Aprendizaje reforzado con Super Mario

José Carlos Batista Rivero

Índice

- Introducción

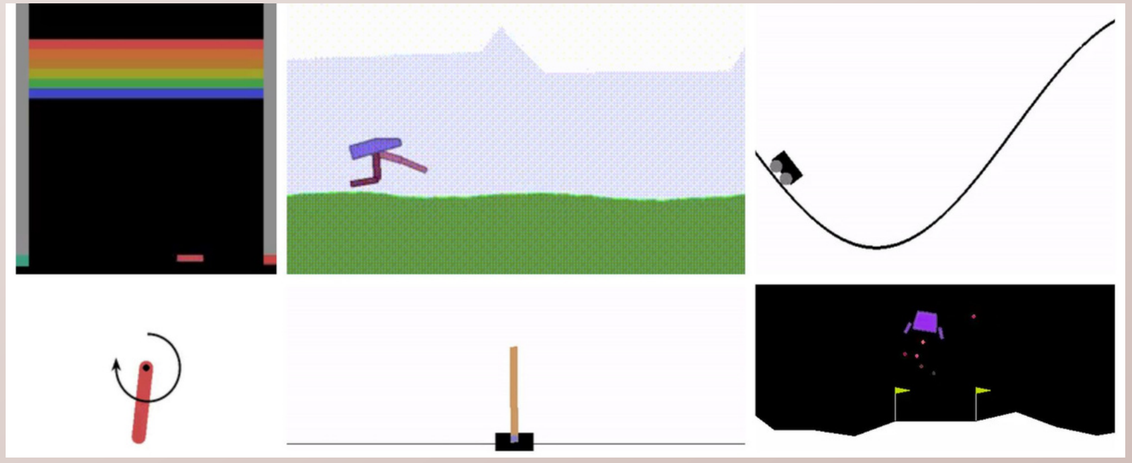
- Aprendizaje reforzado

-DQN (Deep Q Learning)

- Demo

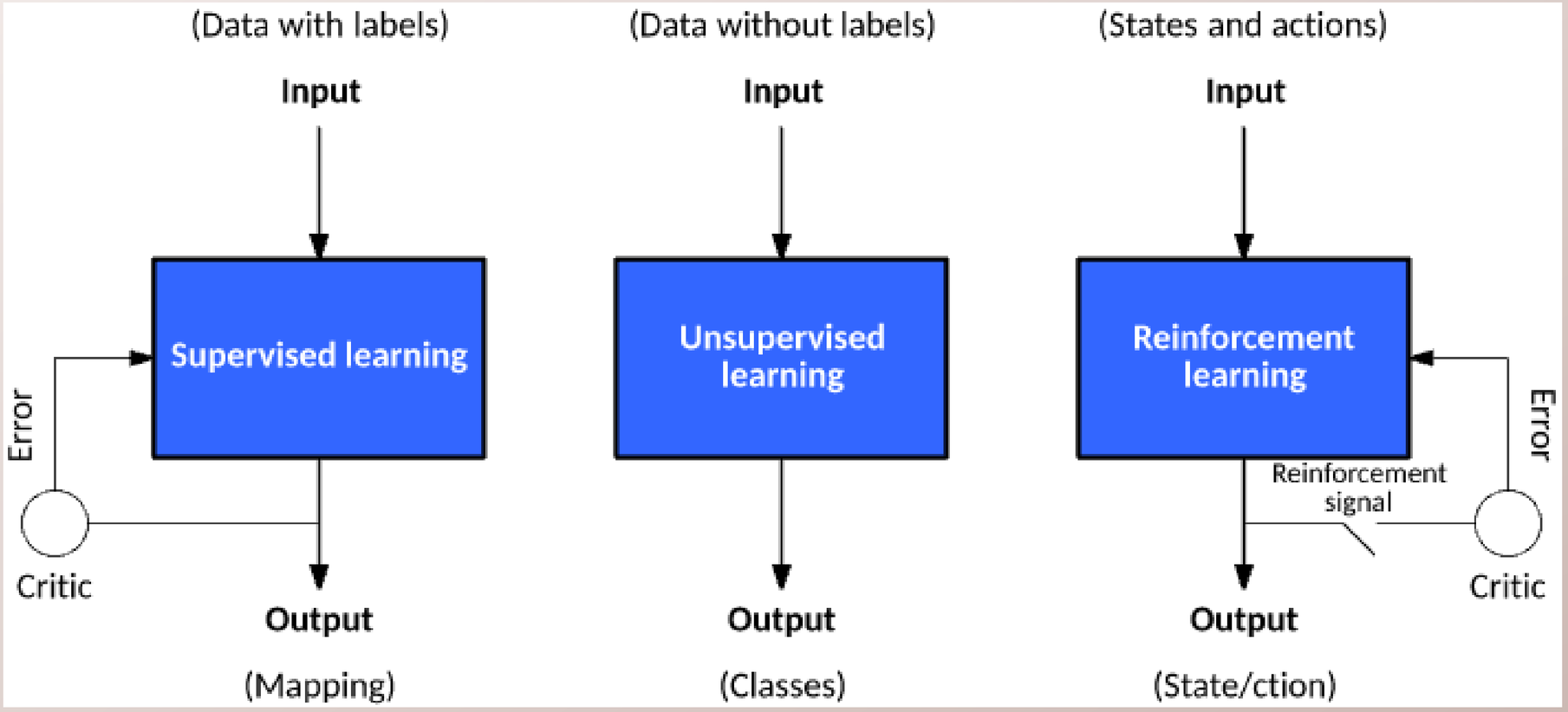
Introducción

OpenAI Gym

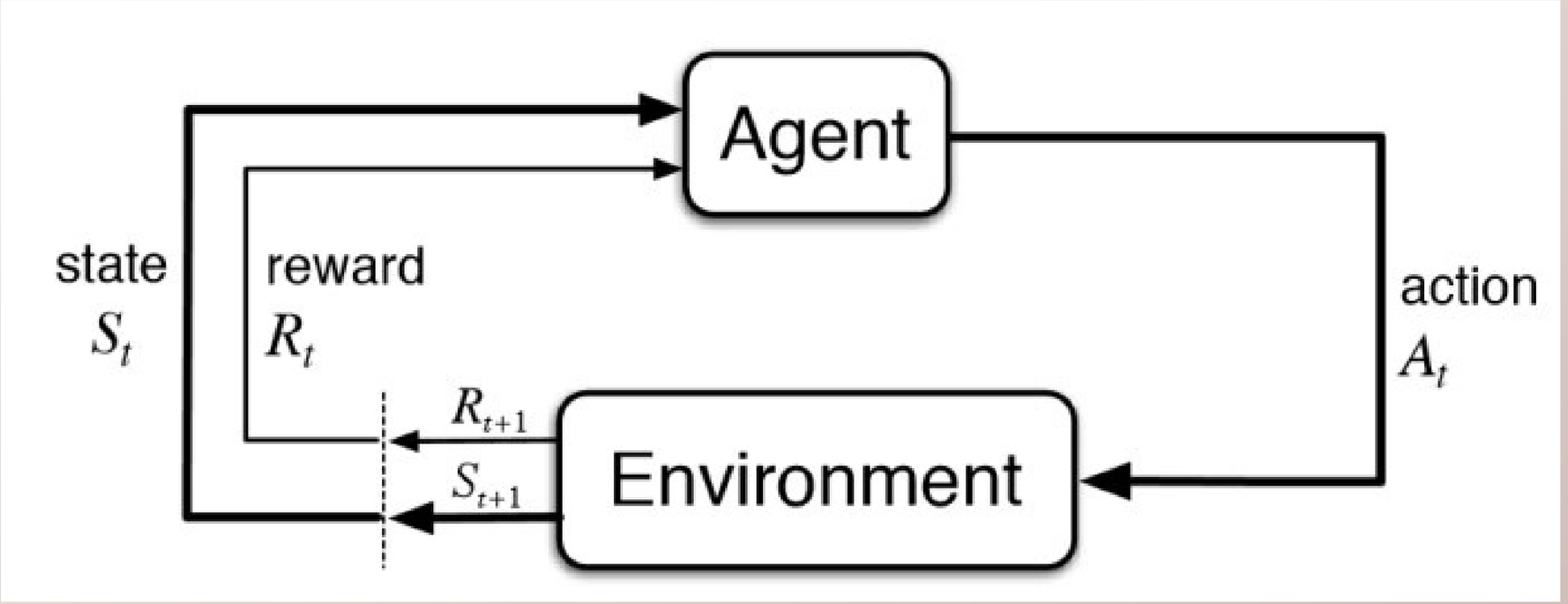


Aprendizaje reforzado

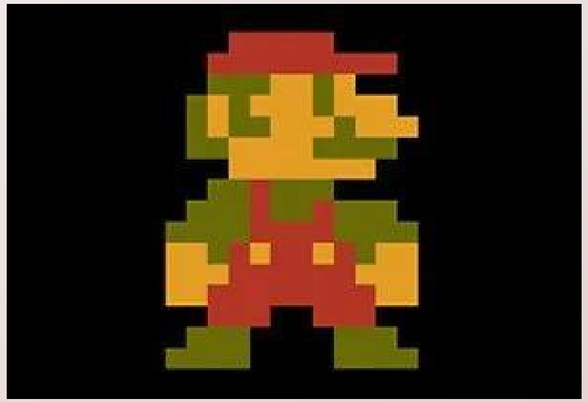
¿Qué es el aprendizaje reforzado?



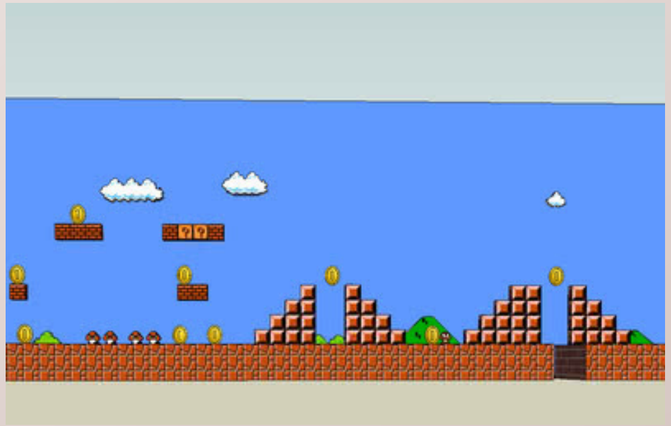
Filosofía



Agente



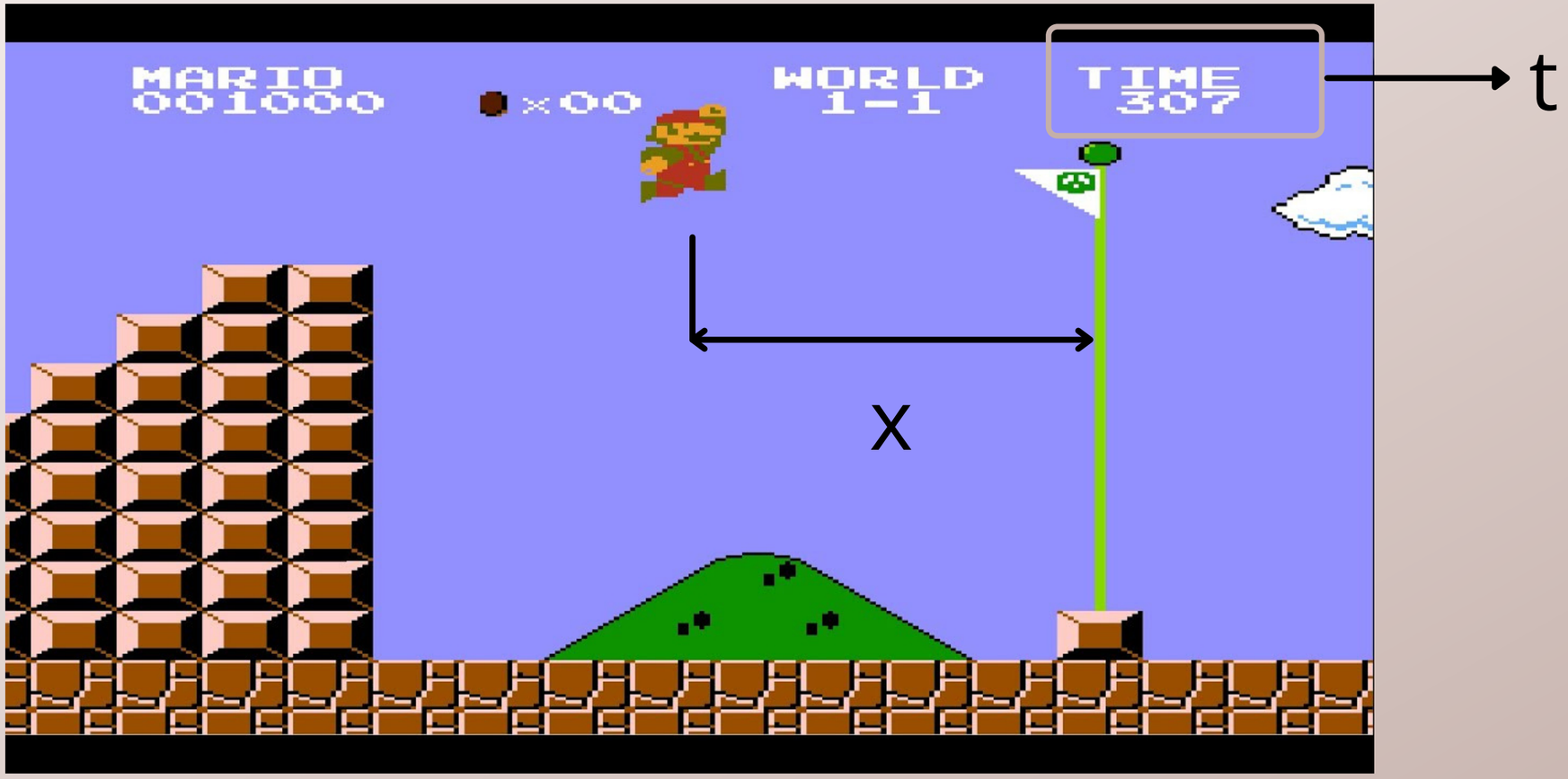
Entorno



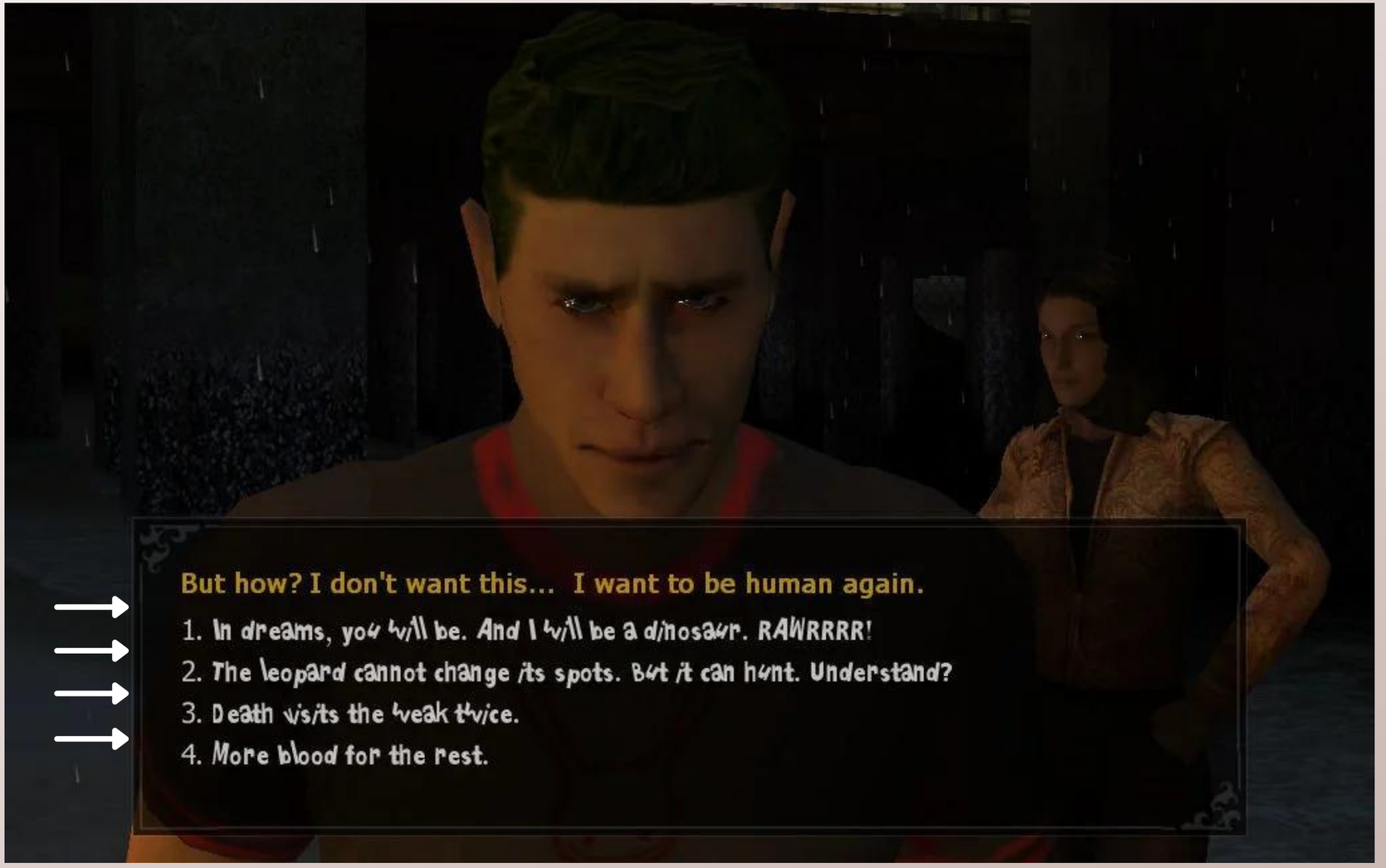
Acción

Recompensa

$$r = x + t + d$$

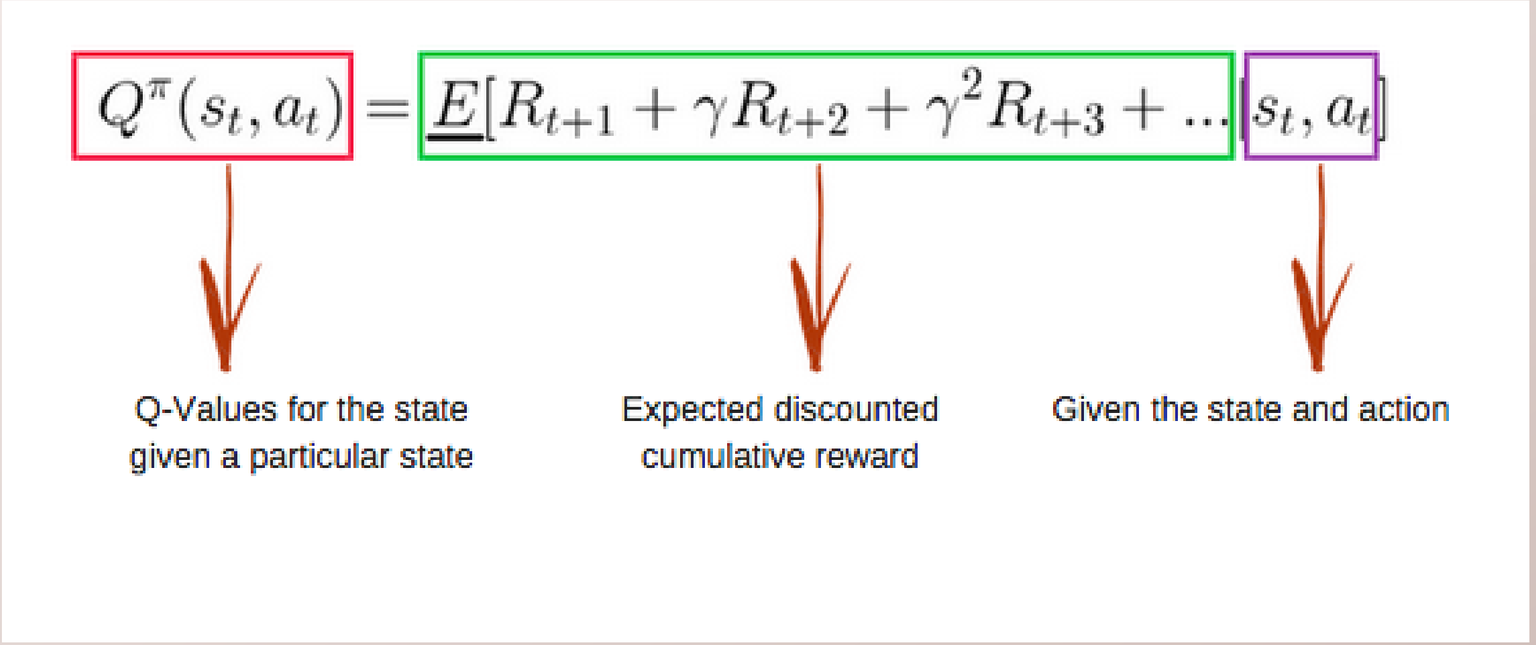


Recompensa


But how? I don't want this... I want to be human again.
1. In dreams, you will be. And I will be a dinosaur. RAWRRRR!
2. The leopard cannot change its spots. But it can hunt. Understand?
3. Death visits the weak twice.
4. More blood for the rest.

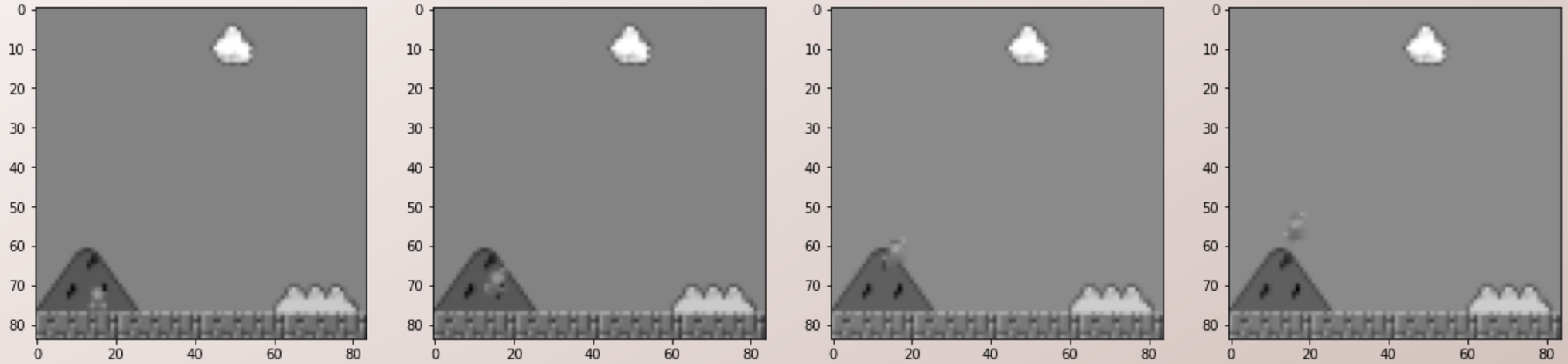
Deep Q Learning

Valor Q



$0 < \gamma < 1$

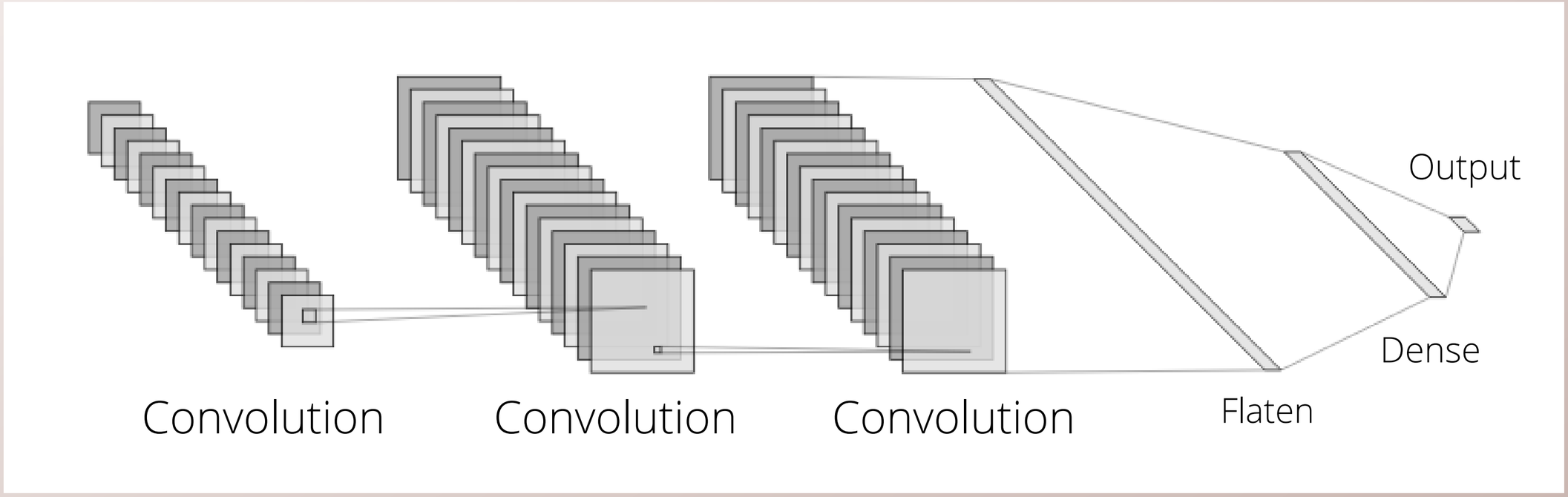
¿Por qué una red neuronal?



$$256^{4 \cdot 84 \cdot 84} = 256^{28224}$$

F(S)

S → A


Output
Dense
Convolution
Convolution
Convolution
Flaten

Pytorch vs Keras

Keras

```python
def build_model(input_shape, n_actions):
    model = keras.Sequential()
    model.add(keras.layers.Conv2D(32, kernel_size=32, strides=4, activation='ReLU', input_shape=input_shape))
    model.add(keras.layers.Conv2D(32, kernel_size=64, strides=2, activation='ReLU'))
    model.add(keras.layers.Conv2D(32, kernel_size=64, strides=1, activation='ReLU'))
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(512, activation='ReLU'))
    model.add(keras.layers.Conv2D(n_actions, activation='linear'))

    return model
```

Pytorch

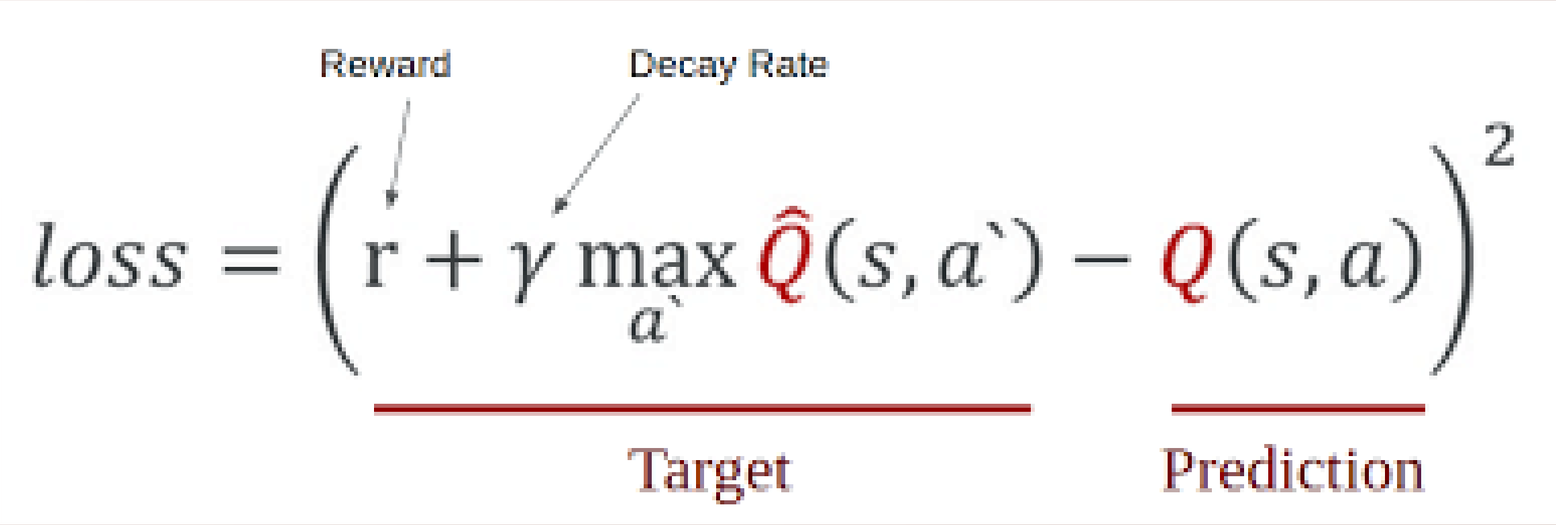
```python
class ConvNet(nn.Module):
    def __init__(self, input_shape, n_actions):
        super(ConvNet, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(input_shape[0], 32, kernel_size=8, stride=4),
            nn.ReLU(),
            nn.Conv2d(32, 64, kernel_size=4, stride=2),
            nn.ReLU(),
            nn.Conv2d(64, 64, kernel_size=3, stride=1),
            nn.ReLU()
        )

        conv_out_size = self._get_conv_out(input_shape)
        self.fc = nn.Sequential(
            nn.Linear(conv_out_size, 512),
            nn.ReLU(),
            nn.Linear(512, n_actions)
        )

    def _get_conv_out(self, shape):
        o = self.conv(torch.zeros(1, *shape))
        return int(np.prod(o.size()))

    def forward(self, x):
        conv_out = self.conv(x).view(x.size()[0], -1)
        return self.fc(conv_out)
```

Función de coste



Mejorar el entrenamiento

- Exploración vs Predicción

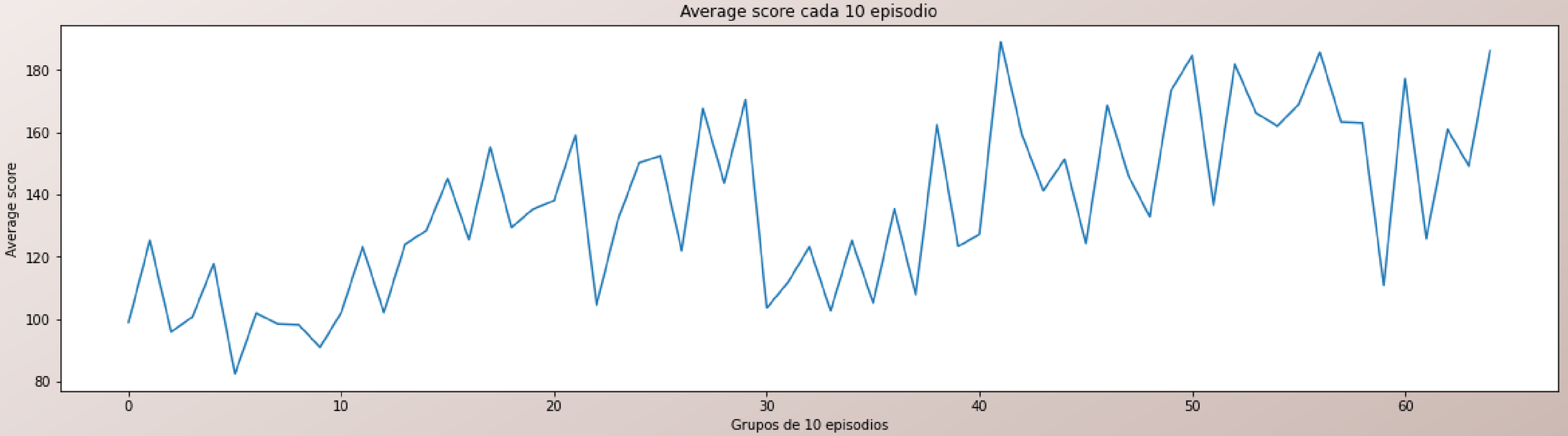
- Memory replay

- Doble red neuronal

Parámetros

	Modelo	learning_rate	gamma	epsilon_decay	score_episode
0	Final	0.00050	0.95	0.999	21.974
1	Prueba1	0.10000	0.92	0.995	-38.436
2	Prueba2	0.00100	0.97	0.990	19.740
3	Prueba3	0.00010	0.99	0.950	21.736
4	Prueba4	0.00001	1.00	0.900	17.922

Entrenamiento



Demo

"Lo siento Mario, pero la princesa está en otro castillo "

Un Toad

¡Muchas gracias!

Autor : José Carlos Batista Rivero
Correo electrónico : josecarlosbatistarivero@gmail.com
github : https://github.com/JoseKuru