```
import numpy as np
import gym
import matplotlib.pyplot as plt
import keras
from keras.models import Sequential
from keras.layers import Dense
from collections import deque
import random
import tensorflow as tf

env = gym.make("CartPole-v1")
state = env.reset()


model = Sequential()
model.add(Dense(units = 50, input_dim=4, activation='relu'))
model.add(Dense(units = 50, activation = "relu"))
model.add(Dense(units = 2, activation = "linear"))

opt = tf.keras.optimizers.Adam(learning_rate=0.001)
#opt = tf.keras.optimizers.SGD(learning_rate=0.001)

model.compile(loss='MSE',optimizer=opt)
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                250

 dense_1 (Dense)             (None, 50)                2550

 dense_2 (Dense)             (None, 2)                 102

=================================================================
Total params: 2,902
Trainable params: 2,902
Non-trainable params: 0
_________________________________________________________________
```

```
train_episodes = 200
epsilon = 0.3
gamma = 0.99
max_steps = 200
state = env.reset()


Loss = []
Rewards = []

for e in range(1, train_episodes+1):
  epsilon = epsilon - (1/train_episodes)
  total_reward = 0
  t = 0

  state = env.reset()
  state = np.reshape(state, [1, 4])

  done = False
  while t < max_steps and done == False:

    Qs = model.predict(state)[0]

    if np.random.rand()<epsilon:
      action = env.action_space.sample()
    else:
      action = np.argmax(Qs)

    next_state, reward, done, _ = env.step(action)
    next_state = np.reshape(next_state, [1, 4])

    total_reward += reward


    if done:
      y = reward
    else:
```

```
        y = reward + gamma*np.max(model.predict(next_state)[0])

      Q_target = model.predict(state)
      Q_target[0][action] = y

      h = model.fit(state,Q_target,epochs=1,verbose=0)

      loss = h.history['loss'][0]

      state = next_state
      t+=1

    print(e," R=",total_reward," L=",loss)
    Rewards.append(total_reward)
    Loss.append(loss)
```

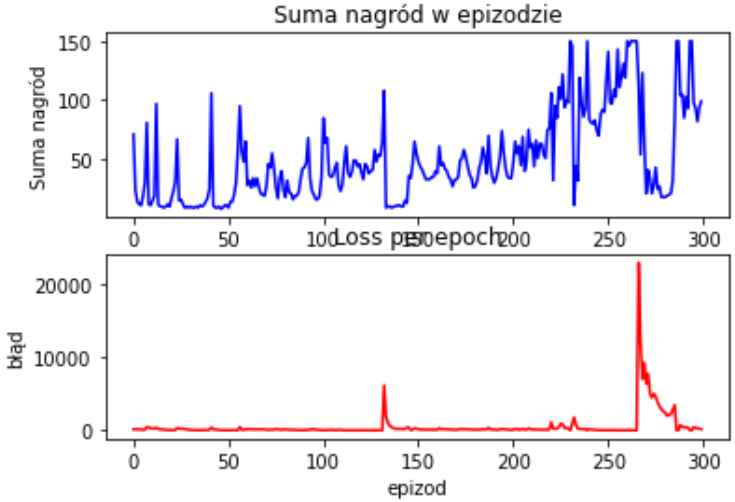
```
 244  R= 83.0  L= 54.84345626831055
 245  R= 74.0  L= 65.49209594726562
 246  R= 70.0  L= 22.112791061401367
 247  R= 88.0  L= 14.043977737426758
 248  R= 92.0  L= 5.582646369934082
 249  R= 90.0  L= 10.464609146118164
 250  R= 117.0  L= 11.7564697265625
 251  R= 141.0  L= 13.428383827209473
 252  R= 98.0  L= 29.028762817382812
 253  R= 97.0  L= 20.156414031982422
 254  R= 109.0  L= 27.710723876953125
 255  R= 103.0  L= 27.634885787963867
 256  R= 143.0  L= 24.0050106048584
 257  R= 111.0  L= 17.484708786010742
 258  R= 120.0  L= 18.53184700012207
 259  R= 131.0  L= 31.703706741333008
 260  R= 119.0  L= 2.9838271141052246
 261  R= 150.0  L= 0.006170779466629028
 262  R= 146.0  L= 45.91840744018555
 263  R= 150.0  L= 0.03624185174703598
 264  R= 150.0  L= 0.003223076229915023
 265  R= 150.0  L= 0.18602710962295532
 266  R= 150.0  L= 0.01925271935760975
 267  R= 114.0  L= 22892.521484375
 268  R= 54.0  L= 12187.412109375
 269  R= 123.0  L= 6994.80126953125
```

```
270  R= 65.0  L= 9211.603515625
271  R= 21.0  L= 6349.32470703125
272  R= 41.0  L= 7736.923828125
273  R= 34.0  L= 5092.65869140625
274  R= 21.0  L= 4446.3828125
275  R= 28.0  L= 4984.30517578125
276  R= 43.0  L= 4644.6357421875
277  R= 25.0  L= 3807.314453125
278  R= 27.0  L= 3268.956298828125
279  R= 18.0  L= 2869.541015625
280  R= 18.0  L= 2610.567626953125
281  R= 18.0  L= 2404.203125
282  R= 19.0  L= 1970.218017578125
283  R= 20.0  L= 2122.2021484375
284  R= 21.0  L= 2151.3740234375
285  R= 32.0  L= 2816.465576171875
286  R= 82.0  L= 3442.446044921875
287  R= 150.0  L= 0.5696285963058472
288  R= 150.0  L= 0.6133140921592712
289  R= 104.0  L= 718.2807006835938
290  R= 105.0  L= 449.3679504394531
291  R= 85.0  L= 415.5384826660156
292  R= 103.0  L= 386.5171813964844
293  R= 93.0  L= 376.5469970703125
294  R= 150.0  L= 0.0018701031804084778
295  R= 150.0  L= 0.01939445547759533
296  R= 98.0  L= 437.73248291015625
297  R= 93.0  L= 325.9156494140625
298  R= 82.0  L= 289.3359375
299  R= 94.0  L= 203.66070556640625
300  R= 99.0  L= 141.95652770996094
```

```
plt.subplot(211)
plt.ylabel('Suma nagród')
plt.title('Suma nagród w epizodzie')
plt.plot(list(range(train_episodes)),Rewards,"b")

plt.subplot(212)
plt.xlabel('epizod')
plt.ylabel('błąd')
plt.title('Loss per epoch')
plt.plot(list(range(train_episodes)),Loss,"r")
```

```
plt.show()
```



✓ 0 s ukończono o 10:49