Ch_2(Pong with DQN)

Description:

- Buliding DQN network and train agent to play Pong.

Libraries :

- Installing all necessary packages like (opencv-python _ onnx onnx2pytorch_ pyvirtualdisplay_ gym_minigrid==1.1.0 _ ---- etc)

1)Mini Grid DQN :

- This code trains an agent using the Deep Q-Network (DQN) algorithm for playing Pong. It loads a pre-trained model checkpoint and initializes an optimizer(Adam). It then performs training episodes, where the agent interacts with the environment, collects experiences, and updates the model based on the temporal difference learning rule. The agent's performance and average score are printed during training, and checkpoints of the trained model are saved periodically.
- Recording video of Agent with Episode return 0.955 .

2)PONG DQN _Training:

- **DQN Atari Network**
  - This code defines a Deep Q-Network (DQN) model architecture for playing Atari games. It consists of convolutional layers(num=2) followed by fully connected layers. The convolutional layers extract features from the input image, and the fully connected layers map the features to the Q-values for each possible action. The "forward_"method performs the forward pass of the network, taking an input tensor "x", applying the layers sequentially, and returning the output tensor representing the Q-values for the input state.

- **DQN Algoritm**
  - Using the Deep Q-Network (DQN) algorithm for playing the game of Pong. It initializes variables, performs the training loop over multiple episodes, selects actions based on an epsilon-greedy policy, stores transitions in a replay buffer, updates the policy using temporal difference learning, and prints the episode returns and average score every 50 episodes.

  - **Hyper parameters:**
    - env_id = "PongNoFrameskip-v4"
    - num_episodes = 500 _ buffer_size = 1e5 _ epsilon = 1.0 _ timesteps = 0 _minibatch_size = 128
    - gamma = 0.99 _ eval_episode = 100_ num_eval = 10 _ tau = 1e-3 _learning_rate = 0.00001
    - update_after = 200 _ epsilon_decay = 10**5_ epsilon_ub = 1.0_ epsilon_lb = 0.02

- **Saving model parameters and test on env using 0nnx model**
- **Average return : 18.6**