**Appendix C: Pseudocode of the A2CPER algorithm**

|  |
| --- |
| **Pseudocode of A2CPER Algorithm: (eg:CartPole-v1**) |
| 0: repeat |
| 1: procedure PRIORITIZED\_ACTOR\_CRITIC\_SELF\_ATTENTION(CartPole-v1) |
| 2: Initialize env ← gym.make('CartPole-v1').unwrapped |
| 3: Set seed for reproducibility: env. seed(1), torch.manual\_seed(1) |
| 4: Define state\_space, action\_space, MEMORY\_CAPACITY, batch\_size,  , episodes |
| 5: Initialize Policy Network with Self-Attention (state\_space, action\_space) |
| 6: optimizer ← Adam(policy.parameters, lr = 0.01) |
| 7: prioritized\_replay\_buffer ←  PrioritizedReplayBuffer(MEMORY\_CAPACITY, ) |
| 8: for i\_episode ∈ {1, 2, ..., episodes} do |
| 9: state ← env. reset() |
| 10: for t ← 1 to T do |
| 11: ← policy(state) |
| 12: action ← sample() |
| 13: next\_state, reward, done ← env.step(action) |
| 14: error ← |
| 15: prioritized\_replay\_buffer.store(  (state, action, reward, next\_state, done), error) |
| 16: if len(prioritized\_replay\_buffer) ≥ batch\_size then |
| 17: B, indices, weights ← prioritized\_replay\_buffer.sample(batch\_size) |
| 18: for each idx ∈ indices do |
| 19: ← B[idx] |
| 20: Q\_targets ← |
| 21: ← |
| 22: policy\_loss ← policy\_loss + Δθ |
| 23: end for |
| 24: Update policy using gradient descent: ∇\_θ J(θ) |
| 25: Update priorities in prioritized\_replay\_buffer for indices |
| 26: end if |
| 27: if i\_episode mod plot\_frequency = 0 then |
| 28: PlotProgress(episode\_rewards) |
| 29: end for |
| 30: Evaluate policy by testing it on the environment. |
| 31: end procedure |