Pseudo-Code:

1. Initialise policy parameters θ and state value function parameters W.

2. For each episode:

    1. Initialise state s

    2. while(s is not terminal) do

        1. Sample action a based on actors policy μ_θ,

            $a \sim \mu_\theta(s, a, \theta)$

        2. Take action a and receive reward r and move to next state s'

        3. Add reward to the episodes total rewards,

            $$E_r = E_r + r$$

            where,

            $E_r = Total\, rewards\, obtained\, for\, the\, current\, episode.$

        4. Calculate $td(0)$ error,

            $$\delta = r + \gamma * V(s') - V(s)$$

        5. Update networks,

            1. Compute total value loss,

                a. $loss_{value} = \delta^2$

            2. Compute total policy loss,

                a. $loss_{policy} = - log(\pi(s, a)) * \delta$

            3. Compute total loss,

                a. $total_{loss} = loss_{value} + loss_{policy}$

4. Update parameters θ, W based on $total_{loss}$

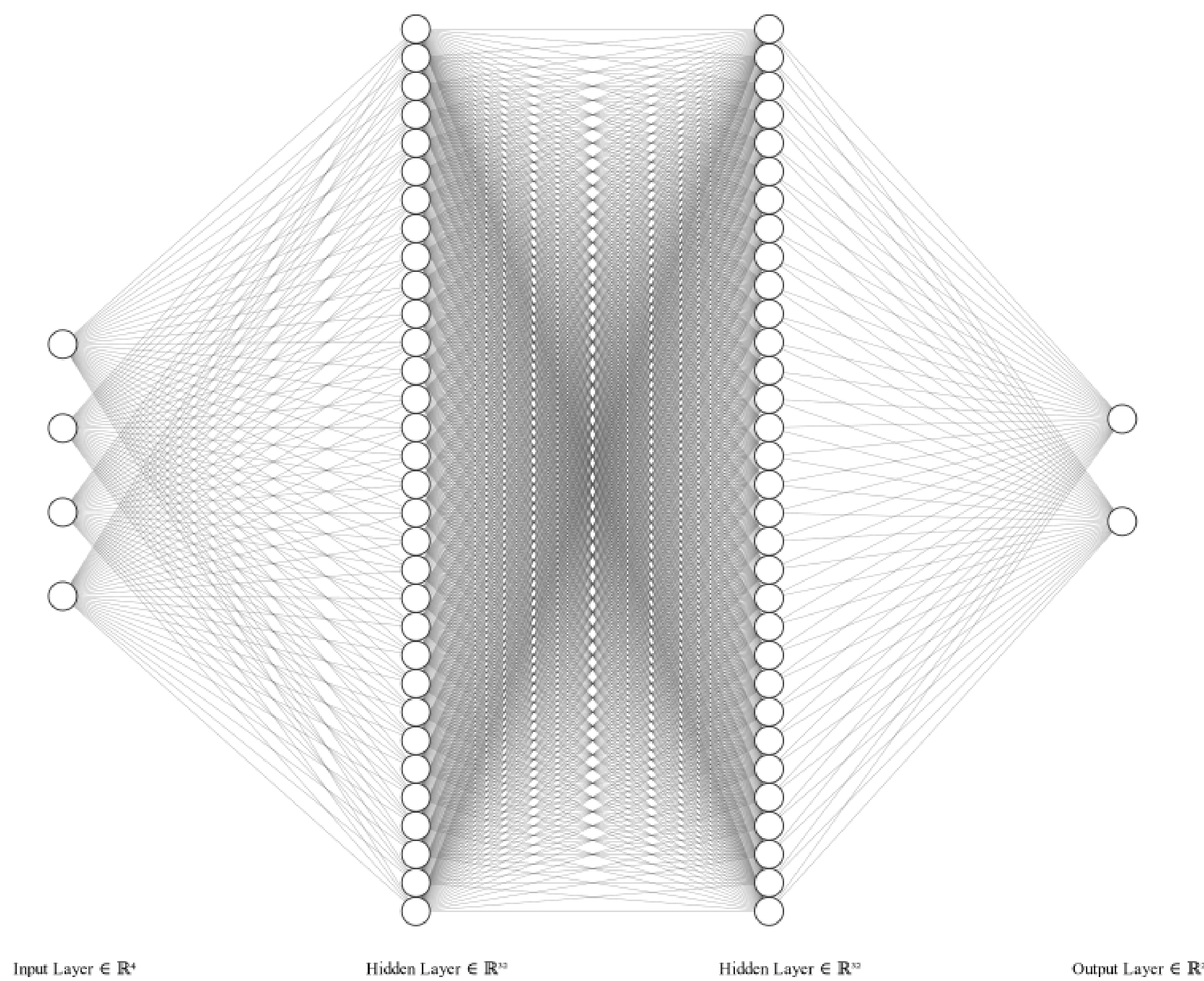
3. Store total episode rewards,

$$episode_{rewards}.append\ (E_r)$$

Network Architectures used:

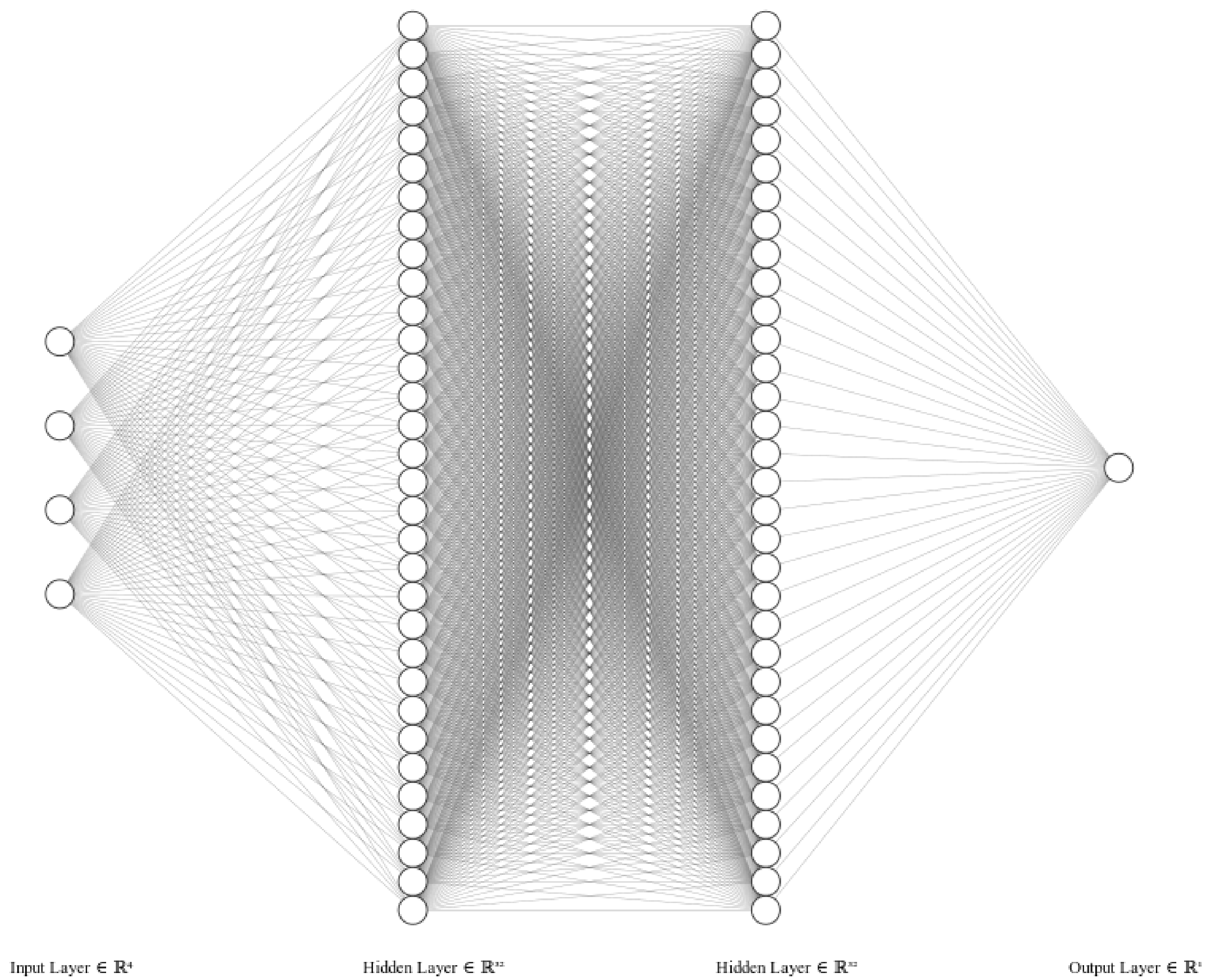
Policy Network: (input: state vector $\phi(s)$ output: probability of the 2 actions $\pi(s, a)$ **)**

- Input layer: 4 nodes
- Hidden layer 1: 32 nodes
- Hidden layer 2: 32 nodes
- Output layer: 2 nodes



Value Network: (input: state vector $\phi(s)$ output: value of state $v(s)$**)**

- Input layer: 4 nodes
- Hidden layer 1: 32 nodes
- Hidden layer 2: 32 nodes
- Output layer: 1 node



Note: Networks do not share any parameters.

Note: Loss functions.

$$L_{critic}(w) = 0.5 \; * \; ||\hat{V}_w - V||^2$$

Where,

$\hat{V}_w = \; Estimated\ value\ function$

$V = \; True\ value\ function$

In our case(td(0) critic updates),

$$V(s_t) = r + \gamma * \hat{V}_w(s_{t+1})$$

$$L_{actor}(\theta) = log(\pi(\theta)) \; * \; \delta$$

Where,

$\pi\,(\theta) \; = \; probability\ of\ taking\ action\ a_t\ at\ time\ t$

$\delta = td(0)\ error\ at\ current\ time\ step$