Algorithm 1 MERLIN Worker Pseudocode

// Assume global shared parameter vectors θ for the policy network and χ for the memory-based predictor; global shared counter $T := 0$
// Assume thread-specific parameter vectors θ', χ'
// Assume discount factor $\gamma \in (0, 1]$ and bootstrapping parameter $\lambda \in [0, 1]$
Initialize thread step counter $t := 1$
repeat
    Synchronize thread-specific parameters $\theta' := \theta; \chi' := \chi$
    Zero model's memory & recurrent state if new episode begins
    $t_{\text{start}} := t$
    repeat
        Prior $\mathcal{N}(\mu_t^{\text{p}}, \log \Sigma_t^{\text{p}}) = p(h_{t-1}, m_{t-1})$
        $e_t = \text{enc}(o_t)$
        Posterior $\mathcal{N}(\mu_t^{\text{q}}, \log \Sigma_t^{\text{q}}) = q(e_t, h_{t-1}, m_{t-1}, \mu_t^{\text{p}}, \log \Sigma_t^{\text{p}})$
        Sample $z_t \sim \mathcal{N}(\mu_t^{\text{q}}, \log \Sigma_t^{\text{q}})$
        Policy network update $\tilde{h}_t = \text{rec}(\tilde{h}_{t-1}, \tilde{m}_t, \text{StopGradient}(z_t))$
        Policy distribution $\pi_t = \pi(\tilde{h}_t, \text{StopGradient}(z_t))$
        Sample $a_t \sim \pi_t$
        $h_t = \text{rec}(h_{t-1}, m_t, z_t)$
        Update memory with z_t by Methods Eq. 2
        $R_t, o_t^r = \text{dec}(z_t, \pi_t, a_t)$
        Apply a_t to environment and receive reward r_t and observation o_{t+1}
        $t := t + 1; T := T + 1$
    until environment termination or $t - t_{\text{start}} == \tau_{\text{window}}$
    If not terminated, run additional step to compute $V_\nu^\pi(z_{t+1}, \log \pi_{t+1})$
    and set $R_{t+1} := V^\pi(z_{t+1}, \log \pi_{t+1})$ // (but don't increment counters)
    Reset performance accumulators $\mathcal{A} := 0; \mathcal{L} := 0; \mathcal{H} := 0$
    for k from t down to t_{start} **do**
        $\gamma_t := \begin{cases} 0, & \text{if } k \text{ is environment termination} \\ \gamma, & \text{otherwise} \end{cases}$
        $R_k := r_k + \gamma_t R_{k+1}$
        $\delta_k := r_k + \gamma_t V^\pi(z_{k+1}, \log \pi_{k+1}) - V^\pi(z_k, \log \pi_k)$
        $A_k := \delta_k + (\gamma\lambda) A_{k+1}$
        $\mathcal{L} := \mathcal{L} + \mathcal{L}_k$ (Eq. 7)
        $\mathcal{A} := \mathcal{A} + A_k \log \pi_k[a_k]$
        $\mathcal{H} := \mathcal{H} - \alpha_{\text{entropy}} \sum_i \pi_k[i] \log \pi_k[i]$ (Entropy loss)
    end for
    $d\chi' := \nabla_{\chi'} \mathcal{L}$
    $d\theta' := \nabla_{\theta'}(\mathcal{A} + \mathcal{H})$
    Asynchronously update via gradient ascent θ using $d\theta'$ and χ using $d\chi'$
until $T > T_{\max}$