Algorithm 1 Deep Q-learning with experience replay

Input: $r_{\text{learn}}, n_{\text{rmem}}, \epsilon_{\text{final}}, l_{\text{expl}}, f_{\text{learn}}, f_{\text{update}}, \gamma, n_{\text{batch}}, M$

Replay Memory D を初期化
Q-Network Q をランダムな重み θ で初期化
Target network Q^- を重み $\theta^- = \theta$ で初期化
for episode $= 1, \cdots, M$ **do**
  $t = 1$
  while not *done* **do**
    ε -greedy に従って行動 a_t を選択
    $\epsilon = \max(\epsilon_{\text{final}}, \epsilon - \frac{1-\epsilon_{\text{final}}}{l_{\text{expl}}}) \rightarrow \varepsilon$ を線形減少
    行動 a_t を実行し，報酬 r_t と次の画面 x_{t+1} と *done* を観測
    前処理して次の状態 s_{t+1} を生成
    D に $(s_t, a_t, r_t, s_{t+1}, done)$ を追加，$|D| > n_{\text{nmem}}$ なら古いものを削除する．
    if $t > n_{\text{rpstart}}$ **then**
      if $(t-1)\%f_{\text{learn}} = 0$ **then**
        D からランダムに $(s_j, a_j, r_j, s_{j+1}, done)$ を n_{batch} 個の履歴をサンプル
        $y_j = \begin{cases} r_j & (done) \\ r_j + \gamma \max_{a'} Q^-(s_{j+1}, a'; \theta^-) & (\text{otherwise}) \end{cases}$
        θ を $(y_j - Q(s_j, a_j; \theta))^2$ の勾配方向に学習率 r_{learn} で更新（勾配計算の後の更新の際，r_j は $[-1, 1]$ にクリップされる）
      end if
      if $(t-1)\%f_{\text{update}} = 0$ **then**
        $Q^- = Q$
      end if
    end if
    $t = t + 1$
  end while
end for

Algorithm 2 $reset_{\text{noop}}()$

Input: env, l_{nomax}

エピソードの開始時に，数フレーム何もしない行動を取り，初期状態を決定する．
$T \sim U(1, l_{\text{nomax}})$
for $t' = 1, \cdots, T$ **do**
  $a_{t'}$ = (do nothing) の実行
end for

Output: 初期状態が決定した環境 *env*

Algorithm 3 $step_{\text{repeat}}(a_t)$

Input: $env, l_{\text{history}}, a_t$

1: 1 回行動を取ると，同じ行動を指定フレーム続ける．指定数分行動を繰り返したら，直前のフレームの観測との最大値を状態として返す.
2: ※ a_t は選択したい行動とする．
3: $r_{\text{total}} = 0$
4: **for** $t' = 1, \cdots, l_{\text{history}}$ **do**
5: $\quad s_{\text{prev}} = s_{t'}$
6: $\quad a_{t'} = a_t$ として行動を選択し，環境 env を更新，$(s_{t'+1}, r_{t'}, done_{t'})$ を観測する．
7: $\quad r_{\text{total}} = r_{\text{total}} + r_{t'}$
8: $\quad s_{\max} = \max(s_{\text{prev}}, s_{t'+1})$
9: $\quad done = done_{t'}$
10: **end for**

Output: $s_{\max}, r_{\text{total}}, done$

Algorithm 4 $observe_{\text{gray84}}()$

Input: s_t

1: 観測した画面を (84,84) サイズのグレースケール画像に変換して返す.
2: s_t をグレースケール画像に変換
3: 変換後の s_t をさらに (84,84) に reshape

Output: 変換後の s_t

Algorithm 5 $step_{\text{stack}}(a_t)$

Input: env, k, a_t, S

1: k ステップ数分の観測の履歴を状態として返す.
2: S は観測の履歴とする．
3: a_t を行動として選択し，環境 env を更新，$(s_{t+1}, r_t, done_t)$ を観測する．
4: S に s_{t+1} を追加．$|S| > k$ なら，一番古い履歴を S から削除する．

Output: $S, r_t, done_t$

Algorithm 6 $reward_{\text{clip}}()$

Input: r_t

1: 報酬 r_t が正なら +1 に，負なら −1 に，0 なら 0 として返す.
2: **if** $r_t > 0$ **then**
3: $\quad r_t = 1$
4: **else if** $r_t < 0$ **then**
5: $\quad r_t = -1$
6: **else**
7: $\quad r_t = 0$
8: **end if**

Output: r_t