In [*]:

```
import gym
import random
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
import numpy as np
import torch.optim as optim
import torch as T
import collections
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')


env = gym.make('SpaceInvaders-v0')
env.reset()
height, width, channels = env.observation_space.shape
actions = env.action_space
observation, reward, done, info = env.step(3)
#imgplot = plt.imshow(observation)

print("Enter 'e' for Experience Replay and 'p' for Prioritized Experience Replay:")
ans = input()
```

```
Enter 'e' for Experience Replay and 'p' for Prioritized Experience Replay:
```

In [3]:
```
class DQN(nn.Module):
    def __init__(self, lr, input_dims, fc1_dims, fc2_dims,
                 n_actions):
        super(DQN, self).__init__()
        self.input_dims = input_dims
        self.fc1_dims = fc1_dims
        self.fc2_dims = fc2_dims
        self.n_actions = n_actions
        self.conv1 = nn.Conv2d(in_channels= 3, out_channels= 32, kernel_size= (5,5), str
        self.conv2 = nn.Conv2d(in_channels= 32, out_channels= 64, kernel_size= (3,3), st
        self.conv3 = nn.Conv2d(in_channels= 64, out_channels= 64, kernel_size= (3,3), st
        self.fc1 = nn.Linear(128, self.fc1_dims)
        self.fc2 = nn.Linear(self.fc1_dims, self.fc2_dims)
        self.fc3 = nn.Linear(self.fc2_dims, self.n_actions)

        self.optimizer = optim.Adam(self.parameters(), lr=lr)
        self.loss = nn.MSELoss()
        self.device = 'cpu'
        self.to(self.device)

    def forward(self, state):
        x = self.conv1(state)
        x = F.relu(x)
        x = F.max_pool2d(x,(2, 2))

        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x,(2, 2))

        x = self.conv3(x)
        x = F.relu(x)

        x = torch.flatten(x)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        actions = self.fc3(x)

        return actions
```

In [4]:
```
class Agent():
    def __init__(self,gamma,lr,epsilon,input_dims,n_actions,batch_size,max_mem_size=100
        self.input_dims = input_dims
        self.gamma = gamma
        self.epsilon = epsilon
        self.eps_min = eps_end
        self.lr = lr
        self.n_actions = n_actions
        self.action_space = [0,1,2,3,4,5] #Change this to n_actions for generalised appl
        self.mem_size = max_mem_size
        self.batch_size = batch_size
        self.mem_cntr = 0
        self.beta = 0.4


        self.Q_eval = DQN(lr, input_dims= [100800], fc1_dims = 300, fc2_dims = 100, n_ac

        self.state_memory = np.zeros((self.mem_size,*self.input_dims), dtype = np.float3
        self.new_state_memory = np.zeros((self.mem_size,*self.input_dims), dtype = np.fl
        self.action_memory = np.zeros(self.mem_size, dtype = np.int32)
        self.reward_memory = np.zeros(self.mem_size, dtype = np.float32)
        self.terminal_memory = np.zeros(self.mem_size,dtype= bool)

        self.priorities = collections.deque(maxlen=self.mem_size)

    def store_transition(self,state,action,reward,state_,done):
        index = self.mem_cntr % self.mem_size
        self.state_memory[index] = state
        self.new_state_memory[index] = state_
        self.action_memory[index] = action
        self.reward_memory[index] = reward
        self.terminal_memory[index] = done

        if ans == 'p':
            self.priorities.append(max(self.priorities, default=1))

        self.mem_cntr += 1

    def scaled_prob(self):
        P = np.array(self.priorities, dtype = np.float64)
        P /= P.sum()
        return P

    def prob_imp(self, prob, beta):
        self.beta = beta
        i =  (1/self.mem_size * 1/prob)**(-self.beta)
        i /= max(i)
        return i

    def choose_action(self,state):
        if np.random.random() > self.epsilon:
            state = T.tensor([state]).to(self.Q_eval.device)
            actions = self.Q_eval.forward(state)
            action = T.argmax(actions).item()
        else:
            action = np.random.choice(self.action_space)

        return action

    def sample(self):
        self.beta = np.min([1., 0.001 + self.beta])

        max_mem = min(self.mem_cntr, self.mem_size)
```

```
        probability = self.scaled_prob()
        info = np.random.choice(max_mem, self.batch_size, replace=False, p = probability
        imp = self.prob_imp(probability[info], self.beta)

        return imp, info

    def prop_priority(self, i, err, c = 1.1, alpha_value = 0.7): #for proportional prior
        self.priorities[i] = (np.abs(err) + c)** alpha_value

    def learn(self):
        if self.mem_cntr < self.batch_size:
            return

        self.Q_eval.optimizer.zero_grad()

        if ans == 'p':
            imp, batch = self.sample()

        if ans == 'e':
            max_mem = min(self.mem_cntr, self.mem_size)
            batch = np.random.choice(max_mem, self.batch_size, replace=False)
            batch_index = np.arange(self.batch_size, dtype=np.int32)

        state_batch = T.tensor(self.state_memory[batch]).to(self.Q_eval.device)
        new_state_batch = T.tensor(self.new_state_memory[batch]).to(self.Q_eval.device)

        action_batch = self.action_memory[batch]

        reward_batch = T.tensor(self.reward_memory[batch]).to(self.Q_eval.device)
        terminal_batch = T.tensor(self.terminal_memory[batch]).to(self.Q_eval.device)

        q_eval = self.Q_eval.forward(state_batch)
        q_next = self.Q_eval.forward(new_state_batch)

        q_target = reward_batch + self.gamma*T.max(q_next, dim=0)[0]

        if ans == 'p':
            i = np.arange(self.batch_size)
            diff = T.abs(q_eval - q_target)
            for i in range(self.batch_size):
                idx = batch[i]
                self.prop_priority(idx, diff[i].detach().numpy())

        loss = self.Q_eval.loss(q_target, q_eval).to(self.Q_eval.device)
        loss.backward()
        self.Q_eval.optimizer.step()
```

In [5]:
```
agent = Agent(gamma=0.99,lr=0.001,epsilon=0.1,input_dims= env.observation_space.shape,n_

Q_eval = DQN(lr = 0.001, input_dims= [100800], fc1_dims = 300, fc2_dims = 100, n_actions

x1 = observation[:,:,0]
x2 = observation[:,:,1]
x3 = observation[:,:,2]
observation_new = np.zeros((3,210,160), dtype = np.float32)
observation_new[0] = x1
observation_new[1] = x2
observation_new[2] = x3

state = T.tensor([observation_new]).to(Q_eval.device)
actions = Q_eval.forward(state)
action = T.argmax(actions).item()

action_space = [0,1,2,3,4,5]
action = np.random.choice(action_space)
```

```
C:\Users\a4ama\AppData\Local\Temp/ipykernel_19732/2343478851.py:13: UserWarning: Creat
ing a tensor from a list of numpy.ndarrays is extremely slow. Please consider converti
ng the list to a single numpy.ndarray with numpy.array() before converting to a tenso
r. (Triggered internally at  ..\torch\csrc\utils\tensor_new.cpp:201.)
  state = T.tensor([observation_new]).to(Q_eval.device)
```

In [7]:
```
env.reset()

agent = Agent(gamma=0.99,lr=0.001,epsilon=0.1,input_dims= observation_new.shape ,n_actio
scores,eps_history = [], []
n_games = 100
reward_ = np.zeros(n_games)
reward_avg = np.zeros(n_games)
epsilon = np.zeros(n_games)

for i in range(n_games):
    score = 0
    done = False
    observation = env.reset()
    x1 = observation[:,:,0]
    x2 = observation[:,:,1]
    x3 = observation[:,:,2]
    observation_new = np.zeros((3,210,160), dtype = np.float32)
    observation_new[0] = x1
    observation_new[1] = x2
    observation_new[2] = x3

    while not done:

        action = agent.choose_action(observation_new)
        observation_, reward, done, info = env.step(action)
        x1 = observation_[:,:,0]
        x2 = observation_[:,:,1]
        x3 = observation_[:,:,2]
        observation_new_ = np.zeros((3,210,160), dtype = np.float32)
        observation_new_[0] = x1
        observation_new_[1] = x2
        observation_new_[2] = x3
        #env.render()
        score += reward
        agent.store_transition(observation_new, action, reward, observation_new_, done)
        agent.learn()
        observation_new = observation_new_

    scores.append(score)
    eps_history.append(agent.epsilon)
    avg_score = np.mean(scores[-100:])
    reward_avg[i] = avg_score
    epsilon[i] = agent.epsilon
    reward_[i] = score

    print('episode ', i, 'score%.2f' % score,
          'average score %.2f' % avg_score,
          'epsilon %.2f' % agent.epsilon)
```

```
episode  0 score105.00 average score 105.00 epsilon 0.10
episode  1 score205.00 average score 155.00 epsilon 0.10
episode  2 score125.00 average score 145.00 epsilon 0.10
episode  3 score105.00 average score 135.00 epsilon 0.10
episode  4 score210.00 average score 150.00 epsilon 0.10
episode  5 score250.00 average score 166.67 epsilon 0.10
episode  6 score175.00 average score 167.86 epsilon 0.10
episode  7 score205.00 average score 172.50 epsilon 0.10
episode  8 score75.00 average score 161.67 epsilon 0.10
episode  9 score295.00 average score 175.00 epsilon 0.10
```

```
episode  10 score190.00 average score 176.36 epsilon 0.10
episode  11 score600.00 average score 211.67 epsilon 0.10
episode  12 score450.00 average score 230.00 epsilon 0.10
episode  13 score320.00 average score 236.43 epsilon 0.10
episode  14 score230.00 average score 236.00 epsilon 0.10
episode  15 score125.00 average score 229.06 epsilon 0.10
episode  16 score110.00 average score 222.06 epsilon 0.10
episode  17 score330.00 average score 228.06 epsilon 0.10
episode  18 score125.00 average score 222.63 epsilon 0.10
episode  19 score90.00 average score 216.00 epsilon 0.10
episode  20 score155.00 average score 213.10 epsilon 0.10
episode  21 score255.00 average score 215.00 epsilon 0.10
episode  22 score510.00 average score 227.83 epsilon 0.10
episode  23 score175.00 average score 225.62 epsilon 0.10
episode  24 score70.00 average score 219.40 epsilon 0.10
episode  25 score135.00 average score 216.15 epsilon 0.10
episode  26 score135.00 average score 213.15 epsilon 0.10
episode  27 score215.00 average score 213.21 epsilon 0.10
episode  28 score415.00 average score 220.17 epsilon 0.10
episode  29 score60.00 average score 214.83 epsilon 0.10
episode  30 score175.00 average score 213.55 epsilon 0.10
episode  31 score240.00 average score 214.38 epsilon 0.10
episode  32 score75.00 average score 210.15 epsilon 0.10
episode  33 score475.00 average score 217.94 epsilon 0.10
episode  34 score110.00 average score 214.86 epsilon 0.10
episode  35 score210.00 average score 214.72 epsilon 0.10
episode  36 score115.00 average score 212.03 epsilon 0.10
episode  37 score95.00 average score 208.95 epsilon 0.10
episode  38 score125.00 average score 206.79 epsilon 0.10
episode  39 score55.00 average score 203.00 epsilon 0.10
episode  40 score55.00 average score 199.39 epsilon 0.10
episode  41 score465.00 average score 205.71 epsilon 0.10
episode  42 score215.00 average score 205.93 epsilon 0.10
episode  43 score80.00 average score 203.07 epsilon 0.10
episode  44 score185.00 average score 202.67 epsilon 0.10
episode  45 score275.00 average score 204.24 epsilon 0.10
episode  46 score80.00 average score 201.60 epsilon 0.10
episode  47 score390.00 average score 205.52 epsilon 0.10
episode  48 score155.00 average score 204.49 epsilon 0.10
episode  49 score220.00 average score 204.80 epsilon 0.10
episode  50 score20.00 average score 201.18 epsilon 0.10
episode  51 score305.00 average score 203.17 epsilon 0.10
episode  52 score105.00 average score 201.32 epsilon 0.10
episode  53 score170.00 average score 200.74 epsilon 0.10
episode  54 score160.00 average score 200.00 epsilon 0.10
episode  55 score80.00 average score 197.86 epsilon 0.10
episode  56 score100.00 average score 196.14 epsilon 0.10
episode  57 score480.00 average score 201.03 epsilon 0.10
episode  58 score110.00 average score 199.49 epsilon 0.10
episode  59 score60.00 average score 197.17 epsilon 0.10
episode  60 score55.00 average score 194.84 epsilon 0.10
episode  61 score165.00 average score 194.35 epsilon 0.10
episode  62 score165.00 average score 193.89 epsilon 0.10
episode  63 score1110.00 average score 208.20 epsilon 0.10
episode  64 score425.00 average score 211.54 epsilon 0.10
episode  65 score110.00 average score 210.00 epsilon 0.10
episode  66 score140.00 average score 208.96 epsilon 0.10
episode  67 score380.00 average score 211.47 epsilon 0.10
episode  68 score65.00 average score 209.35 epsilon 0.10
episode  69 score385.00 average score 211.86 epsilon 0.10
episode  70 score105.00 average score 210.35 epsilon 0.10
episode  71 score175.00 average score 209.86 epsilon 0.10
episode  72 score215.00 average score 209.93 epsilon 0.10
```

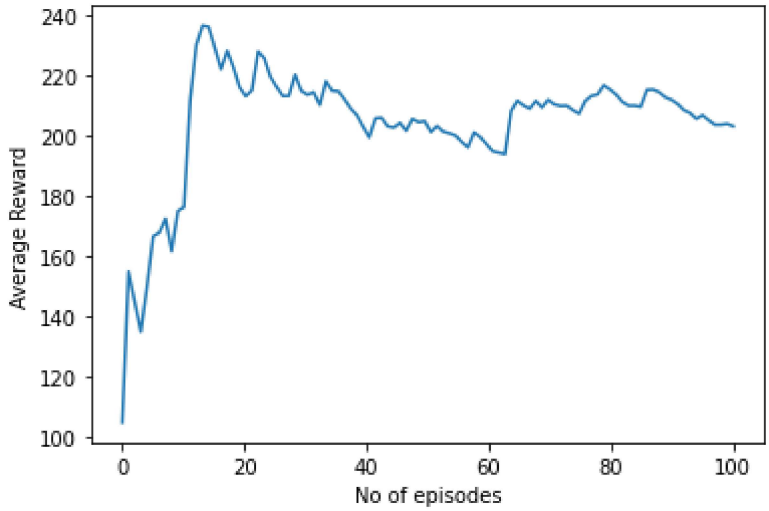
```
episode  73 score100.00 average score 208.45 epsilon 0.10
episode  74 score120.00 average score 207.27 epsilon 0.10
episode  75 score530.00 average score 211.51 epsilon 0.10
episode  76 score340.00 average score 213.18 epsilon 0.10
episode  77 score250.00 average score 213.65 epsilon 0.10
episode  78 score455.00 average score 216.71 epsilon 0.10
episode  79 score105.00 average score 215.31 epsilon 0.10
episode  80 score80.00 average score 213.64 epsilon 0.10
episode  81 score15.00 average score 211.22 epsilon 0.10
episode  82 score105.00 average score 209.94 epsilon 0.10
episode  83 score210.00 average score 209.94 epsilon 0.10
episode  84 score180.00 average score 209.59 epsilon 0.10
episode  85 score690.00 average score 215.17 epsilon 0.10
episode  86 score225.00 average score 215.29 epsilon 0.10
episode  87 score135.00 average score 214.38 epsilon 0.10
episode  88 score65.00 average score 212.70 epsilon 0.10
episode  89 score135.00 average score 211.83 epsilon 0.10
episode  90 score85.00 average score 210.44 epsilon 0.10
episode  91 score25.00 average score 208.42 epsilon 0.10
episode  92 score120.00 average score 207.47 epsilon 0.10
episode  93 score30.00 average score 205.59 epsilon 0.10
episode  94 score315.00 average score 206.74 epsilon 0.10
episode  95 score50.00 average score 205.10 epsilon 0.10
episode  96 score60.00 average score 203.61 epsilon 0.10
episode  97 score205.00 average score 203.62 epsilon 0.10
episode  98 score230.00 average score 203.89 epsilon 0.10
episode  99 score125.00 average score 203.10 epsilon 0.10
```

In [9]:

```
#env.close()
x = np.linspace(0, n_games, num= n_games)
plt.plot(x, reward_avg)
plt.xlabel('No of episodes')
plt.ylabel('Average Reward')
```

Out[9]:

```
Text(0, 0.5, 'Average Reward')
```



In [ ]: