

```

8 dqn = DQNAgent(model=model, nb_actions=nb_actions, policy=policy, memory=memory,
9               processor=processor, nb_steps_warmup=agent_nb_steps_warmup, gamma=.99,
10              #target_model_update=agent_target_model_update,
11              train_interval=agent_train_interval, batch_size=agent_batch_size
12              #enable_double_dqn=True,
13              #enable_dueling_network=True, dueling_type='avg'
14              )
15
16 # https://github.com/keras-rl/keras-rl/tree/master/examples
17 #dqn = DQNAgent(model=model, nb_actions=nb_actions, memory=memory, nb_steps_warmup=10,
18 #              enable_dueling_network=True, dueling_type='avg', target_model_update=1e-2, policy=policy)
19
20 #dqn.show_hyperparams()
21 # TO-DO
22 # model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
23 optimizer = optimizer_type # TO-DO
24 dqn.compile(optimizer, metrics=['mae'])
25
26 '''
27 def huber_loss(a, b, in_keras=True):
28     error = a - b
29     quadratic_term = error*error / 2
30     linear_term = abs(error) - 1/2
31     use_linear_term = (abs(error) > 1.0)
32     if in_keras:
33         # Keras won't let us multiply floats by booleans, so we explicitly cast the booleans to floats
34         use_linear_term = K.cast(use_linear_term, 'float32')
35     return use_linear_term * linear_term + (1-use_linear_term) * quadratic_term
36
37 loss=huber_loss
38 custom_objects={'huber_loss': huber_loss}
39 self.model.compile(optimizer, loss=huber_loss)
40 '''

```

```

❏ "\ndef huber_loss(a, b, in_keras=True):\n    error = a - b\n    quadratic_term = error*error / 2\n    linear_term = abs(er

```

```

1 # Timestamp para el nombre de archivo de pesos
2 now = datetime.now()
3 date_time = now.strftime("%Y%m%d%H%M")
4
5 # Nombre del archivo para guardar los pesos
6 weights_filename = 'dqn_{}_weights.h5f'.format(ENV_NAME)
7 weights_filename = date_time + '_' + weights_filename
8
9 # checkpoint por intervalo
10 saved_model= date_time + '_dqn_PartialRunPong_weight_{step}.h5'
11 checkpoint = ModelIntervalCheckpoint(saved_model, interval=50000)
12
13 # logs para TensorBoard
14 tb_log_dir = 'logs/tmp'
15 tb_callback = TensorBoard(log_dir=tb_log_dir, histogram_freq=0, write_graph=True)
16
17 nb_steps = 1750000
18 nb_steps = train_steps
19 # Entrenamiento con diferentes callbacks
20 #dqn.fit(env, nb_steps=nb_steps, visualize=False, callbacks=[checkpoint, tb_callback], verbose=2)
21 #dqn.fit(env, nb_steps=nb_steps, visualize=False, callbacks=[tb_callback], verbose=2)
22 #dqn.fit(env, nb_steps=nb_steps, visualize=False, callbacks=[checkpoint], verbose=2)
23 train_history = dqn.fit(env, nb_steps=nb_steps, visualize=False, verbose=2)
24
25
26 dqn.save_weights(weights_filename, overwrite=True)
27 print("Pesos guardados en:",weights_filename)
28
29 #nb_episodes = 3
30 #dqn.test(env, nb_episodes=nb_episodes, visualize=False)
31
32 grafica_entrenamiento(train_history)
33
34 calcula_media_test(dqn)
35
36 # colab 397.328 segundos
37 # local 754.133 segundos

```

```

❏

```


[illegible]

