Reinforcement Learning in Gene Regulatory Network Control

Zhaocheng Yang, CSCI 1470 (Deep Learning), Brown University

Introduction

- Living cells behave like vast switching circuits: each gene is either ON (1) or OFF (0), and the pattern evolves stochastically over time. In medicine and biology, controlling such a gene-regulatory network (GRN) means we want to shift these networks from undesirable states (e.g., cancer-like) to healthy target states by finding the minimal set of gene flips.

- This project frames GRN control as a model-free reinforcement learning (RL) problem: an agent learns to intervene on selected genes to maximize long-term rewards, without knowing the full system dynamics. We study this using probabilistic Boolean networks with up to 100 genes (~2^{100} states), where traditional methods fail due to a combinatorial explosion during the matrix transition.

- Our goal: design a double-DQN with prioritized replay (DDQN-PER) agent that can steer these massive networks toward desired attractors through learned, efficient, and interpretable policies.

- Paper: *Deep Reinforcement Learning for Stabilization of Large-Scale Probabilistic Boolean Networks*

Dataset & Preprocessing

- Single-cell RNA-seq (GSE132188)
  - 11122 mouse cells × 27998 genes

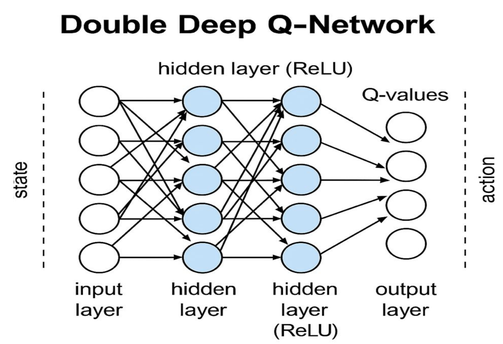
- Preprocessing pipeline
  - Gene Selection: Select the top 100 most variable genes
  - Booleanization: Threshold counts → 0/1 "off / on" states for each gene.
  - Train/Test Split: We sample 8,897 binary cell states for training and 2,225 for evaluation.

n_cells	11122
n_genes_total	27998
selected_genes	100
Train	(8897, 100)
Test	(2225, 100)

SOURCES:
Paper: https://ieeexplore.ieee.org/document/9999487
Dataset: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE132188

Methodology

- Model Overview: We use Double Deep Q-Networks (DDQN) with Prioritized Experience Replay (PER) to learn gene intervention policies in large-scale Probabilistic Boolean Networks (PBNs with N=100).
- RL Formulation:
  a. State: Binary gene expression vector (length 100)
  b. Action: Flip a single gene or take no action (101 choices)
  c. Reward:
     i. +5 for reaching target attractor
     ii. −1 per intervention
     iii. −2 if reaching wrong attractor
  d. Policy: Learned Q-network selects actions to maximize expected return.



- Q-learning & Double DQN: Q-learning learns the action-value function Q(s,a). Double DQN reduces overestimation by separating action selection and value evaluation.
- Network Architecture:
  - Input: 100-dim binary vector
  - Hidden layers: 64 → 64 (ReLU)
  - Output: Q-values for 101 actions
  - Optimizer: Adam (1e−4 learning rate)
- Prioritized Experience Replay: Samples important transitions more often using TD-error. Improves sample efficiency and speeds convergence.
- Training Parameters (some):
  - Buffer size = 200k
  - Batch size = 128
  - ε-greedy policy with ε decay from 1.0 → 0.05

Algorithm 1: DDQN With PER Training Algorithm.

Input: γ, $\min_\epsilon$, $|\mathcal{B}|$, β, ω, α, N, N_{episodes}, N_{epochs}, `horizon`, `batchSize`, c, `updateInterval`
Output: θ^*

$\theta \leftarrow \texttt{rand}([0,1])$, $\theta^- \leftarrow \theta$ ▷ Initialize network weights
$\mathcal{B} \leftarrow \varnothing$, $\texttt{inc}_\beta \leftarrow \frac{\beta}{0.75 \times N_{\text{episodes}} \times N_{\text{epochs}}}$, $\max_p \leftarrow 1$ ▷ Initialize PER
$\epsilon \leftarrow 1$, $\texttt{dec}_\epsilon \leftarrow \frac{1-\min_\epsilon}{N_{\text{episodes}} \times N_{\text{epochs}}}$ ▷ Initialize ϵ-greedy
`trainCount` $\leftarrow 0$
for `epoch` $\in [0, N_{\text{epochs}}]$ **do**
  for `episode` $\in [0, N_{\text{episodes}}]$ **do**
    $t \leftarrow 0$, $s_t \leftarrow \varnothing$
    $\mathcal{X}_t \leftarrow \texttt{rand}(\mathcal{D}^N)$ ▷ Initialize PBN to a random state
    while $s_t \notin \mathcal{Y} \wedge t \neq$ `horizon` **do**
      $s_t \leftarrow \texttt{read}(\mathcal{X}_t)$, $a_t \leftarrow \epsilon\text{-greedy}(\epsilon, s_t)$
      $s_{t+1}, r(s_t, a_t) \leftarrow \texttt{apply}(\text{action})$ ▷ Apply the chosen action to the environment
      `saveToReplayBuffer`$(\mathcal{B}, (s_t, a_t, r(s_t, a_t), s_{t+1}, \max_p))$
      if $|\mathcal{B}| \geq$ `batchSize` **then**
        Sample $(\mathbf{T} = \{\mathbf{S}, \mathbf{A}, \mathbf{R}, \mathbf{S}'\}, \mathbf{W})$ where $\forall i \in \mathbf{T}$, $P(i) = \frac{p_i^\omega}{\sum_{s \in \mathcal{B}} p_s^\omega}$, $\forall w_i \in \mathbf{W}$, $w_i = (|\mathcal{B}| \cdot P(i))^{-\beta}$
        $L(\theta) \leftarrow (\mathbf{R} + \gamma \max_{a'} Q(\mathbf{S}', a'; \theta^-) - Q(\mathbf{S}, \mathbf{A}; \theta)) \cdot \mathbf{W}$
        $\theta \leftarrow \theta - \alpha \nabla_\theta L(\theta)$ ▷ Gradient Descent
        $\forall i \in \mathbf{T}$ update priorities with $p_i' \leftarrow L(\theta) \times c$, $\max_p \leftarrow \max(p_i \forall i \in \mathcal{B})$ ▷ Update PER priorities
        `trainCount` $\leftarrow$ `trainCount` $+ 1$
        if `trainCount` mod `updateInterval` $= 0$ **then** ▷ Update second DQN
          $\theta^- \leftarrow \theta$
        end if
      end if
      $t \leftarrow t + 1$
    end while
    $\beta \leftarrow \min(\beta + \texttt{inc}_\beta, 1)$, $\epsilon \leftarrow \max(\epsilon - \texttt{dec}_\epsilon, \min_\epsilon)$ ▷ Update annealed parameters
  end for
end for

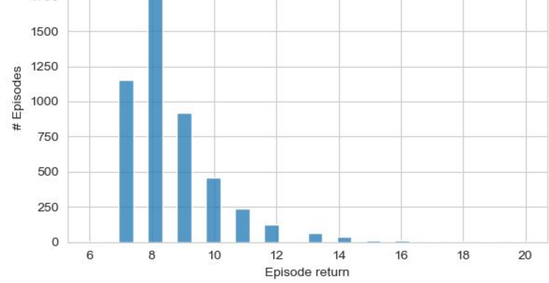
Results

Our results: (5000 evaluation runs)

Metric	Meaning	Value
Success Rate	% of episodes where agent reached the target attractor	100%
Mean Interventions	Avg. number of non-null gene flips before reaching the goal	3.5584
Episode Return	Accumulated reward (goal = +5, intervention cost = –1 per flip)	Avg = 8.5584 Std = 1.5813

Other's results:

Controller	Success rate	Mean # flips	Comment / reference
Dynamic-programming controller (Pal et al., 2006)	100 %	11–100+	Requires solving Bellman equations; scales only to N ≤ 10
Pinning-control heuristics (Lin et al., 2022)	50 – 80 %	10 – 30	Needs pre-selected control nodes; exponential in largest in-degree *d*



- Success ≠ optimality: Even though the agent always succeeds, it doesn't always minimize interventions.
- Interpretability gap: High-performing policy, but hard to biologically interpret why certain flips are made.

Discussion

- Limitations:
  - Fixed intervention cost: The reward function penalizes each gene flip equally, but biological interventions vary in cost, feasibility, and side effects. Our current model does not account for these real-world constraints.
  - Black-box policies: Despite learning effective strategies, deep Q-networks lack interpretability—we don't know *why* certain genes are flipped, or whether the learned policy aligns with known biology.
- Future Directions:
  - Biologically grounded reward shaping: Incorporate *domain knowledge* into the reward structure—e.g., penalize harmful state transitions or prioritize minimal interventions on specific genes of interest.
  - Hybrid symbolic-RL models:Explore *neuro-symbolic systems* that combine learned policies with graph-based regulatory logic or probabilistic model checking for safety-critical control.