Deep Q Network

مقدمه:

یادگیری تقویتی Reinforcement Learning (RL) یکی از حوزه های یادگیری ماشینی (ML) است. بر خلاف سایر پارادایم های ML، مانند یادگیری تحت نظارت و بدون نظارت، RL با تعامل با محیط خود به صورت آزمون و خطا کار می کند.

RL یکی از فعال ترین حوزه های تحقیقاتی در زمینه هوش مصنوعی است و اعتقاد بر این است که RL ما را یک گام به سمت دستیابی به هوش عمومی مصنوعی نزدیک می کند. RL در چند سال گذشته با طیف گسترده ای از کاربردها از ساخت یک سیستم توصیه تا اتومبیل های خودران به سرعت تکامل یافته است.

دلیل اصلی این تکامل، ظهور یادگیری تقویتی عمیق است که ترکیبی از یادگیری عمیق و RL است. با ظهور الگوریتم‌ها و کتابخانه‌های جدید RL، RL به وضوح یکی از امیدوارکننده‌ترین حوزه‌های ML است.

مراحل درگیر در یک الگوریتم معمولی RL به شرح زیر است:

1. ابتدا عامل با انجام یک عمل با محیط تعامل می کند.
2. با انجام یک عمل عامل از حالتی به حالت دیگر حرکت می کند.
3. سپس عامل بر اساس اقدامی که انجام داده است، پاداش دریافت می کند.
4. بر اساس پاداش، عامل متوجه خوب یا بد بودن عمل می شود.
5. اگر عمل خوب بود، یعنی اگر عامل پاداش مثبتی دریافت کرد، در آن صورت عامل ترجیح می‌دهد آن عمل را انجام دهد، در غیر این صورت، عامل برای جستجوی پاداش مثبت، اقدامات دیگری را انجام می‌دهد.

یک خط مشی **Policy** رفتار عامل را در یک محیط تعریف می کند. خط مشی به عامل می گوید که در هر حالت چه اقدامی را انجام دهد.

عامل با انجام اعمالی با محیط در تعامل است و از حالت اولیه شروع و به حالت نهایی می رسد. این برهمکنش عامل-محیط که از حالت اولیه شروع می شود تا حالت نهایی، یک قسمت**Episode** نامیده می شودوبا T نشان داده می شود.

تابع Q که تابع مقدار حالت-عمل state-action نیز نامیده می شود، مقدار یک جفت حالت-عمل را نشان می دهد. مقدار یک جفت state-action بازدهی است که عامل با شروع از حالت s و انجام عمل a به خط مشی π به دست می آورد. مقدار یک جفت state-action یا تابع Q معمولا با Q(s,a) نشان داده می شود و به عنوان مقدار Q یا مقدار حالت-عمل شناخته می شود. به صورت زیر بیان می شود:

$$Q^{\pi}(s,a) = [R(\tau)|s_0 = s, a_0 = a]$$

هدف از یادگیری تقویتی یافتن خط مشی بهینه است، یعنی سیاستی که حداکثر بازدهی را به ما می دهد (مجموع پاداش های قسمت). برای محاسبه خط مشی، ابتدا تابع Q را محاسبه می کنیم. هنگامی که تابع Q را داریم، با انتخاب یک عمل در هر حالت که دارای حداکثر مقدار Q است، خط مشی(سیاست) بهینه را استخراج می کنیم

می توان محاسبه مقادیر Q را با استفاده از هر تقریبگر تابعی، مانند شبکه عصبی، تقریب زد. بنابراین، ما فقط وضعیت محیط را به یک شبکه عصبی تغذیه می کنیم و مقدار Q تمام اقدامات ممکن در آن حالت را برمی گرداند.

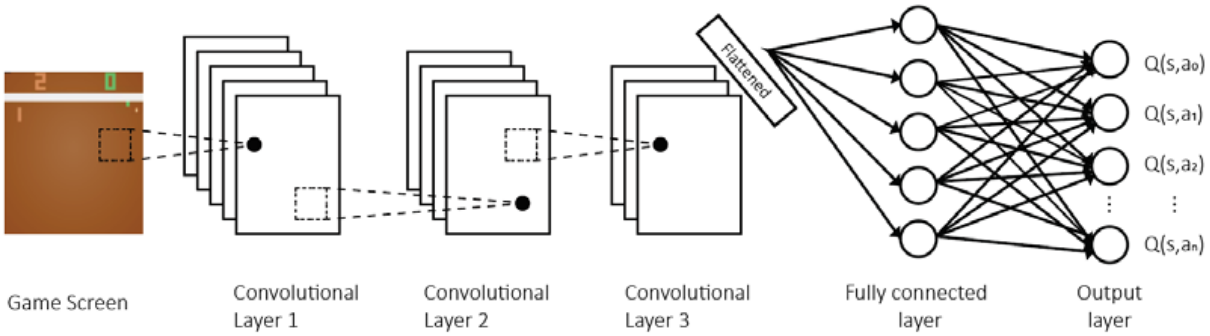
هنگامی که مقادیر Q را به دست آوردیم، می توانیم بهترین عمل را به عنوان عملکردی که حداکثر مقدار Q را دارد انتخاب کنیم

اجرای بازی های Atari2600 با استفاده از DQN

Atari2600 یک کنسول بازی ویدیویی محبوب از یک شرکت بازی سازی به نام Atari است.

کنسول بازی آتاری چندین بازی محبوب مانند Pong، Space Invaders، Ms. Pac-Man، Breakout، Centipede و بسیاری دیگر را ارائه می دهد.

در محیط آتاری تصویر صفحه بازی حالت محیط است. بنابراین، ما فقط تصویر صفحه بازی را به عنوان ورودی به DQN می‌دهیم و مقادیر Q تمام اقدامات موجود در حالت را برمی‌گرداند. از آنجایی که ما با تصاویر سروکار داریم، برای تقریب مقدار Q، می‌توانیم از شبکه عصبی کانولوشنال (CNN) استفاده کنیم



DQN را برای اجرای بازی Ms Pacman پیاده سازی کنیم.

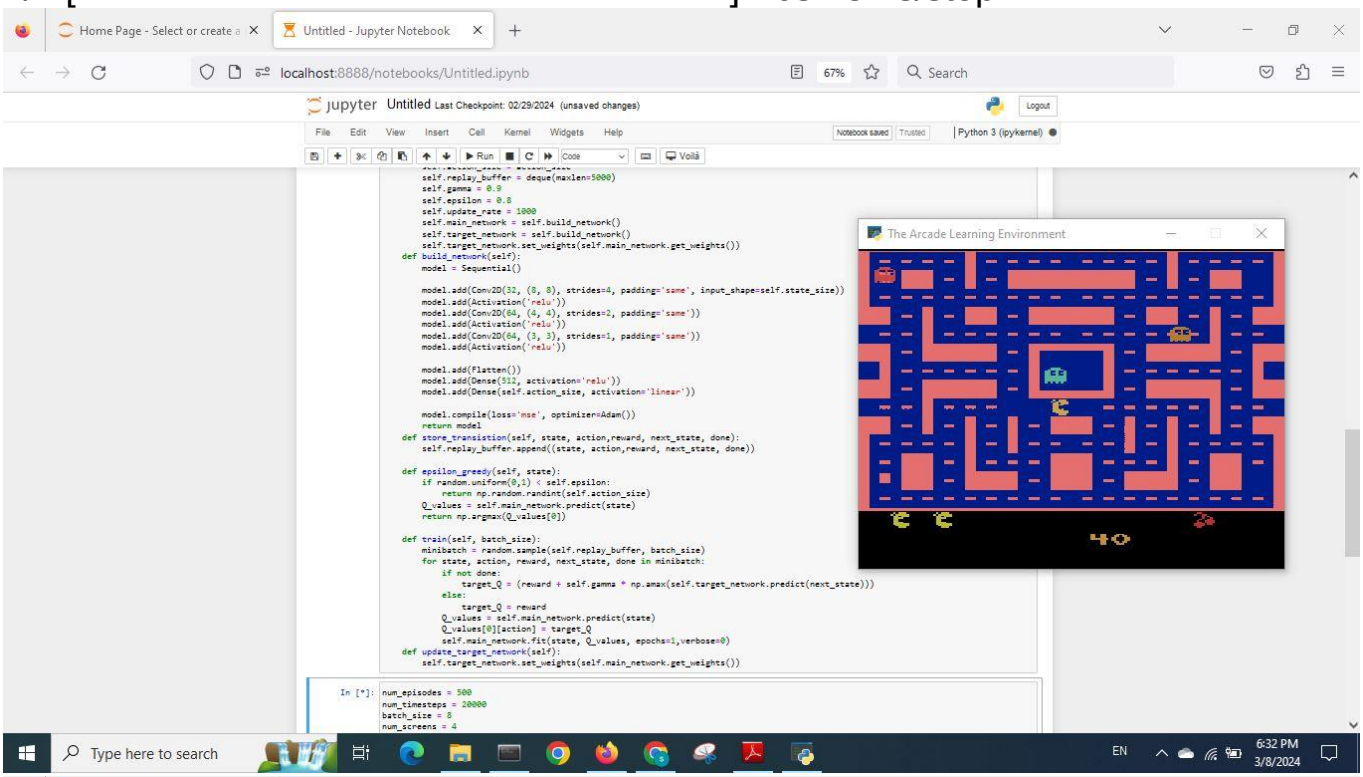
با رندر کردن محیط، همچنین می‌توانیم مشاهده کنیم که عامل چگونه بازی را در یک سری از قسمت‌ها یاد می‌گیرد:

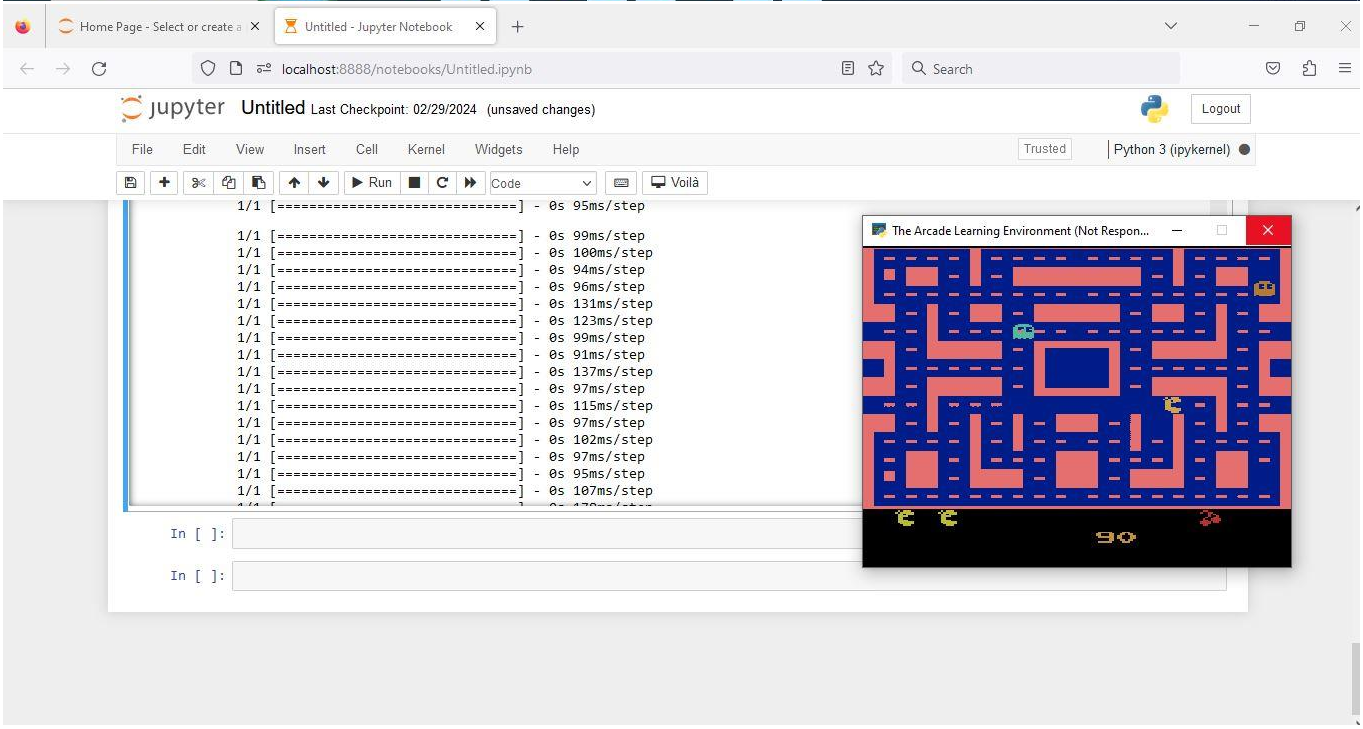
```
import random
import gym
import numpy as np
from collections import deque
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Flatten,
Conv2D,MaxPooling2D
from tensorflow.keras.optimizers import Adam



env = gym.make("MsPacman-v0",render_mode='human')
t1=env.reset()
state_size = (88, 80, 1)
action_size = env.action_space.n
color = np.array([210, 164, 74]).mean()
def preprocess_state(state):
    image = state[1:176:2, ::2]
    image = image.mean(axis=2)
    image[image==color] = 0
    image = (image - 128) / 128 - 1
    image = np.expand_dims(image.reshape(88, 80, 1), axis=0)
    return image


class DQN:
    def __init__(self, state_size, action_size):
        self.state_size = state_size
        self.action_size = action_size
        self.replay_buffer = deque(maxlen=5000)
        self.gamma = 0.9
        self.epsilon = 0.8
        self.update_rate = 1000
        self.main_network = self.build_network()
        self.target_network = self.build_network()
        self.target_network.set_weights(self.main_network.get_weights())
    def build_network(self):
        model = Sequential()
```

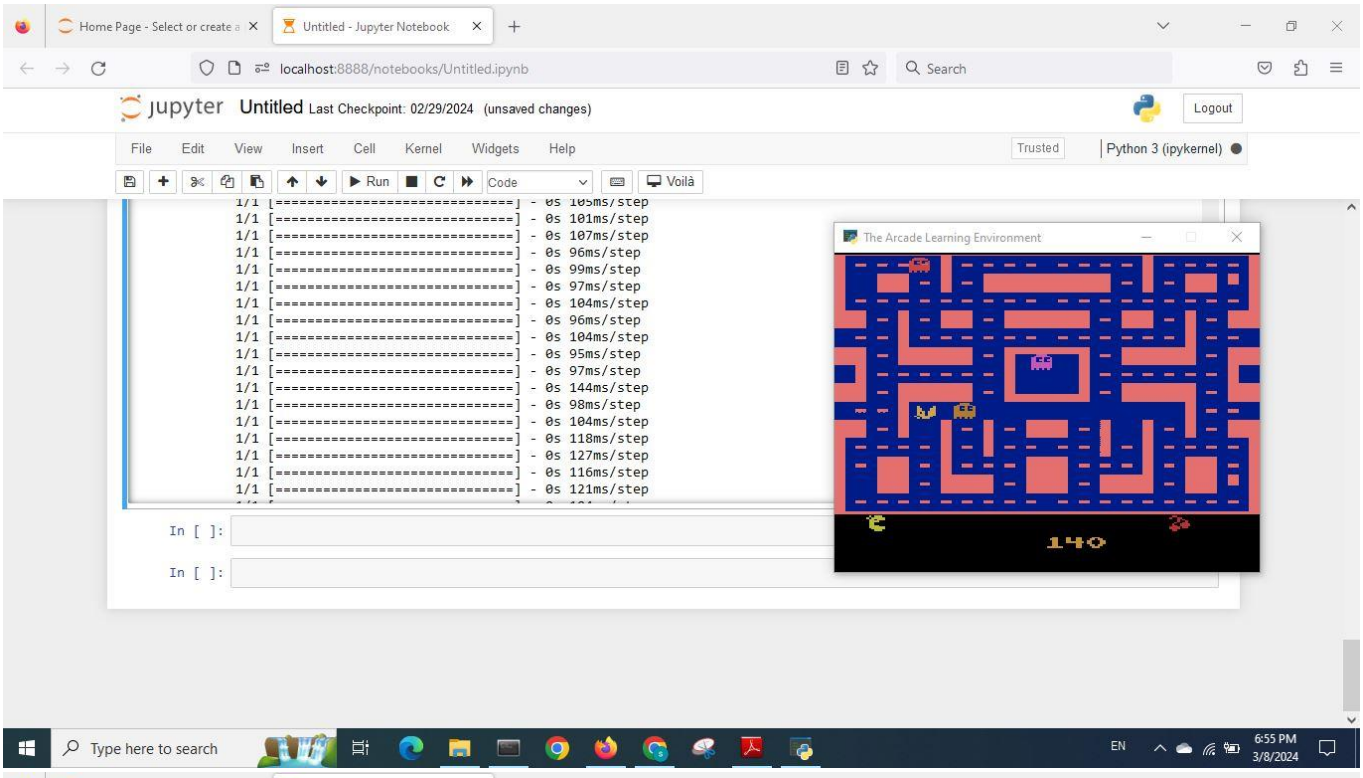
```
        model.add(Conv2D(32, (8, 8), strides=4, padding='same',
input_shape=self.state_size))
        model.add(Activation('relu'))
        model.add(Conv2D(64, (4, 4), strides=2, padding='same'))
        model.add(Activation('relu'))
        model.add(Conv2D(64, (3, 3), strides=1, padding='same'))
        model.add(Activation('relu'))

        model.add(Flatten())
        model.add(Dense(512, activation='relu'))
        model.add(Dense(self.action_size, activation='linear'))

        model.compile(loss='mse', optimizer=Adam())
        return model
    def store_transistion(self, state, action,reward, next_state, done):
        self.replay_buffer.append((state, action,reward, next_state, done))

    def epsilon_greedy(self, state):
        if random.uniform(0,1) < self.epsilon:
            return np.random.randint(self.action_size)
        Q_values = self.main_network.predict(state)
        return np.argmax(Q_values[0])

    def train(self, batch_size):
        minibatch = random.sample(self.replay_buffer, batch_size)
        for state, action, reward, next_state, done in minibatch:
            if not done:
                target_Q = (reward + self.gamma *
np.amax(self.target_network.predict(next_state)))
            else:
                target_Q = reward
            Q_values = self.main_network.predict(state)
            Q_values[0][action] = target_Q
            self.main_network.fit(state, Q_values, epochs=1,verbose=0)
    def update_target_network(self):
        self.target_network.set_weights(self.main_network.get_weights())
num_episodes = 500
num_timesteps = 20000
batch_size = 8
num_screens = 4
```

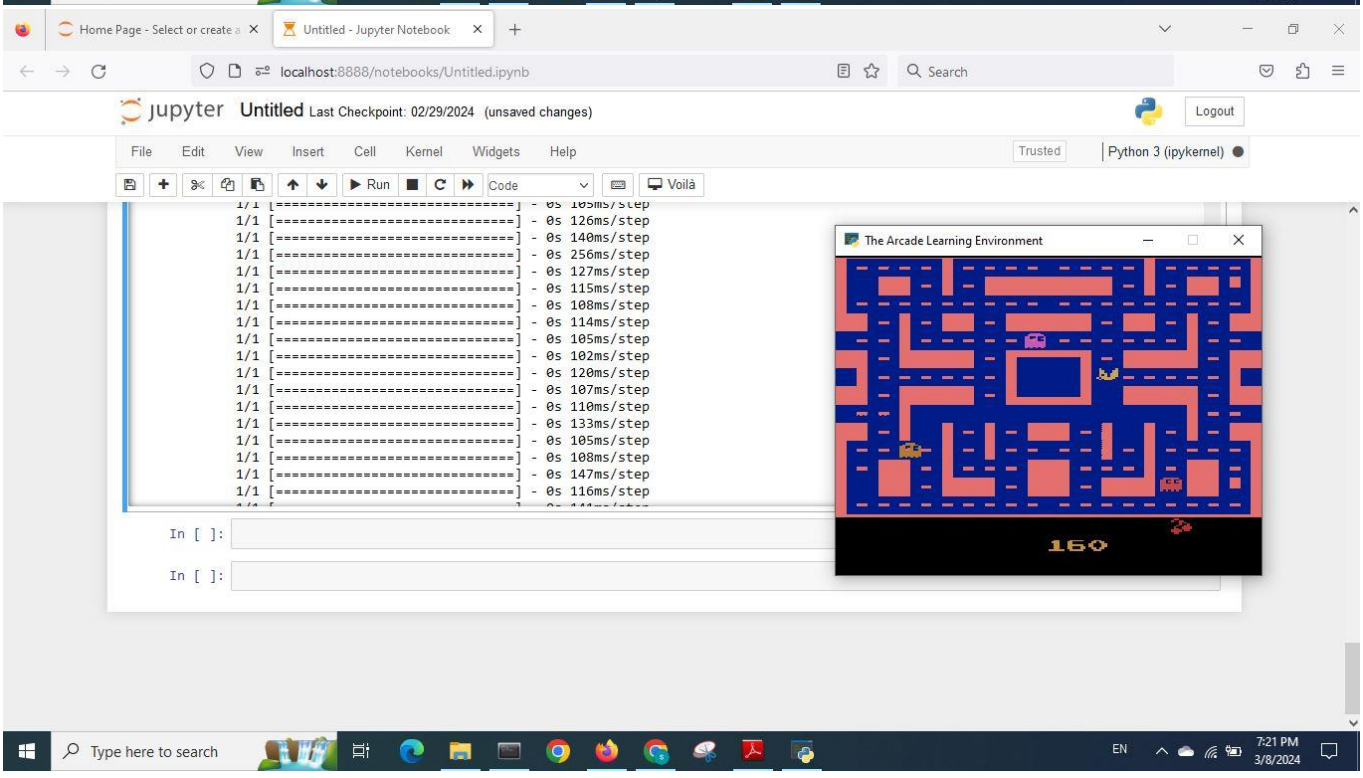
```
dqn = DQN(state_size, action_size)

done = False

time_step = 0
for i in range(num_episodes):
    Return = 0
    state = preprocess_state((env.reset()[0]))
    for t in range(num_timesteps):
        #env.reset()
        env.render()
        time_step += 1
        if time_step % dqn.update_rate == 0:
            dqn.update_target_network()
        action = dqn.epsilon_greedy(state)
        next_state, reward, done,inf,_= env.step(action)
        next_state = preprocess_state(next_state)
        dqn.store_transistion(state, action, reward, next_state, done)
        state = next_state
        Return += reward
        if done:
            print('Episode: ',i, ',' 'Return', Return)
            break
        if len(dqn.replay_buffer) > batch_size:
            dqn.train(batch_size)

1/1 [==============================] - 0s 290ms/step
1/1 [==============================] - 0s 78ms/step
1/1 [==============================] - 1s 703ms/step
1/1 [==============================] - 0s 124ms/step
1/1 [==============================] - 0s 177ms/step
1/1 [==============================] - 0s 135ms/step
1/1 [==============================] - 0s 102ms/step
1/1 [==============================] - 0s 104ms/step
1/1 [==============================] - 0s 81ms/step
1/1 [==============================] - 0s 82ms/step
1/1 [==============================] - 0s 80ms/step
1/1 [==============================] - 0s 85ms/step
1/1 [==============================] - 0s 81ms/step
1/1 [==============================] - 0s 105ms/step
```

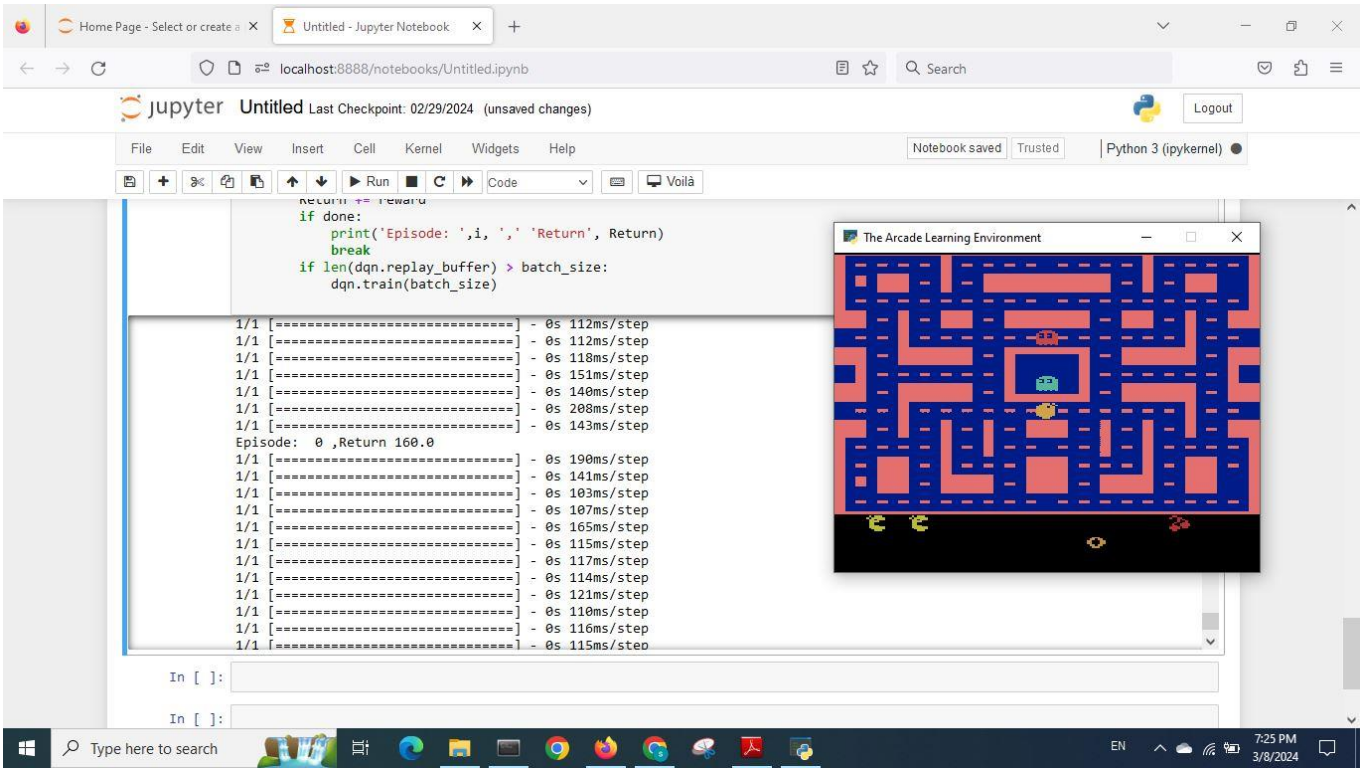
1/1 [==============================] - 0s 76ms/step
1/1 [==============================] - 0s 82ms/step
1/1 [==============================] - 0s 78ms/step






Home Page - Select or create a
Untitled - Jupyter Notebook
localhost:8888/notebooks/Untitled.ipynb
Search
jupyter Untitled Last Checkpoint: 02/29/2024 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Voilà
1/1 [==========] - 0s 101ms/step
1/1 [==========] - 0s 107ms/step
1/1 [==========] - 0s 96ms/step
1/1 [==========] - 0s 99ms/step
1/1 [==========] - 0s 97ms/step
1/1 [==========] - 0s 104ms/step
1/1 [==========] - 0s 96ms/step
1/1 [==========] - 0s 104ms/step
1/1 [==========] - 0s 95ms/step
1/1 [==========] - 0s 97ms/step
1/1 [==========] - 0s 144ms/step
1/1 [==========] - 0s 98ms/step
1/1 [==========] - 0s 104ms/step
1/1 [==========] - 0s 118ms/step
1/1 [==========] - 0s 127ms/step
1/1 [==========] - 0s 116ms/step
1/1 [==========] - 0s 121ms/step
In [ ]:
In [ ]:
The Arcade Learning Environment
140
Type here to search
EN
6:55 PM
3/8/2024


Home Page - Select or create a
Untitled - Jupyter Notebook
localhost:8888/notebooks/Untitled.ipynb
Search
jupyter Untitled Last Checkpoint: 02/29/2024 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Voilà
1/1 [==========] - 0s 126ms/step
1/1 [==========] - 0s 140ms/step
1/1 [==========] - 0s 256ms/step
1/1 [==========] - 0s 127ms/step
1/1 [==========] - 0s 115ms/step
1/1 [==========] - 0s 108ms/step
1/1 [==========] - 0s 114ms/step
1/1 [==========] - 0s 105ms/step
1/1 [==========] - 0s 102ms/step
1/1 [==========] - 0s 120ms/step
1/1 [==========] - 0s 107ms/step
1/1 [==========] - 0s 110ms/step
1/1 [==========] - 0s 133ms/step
1/1 [==========] - 0s 105ms/step
1/1 [==========] - 0s 108ms/step
1/1 [==========] - 0s 147ms/step
1/1 [==========] - 0s 116ms/step
In [ ]:
In [ ]:
The Arcade Learning Environment
160
Type here to search
EN
7:21 PM
3/8/2024



نویسنده رضا افشارنیاکان

Reza.afsharniakan@gmail.com