Entrega final tarea 2: Deep Q-Network

Código: EL7021-1

Nombre: José Luis Cádiz Sejas

1. **Deep Q-Network (i):** Código adjunto.

2. **Replay Buffer**.
   2.1. Código adjunto.
   2.2. Código adjunto.
   2.3. Código adjunto.
   2.4. Corroboración de funcionamiento:

   Se genera un Buffer de tamaño 5 en donde el muestreo será de tamaño 3. Se simula la obtención de 6 transiciones, las cuales se van almacenando dentro del Buffer.

```
# Inicialización de memory Buffer
max_size=5
sample_size=3
memory=ReplayBuffer(dim_states, dim_actions, max_size, sample_size)
```
✓ 0.0s

```
# Simulación de 6 transiciones
s_t=env.reset()

for i in range(6):

    a_t=np.random.randint(2)
    s_t1, r_t, done_t, _ = env.step(a_t)
    print(s_t1, r_t, done_t)

    # Guardar
    memory.store_transition(s_t, a_t, r_t, s_t1, done_t)

    s_t = s_t1
```
✓ 0.0s

```
[ 0.00593574 -0.17127012 -0.02576173  0.2443883 ] 1.0 False
[ 0.00251034  0.02421015 -0.02087396 -0.0563079 ] 1.0 False
[ 0.00299455 -0.17060639 -0.02200012  0.22971673] 1.0 False
[-4.1758240e-04 -3.6540717e-01 -1.7405786e-02  5.1537967e-01] 1.0 False
[-0.00772573 -0.17004447 -0.00709819  0.217263  ] 1.0 False
[-0.01112662  0.02517823 -0.00275293 -0.07765053] 1.0 False
```

   Se observa que el sexto elemento de la transición de estados coincide con el primer elemento del conjunto de estados almacenados. Esto debido a que la data se va sobrescribiendo en la medida de que hay nuevos registros.

```
# Datos almacenados: Conjunto de estados almacenados
memory._s_t1_array
```
✓ 0.0s

```
array([[-1.11266151e-02,  2.51782257e-02, -2.75293179e-03,
        -7.76505321e-02],
       [ 2.51034228e-03,  2.42101457e-02, -2.08739620e-02,
        -5.63078970e-02],
       [ 2.99454527e-03, -1.70606390e-01, -2.20001191e-02,
         2.29716733e-01],
       [-4.17582400e-04, -3.65407169e-01, -1.74057856e-02,
         5.15379667e-01],
       [-7.72572542e-03, -1.70044467e-01, -7.09819188e-03,
         2.17262998e-01]])
```

Se comprueba adicionalmente que el tamaño del muestreo es el mismo que el seteado para una variable en particular, en este caso para s_t.

```
# Se guardan a los 5 elementos, los cuales son el max_size del buffer
memory._s_t1_array.shape
```
✓ 0.0s

```
(5, 4)
```

```
# Sample
memory.sample_transitions()
```
✓ 0.0s

```
(array([[ 2.99454527e-03, -1.70606390e-01, -2.20001191e-02,
          2.29716733e-01],
        [-4.17582400e-04, -3.65407169e-01, -1.74057856e-02,
          5.15379667e-01],
        [-7.72572542e-03, -1.70044467e-01, -7.09819188e-03,
          2.17262998e-01]]),
 array([0., 1., 1.]),
 array([1., 1., 1.]),
 array([[-4.17582400e-04, -3.65407169e-01, -1.74057856e-02,
          5.15379667e-01],
        [-7.72572542e-03, -1.70044467e-01, -7.09819188e-03,
          2.17262998e-01],
        [-1.11266151e-02,  2.51782257e-02, -2.75293179e-03,
         -7.76505321e-02]]),
 array([False, False, False]))
```

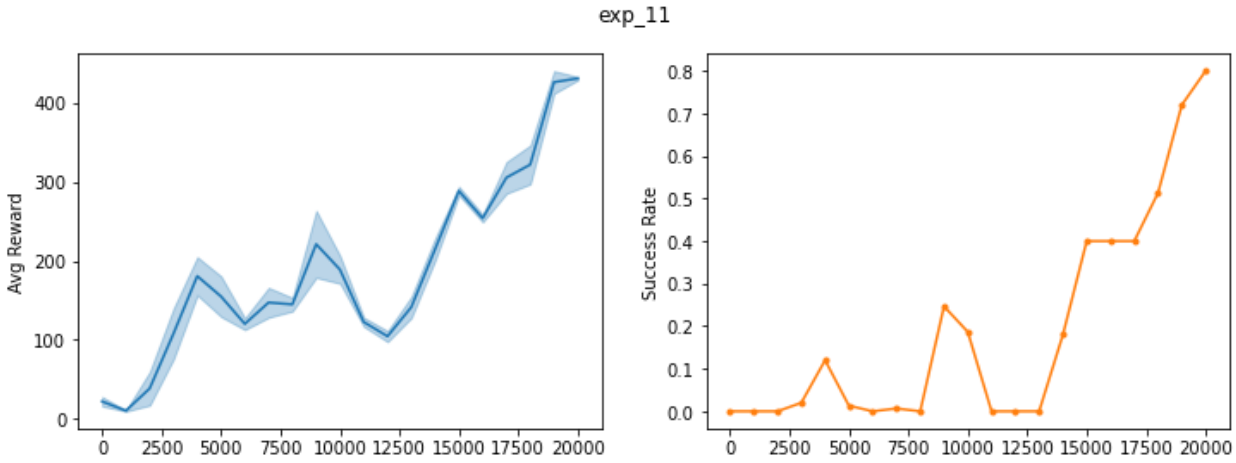
```
# Se observa el muestreo de 3 elementos para el conjunto de estados
len(memory.sample_transitions()[0])
```
✓ 0.0s

```
3
```

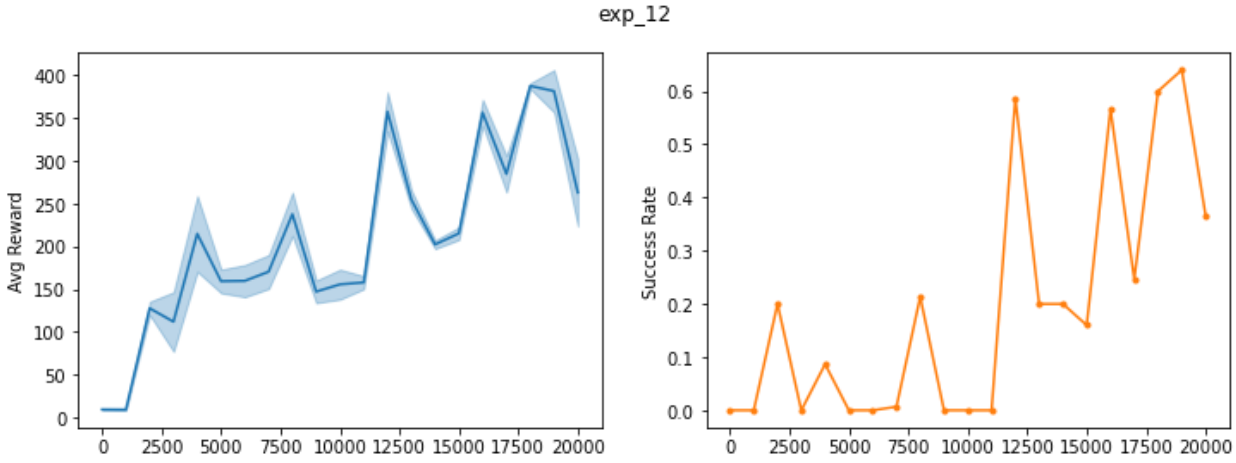
3. **Deep Q-Network (ii):** Código adjunto.

4. **Experimentos**.
   4.1. Código adjunto.
   4.2. Experimentos y análisis:

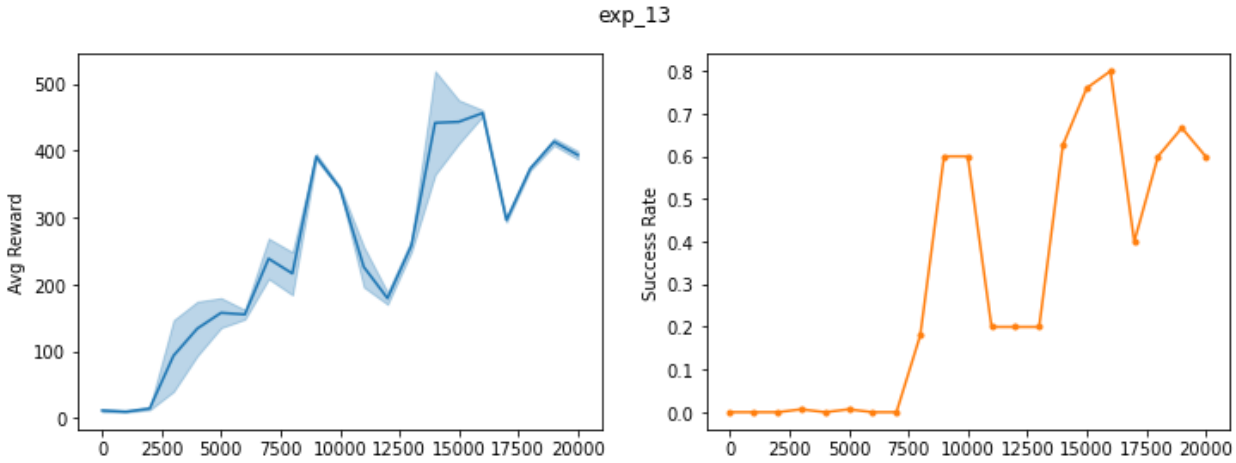
- Exp_11: {"name":"exp_11","replay_buffer_size":1000, "batch_size":16, "nb_steps_target_replace":100}



- Exp_12: {"name":"exp_12","replay_buffer_size":2500, "batch_size":16, "nb_steps_target_replace":100}

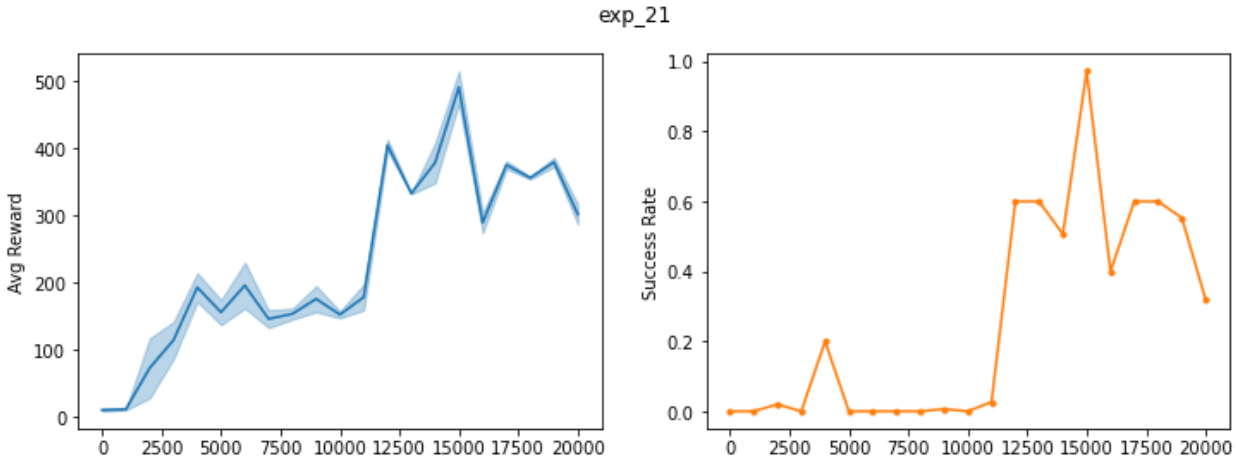


- Exp_13: {"name":"exp_13","replay_buffer_size":5000, "batch_size":16, "nb_steps_target_replace":100}

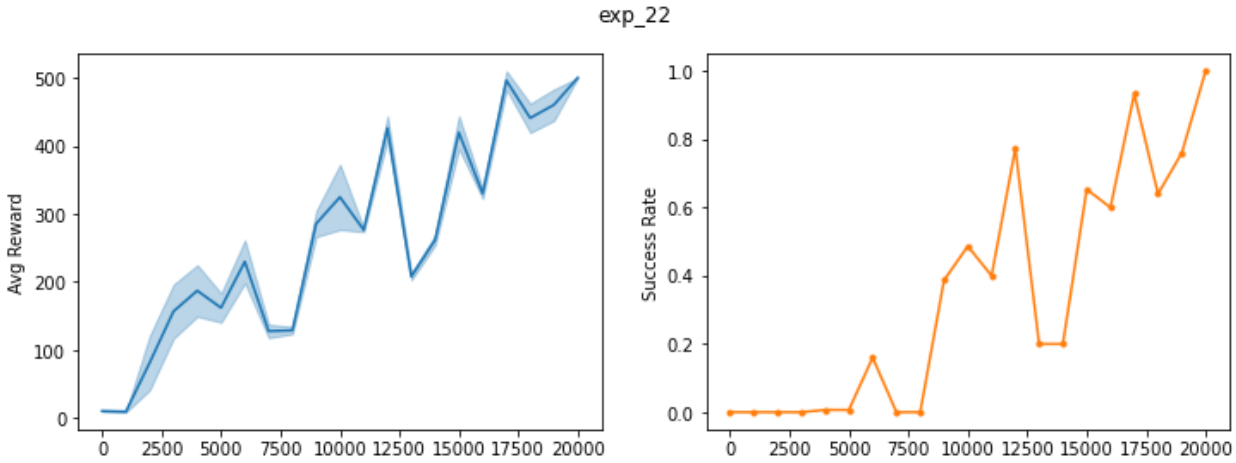


A partir del experimento 1, se puede observar que se obtiene un mejor Success Rate en un número menor de episodios cuando el Buffer posee un mayor tamaño (ver experimento 13). Esto se asocia a que es posible almacenar una mayor variabilidad de información, la cual es valiosa para el entrenamiento de la red neuronal.

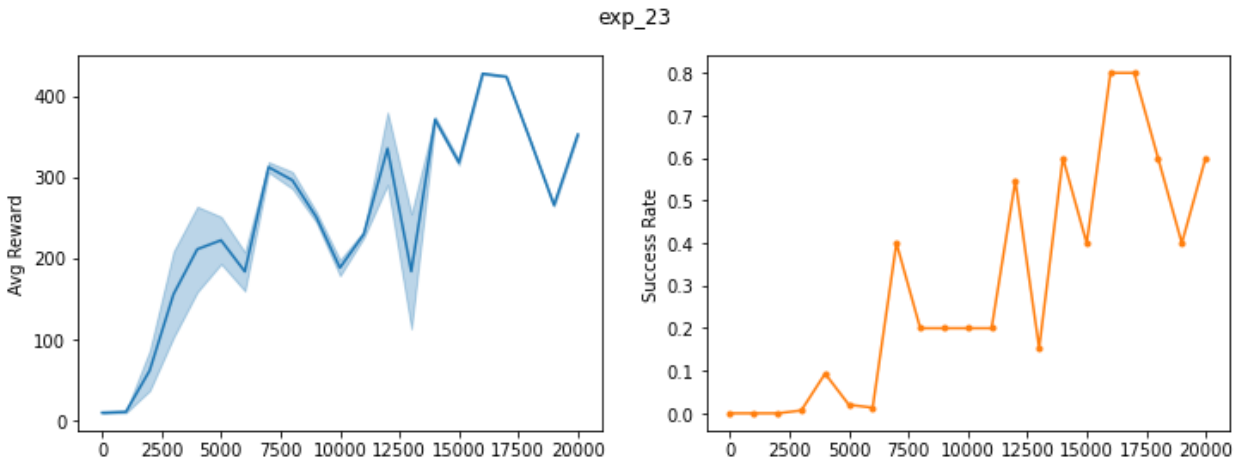
- Exp_21: {"name":"exp_21","replay_buffer_size":5000, "batch_size":32, "nb_steps_target_replace":100}



- Exp_22: {"name":"exp_22","replay_buffer_size":5000, "batch_size":64, "nb_steps_target_replace":100}

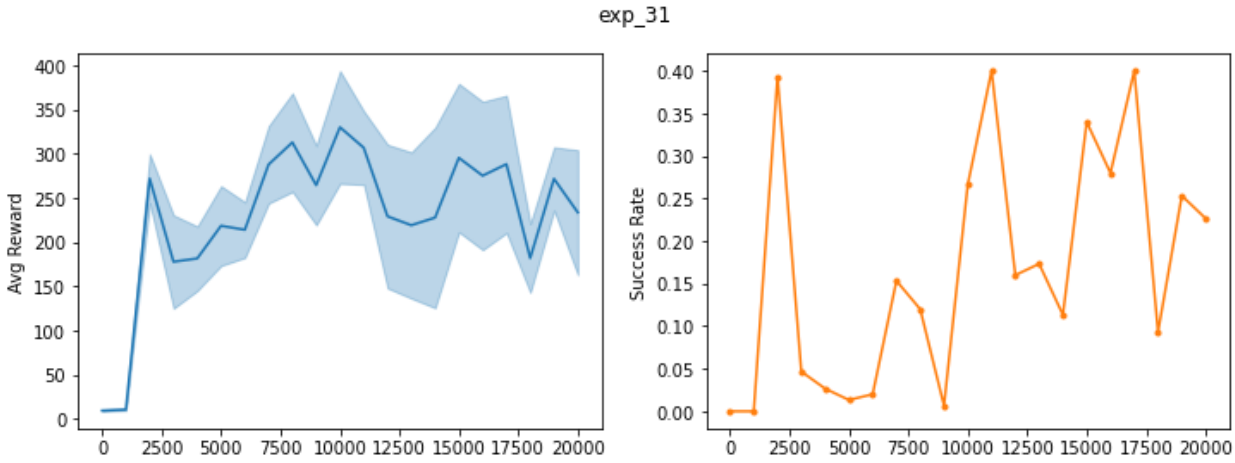


- Exp_23: {"name":"exp_23","replay_buffer_size":5000, "batch_size":128, "nb_steps_target_replace":100}

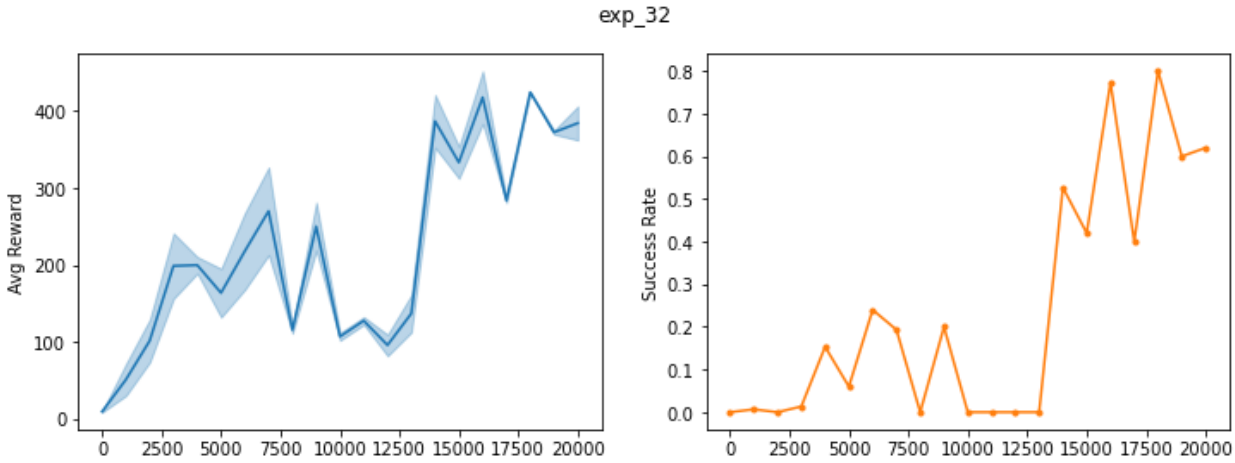


A partir del experimento 2, no se observa una mayor variabilidad en el rendimiento de la política óptima, pero si observa una disminución en la varianza del Reward obtenido cuando hay un tamaño mayor del batch (ver experimento 23), esto en la medida de que van avanzados los episodios de entrenamiento.

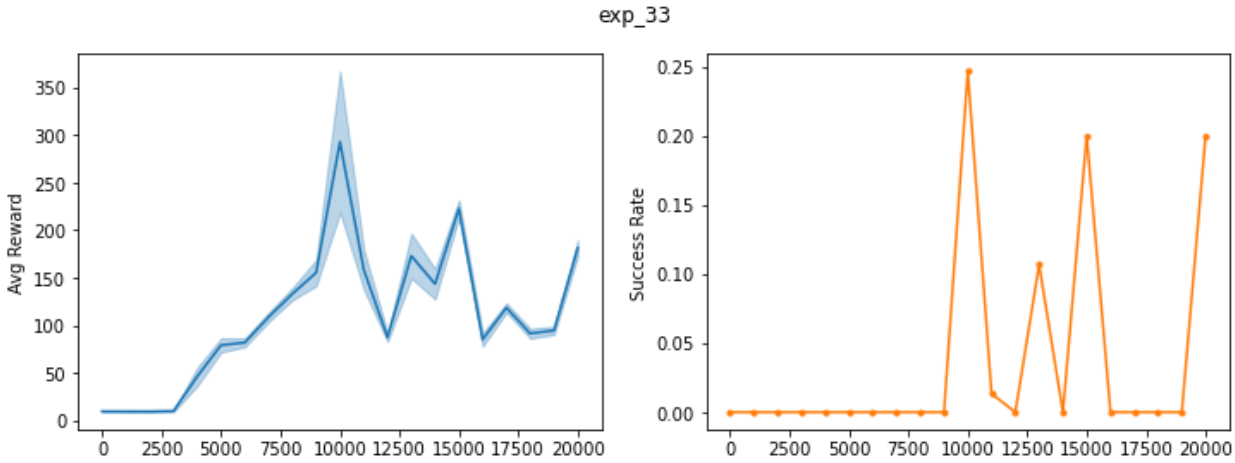
- Exp_31: {"name":"exp_31","replay_buffer_size":5000, "batch_size":128, "nb_steps_target_replace":1}



- Exp_32: {"name":"exp_32","replay_buffer_size":5000, "batch_size":128, "nb_steps_target_replace":100}



- Exp_33: {"name":"exp_33","replay_buffer_size":5000, "batch_size":128, "nb_steps_target_replace":1000}



A partir del experimento 3, se observa que para un nb_steps_target_replace igual 1 (ver experimento 31) se genera una inestabilidad en el aprendizaje de la red neuronal, esto debido a que estamos evaluando la red con las predicciones de esta misma. La inestabilidad del aprendizaje de la red se puede ver en la alta varianza del Reward obtenido.

Por otro lado, si nb_steps_target_replace posee un valor muy alto, el proceso de aprendizaje tampoco es bueno (ver experimento 33). A partir de esto, se concluye que para lograr un buen Success Rate nb_steps_target_replace no puede ser un valor ni muy bajo ni tampoco muy alto (ver experimento 32).

En definitiva, para que el agente pueda aprender las políticas óptimas del ambiente en cuestión, es importante tener un buen tamaño de Replay Buffer y un valor intermedio para la actualización de los parámetros de la red neuronal target. Por otro lado, el tamaño del batch podría afectar en la varianza del Reward pero no de manera significativa en el rendimiento en termino de Success Rate.