

Avance tarea 3: Policy - Gradient

Código: EL7021-1

Nombre: José Luis Cádiz Sejas

1. Parametrización de la política: Código adjunto.

- 1.1. Código adjunto.
- 1.2. Código adjunto.

2. Muestreo de trayectorias.

- 2.1. **Perform_single_rollout y verificación de funcionamiento:** Para verificar su funcionamiento se considera que el largo del episodio vine dado por `print(f"Largo del episodio {nb_steps}")` cuando la tarea está terminada.

2.1.1. CartPole:

```
# Rollout
x1=perform_single_rollout(env, policy_gradients_agent, 1000, render=False)
print(x1[0].shape)
print(x1[1].shape)
print(x1[2].shape)
```

[56]

```
... Largo del episodio 22
(22, 4)
(22,)
(22,)
```

El número de filas de las observaciones es igual con el largo del episodio, por lo que se concluye el correcto funcionamiento de la función.

2.1.2. Pendulum:

```
# Rollout
x1=perform_single_rollout(env, policy_gradients_agent, 1000, render=False)
print(x1[0].shape)
print(x1[1].shape)
print(x1[2].shape)
```


[12] ✓ 0.2s

```
... Largo del episodio 200
(200, 3)
(200,)
(200,)
```

El número de filas de las observaciones es igual con el largo del episodio, por lo que se concluye el correcto funcionamiento de la función.

- 2.2. **Sample_rollouts y verificación de funcionamiento:** Se verifica obteniendo un número de sample rollouts al menos igual o mayor que el tamaño del Batch.

2.2.1. CartPole:

```
>  # Sample rollouts  
x2=sample_rollouts(env, policy_gradients_agent, 1000, 5000)  
[57]  
... Largo del episodio 21  
Largo del episodio 11  
Largo del episodio 25  
Largo del episodio 34  
Largo del episodio 23  
Largo del episodio 57  
  
sampled_obs = np.concatenate([x2[i][0] for i in range(len(x2))])  
sampled_action = np.concatenate([x2[i][1] for i in range(len(x2))])  
sampled_reward = np.concatenate([x2[i][2] for i in range(len(x2))])  
print(sampled_obs.shape)  
print(sampled_action.shape)  
print(sampled_reward.shape)  
[58]  
... (5017, 4)  
(5017,)  
(5017,)
```

El largo del registro de sample rollout es al menos el número de sample mini batch, se concluye que la función funciona.

2.2.2. Pendulum:

```
# Sample rollouts  
x2=sample_rollouts(env, policy_gradients_agent, 1000, 5000)  
[53]  
... Largo del episodio 200  
Largo del episodio 200  
Largo del episodio 200  
Largo del episodio 200  
  
sampled_obs = np.concatenate([x2[i][0] for i in range(len(x2))])  
sampled_action = np.concatenate([x2[i][1] for i in range(len(x2))])  
sampled_reward = np.concatenate([x2[i][2] for i in range(len(x2))])  
print(sampled_obs.shape)  
print(sampled_action.shape)  
print(sampled_reward.shape)  
[54]  
... (5000, 3)  
(5000,)  
(5000,)
```

El largo del registro de sample rollout es al menos el número de sample mini batch, se concluye que la función funciona.

3. Estimación de retornos:

3.1. Código adjunto.

- 3.2. **Verificación de funcionamiento:** Se verifica mediante calculo manual del retorno descontado. Si este el mismo dentro de los steps de cada episodio, se verifica el correcto funcionamiento.

3.2.1. CartPole:

```
env = gym.make('CartPole-v1')

dim_states = env.observation_space.shape[0]

continuous_control = isinstance(env.action_space, gym.spaces.Box)
dim_actions = env.action_space.shape[0] if continuous_control else env.action_space.n

policy_gradients_agent = PolicyGradients(dim_states=dim_states,
                                         dim_actions=dim_actions,
                                         lr=0.005,
                                         gamma=0.99,
                                         continuous_control=continuous_control,
                                         reward_to_go=False,
                                         use_baseline=False)

# Sample rollouts (2 episodios): Ejecutar hasta que se generen solo 2 episodios!!
x2=sample_rollouts(env, policy_gradients_agent, 1000, 22)
sampled_rew = [x2[i][2] for i in range(len(x2))]

print("")
print("Vector de retorno")
print(estimate_returns(sampled_rew))
print("")
retorno=0
for t,reward in enumerate(x2[0][2]):
    retorno=retorno+(_gamma**t)*reward
print("Retorno Ep 1")
print(retorno)
print("")
retorno=0
for t,reward in enumerate(x2[1][2]):
    retorno=retorno+(_gamma**t)*reward
print("Retorno Ep 2")
print(retorno)
print("")
```

Largo del episodio 15

Largo del episodio 32

Vector de retorno

```
[13.994164 13.994164 13.994164 13.994164 13.994164 13.994164 13.994164
13.994164 13.994164 13.994164 13.994164 13.994164 13.994164 13.994164
13.994164 27.501966 27.501966 27.501966 27.501966 27.501966 27.501966
27.501966 27.501966 27.501966 27.501966 27.501966 27.501966 27.501966
27.501966 27.501966 27.501966 27.501966 27.501966 27.501966 27.501966
27.501966 27.501966 27.501966 27.501966 27.501966]
```

Retorno Ep 1

13.994164535871148

Retorno Ep 2

27.501966404214624

3.2.2. Pendulum:

```
env=gym.make('Pendulum-v1')

dim_states = env.observation_space.shape[0]

continuous_control = isinstance(env.action_space, gym.spaces.Box)
dim_actions = env.action_space.shape[0] if continuous_control else env.action_space.n

policy_gradients_agent = PolicyGradients(dim_states=dim_states,
                                         dim_actions=dim_actions,
                                         lr=0.005,
                                         gamma=0.99,
                                         continuous_control=continuous_control,
                                         reward_to_go=False,
                                         use_baseline=False)

# Sample rollouts (2 episodios): Ejecutar hasta que se generen solo 2 episodios!!
x2=sample_rollouts(env, policy_gradients_agent, 1000, 220)
sampled_rew = [x2[i][2] for i in range(len(x2))]
index_episodio_1=x2[0][1].shape[0]

print("")
print("Muestra vector de retorno ep 1")
print(estimate_returns(sampled_rew)[index_episodio_1-1])
print("")
print("Muestra vector de retorno ep 2")
print(estimate_returns(sampled_rew)[index_episodio_1+1])
print("")

retorno=0
for t,reward in enumerate(x2[0][2]):
    retorno=retorno+(_gamma**t)*reward
print("Retorno Ep 1")
print(retorno)
print("")

retorno=0
for t,reward in enumerate(x2[1][2]):
    retorno=retorno+(_gamma**t)*reward
print("Retorno Ep 2")
print(retorno)
print("")
```

Largo del episodio 200
Largo del episodio 200

Muestra vector de retorno ep 1
-758.4204

Muestra vector de retorno ep 2
-679.38983

Retorno Ep 1
-758.4204328420228

Retorno Ep 2
-679.3898158058794

4. Policy Gradients:

4.1. Código adjunto.

5. Reducción de la varianza.

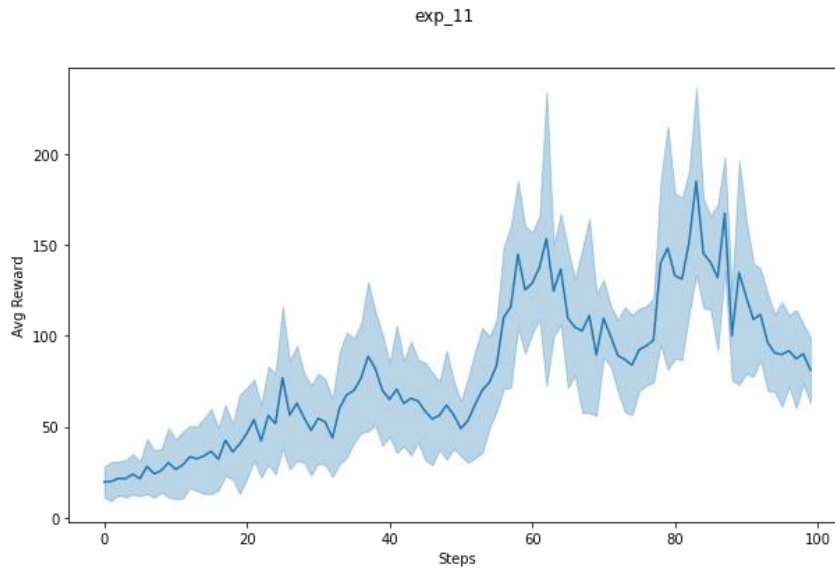
5.1. Código adjunto.

5.2. Código adjunto.

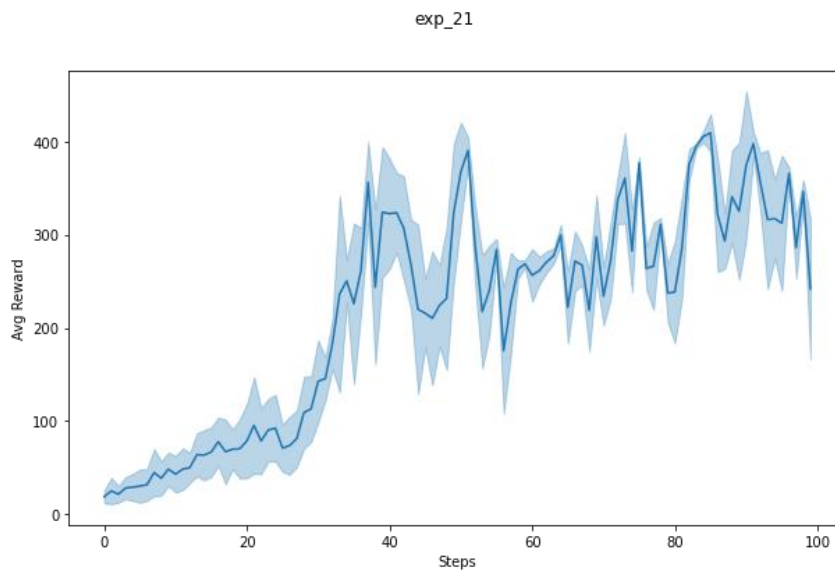
6. Evaluación del algoritmo:

6.1. Entrenamiento y reporte de resultados Cartpole.

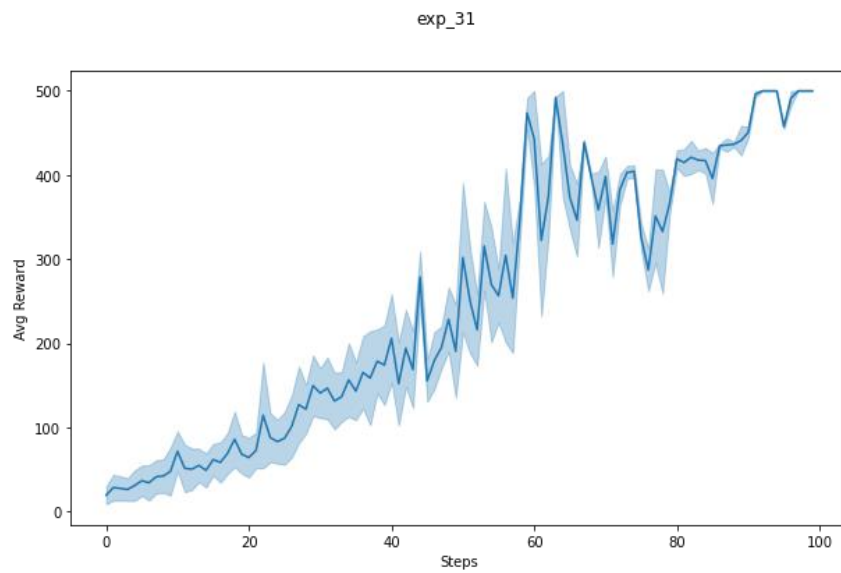
6.1.1. Exp 11: {"name": "exp_11", "batch_size": 500,
"use_baseline": False, "reward_to_go": False}



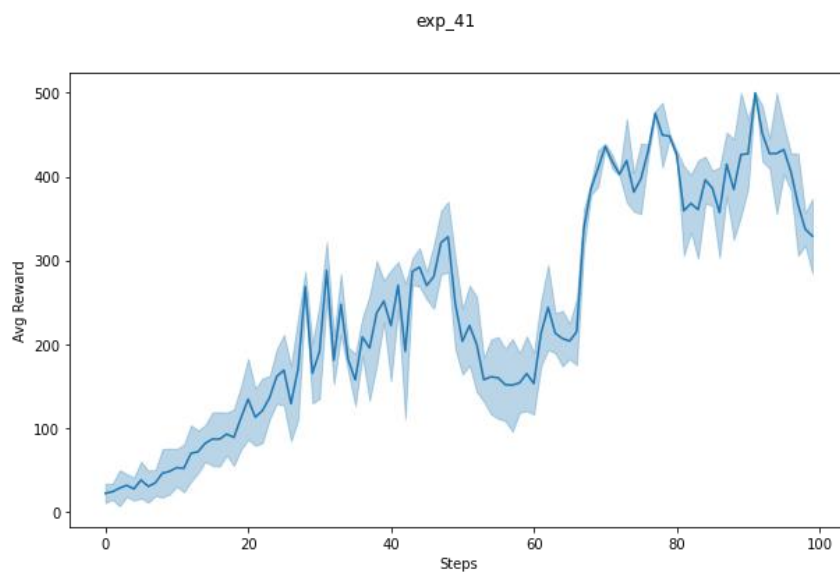
6.1.2. Exp 21: {"name": "exp_21", "batch_size": 500,
"use_baseline": False, "reward_to_go": True}



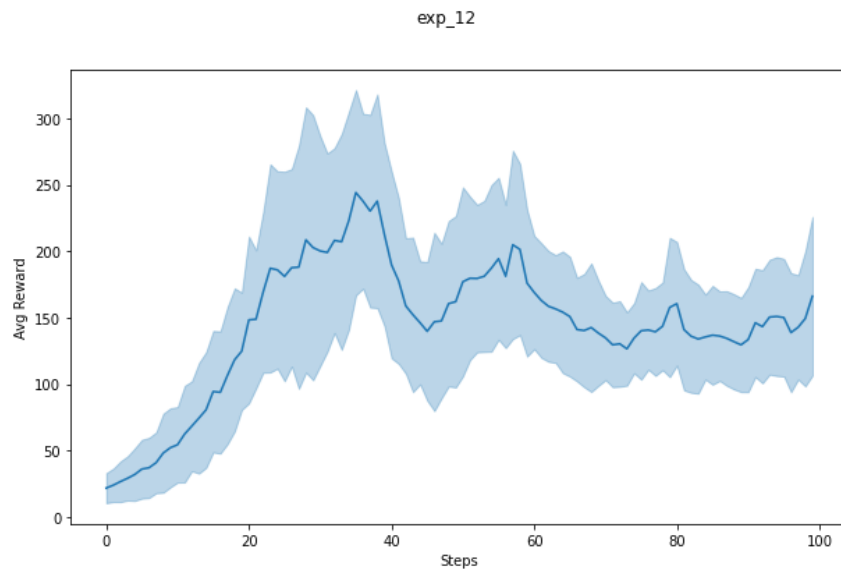
6.1.3. Exp 31: {"name": "exp_31", "batch_size": 500,
"use_baseline": True, "reward_to_go": False}



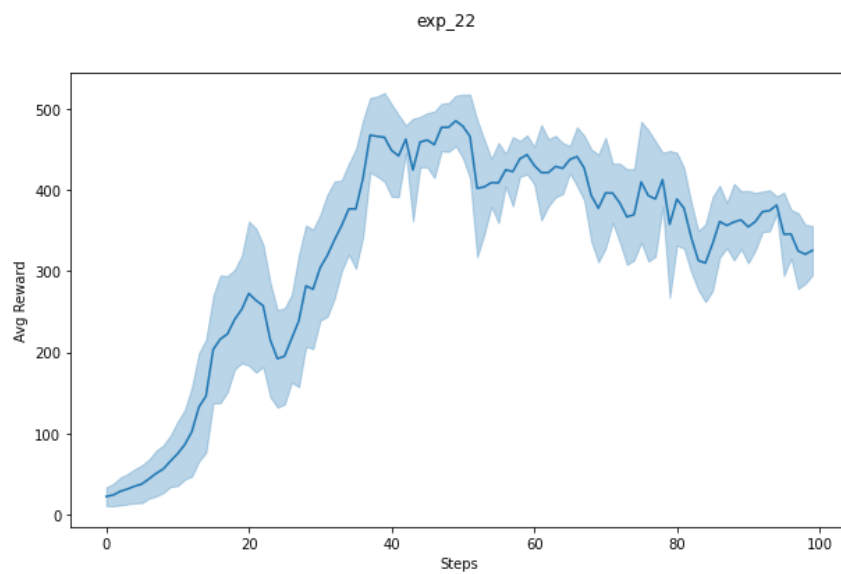
6.1.4. Exp 41: {"name": "exp_41", "batch_size": 500,
"use_baseline": True, "reward_to_go": True}



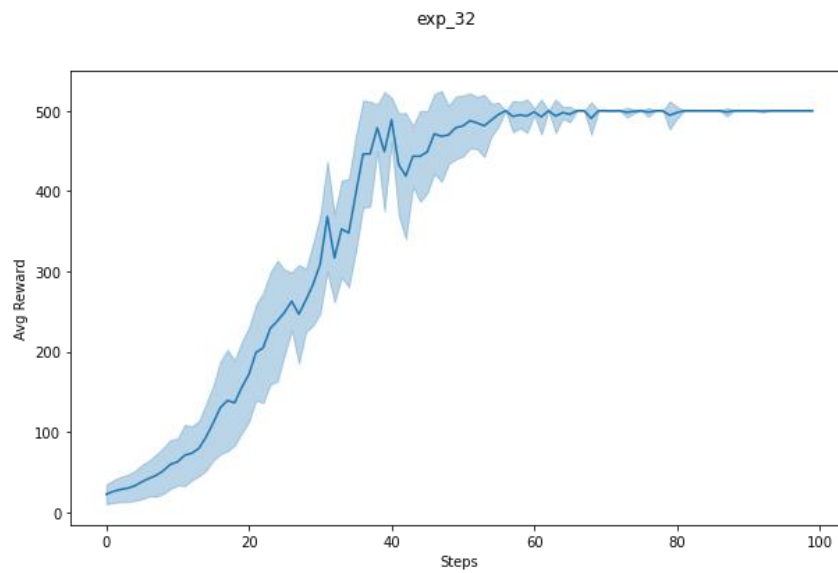
6.1.5. Exp 12: {"name": "exp_12", "batch_size": 5000,
"use_baseline": False, "reward_to_go": False}



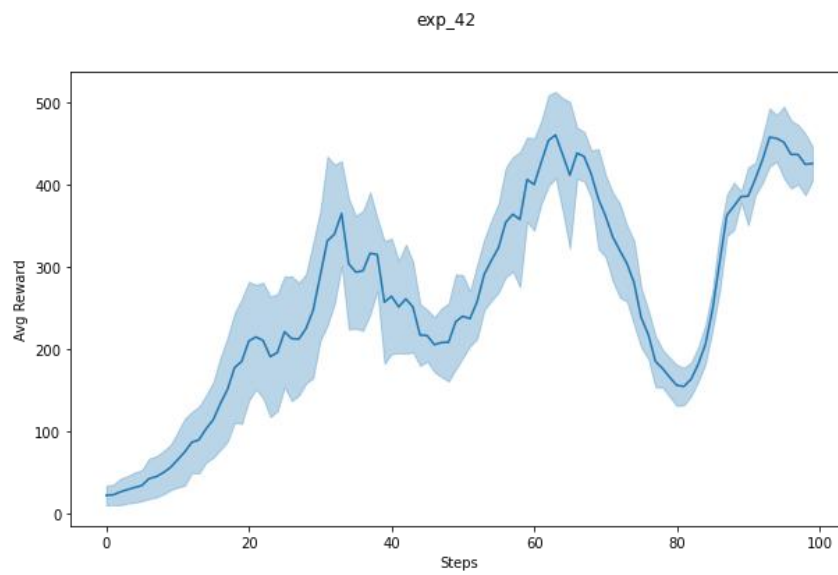
6.1.6. Exp 22: {"name": "exp_22", "batch_size": 5000,
"use_baseline": False, "reward_to_go": True}



6.1.7. Exp 32: {"name": "exp_32", "batch_size": 5000,
"use_baseline": True, "reward_to_go": False}

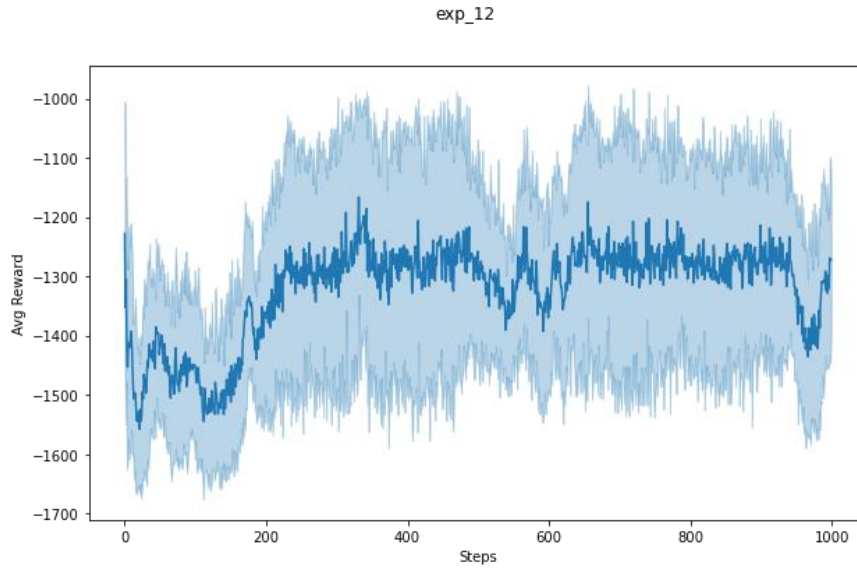


6.1.8. Exp 42: {"name": "exp_42", "batch_size": 5000,
"use_baseline": True, "reward_to_go": True}

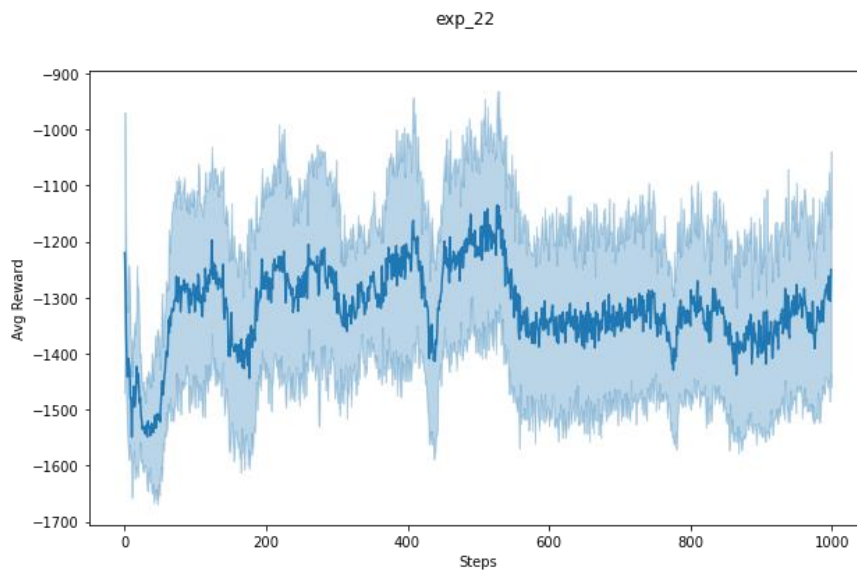


6.2. Entrenamiento y reporte de resultados Pendulum.

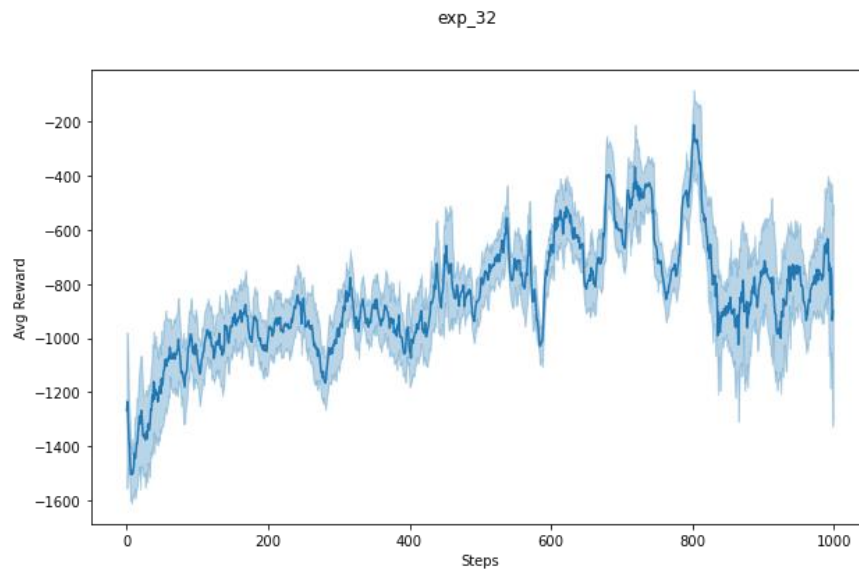
6.2.1. Exp 12: {"name": "exp_12", "batch_size": 5000,
"use_baseline": False, "reward_to_go": False}



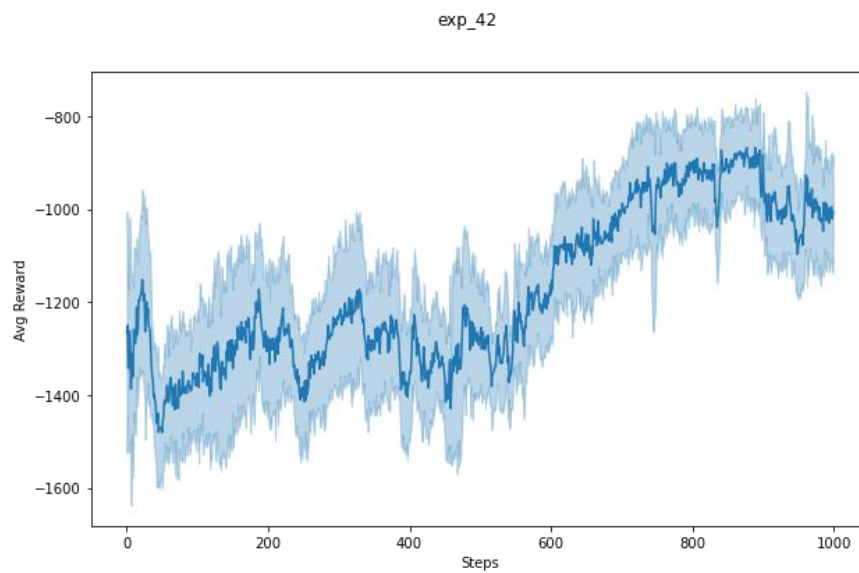
6.2.2. Exp 22: {"name": "exp_22", "batch_size": 5000,
"use_baseline": False, "reward_to_go": True}



6.2.3. Exp 32: {"name": "exp_32", "batch_size": 5000,
"use_baseline": True, "reward_to_go": False}



6.2.4. Exp 42: {"name": "exp_42", "batch_size": 5000,
"use_baseline": True, "reward_to_go": True}



- 6.3. **Análisis CartPole:** Respecto al tamaño de Batches, se observa que el desempeño de los modelos mejora con el aumento de Batches y disminuye la variabilidad del rendimiento de un entrenamiento a otro, sin embargo, aumenta la varianza dentro de un entrenamiento en particular.

Por otro lado, se observa que al incorporar uso de reward_to_go o baseline, aumenta el rendimiento del agente y disminuye la varianza de los entrenamientos, sin embargo, existe una anomalía en el experimento Exp_42 al utilizar ambos enfoques.

- 6.4. **Análisis Pendulum:** A partir de los experimentos se observa que el uso de reward_to_go y baseline en conjunto aumentan considerablemente el rendimiento del agente y además disminuye en gran medida la varianza y variabilidad de los resultados.