```
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()
```

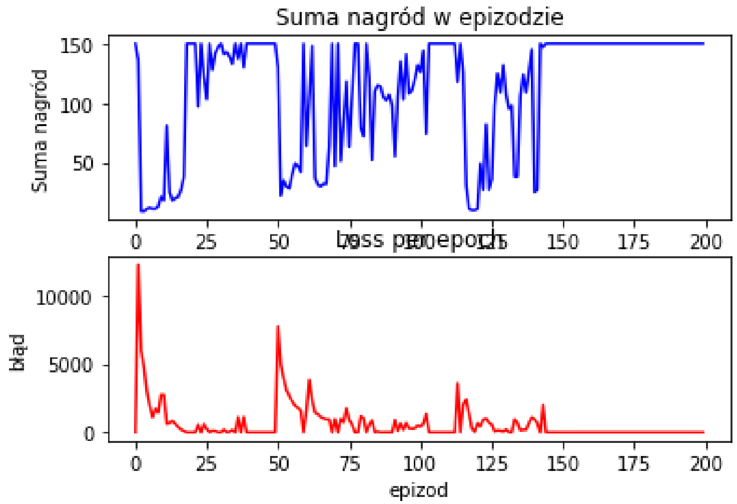
```
    Model: "sequential"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #
    =================================================================
     dense (Dense)               (None, 50)                250

     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 = 0.25
gamma = 0.99
max_steps = 150
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)
```

```
143  R= 150.0  L= 0.11849882453680038
144  R= 147.0  L= 2013.1258544921875
145  R= 150.0  L= 1.2475459575653076
146  R= 150.0  L= 0.003960222937166691
147  R= 150.0  L= 0.02054009772837162
148  R= 150.0  L= 7.553378236480057e-05
149  R= 150.0  L= 0.0029191207140684128
150  R= 150.0  L= 0.0003694722254294902
151  R= 150.0  L= 0.019431287422776222
152  R= 150.0  L= 0.030050568282604218
153  R= 150.0  L= 0.019200356677174568
154  R= 150.0  L= 0.0009016470285132527
155  R= 150.0  L= 0.0863410010933876
156  R= 150.0  L= 0.005154371727257967
157  R= 150.0  L= 0.0020899074152112007
158  R= 150.0  L= 0.07917571812868118
159  R= 150.0  L= 0.13496428728103638
160  R= 150.0  L= 0.04520944133400917
161  R= 150.0  L= 0.19915544986724854
162  R= 150.0  L= 0.025776028633117676
163  R= 150.0  L= 0.18915341794490814
164  R= 150.0  L= 0.010797116905450821
165  R= 150.0  L= 0.03614896535873413
166  R= 150.0  L= 0.013002798892557621
167  R= 150.0  L= 0.05153247341513634
168  R= 150.0  L= 0.012867813929915428
169  R= 150.0  L= 0.07815288007259369
```

```
    169  R= 150.0  L= 0.07815286007259509
    170  R= 150.0  L= 0.08581867814064026
    171  R= 150.0  L= 0.07210041582584381
    172  R= 150.0  L= 0.10893897712230682
    173  R= 150.0  L= 0.11566350609064102
    174  R= 150.0  L= 0.05209735408425331
    175  R= 150.0  L= 0.23023107647895813
    176  R= 150.0  L= 0.05262323096394539
    177  R= 150.0  L= 0.03142381086945534
    178  R= 150.0  L= 0.17996704578399658
    179  R= 150.0  L= 0.024338100105524063
    180  R= 150.0  L= 0.06861201673746109
    181  R= 150.0  L= 0.06600603461265564
    182  R= 150.0  L= 0.06704125553369522
    183  R= 150.0  L= 0.0007839436875656247
    184  R= 150.0  L= 0.0016101357759907842
    185  R= 150.0  L= 0.5003967881202698
    186  R= 150.0  L= 6.246336852200329e-05
    187  R= 150.0  L= 0.10877522081136703
    188  R= 150.0  L= 0.009823426604270935
    189  R= 150.0  L= 0.008581067435443401
    190  R= 150.0  L= 0.0029302071779966354
    191  R= 150.0  L= 0.0758029967546463
    192  R= 150.0  L= 0.12211009860038757
    193  R= 150.0  L= 0.02781561389565468
    194  R= 150.0  L= 0.006660597398877144
    195  R= 150.0  L= 0.013092768378555775
    196  R= 150.0  L= 0.00022573585738427937
    197  R= 150.0  L= 0.020715204998850822
    198  R= 150.0  L= 0.008053706027567387
    199  R= 150.0  L= 0.00046496305731125176
    200  R= 150.0  L= 0.008598066866397858
```

```
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()
```


Suma nagród w epizodzie
Suma nagród
150
100
50
0
25
50
75
100
125
150
175
200
Loss per epoch
błąd
10000
5000
0
0
25
50
75
100
125
150
175
200
epizod