1 Appendix.B Persudo Code of Neural Q-Learning

Algorithm 1 Asynchronous NAF - N collector threads and 1 trainer thread

Initialize replay memory D and memory size N, training step M
Initialize Q value approximator network weights with random values ω
Initialize the last experience number to train EXP
Initialize Training batch size L
for step $= 1, M$ **do**
  \\collect experience
  action $a_t = \epsilon - \text{greedy}(actionlist)$
  execute a_t and observe state s_{t+1} and reward r_t
  append (s_t, a_t, r_t, s_{t+1}) into memory D
  $s_t = s_{t+1}$
  if step $> EXP$ **then**
    randomly sample a batch of transitions(s_i, a_i, r_i, s_{i+1}) from replay memory D
    $y_i = r_i + \gamma \max'_a Q(s_{i+1}, a'|\omega)$
    train Q value approximator withy_i, s_i pair batch
  end if
end for
for t=1,T **do**
  Execute $\boldsymbol{u}_t$ and observe r_t and $\boldsymbol{x}_{t+1}$
end for

In our experiments, we set training step $M = 30000$. We build a 5-layer feedforward network to serve as the Q-value approximator, the size of each layer is: 135,270,108,52,8. Gredient descent is used to train the network.