Pázmány Péter Catholic University

Faculty of Information Technology and Bionics

Tutored Research Project

Simulation of Cancer Growth with Cellular Automata

Richárd Bence Rózsa

Bioinformatics MSc

2026

Supervisor: Dr. András Horváth

Abstract

Modeling the spread and growth of tumor cells is an important tool for understanding the mechanisms underlying biological processes in cancer, as well as for studying therapeutic strategies. Cellular automata provide a discrete modeling method based on simple local rules, capable of reproducing complex emergent patterns.

In this work, I analyzed a cellular automaton model for tumor growth from a 2022 publication and further developed the model with new features, such as an immune response and mutations. I also coded the model in Python, creating an easy-to-use program with visualization, animation, as well as a dashboard that acts as a graphical interface. I developed a Python module with detailed documentation and PyPI upload for simple reusability, and created an online version of the dashboard via Streamlit Community Cloud, which can be used without writing a single line of code.

With multiple parameters, like the size of the field, probabilities for certain actions, immune strength, etc. the user can create their own simulations, and visualize the results through growth plots, proliferation potential histograms, and growth animation. Simulations can be run with a couple lines of code or on a streamlit dashboard. Statistical properties of the simulations can be displayed, analyzed, and saved alongside the growth field itself. Machine learning features are also available for data set generation with randomized parameters, model training, and prediction of tumor size or confluence.

Contents

1. Introduction

1.1. Cellular Automata

Cellular automata (CA) are mathematical models used to study complex systems through simple, local rules. A cellular automaton consists of a grid of cells, each of which can exist in one of a finite number of states (for example, "on" or "off," or multiple colors or values). Time progresses in discrete steps, and at each step, every cell updates its state simultaneously based on a set of rules that typically depend only on its own current state and the states of its neighboring cells. Despite this simplicity, cellular automata can generate surprisingly complex and diverse patterns of behavior, ranging from stable structures and repetitive cycles to chaotic, unpredictable dynamics. As I describe later, I sometimes minimally cross the border between true CA and hybrid models, because I do rely on global rules in certain cases, so technically my apporach is a hybrid cellular automaton.

1.2. Motivation

Cellular automaton models are particularly valuable for simulating natural and artificial events because they capture how local interactions give rise to large-scale behaviors. For example, they have been used to model traffic flow, biological growth, the spread of diseases, forest fires, and fluid dynamics. One of the most famous examples is Conway's Game of Life, a two-dimensional cellular automaton where simple birth and survival rules produce intricate, evolving patterns that resemble living systems. More generally, CA models provide a framework for exploring emergence, self-organization, and complexity in systems where global order arises from local interactions. This makes them an important tool in fields like physics, computer science, ecology, and complex systems research.

In this work I develop and code a (hybrid) cellular automaton model for tumor growth. Cellular automaton models capture spatial and local interactions at the level of individual cells, while still being computationally efficient. They represent emergent behavior: how local cell decisions, like dividing or dying, give rise to large-scale tumor patterns. They can integrate enviremental factors such as nutrients, immune cells or blood supply. However, there are also disadvantages, as the cellular automaton approach can lead to oversimplified models and many constraints due to lattice dependence and limited resolution.

2. The Model

2.1. Tumor Growth and Cell Types

The theoretical background for this model is based on the work of Carlos A Valentim, José A Rabi and Sergio A David.[1] This is a 2D cellular automaton model; the area is a square grid, with each cell being able to have discrate values ranging from 0 to a 'pmax'+1 (see below). If a square is 0, it is empty, if not 0, a tumor cell exists there with a proliferation potential of the square's value.

In the model, we have two types of tumor cells. RTCs (regular tumor cells) can only divide a limited amount of times. This amount is their proliferation potential, which gets smaller with every division. These cells can only proliferate into other RTCs, and the daughter cell will have the same proliferation potential value as the mother cell after the division (So an RTC with pp = 5 will create two cells, whose pp = 4). STCs (stem tumor cells) have infinite divisions. Their daughter cell is either an RTC with the highest proliferation potential value (pmax) or (by a small chance) another STC. The value which we represent these cells with in the model is pmax+1.

At every cycle each cell chooses from four different options. First, they can die by apoptosis (low chance for RTCs only). If they survive, they can proliferate with the probability of $CCT \cdot dt/24$ (where CCT is the cell cycle time, dt is the time step in the model). In case of no proliferation, they can migrate to one of the eight neighbouring cells. The probability of this action is $\mu \cdot dt$ (μ is the migration capacity of the cell). If none of these actions happen, the cell stays quiescent. Quiescence is forced if there is no free space around the a surviving cell. In the python code all these probabilites, the maximum proliferation potential value, model duration and area size is a parameter, which can be set by the user.

This was the entire model explained in the article mentioned above. Although it models tumor movements and proliferation well, while being relatively cheap computationally, it lacks certain features, which would be desirable to add to achieve more realistic outcomes. In the next sections I detail how I enhanced this model by adding immune cells and tumor mutations.

2.2. Immune Response

I expanded this model by simulating an immune response during the growth of the tumor cells. The basic idea was the following: in each cycle, after the tumor growth, immune cells may also appear and act. They spawn randomly on the inner edge of the field and every time step they move to a free neighbouring spot. If they get in contact with a cancer cell (they move next to one), they might kill it. After a certain amount of life, the immune cells die. The spawn rate, kill chance and lifespan is influenced by a single 'immune strength (I)' parameter, and the spawn rate also increases with the size of the tumor (but has a maximum value based on I).

In this first implementation, quite a few problems were recognized: the spawnpoints were always static, immune cells did not move towards the tumor, and the results (like immune-tumor ratio, kill-rate, etc.) were not realistic at all. In order to improve the immune response and make it more realistic, I looked through multiple publications on immune-tumor interactions. Here I list the conclusions:

- Immune cells migrate through tissue, forming motile kinapses rather than fixed synapses when contacting targets. [2] [3]
- The per-cell killing capacity of immune cells in vivo is modest: one CTL (cytotoxic T lympocyte) on average kills ∼2–16 infected cells per day. [2] [4]
- Killing by CTLs often requires multiple hits or prolonged contact: in solid-tumor settings, single CTL–tumor cell contacts frequently fail. The lag time between CTL binding and target cell death can be hours (e.g., 1.8 ± 1.5 h) in some intravital studies. [4] [5]
- The probability of tumor-cell death increases when the local density of CTLs is higher (i.e., multiple CTLs cooperating). [2] [5]
- Tumor mutations reduce immune-cell kill probability: tumor cells may repair sub-lethal damage, or require repeated hits before elimination. [6]
- Immune motility (random walk + chemotaxis) in tissue varies (1–20 µm/m) and is reduced in dense tumor microenvironments, but is much higher than tumor motility.[7] [8]
- Immune recruitment is influenced by tumor size, chemokine gradients, and immune-strength; infiltration is often lower in "cold" tumors and higher in "hot" tumors. [4]
- Immune exhaustion/time-dependent decline in functional cytotoxicity occurs: even for the same immune-cell count, kill efficacy may drop over time. [9] [10]

Based on these findings, here is how I significantly enhanced the immune response in my model. First, I wanted to change the spawnpoints of the cells, so that they move dinamically with the tumor, instead of being locked to the edge, which could lead to too much or too little distance from the cancer cells. A computationally cheap approach was to find a "frame", which consists of two rows and two columns (one on each side) that is always a certain distance away from the closest tumor cell. This is basically a square frame that moves and grows with the tumor.

I also wanted the immune cells to move towards the tumor instead of randomly walking on the field. For this, I calculate tumor density in each cycle (using Gaussian filter from Scipy) and have each immune cell decide it's target when moving based on this, so they move to cells that are closer the tumor with higher probability. As immune cells are much faster than tumor cells, they are allowed to migrate multiple times in each cycle based on their distance from the tumor (using the calculated tumor density map). The following equation gives us the number of steps a cell takes per cycle:

$$\text{moves (per cycle)} = int(1 + I \cdot (1 - \text{tumor\_density}))$$

I also modified calculations for immune spawn chance and kill chance on contact to match more realistic values. The goal for spawn rate is a sigmoid-like saturating spawn, with a delayed onset. My target average immune-tumor ratio was between 0.1 and 0.3 and target kill probability was 0.15-0.3 (including the effect of mutations) at $I = 5$. These values are calculated with the following equations (considering spawn chance is 0, if we are above target immune-tumor ratio):

$$\text{spawn\_chance (per cell)} = I \cdot (\text{tumor\_size}/(\text{tumor\_size} + I \cdot 500)) \cdot IE$$

$$\text{kill\_chance (per contact)} = min[(0.05 \cdot I) \cdot exp(-0.25 \cdot \text{mutation\_state(target)}), 0.3] \cdot IE$$

As you can see both spawn and kill probability is influenced by immune exhaustion (IE): a time dependent decline of immune function, calculated (with the α parameter) as:

$$IE = max(0.2, 1.0/(1.0 + \alpha \cdot \text{cycle\_count}))$$

Using these equations, we can model the effects of immune exhaustion as the decline of the entire immune population. We also increased immune cell speed compared to tumor cells, and scaled immune recruitment by tumor size and kill chance by mutation state. With default parameters (seen in table 2.1) we reach a $\sim$ 2.3 average kill/immune cell/day. (Immune cell lifespan is a random integer between $I - 1$ weeks and $I + 1$ weeks.)

As a small but important note, with some of these implementations, this model minimally crosses the line between a true cellular automaton and a hybrid cellular automaton. Sometimes (for example, during immune spawn calculations) we rely on global parameters and continuous-valued or time-dependent parameters (like chemoattractant map), which are outside the cellular automaton rules. Still, the core of the model, the tumor behaviour, the immune killing, movements and mutations are CA, while some of these other factors can be considered hybrid. Though we can also think of the gaussian map and immune spawning mechanism as local rules with extended range, this model, to be exact, is a hybrid cellular automaton (HCA).

2.3. Tumor Mutations

I also added random mutations to the model that influence the proliferation rate, immune resistance and apoptosis chance of the cells. These mutations can be beneficial for the cancer cells, but negative mutations can also appear that slow the growth of the tumor.

In this basic implementation each tumor cell has a *mutation state* that influences the parameters mentioned above. This state ranges from -3 to +3: negative values reduce the survivability of cells by increaseing the probability of apoptosis and make them more vulnerable to immune attacks. The chance of a division also decreases. In the case of a positive value, the effects are the exact opposite. The higher the absolute value of the mutation state, the stronger the effects are. New mutations can arise at each proliferation based on the *Mutation chance (M)* parameter of the model, and existing ones are inherited during a division.

2.4. Model Parameters

Below you can see the list of the parameters of the model with their default values. All other values used in the model (like probability of proliferation: PP and probability of migration: PM) are calculated from these parameters, as explained in section 2.1.

Parameter	Datatype	Description	Default Value
cycle	int	duration of the model given in hours	500
side	int	the length of the side of field (10um)	50
pmax	int	maximum proliferation potential of RTC	15
PA	int	chance for apoptosis of RTC (in percent)	1
CCT	int	cell cycle time of cells given in hours	24
Dt	float	time step of the model given in days	1/24
PS	int	STC-STC division chance (in percent)	15
mu	int	migration capacity of cancer cells	4
I	int	strength of the immune cells (1-10)	5
M	int	tumor mutation chance (in percent)	10

Table 2.1: Base model parameters

The I parameter decides how strong immune cells are: their spawn rate, kill chance, lifespan and maximum motility as shown in the formulas above. Other parameters (that are not about the cells themselves, not calculated by I) of the immune response can be seen in table 2.2 below.

Parameter	Datatype	Description	Default Value
offset	int	distance of spawnpoints ("frame") from the tumor	20
alpha	float	controls strength of immune (population) exhaustion	0.002
it_targ	float	desired mean immune/tumor ratio during simulation	0.1
infil	float	threshold for what cell counts as infiltrating (0-1)*	0.3

Table 2.2: Immune response parameters *based on chemokine gradient, only matters for stats

3. The Python Package

3.1. Introduction to TCAMpy

I created TCAMpy, a python module for my cellular automaton model. The documentation (with links to the github repository, example codes and results, as well as example files) can be found on the following link: `https://tcampy.readthedocs.io/en/latest/` and it can be used online very easily as a Streamlit dashboard via Streamlit Community Cloud here: `https://tcampy.streamlit.app/`. The module itself can also be installed from PyPI using *pip install* and the documentation gives a detailed guide on usage.

Figure 3.1 is an example of the plots the user can create from one simulation. It includes the growth field of the tumor, growth curves, a mutation heatmap and a histogram of the final proliferation potentails.

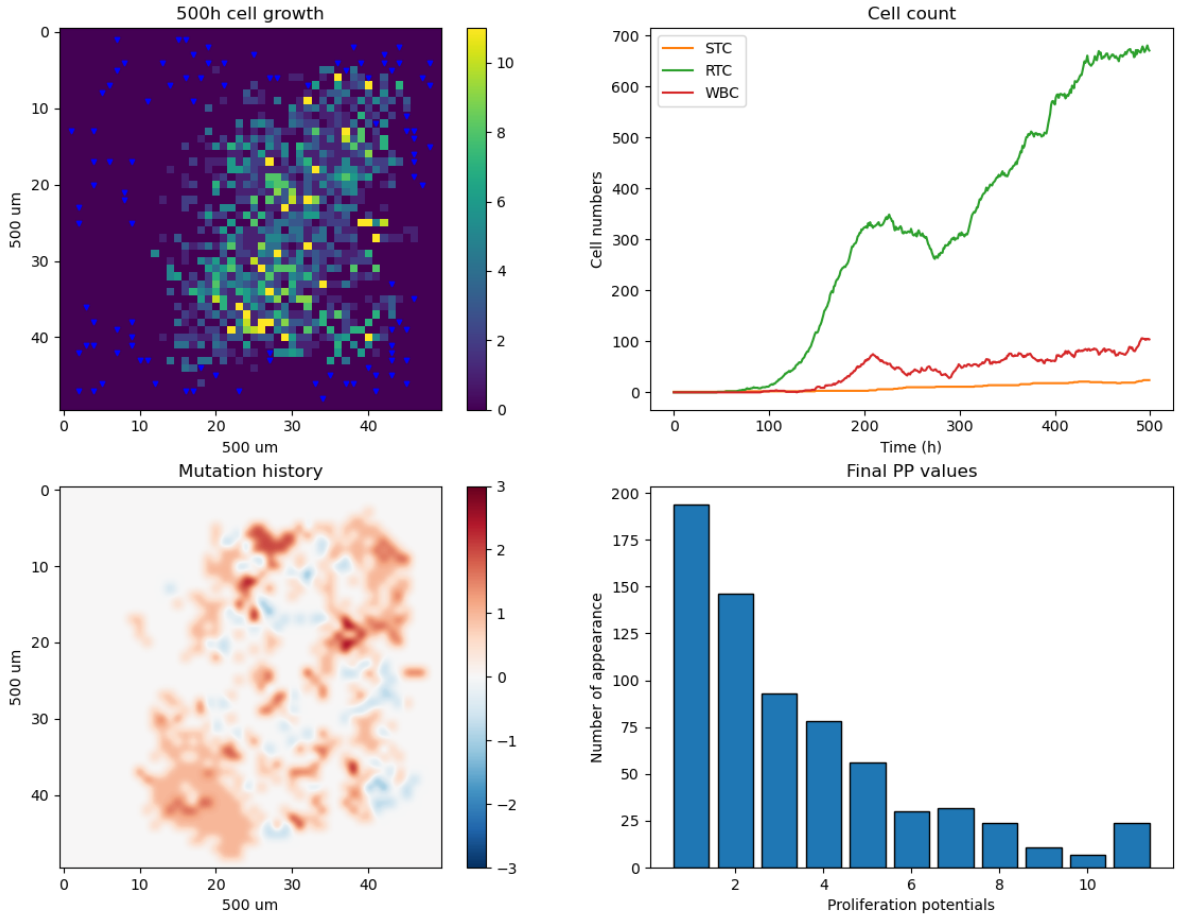

Figure 3.1: Result of a single simulation

With TCAMpy the user can set the parameters, create unique initial states, view and save statistics, save data and also use a streamlit dashboard as a graphical interface. Growth plots, histograms and animation is available for visualization in an easy to use way. The source code and API details are also available in the documentation, and here I am going to explain the key parts of the model's implementation.

The code consists of three classes/objects, each for a different aspect. The *TModel* class has all the functions responsible for the model itself and for running simulations, as well as visualizing the results: creating initial state, simulating tumor and immune cell behaviour, counting and plotting cells, displaying heatmaps and saving stitistics. As I have limited space in this report and a large number of functions in the code, I am not able to list all of these here, but their complete description (and their source code) can be seen in the documentation. When creating a *TModel* object, the user must also define the parameter values for it. Then, with a simple *run()* function, a simulation can begin.

3.2. Dashboard

The module offers the *TDashboard* object for the user to create a graphical interface for simple usage through a Streamlit dashboard. This can be done with a couple lines of code or opened via Streamlit Community Cloud. Both options provide the same GUI, where sliders, buttons and numeric inputs are available to set parameter values, create the initial state and run simulations. Although animation for the growth is not currently possible here, the same plots as seen on figure 3.1 can be displayed alongside a table with statistical properties and growth curves for average values (in case of multiple simulations). All simulation statistics and growth fields can be downloaded for further analysis.

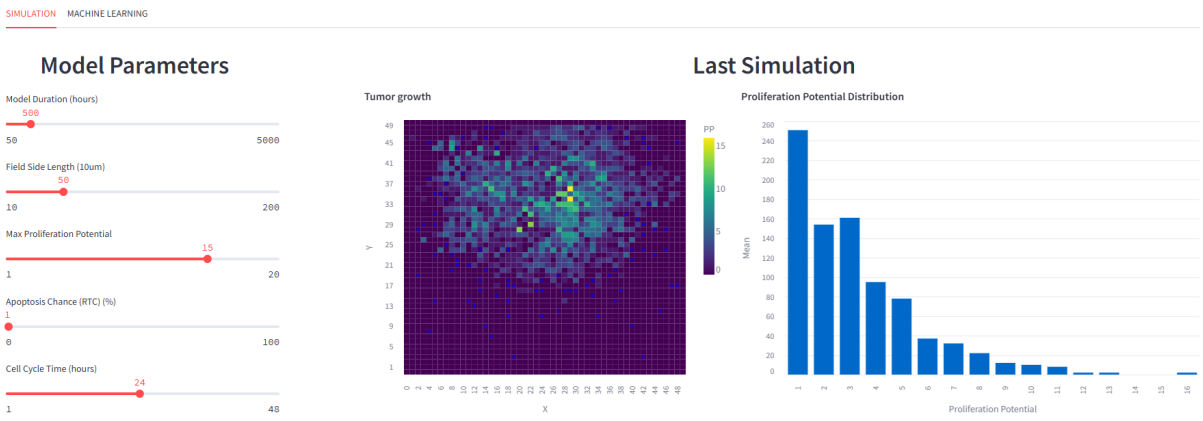

Figure 3.2: Dashboard Snippet

3.3. Machine Learning

These simulations can take a long time, especially with large area size and many cycles (with default parameters shown in the examples it takes a couple seconds to run one simulation). With machine learning we can predict features like tumor size much faster as long as we have a dataset to use for model training. With my python module, the user can generate datasets with randomzied parameters, train a model for future prediction, list the parameters that influence the growth the most, and predict a selected numeric attribute for a new instance.

In my *TML* class I use python's *sklearn* package to implement these functions. Usage is relatively simple, and guides are available in the documentation. For dataset generation, a dictionary of parameters is needed with the randomization ranges, and the results include the original parameters and all statistical properites about the simulations, which are saved to a file. This file can be used for model training and with the trained model statistics (like tumor size or confluence) can be predicted for a new instance.

3.4. PyPI and Documentation

The module has been uploaded to PyPI so it can be easily used by anyone, even outside of Streamlit Cloud. It can be installed, alongside all the dependencies, with *pip install TCAMpy*. (Dependencies are the following packages: numpy, matplotlib, pandas, scipy, altair, streamlit, streamlit-javascript, scikit-learn, openpyxl, tqdm. These are automatically installed with TCAMpy.)

The Read The Docs documentation explains the theoretical background of the model and gives a short explanation about cell behaviours. A detailed guide about usage is part of it, with example codes and files. The API documentation section shows all classes and functions, with the option to view the source code.

4. Summary

4.1. The Current State of the Model

Throughout this semester I have analyzed and coded a cellular automaton model for cancer growth, and expanded it with simulation of an immune response and tumor mutations. The behaviour of immune cells is relatively complex and has been tuned based on a number of publications, while mutations remain simple and basic for now, with improvements planned for the near future.

My python module, TCAMpy, allows the user to easily run their own simulations with custom parameters and display statistics, plots and even animate growth. A dashboard can be created or used online on Streamlit CLoud as a coding-free option. The 1.0 version of the module has been uploaded to PyPI, which was one of my major goals for this semester.

4.2. Plans for the Future

Although they are quite complex now, TCAMpy and the model itself are far from finished. More publications have to be analyzed and many parameters await further tuning. Experimental validation is necessary to finalize the approach and conclude the default parameter values. The modeling of tumor mutations also has to be improved as it is in a very early state.

The code (alongside the documentation) will always need to be updated to reflect the continued development, including all of the changes to the different aspects. A (partial) CUDA/PyTorch based implementation is also a possible approach for the future to speed up simulations, especially in the case of large area size. The machine learning features are present with a very basic implementation, which will be expanded in the future.

Bibliography

[1] C. A. Valentim, J. A. Rabi, and S. A. David, "Cellular-automaton model for tumor growth dynamics: Virtualization of different scenarios," en, *Comput Biol Med*, vol. 153:106481, Feb. 2023. DOI: 10.1016/j.compbiomed.2022.106481. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/36587567/.

[2] H. S et al., "In vivo killing capacity of cytotoxic t cells is limited and involves dynamic interactions and t cell cooperativity," en, *Immunity*, vol. 44, no. 2, pp. 233–45, Feb. 2016. DOI: 10.1016/j.immuni.2016.01.010. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/26872694/.

[3] S. Halle, O. Halle, and R. Förster, "Mechanisms and dynamics of t cell-mediated cytotoxicity (review)," en, *Trends Cancer*, vol. 38, no. 6, pp. 432–443, Jun. 2017. DOI: 10.1016/j.it.2017.04.002. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/28499492/.

[4] B. Weigelin et al., "Cytotoxic t cells are able to efficiently eliminate cancer cells by additive cytotoxicity," en, *Nature Communications*, vol. 12, no. 5217, Dec. 2021. DOI: 10.1038/s41467-021-25282-3. [Online]. Available: https://doi.org/10.1038/s41467-021-25282-3.

[5] B. Weigelin and P. Friedl, "T cell-mediated additive cytotoxicity – death by multiple hits," en, *Trends Cancer*, vol. 8, no. 12, pp. 980–987, Dec. 2022. DOI: 10.1016/j.trecan.2022.07.007. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/35965200/.

[6] B. McKenzie, R. Khazen, and S. Valitutti, "Greek fire, poison arrows, and scorpion bombs: How tumor cells defend against the siege weapons of cytotoxic t lymphocytes," en, *Front Immunol.*, vol. 3, no. 13, May 2022. DOI: 10.3389/fimmu.2022.894306. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/35592329/.

[7] D. Vesperini, G. Montalvo, B. Qu, and F. Lautenschläger, "Characterization of immune cell migration using microfabrication," en, *Biophys Rev*, vol. 13, no. 2, pp. 185–202, Feb. 2021. DOI: 10.1007/s12551-021-00787-9. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC8285443/.

[8] W. Du, P. Nair, A. Johnston, P.-H. Wu, and D. Wirtz, "Cell trafficking at the intersection of the tumor-immune compartments," en, *Annu Rev Biomed Eng*, vol. 6, no. 24, pp. 275–305,

Jun. 2022. DOI: 10.1146/annurev-bioeng-110320-110749. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/35385679/.

[9] C. C. Zebley and B. Youngblood, "Mechanisms of t cell exhaustion guiding next-generation immunotherapy," en, *Trends Cancer*, vol. 8, no. 9, pp. 726–734, Sep. 2022. DOI: 10.1016/j.trecan.2022.04.004. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/35570136/.

[10] R. Nair et al., "Deciphering t-cell exhaustion in the tumor microenvironment: Paving the way for innovative solid tumor therapies," en, *Front Immunol.*, vol. 1, no. 16, Apr. 2025. DOI: 10.3389/fimmu.2025.1548234. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/40236693/.