Algorithm 2: Pseudocode: Deep Q Learning

Input: Initial ϵ, Final ϵ, ϵ-decay rate, learning_rate α, Reward Discounting Factor γ, Number of Steps n_steps, $n\_rollouts$, $n\_iterations$, $batch\_size$, n_epochs, $target\_update\_freq$, Environment

Initialize Replay Memory $\mathcal{RM}$ to capacity N

Initialize Q and $\hat{Q}$ Network with random weights w

$total\_steps = 0$

$k = 0$

Procedure `rollout()`

  Reset Environment and get x_k

  k = 0

  while $k < n\_rollouts$ **do**

    $u_k \leftarrow \begin{cases} \arg\max_u Q(x_k, u) & \text{probability } 1 - \epsilon \\ \text{Random action} & \text{probability } \epsilon \end{cases}$

    Step Environment with action u_k and get x_{k+1} and r_k

    Add transition (x_k, u_k, r_k, x_{k+1}) to $\mathcal{RM}$

    $total\_steps \leftarrow total\_steps + 1$

    if $total\_steps \,\%\, target\_update\_freq == 0$ **then**

      $\hat{Q} \leftarrow Q$

    end

    $\epsilon \leftarrow \epsilon \times (decay)$

    $k \leftarrow k + 1$

  end

  return

Procedure `learn()`

  $i = 0$

  while $i < n\_epochs$ **do**

    Sample random $batch\_size$ number of transitions(x_j, u_j, r_j, x_{j+1}) from $\mathcal{RM}$

    $\hat{y}_j = \begin{cases} r_j & \text{if } x_{j+1} \text{ is terminal} \\ r_j + \gamma \max_u \hat{Q}_u(x_j, u) & \text{if } x_{j+1} \text{ is not terminal} \end{cases}$

    $y_j = Q(x_j, u_j)$

    Perform gradient descent on $(\hat{y}_j - y)^2$

    $i \leftarrow i + 1$

  end

  return

while $k < n\_iterations$ **do**

  `rollout()`

  `learn()`

  $k \leftarrow k + 1$

end