Algorithm 1 DRL-based Decentralized CW Optimization

▷ ### Initialization ###
Initialize the observation buffer, O, with zeroes
Initialize the weights, θ, of the agent
Get the action function, A_θ, which the agent uses to choose the action according to the current stage
Initialize the algorithm's interaction period with the environment, *envStepTime*
Initialize the training stage period, *trainingPeriod*
Set *trainingFlag* $\leftarrow$ *True* to tell the algorithm is in the training stage
Initialize the experience replay buffer, E, with zeroes.
trainingStartTime $\leftarrow$ *currentTime*
lastUpdate $\leftarrow$ *currentTime*
$\mu_{prev}(i) \leftarrow 0$ (previous mean value)
$\sigma^2_{prev}(i) \leftarrow 0$ (previous variance value)
Set *useQueueLevelFlag* $\leftarrow$ *True* to use the averaged normalized transmission queues' level as observation.
$CW \leftarrow 15$

for $t = 1, ..., \infty$ **do**
  for $t = 1, ..., \infty$ **do**

▷ ### Pre-learning stage ###
    $N_t(i) \leftarrow$ get number of transmitted frames
    $N_r(i) \leftarrow$ get number of received frames
    $observation(i) \leftarrow \frac{N_t(i) - N_r(i)}{N_t(i)}$
    O(i).append($observation(i)$)

    if $currentTime \geq lastUpdate + envStepTime$ **then**

▷ ### Learning and operational stages ###
      $\mu(i), \sigma^2(i) \leftarrow preprocess(O(i))$
      $a(i) \leftarrow A_{\theta(i)}(\mu(i), \sigma^2(i), trainingFlag)$
      $CW(i) \leftarrow 2^{a(i)+4} - 1$

      if *trainingFlag* == *True* **then**
        $N_{RP}(i) \leftarrow$ get the number of received packets.
        $tput(i) \leftarrow \frac{N_{RP}}{envStepTime}$
        Send the throughput of each station to the access point.
        $r \leftarrow normalize(tput(i))$
        Broadcast the new r reward value to all associated stations
        $E(i)$.append($(\mu(i), \sigma^2(i), a(i), r, \mu_{prev(i)}, \sigma^2_{prev(i)})$)
        $\mu_{prev(i)} \leftarrow \mu(i)$
        $\sigma^2_{prev(i)} \leftarrow \sigma^2(i)$
        $mb(i) \leftarrow$ get random mini-batch from $E(i)$
        Update $\theta(i)$ based on $mb(i)$
      end if

      lastUpdate $\leftarrow$ *currentTime*
    end if

▷ ### Makes the transition between learning and operational stages ###
    if $currentTime \geq trainingStartTime + trainingPeriod$ **then**
      trainingFlag $\leftarrow$ *False*
    end if
  end for