1+\epsilon)A_t$	yes	$+$	0
$p_t(\theta) > 1+\epsilon$	$-$	$p_t(\theta)A_t$	**no**	$-$	✓

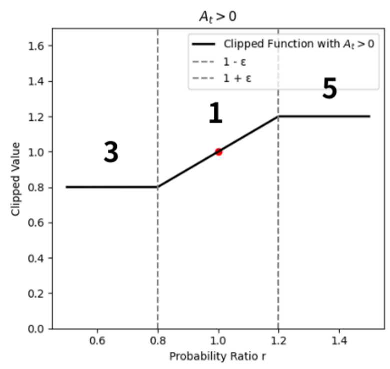


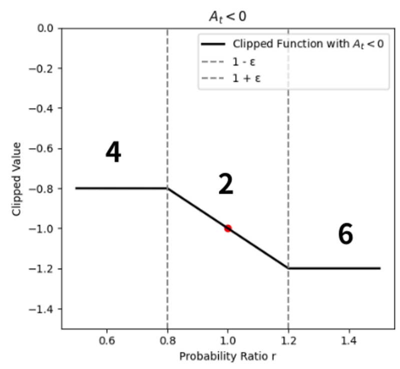


Variant PPO-Clip

$$\tilde{L}_{s,a}^{\text{clip}}(\theta;\theta_k) = \text{clip}\left(\frac{\pi_\theta(a\mid s)}{\pi_{\theta_k}(a\mid s)}, 1-\epsilon, 1+\epsilon\right) A^{\theta_k}(s,a)$$

$p_t(\theta) > 0$	A_t	Return Value of *min*	Objective is Clipped	Sign of Objective
$p_t(\theta) \in [1-\epsilon, 1+\epsilon]$	$+$	$p_t(\theta)A_t$	no	$+$
$p_t(\theta) \in [1-\epsilon, 1+\epsilon]$	$-$	$p_t(\theta)A_t$	no	$-$
$p_t(\theta) < 1-\epsilon$	$+$	$(1-\epsilon)A_t$	**yes**	$+$
$p_t(\theta) < 1-\epsilon$	$-$	$(1-\epsilon)A_t$	yes	$-$
$p_t(\theta) > 1+\epsilon$	$+$	$(1+\epsilon)A_t$	yes	$+$
$p_t(\theta) > 1+\epsilon$	$-$	$(1+\epsilon)A_t$	**yes**	$-$





The purpose of applying the clipping mechanism is to prevent the policy from updating too significantly in the correct direction. Therefore, there is no need to clip when the updates move in the wrong direction. Taking the minimum of the original value ensures this; thus, the function will continuously decrease.

Problem 4

a. Pendulum-v1

- **NN architecture**
  - **Actor**

```
nn.Sequential(
        nn.Linear(num_inputs, 40),
        nn.ReLU(),
        nn.Linear(400, 300),
        nn.ReLU(),
        nn.Linear(300, num_outputs),
        nn.Tanh()
)
```

  - **Critic**

```
nn.Sequential(
    nn.Linear(num_inputs, 400),
    nn.ReLU()
)
nn.Sequential(
    nn.Linear(400 + num_outputs, 300),
    nn.ReLU(),
    nn.Linear(300, 1)
)
```

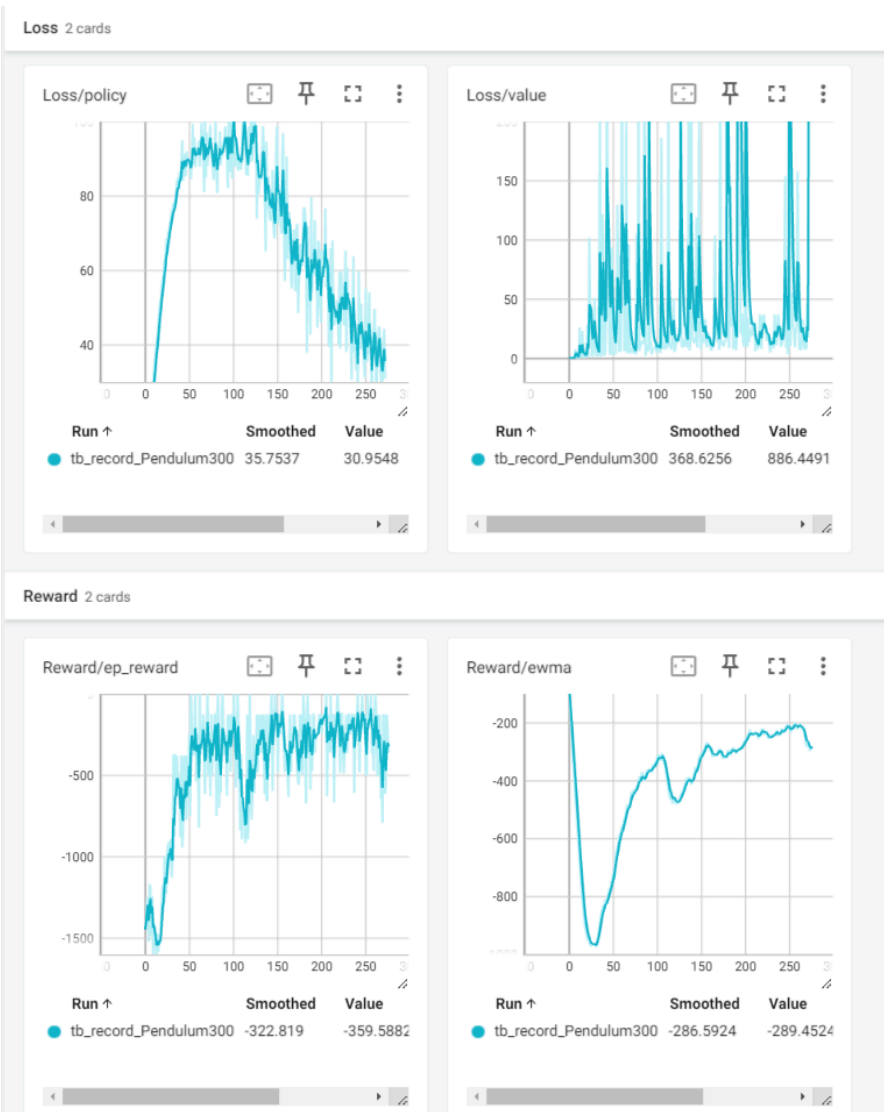
- **Training Result**

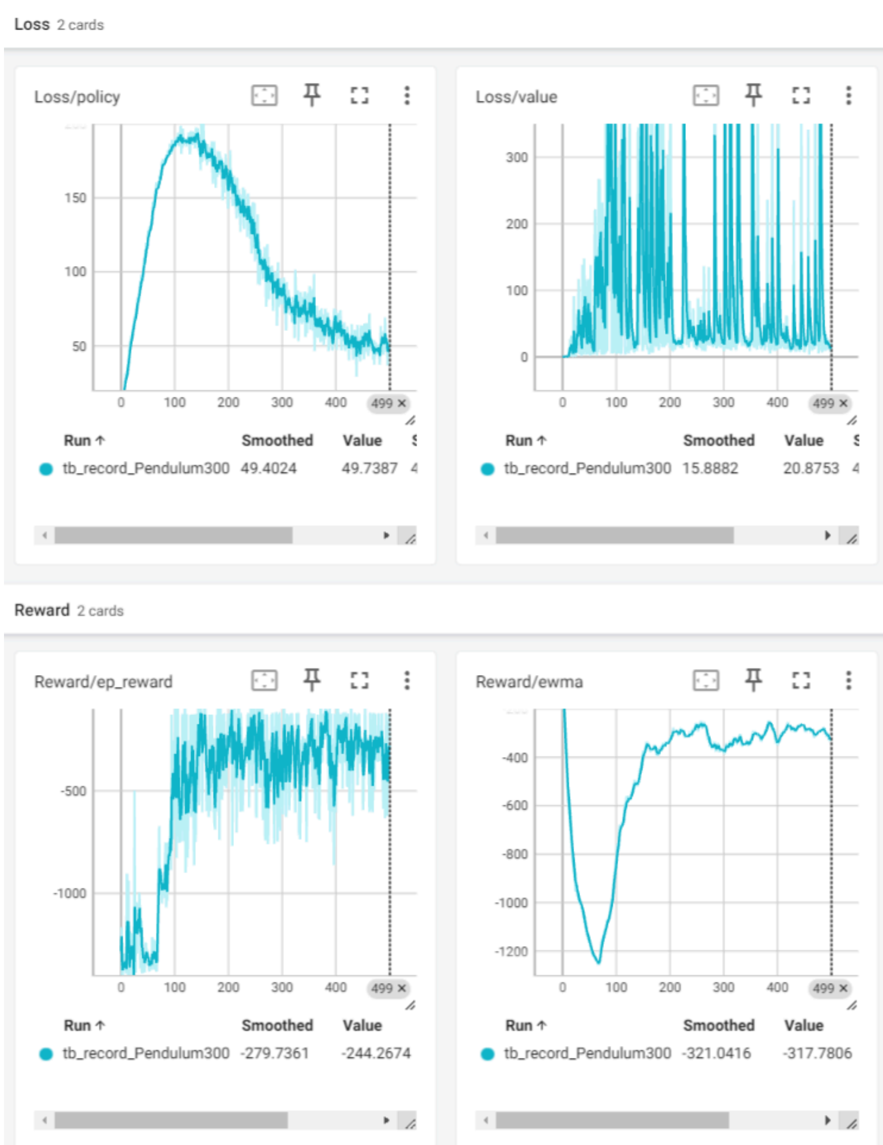
Hyperparameter

Hyperparameter

```
num_episodes = 300
gamma = 0.995
tau = 0.002
hidden_size = 128
noise_scale = 0.1
replay_size = 100000
batch_size = 128
updates_per_step = 1
```

```
num_episodes = 500
gamma = 0.995
tau = 0.002
hidden_size = 128
noise_scale = 0.3
replay_size = 100000
batch_size = 128
updates_per_step = 1
```






```
(py38) chueat@ChuEating-PC:/mnt/c/Users/ChuEating/Reinforcement-Learning/HW3$ python ddpg.py
/home/chueat/anaconda3/envs/py38/lib/python3.8/site-packages/pygame/pkgdata.py:25: DeprecationWarning: pkg_resources is depr
ecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html
  from pkg_resources import resource_stream, resource_exists
/home/chueat/anaconda3/envs/py38/lib/python3.8/site-packages/pkg_resources/__init__.py:2832: DeprecationWarning: Deprecated
call to `pkg_resources.declare_namespace('sphinxcontrib')`.
Implementing implicit namespace packages (as specified in PEP 420) is preferred to `pkg_resources.declare_namespace`. See ht
tps://setuptools.pypa.io/en/latest/references/keywords.html#keyword-namespace-packages
  declare_namespace(pkg)
/home/chueat/anaconda3/envs/py38/lib/python3.8/site-packages/gym/core.py:172: DeprecationWarning: WARN: Function `env.seed(s
eed)` is marked as deprecated and will be removed in the future. Please use `env.reset(seed=seed) instead.
  deprecation(
ed=seed) instead.
  deprecation(
Loading models from ./preTrained/ddpg_actor_Pendulum-v1_05082024_024340_DDPG and ./preTrained/ddpg_critic_Pendulum-v1_050820
24_024340_DDPG
ddpg.py:364: UserWarning: Creating a tensor from a list of numpy.ndarrays is extremely slow. Please consider converting the
list to a single numpy.ndarray with numpy.array() before converting to a tensor. (Triggered internally at /opt/conda/conda-b
ld/pytorch_1712608851799/work/torch/csrc/utils/tensor_new.cpp:274.)
  state = torch.Tensor([env.reset()])
Episode: 1, reward: -126.13
Episode: 2, reward: -480.78
Episode: 3, reward: -1.05
Episode: 4, reward: -127.63
Episode: 5, reward: -246.94
Episode: 6, reward: -243.06
Episode: 7, reward: -128.36
Episode: 8, reward: -121.33
Episode: 9, reward: -1.17
Episode: 10, reward: -127.89
(py38) chueat@ChuEating-PC:/mnt/c/Users/ChuEating/Reinforcement-Learning/HW3$
```

```
(py38) chueat@ChuEating-PC:/mnt/c/Users/ChuEating/Reinforcement-Learning/HW3$ python ddpg.py
/home/chueat/anaconda3/envs/py38/lib/python3.8/site-packages/pygame/pkgdata.py:25: DeprecationWarning: pkg_resources
is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html
  from pkg_resources import resource_stream, resource_exists
/home/chueat/anaconda3/envs/py38/lib/python3.8/site-packages/pkg_resources/__init__.py:2832: DeprecationWarning: Depr
ecated call to `pkg_resources.declare_namespace('sphinxcontrib')`.
Implementing implicit namespace packages (as specified in PEP 420) is preferred to `pkg_resources.declare_namespace`.
 See https://setuptools.pypa.io/en/latest/references/keywords.html#keyword-namespace-packages
  declare_namespace(pkg)
/home/chueat/anaconda3/envs/py38/lib/python3.8/site-packages/gym/core.py:172: DeprecationWarning: WARN: Function `env
.seed(seed)` is marked as deprecated and will be removed in the future. Please use `env.reset(seed=seed) instead.
  deprecation(
Loading models from ./preTrained/ddpg_actor_Pendulum-v1_05082024_030212_DDPG and ./preTrained/ddpg_critic_Pendulum-v1
_05082024_030212_DDPG
ddpg.py:364: UserWarning: Creating a tensor from a list of numpy.ndarrays is extremely slow. Please consider converti
ng the list to a single numpy.ndarray with numpy.array() before converting to a tensor. (Triggered internally at /opt
/conda/conda-bld/pytorch_1712608851799/work/torch/csrc/utils/tensor_new.cpp:274.)
  state = torch.Tensor([env.reset()])
Episode: 1, reward: -244.46
Episode: 2, reward: -713.17
Episode: 3, reward: -585.80
Episode: 4, reward: -367.25
Episode: 5, reward: -485.90
Episode: 6, reward: -120.35
Episode: 7, reward: -125.74
Episode: 8, reward: -121.19
Episode: 9, reward: -498.69
Episode: 10, reward: -121.41
```

- **Discussion**

  1. **Loss/Policy**:

     This graph shows the loss associated with the actor network over the course of training episodes. The loss peaks around episode 100 and then generally trends downward, indicating that the actor network is gradually learning a strategy that minimizes error in its predictions of actions.

  2. **Loss/Value**:

     This chart shows the loss of the critic network. The frequent spikes in loss indicate periods where the network's predictions are significantly off from actual returns, but a general downward trend or stabilization would be expected as training progresses. However, the volatility here suggests that the value estimates might still be fluctuating significantly, which could be due to several factors like exploration strategies or parameter settings.

  3. **Reward/ep_reward**:

     The reward per episode chart shows the reward obtained by the agent each episode. It's clear from the chart that the reward dramatically increases at the beginning of training, indicating that the agent quickly learns a reasonably effective strategy. The reward stabilizes after an initial learning phase, with some fluctuations which could be due to exploration or environmental stochasticity.

  4. **Reward/ewma**:

The exponentially weighted moving average (EWMA) of the reward smooths out the episodic rewards to give a clearer picture of the overall trend. This curve, similarly to the episodic reward, shows significant improvement in the initial episodes and then levels off, indicating that the policy has likely converged to a near-optimal strategy for this environment. Comparing the two charts above, we can observe that even after an additional 200 training episodes, the EWMA (Exponential Weighted Moving Average) has not shown a significant increase. Therefore, it can be understood that an EWMA of approximately -200 might be the limit value for this environment.

  - Encountered issue: Initially, when running `ddpg.py` on a laptop, no matter how the code was modified, the policy could not converge. After transferring the same code to a PC, convergence was achieved. It is speculated that the issue was due to insufficient memory on the laptop, leading to a full replay buffer that could not learn anything new.

b. Pendulum-v3

- **NN architecture**
  - **Actor**

```
nn.Sequential(
        nn.Linear(num_inputs, 40),
        nn.ReLU(),
        nn.Linear(400, 300),
        nn.ReLU(),
        nn.Linear(300, num_outputs),
        nn.Tanh()
)
```

  - **Critic**

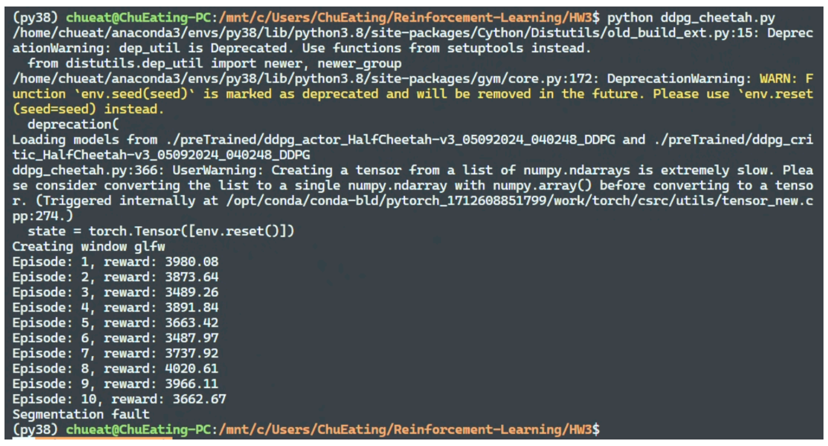
```
nn.Sequential(
    nn.Linear(num_inputs, 400),
    nn.ReLU()
)
nn.Sequential(
    nn.Linear(400 + num_outputs, 300),
    nn.ReLU(),
    nn.Linear(300, 1)
)
```

- **Hyperparameter**

```
gamma = 0.995
tau = 0.002
hidden_size = 128
noise_scale = 0.1
replay_size = 100000
batch_size = 128
updates_per_step = 1
```

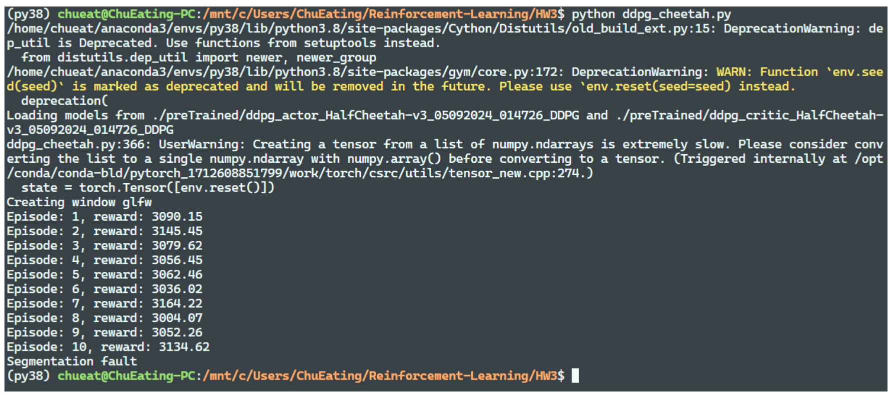
- **Training Result**

Training for 1000 episodes

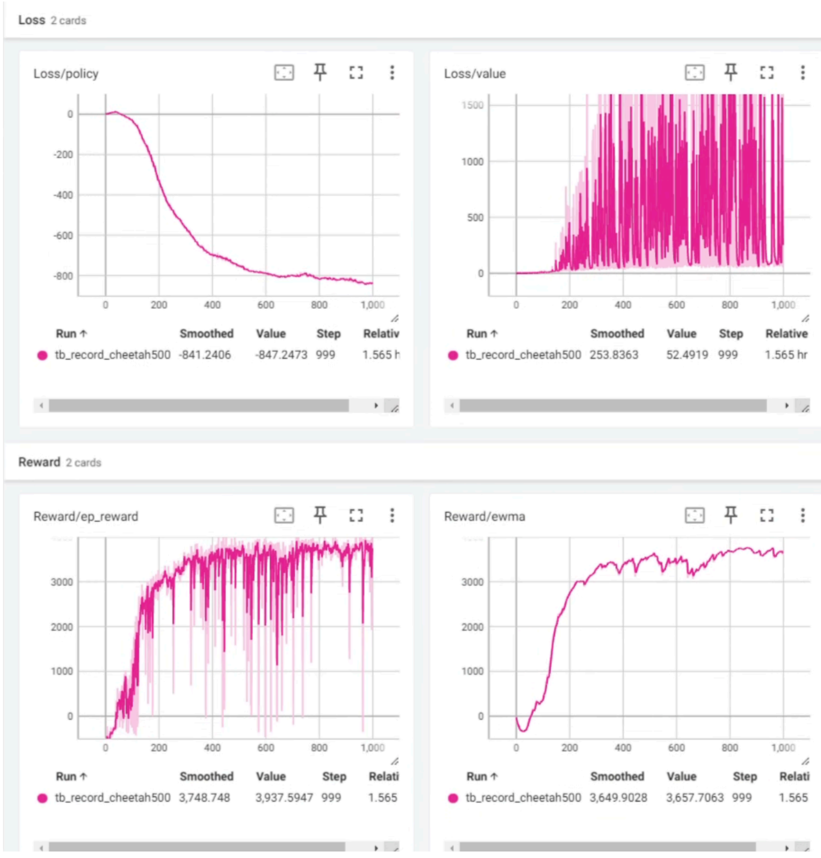


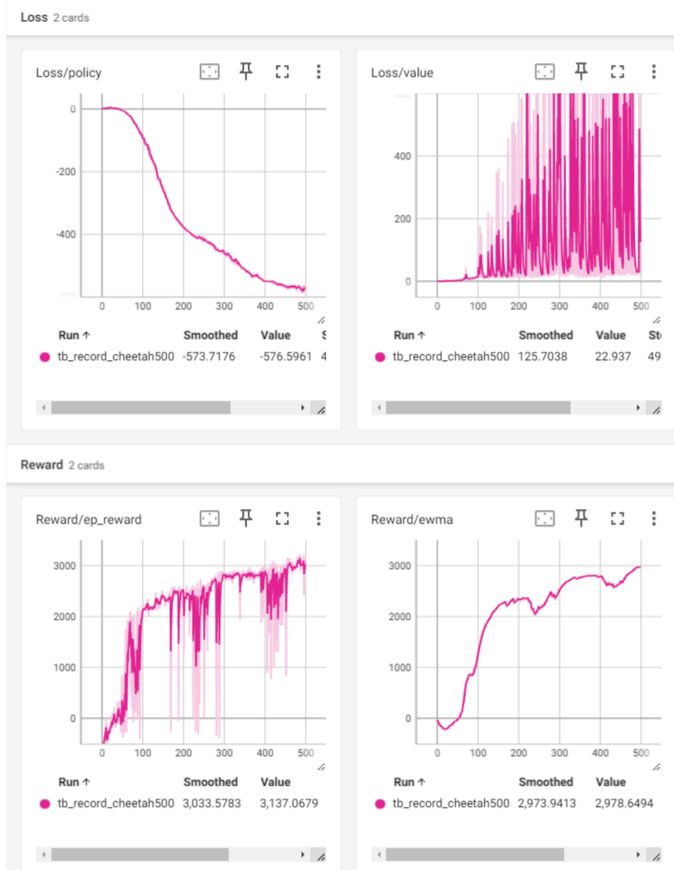
```
(py38) chueat@ChuEating-PC:/mnt/c/Users/ChuEating/Reinforcement-Learning/HW3$ python ddpg_cheetah.py
/home/chueat/anaconda3/envs/py38/lib/python3.8/site-packages/Cython/Distutils/old_build_ext.py:15: Deprec
ationWarning: dep_util is Deprecated. Use functions from setuptools instead.
  from distutils.dep_util import newer, newer_group
/home/chueat/anaconda3/envs/py38/lib/python3.8/site-packages/gym/core.py:172: DeprecationWarning: WARN: F
unction `env.seed(seed)` is marked as deprecated and will be removed in the future. Please use `env.reset
(seed=seed) instead.
  deprecation(
Loading models from ./preTrained/ddpg_actor_HalfCheetah-v3_05092024_040248_DDPG and ./preTrained/ddpg_cri
tic_HalfCheetah-v3_05092024_040248_DDPG
ddpg_cheetah.py:366: UserWarning: Creating a tensor from a list of numpy.ndarrays is extremely slow. Plea
se consider converting the list to a single numpy.ndarray with numpy.array() before converting to a tenso
r. (Triggered internally at /opt/conda/conda-bld/pytorch_1712608851799/work/torch/csrc/utils/tensor_new.c
pp:274.)
  state = torch.Tensor([env.reset()])
Creating window glfw
Episode: 1, reward: 3980.08
Episode: 2, reward: 3873.64
Episode: 3, reward: 3489.26
Episode: 4, reward: 3891.84
Episode: 5, reward: 3663.42
Episode: 6, reward: 3487.97
Episode: 7, reward: 3737.92
Episode: 8, reward: 4020.61
Episode: 9, reward: 3966.11
Episode: 10, reward: 3662.67
Segmentation fault
(py38) chueat@ChuEating-PC:/mnt/c/Users/ChuEating/Reinforcement-Learning/HW3$
```

Training for 500 episodes



```
(py38) chueat@ChuEating-PC:/mnt/c/Users/ChuEating/Reinforcement-Learning/HW3$ python ddpg_cheetah.py
/home/chueat/anaconda3/envs/py38/lib/python3.8/site-packages/Cython/Distutils/old_build_ext.py:15: DeprecationWarning: de
p_util is Deprecated. Use functions from setuptools instead.
  from distutils.dep_util import newer, newer_group
/home/chueat/anaconda3/envs/py38/lib/python3.8/site-packages/gym/core.py:172: DeprecationWarning: WARN: Function `env.see
d(seed)` is marked as deprecated and will be removed in the future. Please use `env.reset(seed=seed) instead.
  deprecation(
Loading models from ./preTrained/ddpg_actor_HalfCheetah-v3_05092024_014726_DDPG and ./preTrained/ddpg_critic_HalfCheetah-
v3_05092024_014726_DDPG
ddpg_cheetah.py:366: UserWarning: Creating a tensor from a list of numpy.ndarrays is extremely slow. Please consider conv
erting the list to a single numpy.ndarray with numpy.array() before converting to a tensor. (Triggered internally at /opt
/conda/conda-bld/pytorch_1712608851799/work/torch/csrc/utils/tensor_new.cpp:274.)
  state = torch.Tensor([env.reset()])
Creating window glfw
Episode: 1, reward: 3090.15
Episode: 2, reward: 3145.45
Episode: 3, reward: 3079.62
Episode: 4, reward: 3056.45
Episode: 5, reward: 3062.46
Episode: 6, reward: 3036.02
Episode: 7, reward: 3164.22
Episode: 8, reward: 3004.07
Episode: 9, reward: 3052.26
Episode: 10, reward: 3134.62
Segmentation fault
(py38) chueat@ChuEating-PC:/mnt/c/Users/ChuEating/Reinforcement-Learning/HW3$
```





- **<u>Discussion</u>**

  The policy loss decreasing significantly over time. The smooth line indicates a consistent reduction in loss, suggesting improvements in policy behavior throughout the training. EWMA of rewards smoothens the fluctuations seen in the per-episode rewards chart. This curve rises steadily, indicating consistent improvement in the model's performance over time, with fewer extremes compared to the per-episode reward graph. We can also observe that before each increase to a new peak in rewards, the EWMA of the rewards first dips. This could be due to the policy escaping from a local maximum.