**Algorithm 1:** The DQN algorithm with experience replay

**Input**: $workflows$; $Amazon\ EC2\ instances$

**Output**: Q-values $Q$, action profile $a$, reward $r$

Initialize replay memory $D$, action-value function $Q$ with random weights;
observe initial state $S$;
**while** *not at max_episode* **do**
  select an action $a$;
  **if** *with probability* $\varepsilon$ **then**
    select a random action;
  **else**
    select replace $a = argmax_{a'} Q(s, a^{'})$ with $r_m$;
  carry out action $a$;
  observe reward $r$ and new state $s^{'}$;
  store experience $< s, a, r, s^{'} >$ in replay memory $D$;
  sample random transitions $< ss, aa, rr, ss^{'} >$ from replay memory $D$;
  calculate target for each minibatch transition;
  **if** $ss^{'}$ *is terminal state* **then**
    $tt = rr$;
  **else**
    $tt$=$rr$+$\gamma$ $max_{a'} Q(ss^{'}, aa^{'})$;
  train the Q-network using $(tt - Q(ss, aa))^2$ as loss;
  $s = s^{'}$
return Q-values $Q$, action profile $a$, reward $r$

**Algorithm 2:** DECENTRALIZED($\Gamma, f, g, \alpha, i$)

**Input**: game $\Gamma$, selection mechanism $f$, decay schedule $g$, learning rate $\alpha$, DQN-based agent $i$

**Output**: values $V$, Q-values $Q$, joint policy $\pi^{i*}$

initialize Q-values $Q$, state $s$, action profile $a$;
**while** *not at max_episode* **do**
  simulate action $a_i$ in state $s$;
  observe action profile $a_{-i}$, rewards $R(s, a)$, and next state $s^{'}$;
  select $\pi^{i*}_{s^{'}} \in f(Q(s^{'}))$;
  **for** *all DQN-based agent* $j$ **do**
    update $V_j(s^{'})$;
    update $Q_j(s, a)$;
  choose action $a_i^{'}$;
  update $s = s^{'}, a = a^{'}$;
  decay $\alpha$ via $g$;
return values $V$, Q-values $Q$, joint policy $\pi^{i*}$ ;