Reinforcement learning-guided control strategies for CAR T-cell activation and expansion

Sakib Ferdous[1], Ibne Farabi Shihab[2] , Ratul Chowdhury[1] and Nigel F. Reuel[1]*

1. Department of Chemical and Biological Engineering, Iowa State University
2. Department of Computer Science, Iowa State University

Corresponding Author – Nigel F Reuel

Email – *reuel@iastate.edu

Address –

3051 Sweeney

618 Bissell Rd.

Ames, IA 50011-1098

Phone: 515-294-4592

Sakib Ferdous – ferdous@iastate.edu

Ibne Farabi Shihab – ishihab@iastate.edu

Ratul Chowdhury – ratul@iastate.edu

Running Title - Reinforcement Learning guided CAR T-cell activation

Abstract

Reinforcement learning (RL), a subset of machine learning (ML), could optimize and control biomanufacturing processes, such as improved production of therapeutic cells. Here, the process of CAR-T cell activation by antigen presenting beads and their subsequent expansion is formulated *in-silico*. The simulation is used as an environment to train RL-agents to dynamically control the number of beads in culture to maximize the population of robust effector cells at the end of the culture. We make periodic decisions of incremental bead addition or complete removal. The simulation is designed to operate in OpenAI Gym, enabling testing of different environments, cell types, RL-agent algorithms, and state inputs to the RL-agent. RL-agent training is demonstrated with three different algorithms (PPO, A2C and DQN), each sampling three different state input types (tabular, image, mixed); PPO-tabular performs best for this simulation environment. Using this approach, training of the RL-agent on different cell types is demonstrated, resulting in unique control strategies for each type. Sensitivity to input-noise (sensor performance), number of control step interventions, and advantages of pre-trained RL-agents are also evaluated. Therefore, we present an RL framework to maximize the population of robust effector cells in CAR-T cell therapy production.

Keywords

CAR T-cell, Reinforcement Learning, Autonomous Process Control, Cell Culture, Deep Reinforcement Learning, T-cell Activation

Introduction

CAR T-cell activation is a critical production step for therapeutic cells that is a prime candidate for adaptive control strategies due to their stochastic behavior. As a brief review (Fig 1a), CAR-T cell therapeutics involve the collection and separation of naïve T cells from the patient, transfecting them to produce Chimeric Antigen Receptors (CARs) and expanding them to provide a suitable count of activated cells. These are then infused back into the patient, where they efficiently attack the malignant cells (Finck et al., 2020). Activated T-cells proliferate more rapidly than naïve, so CARs are more readily expressed (Watanabe et al., 2018). One popular approach to activate the cells is by using artificial antigen-presenting beads (aAPC). However, prolonged proximity to aAPCs can lead to cell exhaustion (Gattinoni et al., 2011; Kouro et al., 2022; Piscopo et al., 2018; Wherry & Kurachi, 2015a). Exhausted cells consequently lose reproductive and therapeutic capacity (Wherry & Kurachi, 2015b). The objective of an activation and expansion campaign is to have the maximum number of robust active cells. There is an optimal strategy for activation (bead addition) such that maximum cells remain activated while the number of exhausted cells is minimized (Watanabe et al., 2018). However, such optimal strategies are difficult to model and predict due to variable activation and propagation rates of donor cells based on age and other genetic factors (J. Jiang & Ahuja, 2021; Mehta et al., 2021).

Static recipes are the current standards for aAPC use in CAR T-cell activation with some new attempts at model predictive control. In most cases, beads are added at the beginning of the culture and removed at the end (Levine et al., 2017; Piscopo et al., 2018; Vormittag et al., 2018). Prolonged signaling causes exhaustion, which can be mitigated by halting expression early (Finck et al., 2020; Kouro et al., 2022). It has been observed, that intermittent exposure to beads yields a greater number of robust effector cells; however, the underlying activation-exhaustion mechanism to inform aAPC dosing patterns across all cell populations remains elusive (Kagoya et al., 2017; Philipp et al., 2022). No monitoring or control is involved in the activation process, which could partially explain the loss of potency of manufactured CAR T-cells (Gumber & Wang, 2022).

Model predictive control (MPC) informed by process sensors have the potential to optimize CAR T-cell manufacturing (Mc Laughlin et al., 2023); however, their application is limited by the need of a fully developed process model (Rashedi et al., 2023; Sommeregger et al., 2017). Population dynamic models can provide more cell level spatial and temporal resolution than a mechanistic model (Prybutok et al., 2022). The stochastic nature of cells is difficult to fully model in a predictive fashion, and therefore adaptive control strategies that can update their control policy based on observed cell behavior are well suited. Model free RL algorithms optimize a policy, or value function, instead of modeling the environment. It can learn directly from continuous sensor data and is useful in situations where it is difficult to model the environment.

Reinforcement learning (RL) is an adaptive control strategy for complex environments that do not obey analytical models (Sutton & Barto, 2018). The RL agent discussed in this work is a deep neural network. In the neural net, there is an input layer of neurons at the start, an interconnected, hidden layer of neurons in the middle, and an output layer at the end. Each of these neurons contain adjustable constants, or weights, which are initialized before training. The input data array is multiplied with the weights of each layer in turn and produces an array of numbers choosing either of the permitted action at the output layer. The environment receives the action and responds to it. A reward or penalty is assigned to the RL-agent based on progress towards a desired objective. In the training phase, the weights are adjusted iteratively on basis of the reward it achieves in each training run. Through iterative rounds of training, the neural net settles on weights which maximize likelihood of choosing the output action that achieves the highest reward. RL has been widely used for other stochastic environments such as chatbots (Miner et al., 2020), autonomous vehicles ('Safe Driving Cars,' 2022), robot automation (Han et al., 2023), stock price prediction and projections (Meng & Khushi, 2019), and manufacturing and supply chain control (Rolf et al., 2022). RL-agents can perform better in an actual, physical environments after being trained on incrementally complex simulated environments (Cutler & How, 2016).

Despite being a well-established field, the application of RL to optimize biological systems (Neftci & Averbeck, 2019) is largely untapped. The main reason could be the lack of suitable environments, or digital twin simulations, to train the RL agent and the confounding, inherent variability in biological processes. To benchmark new RL algorithms, OpenAI has established a test platform called Gym (Brockman et al., 2016), with several environments on which new policy algorithms can be tested. There are different environments coded for specific control tasks, for example robo-gym for robotic tasks (Lucchi et al., 2020), panda-gym for multi-goal robotic task (Gallouédec et al., n.d.) and MACAD-gym (Palanisamy, 2019) for self-driving bots. Biological processes have an added level of stochasticity over these physics-based systems. Actions by the RL-agent on a biological environment will produce a stochastic outcome rather than a deterministic one. The first step to testing this approach is to build a suitable ‘digital twin’ test environment.

Multiple efforts have been made in modeling T-cell expansion and activation (Molina Paris & Lythe, 2021). Researchers have attempted defined, analytical models with systems of ordinary differential equations (Bidot et al., 2008). Stochastic approaches have also been proposed where each cell is an autonomous entity governed by its own dynamics (Neve-Oz et al., 2018). None of these models can account for system changes not built into the analytic model or covered by the range of modeled variability. Moreover, thy do not provide an interactive process which can respond to intermittent changes and thus cannot be used to train a self-learning algorithm to develop control rules. Such a simulation framework is needed to train and test the RL agent approach.

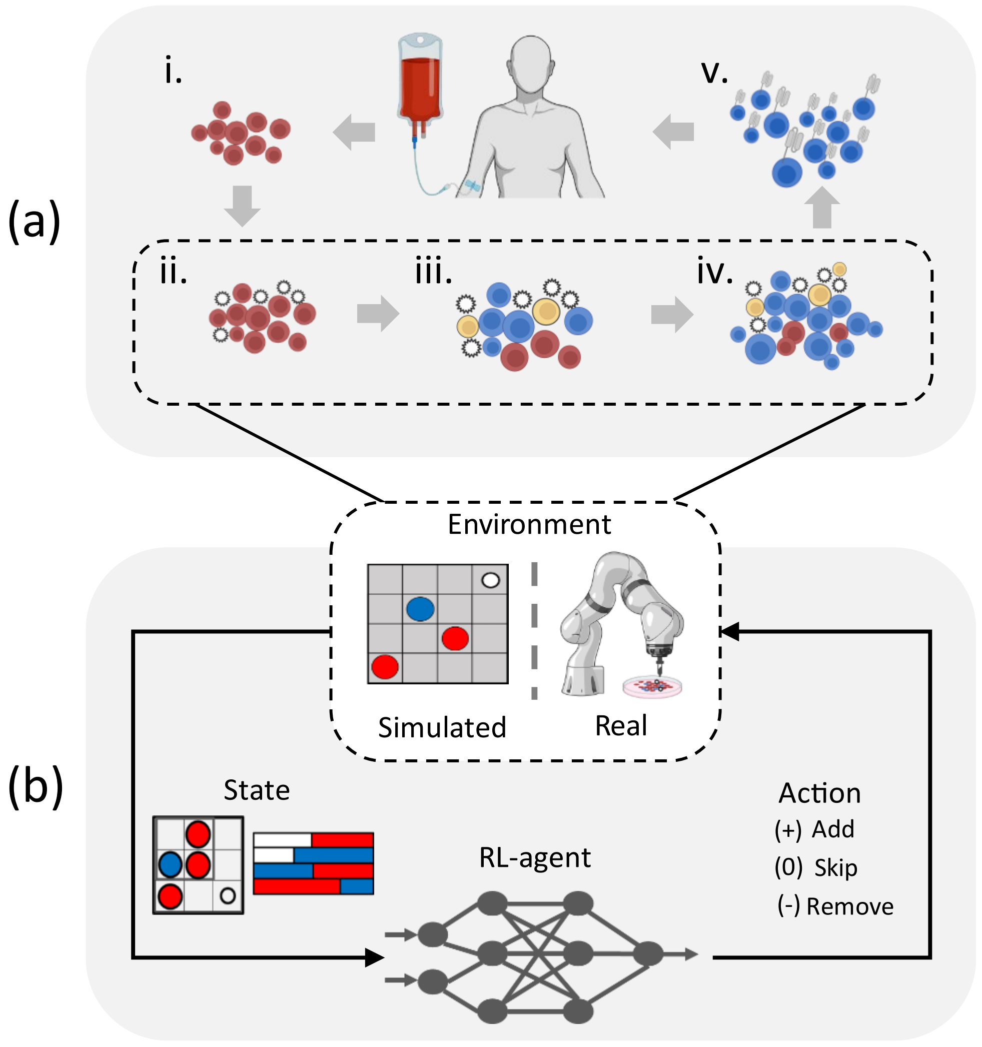


Figure 1: (a) CAR T-cell manufacturing process- i. naïve T-cells (red) are taken out of the body by leukapheresis process, ii. Antigen presenting beads (white spheres with black spikes) are applied to activate the naïve cells, iii. The naïve T-cells are activated (blue), over exposed cells undergo exhaustion (yellow), iv. The activated cells proliferate in number (b) Dynamic, intelligent process control of activation in a simulated cell culture to control real culture with trained policy. The state observation data is collected in tabular, image or combined format as an input to the deep neural network or RL-agent; the agent then selects either of the three permitted actions – add, skip, or remove beads in each control step. Through iterative rounds

of training, the RL-agent learns to map each state to an action which optimizes the end goal of maximum number of robust effector cells.

In this paper, we construct a digital twin test environment for CAR T-cell expansion from individual cell properties and explore the ability of RL agents to determine optimal aAPC bead exposure for varying cell types. CAR T-cell activation and expansion is first coded as a 2D simulation where the RL-agent can decide to add, skip, or remove aAPC to a given population of T cells, with the objective of maximizing count of activated cells at simulation end. The simulation is then converted into a customized gym environment in OpenAI Gym, enabling the testing of several RL algorithms to benchmark policies for this custom environment. An RL agent then settles on an optimized strategy by repeatedly interacting with the environment. Three model-free algorithms –proximal policy optimization (PPO) (Schulman et al., 2017), actor-critic algorithm (A2C) (Mnih et al., 2016), and deep Q-learning network (DQN) (Cruz et al., 2023) are selected as candidate algorithms and are trained in this environment using three different observation space inputs: 1) list of cell counts and other measurable features, 2) image of 2D cell environment, and 3) a combined list-and-image approach (Supplement 5). Different cell types are then used to test how the policies adapt their control strategies of bead dosing. The effect of noise from poor measurement sensors on training efficiency is also tested with observation variables corrupted with Gaussian noise. Finally, the effects of changing the number of times the the RL-agent is allowed to interact with the environment and effects of pre-training agents on control performance are also tested and discussed.

Results

Design of CAR T-cell Simulation

Before attempting to optimize and control a physical system, the bead-based CAR T-cell activation process is simulated as an RL environment in this work (Figure 1a, 2). The simulation is used as a training ground ('environment' in RL language) for the RL-agent algorithm. The objective of the training is to maximize the number of activated CAR T-cells through dynamic control of bead addition and removal. At specified sampling time points, measurable features (or the observation space) from environment are provided to the RL-agent. The observation space includes statistics of the environments (number of naïve or activated cells and robustness of cells from morphology) and process parameters (time elapsed and bead added and/or microscopic image of the culture). The agent policy maps the observations to preferred actions and is iteratively developed by the agent in the training steps. Using the observation space, the agent can decide to add more beads, take away all beads, or refrain from acting at that step (Figure 1b) based on its current policy. The RL-agent then receives a reward or penalty based on number of activated cells which is used to adjust the weights of the neural net underlying the policy.

A 2D surface (Figure 2a) for cell growth is simulated as a continuous $n \times n$ grid with a spacing of 10 microns to match the approximate cell diameter (X. Jiang et al., 2020). In all the simulations, a 50×50 grid corresponding to a 500 by 500 sq-micron area is used. For better clarity in observing the cells (in Figure 2a), a 20 × 20 grid subset is shown for demonstration. The simulated expansion area is made continuous (periodic boundary) to decrease computational cost and approximate a larger area. If a cell exits the simulation grid through one end, it reappears on the opposite end.

All defined parameters for this simulation are described in Table 2. Although attempts were made to associate these parameters with literature values, some assumptions were made for cases where literature or experimental value are yet to be published. It is important to note, that the modular simulation and RL training presented here can be readily updated as more measured values are determined through experiments. A fixed time method (Ruiz Barlett et al., 2009) is used with a value of 6 min per step, derived from the approximate time a cell translates one diameter away or to the next grid spacing (velocity of the cell is ~2 microns per minute (Azarov et al., 2019)). Other factors affecting cellular migration, like media viscosity, age of the cell, size of the cell, etc., are neglected in this simplified model. The total simulation lasts for a 7-day expansion campaign, equivalent to 1600 simulation steps. Bead-to-cell contact, bead-to-cell ratio, and confluence are considered in the simulation rules, considering their role in the activation efficiency (Arman Aksoy et al., n.d.).

At the start of the simulation, the grid is randomly seeded (Figure 2a, n = 1) with a specified number of naïve T-cells indicated as red cells. The following steps are iterated for each cell in the simulation: *Step 1.* It can propagate to any of the eight adjacent cells if it satisfies movement conditions, namely vacancy at the chosen grid and probability of making a move at that step determined stochastically (Figure 2b and Supplement 2). *Step 2:* If a naïve cell occupies a position where an activation bead (coupled to anti-CD3 and anti-CD28 antibodies) is present and if certain conditions (probability of activation at that step beyond a threshold determined stochastically, detailed in Supplement 2) are met, the naïve cell is activated and turns blue in the simulation (Figure 2b). *Step 3:* If an already activated cell gets in a position with a bead, it gets exhausted depending on the value of the specified exhaustion rate (Figure

2b and 2c). *Step 4:* At each timestep, the activated cell is exhausted as natural, transient exhaustion rate (Wherry, 2011) which is ($\frac{\text{natural exhaustion}}{\text{total timesteps}}$) times smaller compared to accelerated exhaustion caused by overexposure and stimulation caused by beads (see Table 1 and Figure 2c). Each cell has several attributes tracked through the simulation, such as activated potency, which starts at 0 with naïve cells and steps to the value of one when activated (Figure 2c). *Step 5:* An activated cell can proliferate under conditions of matured age, potency, and stochastic probability (Figure 2b and detailed in Supplement 2).

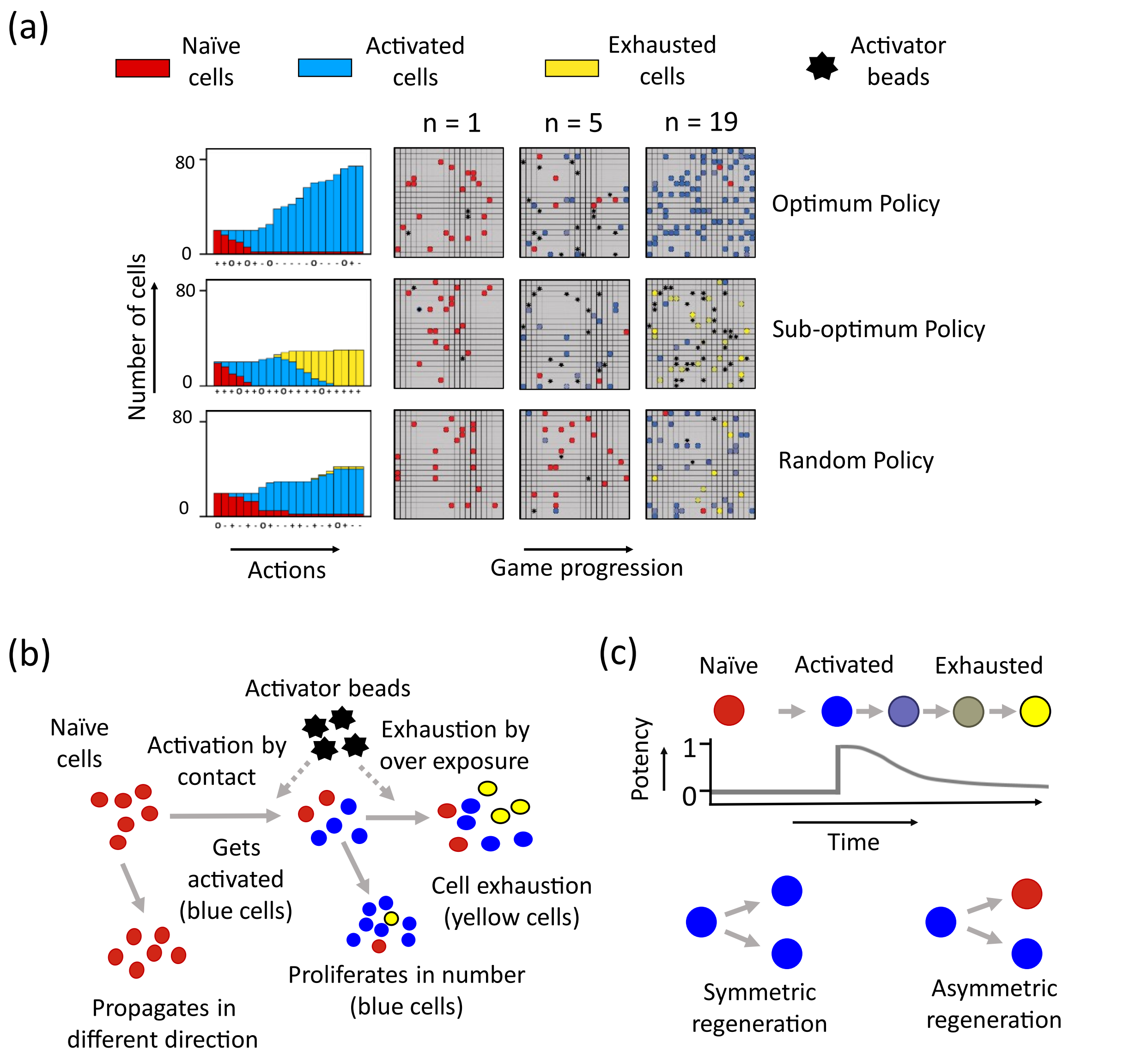


Figure 2: Simulation replicating cell activation and expansion (a) Sample simulation trajectories for three control strategies – top to bottom row depicts optimum, sub-optimum, and random bead additions; the bar plot at left indicates the number of cells separated by type at each simulation step; the symbols at the x-axis represent the action taken: (+) refers to bead addition, (-) refers to the removal and (o) refers to no action; the right three windows are simulation screens at 1, 5 and 19 steps. (b) Process and permitted actions by the cells in each simulated step. (c) Simulated life trajectory of a naïve starting cell to activated with full potency

and natural exhaustion caused by aging. Also defined are two modes of division – symmetric and asymmetric.

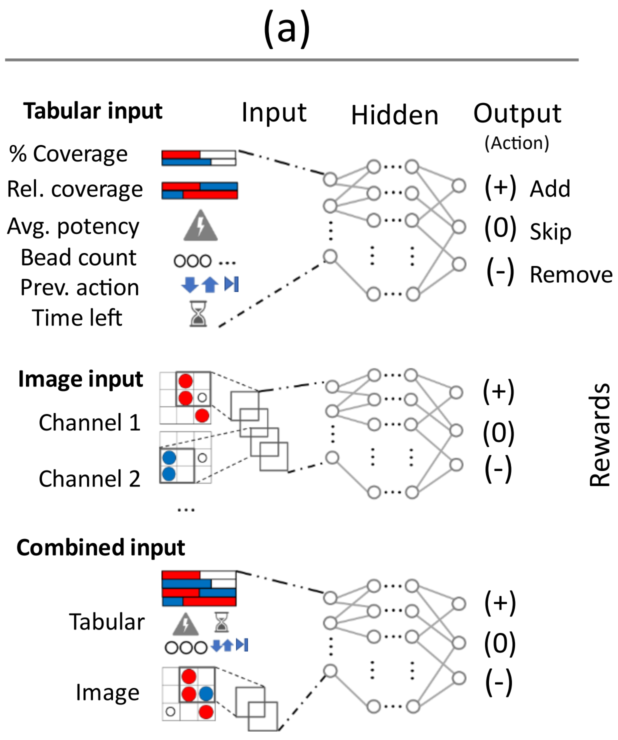
This work has two distinct parts: the CAR T-cell culture simulation (the environment) and the RL algorithm (agent), which updates its policy as it interacts with the environment. The RL-agent can add beads, take out beads, or skip taking any action at the time step. Literature and protocols show that the optimum bead-to-cell ratio varies widely from 3:1 to 9:1 depending on bead and cell type (Trotman-Grant et al., 2021; Zhang et al., 2023). Considering that the system is seeded with 50 cells, ten beads are allowed to be added in each control step (beads can be added in consecutive steps). If a control step occurs every 3.2 or 8 hours, there are 32 and 80 timesteps between actions, and the agent can take a total of 50 and 20 control actions for each seven day simulation respectively. In the case of bead removal, a magnet removes all the beads at once (assuming the use of commercial paramagnetic beads). This is one important real-world constraint where the RL-agent does not have the choice to incrementally add or take out beads; it must add in a specified amount or take out everything at a single step. Based on the properties of the cell (such as regeneration rate, how much it exhausts over time, the chance of getting converted if encountering a bead), the sequence of actions chosen by the agent can be optimal (large population of robust effector cells marked with blue), or sub-optimal (low number of effector cells or low potency effector cells marked with yellow) at the end of all expansion steps (Figure 2b).

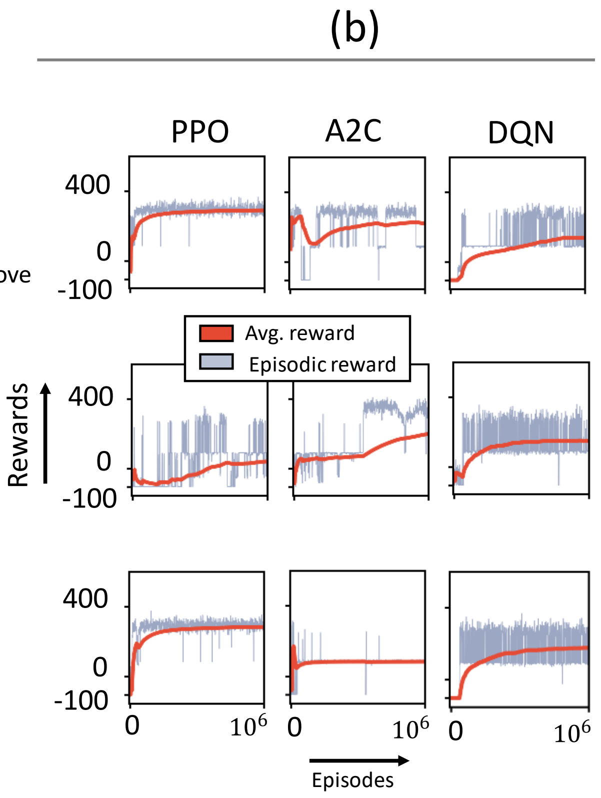
Evaluating RL agent input strategies and algorithms

At each control step, the RL-agent-algorithm takes observational data from the environment and outputs a specified control action using the policy. There are many possible observation data formats that can be provided as input in a real environment. For example, bulk measurements could be made by impedimetric (Liu et al., 2023) (Agilent Xcelligence) or permittivity-based (D'alvia et al., 2022) sensors (Skroot Laboratory). Real-time imaging systems (Espie & Donnadieu, 2023) (Sartorius Incucyte) coupled with Artificial Intelligence (AI) -empowered cell classification tools can specify and quantify cell types based on morphology (Tamiev et al., 2020). Those tools can count naïve and activated cells and other cell properties such as age and robustness. Other data such as time elapsed, quantity of beads in the system and action history can be obtained from the instrument. All the data can be input as a list of measured values to the RL-agent. This method is termed the 'tabular' method in this work (Figure 3a). Another possible observation format can be in the form of an image obtained from high-precision microscopy. In this work, we also try to observe if a three-channel image of the simulation environment, like Figure 2a alone, is enough to provide the agent with enough information to adequately train (Figure 3a) the policy. The third input format tested is the fusion between the above two, where both tabular and image information are provided to the RL-agent (Figure 3a).

Here, we refer to each agent in 'algorithm-input' format; for example, PPO-image refers to an RL-agent trained with PPO algorithm on image data. For three algorithms – PPO, A2C and DQN and three input schemes – tabular, image and combined, in total 9 combinations of 'algorithm-input' is discussed. This analysis aims to demonstrate how RL-agent training depends on algorithms and input schemes.

To improve the decision making of the RL-agent, a reward is tracked through each simulation. Design of a reward function is an empirical, iterative process. The reward function we found to work best is to assign a smaller initial reward to encourage activation by bead addition in the beginning and a large end reward based on number of robust activated cells at simulation end. The summed reward at the end of each episode is the episodic reward, and at each episode we plot the average of all previous episodic rewards, shown in red in Figure 3b. The rising trend of the average reward in the beginning indicates the RL-agent is learning and constantly obtaining a better strategy whereas flattening of the average reward indicates that the RL-agent has settled for an optimized strategy (see PPO-tabular and DQN-tabular input in Figure 3b).





15

Figure 3. Schematic of three different observation space input strategies and learning curves with different RL algorithms used. (a) List of input schemes: tabular input, image input or combined input. (b) Learning curves obtained by training on 3 different reinforcement learning algorithms: PPO, A2C and DQN.

A higher average reward with tight outcome distribution indicates a better-trained RL agent. The policy quality can be determined from the episodic reward distribution of a trained RL-agent. For example, with PPO-tabular and DQN-image (Figure 3b), the RL-agent adopted a stable strategy by 100,000 training episodes as observed from the episodic reward and flattened out average reward. But with PPO-tabular, the episodic reward distribution around the average is +/- 50, whereas it is +/- 250 for DQN-image. That indicates the PPO-tabular RL-agent is better trained, which has a tighter distribution of higher rewards, and the DQN-image is subjected to variability and chance events. The distribution is even tighter for A2C-combined, but the average reward is far less than PPO-combined or DQN-combined. More details on the algorithm are available on Supplement 3-6.

With image input, we tested if it is possible to navigate the environment by simply getting an annotated snapshot of cells and beads with cell type, potency, and age determined from image analysis, and not sending any other data including temporal labels (probing whether the simulation strategy can be step independent). We notice that context information is important. Performance for all algorithms was higher with context data (tabular and combined) than without context data (image only). In all three-input strategies, the nature of DQN was very similar. It settles for a sub-optimal strategy with broader reward distribution (details in *Discussion*). In this work, the default hyperparameters for each neural architecture

(Supplement 3,4,5), as reported in OpenAI Gym, were used without fine-tuning. How an untrained and trained RL-agent navigates the environment is demonstrated in Supplement video 1 and 2 respectively.

Learned control strategies for different cell types and the number of control steps

Next, a PPO-combined RL-agent is tested on each respective cell type, simulating the diversity of patient-derived cells, to assess how the RL-agent can adapt its learned control strategy. The cell parameters are simulated by changing six cell types (Table 1). For each cell type, an RL-agent is first trained for 1M simulations and then used to navigate 1000 simulations on the same 'environment.' The average number of beads in each control step is plotted with standard deviations to reveal the bead addition patterns (Figure 4). The variable actions taken in response to observations (presence of error bars) indicate that the policy is adaptive to navigate different spatial distributions of beads and does not simply memorize and repeat the same actions at each step. In a few instances, there was uniformity of actions (no error bars, same number of beads in all 1000 simulations). The learning curve is also included with the bar plot (insets) indicating that the RL-agent settled for a policy at the end of training (discussed above).

Table 1 Simulated Cell Types

Cell Type	**Exhaustion Rate**	**Activation Probability**	**Natural Exhaustion**	**Reproduction Rate**	**Asymmetric Reproduction**

	$\frac{\text{Unit potency}}{100\text{ collision}})$	$(\frac{\text{Activation}}{100 \times \text{Collisions}})$	$(\frac{\text{Unit potency}}{100 \times \text{timesteps}})$	$(\frac{\text{Regeneration}}{100 \times \text{timesteps}})$	(yes/no)
1(base case)	4	90	1	1	No
2	1	45	1	1	No
3	4	90	10	5	No
4	4	90	10	5	Yes
5	4	90	1	5	No
6	4	90	1	1	Yes

The learned control strategies correlate with intuition for these extreme edge cases. In the base case of Cell 1, to protect the cells from overexposure it removes the beads on the second step after adding the first. The intuitive strategy would be to add the beads in the initial steps and let most of the naïve cells convert and remove the beads when most cells are activated and let them proliferate and increase in number which is what the RL-agent executes with less beads after step 5. With Cell type 2, which has a lower rate of exhaustion than the base case, we observe the RL-agent ramps up a number of beads quicker and maintains a near constant level of exposure until the end when there is another ramp to activate any remaining naïve cells (Figure 4b). Interestingly, the first steps of the RL-agent (the initial ramp) are decisive, with no deviation amongst all runs. Afterwards there are variations in bead number with RL-agent deciding as required to convert the remaining naïve cells. In cell type 3, we simulate a cell with a higher rate of natural exhaustion and regeneration. As exhaustion only applies to active cells, the obvious strategy would be to deliberately delay adding the beads to convert the cells close to the end of the episode. However, as regeneration will be high, the whole region will be crowded with activated cells, so it would be imperative to remove beads and wait for all of them to regenerate as soon as the optimal number of cells get activated. Considering both cases the best strategy would be to add beads in the middle steps and skip the beginning and end steps. This is reflected in the learned strategy of the RL-agent, it skips the first two steps, adds the beads in two repeated steps, then takes out all the beads and waits to make the cells increase in number. With cell type 4, asymmetric regeneration is simulated where an activated cell can produce activated and naïve cells. Beads are required to convert the newly produced naïve cells, but those same beads cause the activated cells to get exhausted. To navigate this system, the RL-agent alternately adds and removes beads, and the overall end score is lower than the other cell types.

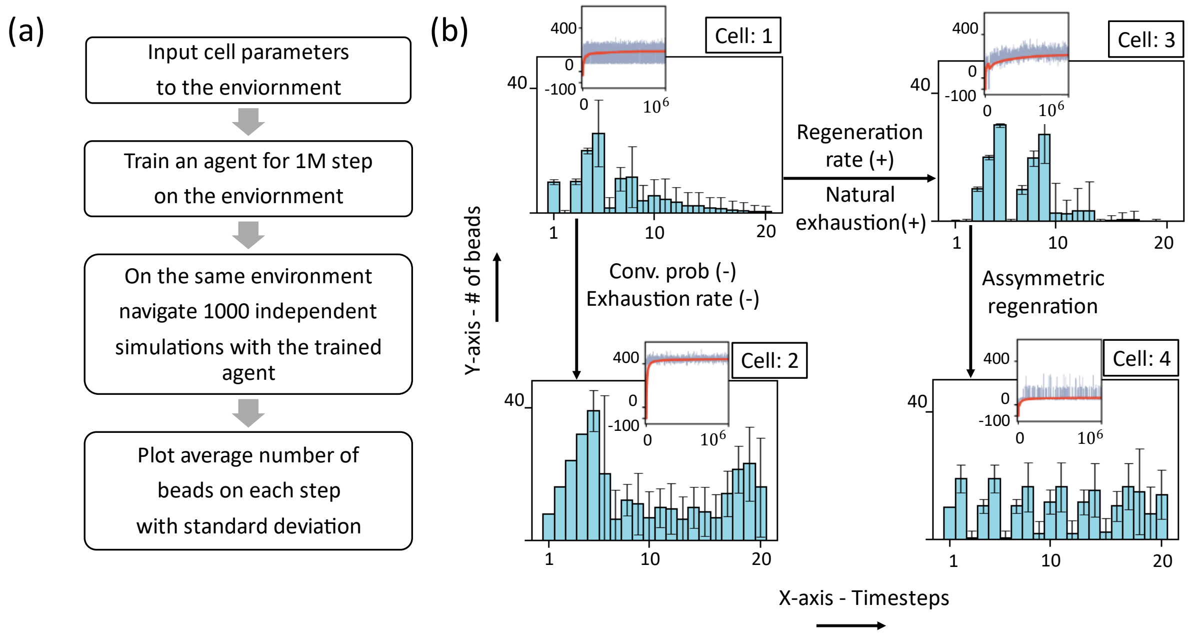


Figure 4: Change of strategy by the RL-agent using 20 control steps for different cell types. (a) Simulation process to obtain control strategy information (b) Strategy of the RL-agent

visualized by average number of beads at each control step (y and x axes respectively). The error bar indicates the standard deviation of beads used at that control step – an indication of simulation variability or constancy (where no bars exist). The learning curve is also attached with each bar plot, axes same as in Figure 3b and snapshot of end stage of a simulation and sample bead and cell population curve is presented for each case at Supplement 9 and Supplement videos. Arrows between plots indicate the change in cell type (also see Table 1).

To test the effect of an RL-agent that has more control over the environment, we repeat the training process with 50 control steps (interacting with the growth vessel every 3.2 hr instead of 8 hr – see justification in Supplement 7) for six cell types (Table 1). The base case behaved the same way, with more dosing of beads in the beginning and reduced in the end (Figure 5). But as it has more frequent control points, the RL-agent skips adding beads at the onset to account for small natural exhaustion, continuously adding beads for the second to the fifth step, then performing the add-remove-skip step depending on the simulated status, with a diminishing number of beads in subsequent steps. For cell type 2, it adds beads for more steps at the outset (Figure 5) than before (Figure 4b) and Cell types 3 and 4 also differ. Cell 5 is simulated with only regeneration increased from the base case, and the RL-agent removes beads in the second half to let the activated cells grow without exhaustion. In cell type 3, the natural exhaustion is increased. To evade the exhaustion the agent adds bead in the later steps and skipping the initial steps. Finally, for cell type 6, we increased the rate of natural exhaustion and added asymmetric regeneration. In this case, the RL-agent alternately adds and removes beads for the first third of the control steps and then ramps the number of beads with variability based on the current cell count; again, the expected outcome (average reward) for this unfortunate cell type is dependent on chance and lower than others.

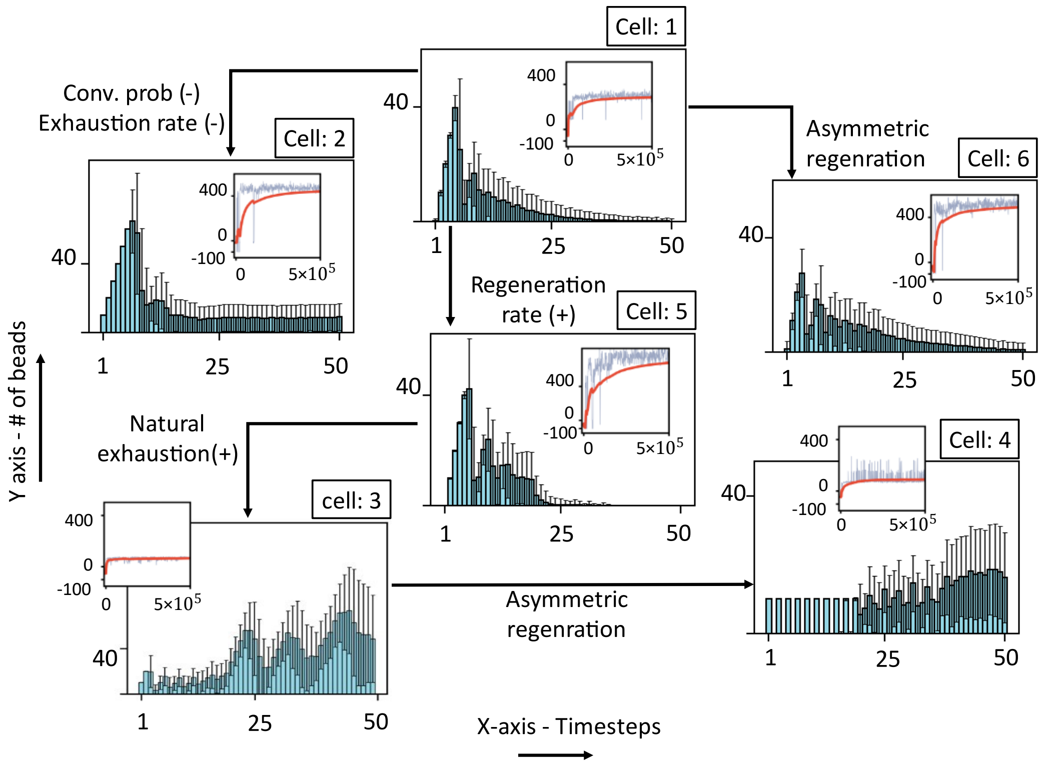


Figure 5: Change of strategy by the RL-agent using 50 control steps learned from training with different cell types. The strategy of the RL-agent visualized by the average number of beads per control step (y and x axes respectively). Error bars indicate one standard deviation, showing variability of steps or uniformity (no error bars). The learning curve is also attached with each bar plot. Arrows indicate the change in cell type; also see Table 1.

Effect of measurement noise, number of control steps, and number of training runs

The ability of an RL-agent to learn unique control strategies for different cell types is a major finding; however, to put this into practice, it will be important to know how accurate the measurements (inputs to the RL-agent) must be as well as the required number of training runs (as 10^6 experiments to determine a unique training regime is not tractable). Here, we explore both topics using the T-cell expansion simulator using the PPO algorithm with combined input.

The observation space for tabular input would be obtained from cell monitoring sensors that distinguish between cell types and estimate potency (optical, impedance, etc.). These devices will not have complete precision. To observe the effect of noise, an RL-agent is trained with 40% of the initial cell number added as Gaussian noise in cell count and potency estimation to simulate measurement error. Interestingly, there is no observable change in the episodic and average reward of the training steps and reward distribution with and without noise (Figure 6a). There are two possible reasons: first, gaussian noise in a stochastic environment does not make a perceivable difference in mapping observation to action, and second, the RL-agent either maps the noise along with the observations or totally disregards the noisy observations and builds its policy on more stable inputs such as time steps. A histogram is also drawn at three stages of training – the zeroth training run, where the RL-agent is fully random, and at 250k and 500k episodes. It is also observed that there is a clear difference in the reward distribution between the random RL-agent at the start and trained RL-agent at 250k runs, but the distribution of rewards at 250k and 500k episodes was indistinguishable.

These experiments (Figures 4 and 5) demonstrate that the RL-agent can perform better with increased interaction with the environment (50 control steps rather than 20). With more interaction, it has better control, and there is a higher reward with less fluctuation, whereas with fewer interactions, it is difficult to control the environment, just like a self-driving car allowed to turn the steering wheel a limited number of time. We investigated if this pattern holds for even further interactions. An RL-agent could interact with a fully automated environment at every observation point. To observe the effect of increased control, we trained an RL-agent with 400 control steps (adding, removing, or maintaining beads every 24 m). In this case, there are an overwhelming 3^{400} possible combinations of action sequences. With such a high number, the agent RL-agent finds it difficult to settle on a control policy, and the learning curve fluctuates more than the 50-control point case (Figure 6b and Supplement 10). This finding indicates that 'real-time' control is likely not as advantageous as a control strategy that is still dynamic yet has a tractable number of possible actions. Agent response with different initial cell numbers is also mentioned in Supplement 11.

In a realized clinical setting, there will likely be a limited number of experiments that can be performed on a new cell type (patient sample) for the RL-agent to self-learn an optimal bead addition strategy. The average learning curve of cell 1 shows 90% of max average reward after 29,000 training sessions for an RL-agent with 50 control steps (Figure 6c). We hypothesized that this number could be further reduced if an RL-agent trained on one cell type is then used as the start point for another cell (e.g., training the RL-agent on a stock cell, before testing with the patient cell sample). To test this approach, the RL-agent is trained on 500k training runs on a base case cell 1 and then used to subsequently train on Cell types 1-4. For cell 1 and cell 2 the optimum strategy is similar – to add beads in the beginning. In that case the RL-agent can adapt faster, and a smaller number of runs (1000 or one updated policy step) is required compared to training from scratch to reach the same level of accuracy. But the optimum strategy is different for cells 3 and 4 – to add beads at the end. In those cases, the RL-agent needs to unlearn the previous strategy and adapt a new strategy. With such a policy change, it takes longer to reach the same level of accuracy rather than starting training from scratch. An alternative or parallel approach to settling on an optimal control strategy would be taking patient cells and performing a series of tests to obtain growth parameters that would allow for building an accurate digital twin to do perform accurate simulation (Figure 1b). Then *in silico* tests, much like this, would augment the physical training data. An *in-silico* test thus can guide if there is a change in policy and weight the choice of – retraining on another cell or training from scratch considering desired yield and resources.

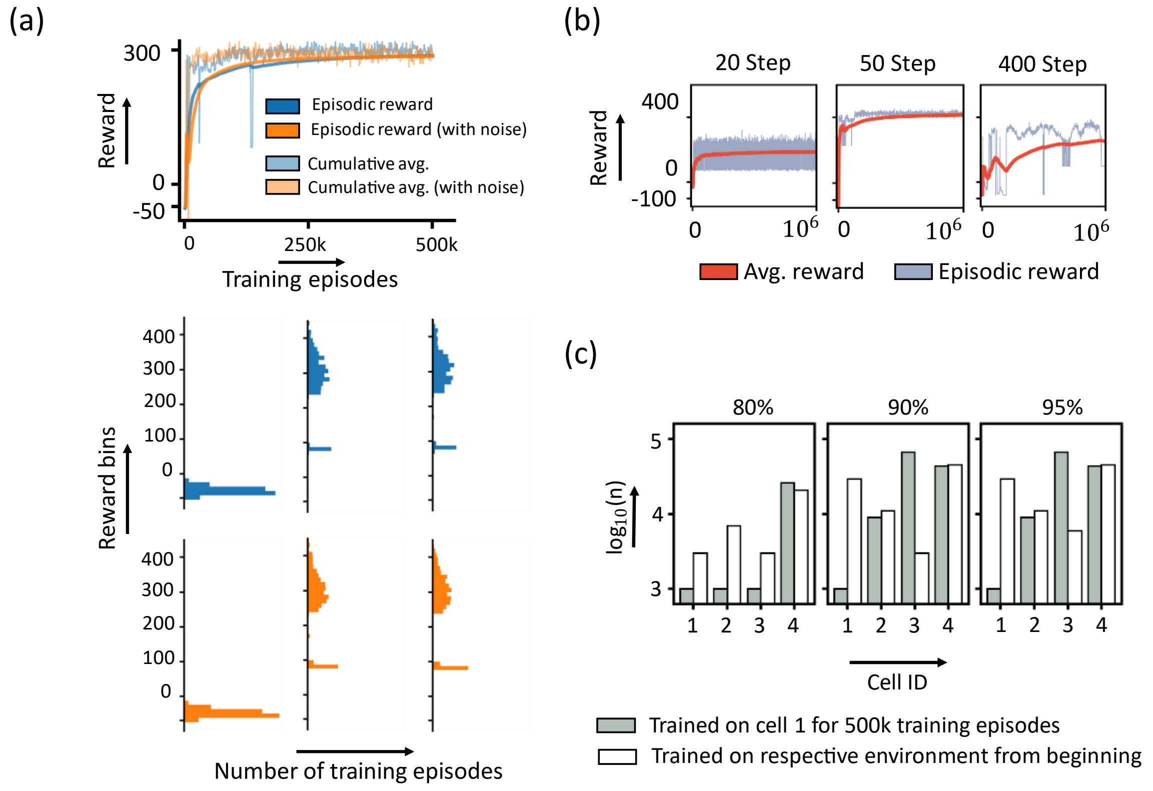


Figure 6: (a) Learning curve for an RL-agent trained with and without noise and reward histogram for simulation conducted with an RL-agent trained on 0, 250k and 500k episodes (b) Learning curve of RL-agent trained with 20, 50 and 400 timesteps (c) Number or training episodes required to reach an accuracy of 80%, 90%, and 95% by RL-agents pre-trained for 500k steps on cell one vs. RL-agents trained on respective cell types from the beginning. Y axis shows the number of training runs required in log base ten scale.

Discussion

Here, we simulate and test an RL-based platform that would help automated cell systems to precisely deploy or remove activator molecules at specific time points during T-cell activation to ensure a maximum number of activated cells (i.e., peak therapeutic potential) before administering them back to the patient. In this work, cell growth parameters were directly inferred from literature to simulate the spatial and temporal stochasticity of CAR T-cell activation and expansion with reasonable fidelity. These simulation parameters should be updated with accurate measurements from the target cell, thereby increasing the accuracy of the simulation. Then before deploying this neural engine (RL agent) for controlling expansion of a patient cell, it would pre-train on the simulated environment thereby reducing the number of training runs required on the physical environment. This work also highlights the utility of non-destructive, continuous measurements from the physical environment (sensor or imaging data) that can be fed as inputs to the RL agent to determine the best dosing policy to maximize activated cells. Continued research on accurate, non-invasive, real-time measurement techniques to enumerate cell types during culture will provide faster training performance. The simulation can inform the type of sensors needed and can also show how much noise the RL-agent can accommodate before it fails to learn anything. With a large amount of measurement noise, the RL-agent will likely (a) disregard the noisy observation parameters (e.g., cell number, cell type, and potency) and (b) fix a redundant policy based only on simulation step count.

One possible reason for the RL-agent's inability to learn solely from discrete image input (Figure 3b) is the lack of connection with the preceding and succeeding time-points. Thus, it becomes impossible to gauge whether a certain action (dosing) helped maximize the number of robust cells. To this end, we anticipate that instead of just providing one disembodied frame, if we exposed the model to short stacks of three to five consecutive frames, the learning rate and gains would improve – but we leave this as an exercise for future work.

This cell-activation routine guided by RL can be used as a template for other model-free, stochastic biological applications. Apart from CAR T-cell activation, this bears promise to control other complex biological policies found in nature, such as the underlying optimization of cell differentiation and proliferation. Improved digital twins of cell culture environments will make this possible. As examples of improvements, this 2D simulation can be updated to a 3D environment representing more realistic growth conditions in static reactors (multilayer growth). Possible further experiments are listed in Supplement 8. In addition, this digital twin model provides a basis to benchmark other machine learning frameworks such as transformer (Vaswani et al., 2017) and DAL-e (Ramesh et al., 2022) based implementations which are finding increasing applicability in different domains of biology. It is foreseeable, that in the near future libraries of pre-trained models would be available to automated cell culturing systems for precision dosing of aAPC to match the range of cell types observed in clinic. Such an approach would de-risk production of therapeutic cells, providing more efficacious therapies to the patients in less time.

Methods

Simulation Design

The simulator of cell expansion was made using the Pygame (Sweigart, 2012) module of Python and is hosted on Zenodo (Ferdous & Shihab, 2023) and GitHub - https://github.com/Sakib1418/Game-of-cells. The simulation was designed to integrate with OpenAI gym (Brockman et al., 2016), a collection of simulated environments and associated toolkits to test and compare RL-agent algorithms. As the new gym environment was made, the Stable Baselines3 module (Raffin et al., 2019) was used on top of the gym to explore current RL algorithms. The properties of the actors (cells) attempt to simulate actual CAR T-cells, for example, movement and regeneration rate. Due to the current lack of measured parameters, such as activation probability on encountering a bead, reasonable estimates are made in this initial work. All simulation values and cell parameters are listed in Table 2. To observe the RL-agent response with different cells, new cell types are conceptualized by changing these cell properties (Table 1). How these parameters are formed into equations governing the fate of the cell and the culture environment or simulation trajectory overall is detailed in the game pseudocode (Supplement 2). The cell parameters could be updated in the script in the repository. The project GitHub repository details the installation of the simulation-game, data analysis, and reproduction of the plots and usage. Reward function design is discussed in Supplement 12.

Table 2: Parameters and their descriptions

Variable Name	**Value**	**Relevant Source**
Simulation Variable		
The initial number of beads	0	(Kagoya et al., 2017)
Grid Number	2500	
Grid dimension	10 Micron	
Confluence	Half of the total grid (1250)	(Arman Aksoy et al., n.d.)
Number of control-steps	20, 50, 400 (variable)	Assumed
Control Time interval	8-hour, 3.2-hour , 24 minutes	(Kagoya et al., 2017)
Number of beads that can be added at each control step	10	(Kagoya et al., 2017; Polonsky et al., 2018; Szopa et al., 2021)
The initial number of naïve cells to begin within the control area	20	Assumed
Total time	160 hours (weeklong growth)	(Piscopo et al., 2018)
Cell Variable		
Mean value of Regeneration Age	Two days	Assumed
Maximum age at which a cell can regenerate	3.5 days	Assumed
Probability of activation	45, 90 ($\frac{\text{Activation}}{100 \times \text{Collisions}}$)	Assumed
Exhaustion Rate	1,4 ($\frac{\text{Unit potency}}{100 \text{ collision}}$)	Assumed
Natural Exhaustion	1, 10 ($\frac{\text{Unit potency}}{100 \times \text{timesteps}}$)	Assumed
Regeneration Rate	1, 5 ($\frac{\text{Regeneration}}{100 \times \text{timesteps}}$)	Assumed
Asymmetric regeneration	True/False	Assumed
Potency value above which a cell is considered robust	0.8	Assumed
Potency value below which a cell is considered exhausted	0.2	Assumed

Reference

Arman Aksoy, B., Czech, E., Paulos, C., & Hammerbacher, J. (n.d.). *Computational and experimental optimization of T cell activation*. https://doi.org/10.1101/629857

Azarov, I., Peskov, K., Helmlinger, G., & Kosinsky, Y. (2019). Role of T Cell-To-Dendritic cell chemoattraction in T Cell priming initiation in the lymph node: An agent-based modeling study. *Frontiers in Immunology*, *10*(JUN). https://doi.org/10.3389/fimmu.2019.01289

Bidot, C., Gruy, F., Haudin, C. S., El Hentati, F., Guy, B., & Lambert, C. (2008). Mathematical modeling of T-cell activation kinetic. *Journal of Computational Biology*, *15*(1), 105–128. https://doi.org/10.1089/cmb.2007.0125

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *CoRR*, *abs/1606.01540*. http://arxiv.org/abs/1606.01540

Cruz, P. J., Vásconez, J. P., Romero, R., Chico, A., Benalcázar, M. E., Álvarez, R., Barona López, L. I., & Valdivieso Caraguay, Á. L. (2023). A Deep Q-Network based hand gesture recognition system for control of robotic platforms. *Scientific Reports*, *13*(1), 7956. https://doi.org/10.1038/s41598-023-34540-x

Cutler, M., & How, J. P. (2016). Autonomous drifting using simulation-aided reinforcement learning. *Proceedings - IEEE International Conference on Robotics and Automation*, *2016-June*, 5442–5448. https://doi.org/10.1109/ICRA.2016.7487756

D'alvia, L., Carraro, S., Peruzzi, B., Urciuoli, E., Palla, L., Prete, Z. Del, & Rizzuto, E. (2022). A Novel Microwave Resonant Sensor for Measuring Cancer Cell Line Aggressiveness. *Sensors*, *22*(12). https://doi.org/10.3390/s22124383

Ferdous, S., & Shihab, I. F. (2023). *CAR T-cell activation control environment in Reinforcement Learning*. https://doi.org/https://doi.org/10.5281/zenodo.7905320

Finck, A., Gill, S. I., & June, C. H. (2020). Cancer immunotherapy comes of age and looks for maturity. In *Nature Communications* (Vol. 11, Issue 1). Nature Research. https://doi.org/10.1038/s41467-020-17140-5

Gallouédec, Q., Cazin, N., Dellandréa, E., & Chen, L. (n.d.). *panda-gym : Open-source goal-conditioned environments for robotic learning*. https://www.franka.de/

Gattinoni, L., Lugli, E., Ji, Y., Pos, Z., Paulos, C. M., Quigley, M. F., Almeida, J. R., Gostick, E., Yu, Z., Carpenito, C., Wang, E., Douek, D. C., Price, D. A., June, C. H., Marincola, F. M., Roederer, M., & Restifo, N. P. (2011). A human memory T cell subset with stem cell-like properties. *Nature Medicine*, *17*(10), 1290–1297. https://doi.org/10.1038/nm.2446

Gumber, D., & Wang, L. D. (2022). *Improving CAR-T immunotherapy: Overcoming the challenges of T cell exhaustion*. https://doi.org/10.1016/j

Han, D., Mulyana, B., Stankovic, V., & Cheng, S. (2023). A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation. *Sensors*, *23*(7), 3762.

Jiang, J., & Ahuja, S. (2021). Addressing Patient to Patient Variability for Autologous CAR T Therapies. *Journal of Pharmaceutical Sciences*, *110*(5), 1871–1876. https://doi.org/https://doi.org/10.1016/j.xphs.2020.12.015

Jiang, X., Dudzinski, S., Beckermann, K. E., Young, K., McKinley, E., J McIntyre, O., Rathmell, J. C., Xu, J., & Gore, J. C. (2020). MRI of tumor T cell infiltration in response to checkpoint inhibitor therapy. *Journal for Immunotherapy of Cancer*, *8*(1). https://doi.org/10.1136/jitc-2019-000328

Kagoya, Y., Nakatsugawa, M., Ochi, T., Cen, Y., Guo, T., Anczurowski, M., Saso, K., Butler, M. O., & Hirano, N. (2017). Transient stimulation expands superior antitumor T cells for adoptive therapy. *JCI Insight*, *2*(2). https://doi.org/10.1172/jci.insight.89580

Kouro, T., Himuro, H., & Sasada, T. (2022). Exhaustion of CAR T cells: potential causes and solutions. In *Journal of Translational Medicine* (Vol. 20, Issue 1). BioMed Central Ltd. https://doi.org/10.1186/s12967-022-03442-3

Levine, B. L., Miskin, J., Wonnacott, K., & Keir, C. (2017). Global Manufacturing of CAR T Cell Therapy. In *Molecular Therapy - Methods and Clinical Development* (Vol. 4, pp. 92–101). Elsevier Inc. https://doi.org/10.1016/j.omtm.2016.12.006

Liu, Z., Jiang, X., Li, S., Chen, J., Jiang, C., Wang, K., Zhang, C., & Wang, B. (2023). A disposable impedance-based sensor for in-line cell growth monitoring in CAR-T cell manufacturing. *Bioelectrochemistry*, *152*, 108416. https://doi.org/https://doi.org/10.1016/j.bioelechem.2023.108416

Lucchi, M., Zindler, F., Muhlbacher-Karrer, S., & Pichler, H. (2020). Robo-gym - An open source toolkit for distributed deep reinforcement learning on real and simulated robots. *IEEE International Conference on Intelligent Robots and Systems*, 5364–5371. https://doi.org/10.1109/IROS45743.2020.9340956

Mc Laughlin, A. M., Milligan, P. A., Yee, C., & Bergstrand, M. (2023). Model-informed drug development of autologous CAR-T cell therapy: Strategies to optimize CAR-T cell exposure leveraging cell kinetic/dynamic modeling. In *CPT: Pharmacometrics and Systems Pharmacology* (Vol. 12, Issue 11, pp. 1577–1590). American Society for Clinical Pharmacology and Therapeutics. https://doi.org/10.1002/psp4.13011

Mehta, P. H., Fiorenza, S., Koldej, R. M., Jaworowski, A., Ritchie, D. S., & Quinn, K. M. (2021). T Cell Fitness and Autologous CAR T Cell Therapy in Haematologic Malignancy. In *Frontiers in Immunology* (Vol. 12). Frontiers Media S.A. https://doi.org/10.3389/fimmu.2021.780442

Meng, T. L., & Khushi, M. (2019). Reinforcement learning in financial markets. *Data*, *4*(3), 110.

Miner, A. S., Laranjo, L., & Kocaballi, A. B. (2020). Chatbots in the fight against the COVID-19 pandemic. In *npj Digital Medicine* (Vol. 3, Issue 1). Nature Research. https://doi.org/10.1038/s41746-020-0280-0

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. *CoRR*, *abs/1602.01783*. http://arxiv.org/abs/1602.01783

molina-paris, C., & Lythe, G. (2021). *Mathematical, Computational and Experimental T Cell Immunology*. https://doi.org/10.1007/978-3-030-57204-4

Neftci, E. O., & Averbeck, B. B. (2019). Reinforcement learning in artificial and biological systems. *Nature Machine Intelligence*, *1*(3), 133–143. https://doi.org/10.1038/s42256-019-0025-4

Neve-Oz, Y., Sajman, J., Razvag, Y., & Sherman, E. (2018). InterCells: A Generic Monte-Carlo simulation of intercellular interfaces captures nanoscale patterning at the immune synapse. *Frontiers in Immunology*, *9*(SEP). https://doi.org/10.3389/fimmu.2018.02051

Palanisamy, P. (2019). Multi-Agent Connected Autonomous Driving using Deep Reinforcement Learning. *CoRR*, *abs/1911.04175*. http://arxiv.org/abs/1911.04175

Philipp, N., Kazerani, M., Nicholls, A., Vick, B., Wulf, J., Straub, T., Scheurer, M., Muth, A., Hänel, G., Nixdorf, D., Sponheimer, M., Ohlmeyer, M., Lacher, S. M., Brauchle, B., Marcinek, A., Rohrbacher, L., Leutbecher, A., Rejeski, K., Weigert, O., … Subklewe, M. (2022). T-cell exhaustion induced by continuous bispecific molecule exposure is ameliorated by treatment-free intervals. *Blood*, *140*(10), 1104–1118. https://doi.org/10.1182/blood.2022015956

Piscopo, N. J., Mueller, K. P., Das, A., Hematti, P., Murphy, W. L., Palecek, S. P., Capitini, C. M., & Saha, K. (2018). Bioengineering Solutions for Manufacturing Challenges in CAR T Cells. In *Biotechnology Journal* (Vol. 13, Issue 2). Wiley-VCH Verlag. https://doi.org/10.1002/biot.201700095

Polonsky, M., Rimer, J., Kern-Perets, A., Zaretsky, I., Miller, S., Bornstein, C., David, E., Kopelman, N. M., Stelzer, G., Porat, Z., Chain, B., & Friedman, N. (2018). Induction of CD4 T cell memory by local cellular collectivity. *Science*, *360*(6394). https://doi.org/10.1126/science.aaj1853

Prybutok, A. N., Yu, J. S., Leonard, J. N., & Bagheri, N. (2022). Mapping CAR T-Cell Design Space Using Agent-Based Models. *Frontiers in Molecular Biosciences*, *9*. https://doi.org/10.3389/fmolb.2022.849363

Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., & Dormann, N. (2019). *Stable baselines3*.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). *Hierarchical Text-Conditional Image Generation with CLIP Latents*. http://arxiv.org/abs/2204.06125

Rashedi, M., Rafiei, M., Demers, M., Khodabandehlou, H., Wang, T., Tulsyan, A., Undey, C., & Garvin, C. (2023). Machine learning-based model predictive controller design for cell culture processes. *Biotechnology and Bioengineering*, *120*(8), 2144–2159. https://doi.org/10.1002/bit.28486

Rolf, B., Jackson, I., Müller, M., Lang, S., Reggelin, T., & Ivanov, D. (2022). A review on reinforcement learning algorithms and applications in supply chain management. *International Journal of Production Research*, 1–29.

Ruiz Barlett, V., Bigeón, J. J., Hoyuelos, M., & Mártin, H. O. (2009). Differences between fixed time step and kinetic Monte Carlo methods for biased diffusion. *Journal of Computational Physics*, *228*(16), 5740–5748. https://doi.org/10.1016/j.jcp.2009.04.035

Safe driving cars. (2022). In *Nature Machine Intelligence* (Vol. 4, Issue 2, pp. 95–96). Nature Research. https://doi.org/10.1038/s42256-022-00456-w

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *CoRR*, *abs/1707.06347*. http://arxiv.org/abs/1707.06347

Sommeregger, W., Sissolak, B., Kandra, K., von Stosch, M., Mayer, M., & Striedner, G. (2017). Quality by control: Towards model predictive control of mammalian cell culture bioprocesses. In *Biotechnology Journal* (Vol. 12, Issue 7). Wiley-VCH Verlag. https://doi.org/10.1002/biot.201600546

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction, 2nd ed. In *Reinforcement learning: An introduction, 2nd ed.* The MIT Press.

Sweigart, A. (2012). *Making Games with Python & Pygame*.

Szopa, I. M., Granica, M., Bujak, J. K., Łabędź, A., Błaszczyk, M., Paulos, C. M., & Majchrzak-Kuligowska, K. (2021). Effective Activation and Expansion of Canine Lymphocytes Using a Novel Nano-Sized Magnetic Beads Approach. *Frontiers in Immunology*, *12*. https://doi.org/10.3389/fimmu.2021.604066

Tamiev, D., Furman, P. E., & Reuel, N. F. (2020). Automated classification of bacterial cell subpopulations with convolutional neural networks. *PLoS ONE*, *15*(10). https://doi.org/10.1371/journal.pone.0241200

Trotman-Grant, A. C., Mohtashami, M., De Sousa Casal, J., Martinez, E. C., Lee, D., Teichman, S., Brauer, P. M., Han, J., Anderson, M. K., & Zúñiga-Pflücker, J. C. (2021). DL4-μbeads induce T cell lineage differentiation from stem cells in a stromal cell-free system. *Nature Communications*, *12*(1). https://doi.org/10.1038/s41467-021-25245-8

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. https://arxiv.org/abs/1706.03762

Vormittag, P., Gunn, R., Ghorashian, S., & Veraitch, F. S. (2018). A guide to manufacturing CAR T cell therapies. In *Current Opinion in Biotechnology* (Vol. 53, pp. 164–181). Elsevier Ltd. https://doi.org/10.1016/j.copbio.2018.01.025

Watanabe, K., Kuramitsu, S., Posey, A. D., & June, C. H. (2018). Expanding the therapeutic window for CAR T cell therapy in solid tumors: The knowns and unknowns of CAR T cell biology. In *Frontiers in Immunology* (Vol. 9, Issue OCT). Frontiers Media S.A. https://doi.org/10.3389/fimmu.2018.02486

Wherry, E. J. (2011). T cell exhaustion. In *Nature Immunology* (Vol. 12, Issue 6, pp. 492–499). https://doi.org/10.1038/ni.2035

Wherry, E. J., & Kurachi, M. (2015a). Molecular and cellular insights into T cell exhaustion. In *Nature Reviews Immunology* (Vol. 15, Issue 8, pp. 486–499). Nature Publishing Group. https://doi.org/10.1038/nri3862

Wherry, E. J., & Kurachi, M. (2015b). Molecular and cellular insights into T cell exhaustion. In *Nature Reviews Immunology* (Vol. 15, Issue 8, pp. 486–499). Nature Publishing Group. https://doi.org/10.1038/nri3862

Zhang, D. K. Y., Adu-Berchie, K., Iyer, S., Liu, Y., Cieri, N., Brockman, J. M., Neuberg, D., Wu, C. J., & Mooney, D. J. (2023). Enhancing CAR-T cell functionality in a patient-specific manner. *Nature Communications*, *14*(1). https://doi.org/10.1038/s41467-023-36126-7

Figure Legends

Figure 1: (a) CAR T-cell manufacturing process- i. naïve T-cells (red) are taken out of the body by leukapheresis process, ii. Antigen presenting beads (white spheres with black spikes) are applied to activate the naïve cells, iii. The naïve T-cells are activated (blue), over exposed cells undergo exhaustion (yellow), iv. The activated cells proliferate in number (b) Dynamic, intelligent process control of activation in a simulated cell culture to control real culture with trained policy. The state observation data is collected in tabular, image or combined format as an input to the deep neural network or RL-agent; the agent then selects either of the three permitted actions – add, skip, or remove beads in each control step. Through iterative rounds of training, the RL-agent learns to map each state to an action which optimizes the end goal of maximum number of robust effector cells.

Figure 2: Proposed simulation replicating cell activation and expansion (a) Sample simulation trajectories for three control strategies – top to bottom row depicts optimum, sub-optimum, and random bead additions; the bar plot at left indicates the number of cells separated by type at each simulation step; the symbols at the x-axis represent the action taken: (+) refers to bead addition, (-) refers to the removal and (o) refers to no action; the right three windows are simulation screens at 1, 5 and 19 steps. (b) Process and permitted actions by the cells in each simulated step. (c) Simulated life trajectory of a naïve starting cell to activated with full potency and natural exhaustion caused by aging. Also defined are two modes of division – symmetric and asymmetric.

Figure 3. Schematic of three different observation space input strategies and learning curve with different algorithms used. (a) List of input schemes –tabular or list input, image input or

combined input (b) learning curves obtained by training on 3 different reinforcement learning algorithms: PPO, A2C and DQN.

Figure 4: Change of strategy by the RL-agent using 20 control steps for different cell types. (a) Simulation process to obtain control strategy information (b) Strategy of the RL-agent visualized by average number of beads at each control step (y and x axes respectively). The error bar indicates the standard deviation of beads used at that control step – an indication of simulation variability or constancy (where no bars exist). The learning curve is also attached with each bar plot, axes same as in Figure 3b. Arrows between plots indicate the change in cell type (also see Table 1).

Figure 5: Change of strategy by the RL-agent using 50 control steps learned from training with different cell types. The strategy of the the RL-agent visualized by the average number of beads per control step (y and x axes respectively). Error bars indicate one standard deviation, showing variability of steps or uniformity (no error bars). The learning curve is also attached with each bar plot. Arrows indicate the change in cell type; also see Table 1.

Figure 6: (a) Learning curve for RL-agent trained with and without noise and reward histogram for simulation conducted with RL-agent trained on 0, 250k and 500k episodes (b) RL-agent trained with 20, 50 and 400 control steps (c) Number or training episodes required to reach an accuracy of 80%, 90%, and 95% by RL-agents pre-trained for 500k steps on cell one vs. RL-agents trained on respective cell types from the beginning. Y axis shows the number of training runs required in log base ten scale.

Funding

This work was supported in part by NSF Award 2042503. Ratul Chowdhury is grateful for a Iowa State University startup grant partially funding this work.

Acknowledgements

We thank Krishanu Saha and his group members at the University of Wisconsin, Madison for useful discussions on CAR-T cell activation and expansion.

Author Contribution

SF performed the literature review, conceptualization, data acquisition, code and software compilation, writing, and reviewing of the manuscript. IFS assisted in code compilation and version control. RC contributed to writing, supervision, and revision. NFR contributed to conceptualization, funding acquisition, writing, reviewing, and overall supervision.

Competing Interest

Nigel Reuel is the scientific founder of Skroot Laboratory Inc., focusing on non-destructive, continuous measurement of cells, and has an equity interest in the company. In addition, Nigel Reuel receives income from Skroot Laboratory Inc. for serving in a leadership role.

Correspondence

All the correspondence should be addressed to Nigel Forrest Reuel at reuel@iastate.edu

Code Availability

Code is available in Zenodo at - https://doi.org/10.5281/zenodo.7905320 and at GitHub at - https://github.com/Sakib1418/Game-of-cells with full details and instructions for reproduction.