Predicting HIV Viral Body Load through Image Analysis

Junjiang Li Ben Maldonado Gabe Skidmore Zhuo Xu Darren Calovini

1 Introduction

The Human Immunodeficiency Virus (HIV) has been a major global health problem for decades. As a result, researchers working in HIV related fields have successfully developed some treatment methods. A commonly-used therapy combines two nucleoside reverse transcriptase inhibitors (NRTI) with a protease inhibitor (PI) [8]. When these inhibitors are absorbed into a cell infected with the HIV virus, they can inhibit viral replication and stop the virus from infecting other cells. The infection process with or without therapy methods, such as the aforementioned NRTI and PI therapy, can be simulated by dividing treatment into discrete time steps and modeling how states of those cells change. Simulations rely on rule sets according to cellular automata (CA) to produce reliable cell states in later stages. CA are simulation models commonly used to reliably model a large number of cells and their interactions. Many researchers apply CA to simulate the dynamics of HIV infection [24]. However, simulations for these therapy methods are computationally costly and take a long time to generate results[11]. The high demands of HIV CA models render them impractical for many patients who would otherwise benefit from them [6]. In addition, classical machine learning methods for this problem focused on features associated with the concentrations of different cell types. In this paper, we attempt a new approach to feature extraction from these CA models by generating images that reflect cell states at each time step from a variety of HIV CA models and extract image features from these images. Then, our regressors forecast the HIV CA models' end states by analyzing these image features from several beginning time steps. These predictions are very useful for further investigating the effectiveness of therapy methods and predicting viral load in the body over time, as it will reduce the time and computational power necessary to test these therapies by cutting out a large section of the simulation entirely.

In this paper, our goal is to utilize machine learning techniques to predict cells' state after 600 weeks of infection and possibly multiple treatments (depending on the model). This objective is addressed via the following three aims.

1. Develop several machine learning regression models that can predict the cell states at week 600 based on extracted image features from the first 200 weeks.
2. Evaluate features from how many time steps can form a solid forecast accuracy.
3. Evaluate the impact of extracted image features on prediction results.

The organization of this paper is as follows. Section 2 provides a basic background on HIV, the process of HIV simulation using cellular automaton, and basic image feature extraction. Section 3 address our particular methods of image extraction and subsequent regression. Section 4 presents the results of our machine learning models and the description of our regressors. Section 5 is the discussion of results. Section 6 is the conclusions we reached from our findings.

2 Background

For this research, *cellular automata* (CA) reproduced the effects of Human Immunodeficiency Virus (HIV) within the human body. HIV's basic mechanics will be explored briefly in section 2.1, while cellular automata will be explored in section 2.2. There are five theoretical rulesets for HIV modeling used in this paper: the dos Santos, González, Moonchai, Precharattana, and Rana models, all described in the aforementioned section 2.2. With all five HIV models, image features collected at each timestep until week 200 were used to predict the end state of the models at week 600 using various regressors; these regressors included the *Decision Tree Regressor* (Tree), the *Lasso Regressor* (Lasso), the *Gaussian Processes Regressor* (GP), and the *Support Vector Regressor* (SVR) . Image featurization of these HIV models is discussed in section 2.3, and the background on the various regressors is given in section 2.4.

2.1 HIV and Acquired Immunodeficiency Syndrome (AIDS)

HIV is a virus spread through bodily fluids that attacks the body's CD4+ lymphocyte immune cells (also known as T cells). According to the World Health Organization

[3], this virus currently affects more than more than 37.9 million people as of 2018 and killed 770,000 people in that same year. The virus itself works in three major stages as described by the CDC [4]:

1. *Acute HIV Infection.* During this stage, the virus attacks the body in large numbers, infecting many T cells. The immune response is mostly equal and opposite to the infection, and the majority of the virus is effectively destroyed. An emphasis must be placed on this surviving minority, however.

2. *Latent HIV Infection.* During this stage, the leftover virus that remains in the body begins to reproduce at a slow, undetected rate. This phase might last a decade, though certain treatments such as the NRTI and PI mentioned in section 1 can extend this period into multiple decades. At the end of this stage, the viral load in the body (that is, the amount of virus) goes up while the T cell count goes down.

3. *AIDS.* When the T cell counts go below 200 cells/mm^3 [21], the infected person enters the Acquired Immunodeficiency Syndrome (AIDS) stage. During this stage, the immune system's capability to defend against infection is greatly diminished, and most patients typically survive as few as three years.

Worst of all the aspects of HIV are its longevity and lack of cure. Once a person becomes infected with the virus, there is no current way to ever rid themselves of it; they can only delay the inevitable with drugs and treatment through a process known as antiretroviral therapy (ART) [22].

2.2 HIV Modeling using Cellular Automata

The idea that HIV could be modeled using CA began in 1990 under Kougias and Schulte [15]. Each new iteration of CA modeling brought with it new ideas that could more closely and accurately mimic the behavior of HIV. While the research is not limited to the five models chosen for this paper, the dos Santos, González, Moonchai, Precharattana, and Rana models represent some of the larger steps forward in HIV CA modeling over the last 30 years [7]. The dos Santos and Coutinho model focuses on recreating the basic behavior of HIV infected cells in lymph node tissue slowly overtaking the immune system across the span of years [29].

González, *et al.* built upon the dos Santos and Coutinho model with the inclusion of two ART treatments, which consisted of two NTRIs and either a PI or a non-nucleoside reverse transcriptase inhibitor (NNRTI). The authors then examined how the different treatments affected the dynamics of HIV within the body [8].

Moonchai and Lenbury introduced the notion of blood transference of the virus, which could be lessened with plasma apheresis. Plasma apheresis is a process by which a patient's blood is removed and stripped of its infected blood plasma. The leftover red blood cells are then reintroduced into the body. Thus, to model this, Moonchai and Lenbury had a pair of CA's: one representing the lymph node tissue where CD4+ cells are located, and the other representing blood [20].

Precharattana, *et al.* focused on the idea that certain HIV cells can remain dormant and not infect healthy cells, thus creating "reservoirs" of infected cells that the immune system does not detect [24].

Rana, *et al.* uses many of the same rules of the dos Santos and Coutinho model with two key differences: first, cells can become infected via the blood in addition to cell-to-cell contact. Second, similar to the González model, the Rana model introduces treatment in the form of no treatment, mono-treatment, and dual-treatment. This modeling of different approaches to treatment allows the authors to examine how multiple types of treatments used in parallel affect the dynamics of the HIV infection [25].

Regardless of the specific ruleset or theoretical approach, each of these HIV models use a form of CA. A cellular automaton is a model that simulates simple discrete components called cells and their interactions with the cells around them. "Discrete" in this context refers to each cell being an individual and associated with its own value or classifier. In all of the HIV CA models listed above, the cells are square-shaped, represent CD4+ T cells (except in the case of Moonchai, where they could also represent blood cells) and all the cells are placed into a 2D grid.[1] The cells with which one cell can interact is called a neighborhood. There are two main approaches to a cell's neighborhood: Von Neumann neighborhoods, which only include the four cells that a given cell shares a direct side with, and Moore neighborhoods, which include the side-sharing cells as well as the cells that share a corner with a target cell. Each of the HIV CA models utilize Moore neighborhoods. Cells along the boundaries of the grid (e.g., first row, last column) are given special cases, since they do not have full Moore neighborhoods. This can cause a model to have irregular simulation of HIV because HIV itself does not exist within a small grid with rigid edges, but rather a continuous human body. All of the models, apart from the Rana Model [25], dealt with this issue by wrapping boundaries (shown in figure 1) from one side onto another. (e.g., the top-left cell now also connects with the top-right cell and the bottom-left

[1]Note that CA grids can be of different shapes such as a triangles or hexagons, and the cells can also be of different sizes. CAs are usually taught as a 2-D shape but they can be reformatted to the *n*th dimension such as 1-D or 3-D.

cell).

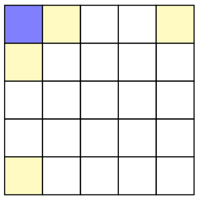


Figure 1: Example of “wrap around”. Von Neumann neighbors of the top-left cell are highlighted in light yellow.

Each cell within the grid has a state (healthy, infected, dead, or some other intermediate or special states such as acute infected or healthy with treatment present) based on a particular ruleset of the simulation model. Since each model may have many different classes of similar cells, such as a healthy cell with any variety of treatment in the González model, cells of similar type can be bundled together (e.g. all healthy cells, regardless of whether or not they have been treated, could be put into the “Healthy” category). By bundling, regression across different HIV CA models with different approaches to simulating the virus is more achievable, since all models will then contain the same classes upon which regression can be performed. In the case of this paper, the cells were bundled into the following four categories: healthy, infected (and detected by the immune system), infected (and not detected by the immune system), and dead.

To start a model, there is a process called seeding. In this kickstarting step, cells within the CA are given a beginning state determined by the ruleset of the model. This seeding process is important as the following steps of the model require a starting point to begin simulations. The model then uses the ruleset to update each cell synchronously over discrete time steps based on its state and the states of cells within its neighborhood. “Discrete” in this context means that the entire grid is updated at one time for each time step. The time step itself represents the passage of a period of real-life time, which is weeks in the case of these HIV CA models. Each model ends at 600 timesteps, which is 600 weeks of HIV simulation. An example of discrete time stepping is shown in figure 2, where 2-D cellular automaton evolved 3 times. The rule set of this CA is given below the time-evolution.

2.3 Image Feature Extraction

At each timestep of the model, the grid of cells and their states can be interpreted as a greyscale image. Image features can then be recognized from these timestep grids, and further processed into features with which regression can be performed. Given the propensity of CA modeling towards a more disorganized image than an actual photo would be and patterns recognized visually within the CA model data (explored in Figures 3, 4, and 5), the authors of this paper focused on simpler image features: blobs, contours, and corners.

- *Blobs* can be described as large masses of similarly colored cells. Figure 3 demonstrates what a blob looks like within one of the HIV CA models.

- *Contours* can be described as the inner or outer edge of a shape formed by similarly colored cells. This shape, unlike a blob, is not filled in with uniformly colored cells, but contains completely different colors within its interior. An example demonstrating a contour can be found in figure 4.

- *Corners* can be described as a sharp turn taken by a strip of similarly colored cells. Figure 5 shows the occurrence of corners.

Note that these figures do not represent the *total* appearance of all potential features in these images. Figure 4, for example, does not highlight all instances of contours, but is simply a visual example for the reader to understand what these image features look like in context.

2.4 Regressors

Once image features have been collected, regression can be performed to determine if these image features have a relationship with the counts of different classes of cells at the end of the simulation. In particular, the authors of this paper used the four regression models. What follows is a simplified explanation of how these regressors work:

- **Support Vector Machine Regressor (SVM/SVR)**: this regressor uses different core functions, called kernels, to calculate its regressions. SVRs also use a value epsilon ϵ, which determines how closely the regressor must fit the data. A regressor then utilizes these two parameters to create a linear function using support vectors from each data point (i.e. vectors drawn from a point to its nearest corresponding point on the line). The function changes in accordance to the lengths of these support vectors. If the lengths go beyond the accepted ϵ value, then the function will shift to be closer to the points until as many points as possible can fit within the given ϵ value [9, p.147-168].

- **Gaussian Processes Regressor (GP)**: much like the SV regressor, GP regressors use kernels. Given a certain dataset, the kernel will help generate covariance matrices Σ, which tell how any two variables or features are related. Σ, when taken with the vector representing the mean of the data (μ) creates a Multivariate Gaussian Distribution (MVD). As new data

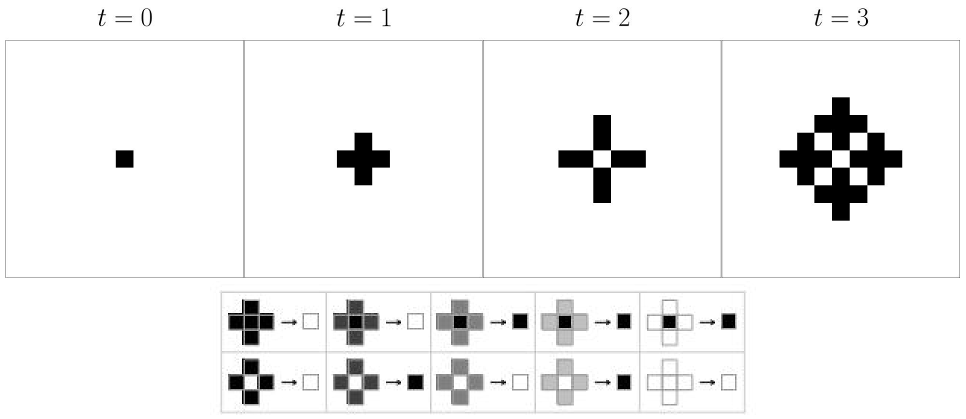


Figure 2: Example cellular automaton evolving by 3 time steps. The rule set of this CA is given below the evolution. In this rule set, each cell can either be black or white, but the shades of gray of the 4 surrounding cells indicate the number of black neighbors around the center cell. When its neighbors are completely white, there are no black neighbors around the center cell. Subsequent darker shades of gray indicate increasing number of black neighbors $(1, 2, 3, 4)$ around the center cell.

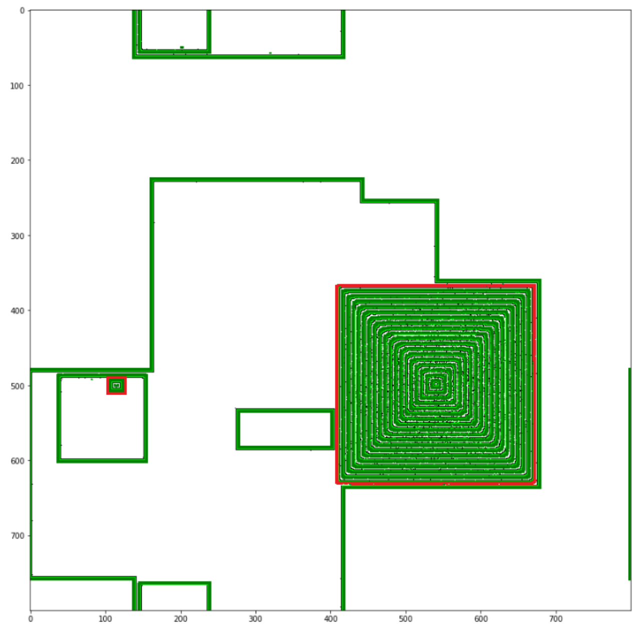


Figure 3: Blobs (highlighted by red borders) that can be detected by image processing libraries.

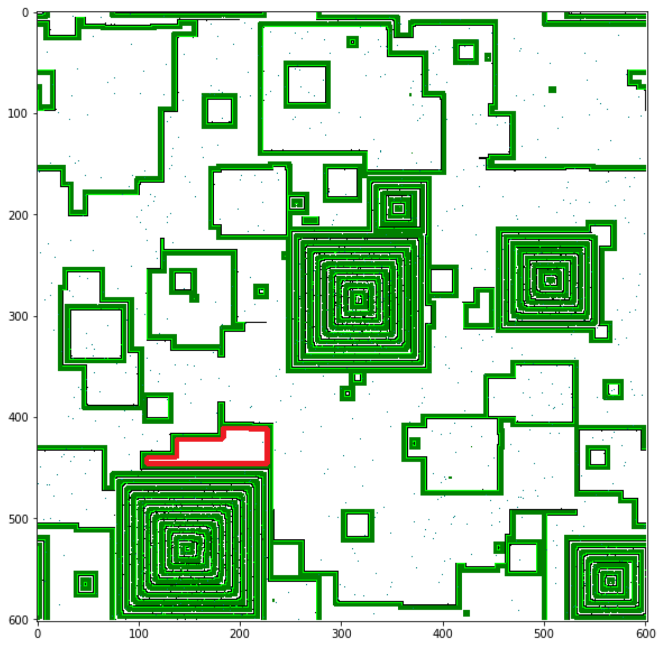


Figure 4: Contours (highlighted by red borders) that can be detected by image processing libraries.

is analyzed, new MVD's are created, and the GP regressors creates a probabilistic distribution of where a data point is likely to land among the many possible MVD's [10]. This probabilistic distribution is then used to perform regression.

- **Lasso Regressor**: The lasso regressor is an extension to the ordinary linear square (OLS) regression. In OLS, we aim to find a vector $\mathbf{w}$ of weights so that the objective function $\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$ is minimized, where $\mathbf{X}$ is a matrix of observations where each row contains all features of a single record arranged in columns, $\mathbf{y}$ is response vector, containing the target for each row, and $\|\cdot\|_2$ denotes the ℓ_2 norm. In this computation, all features are equally considered, which might lead to significant underfitting if a feature is uncorrelated with the target. Lasso

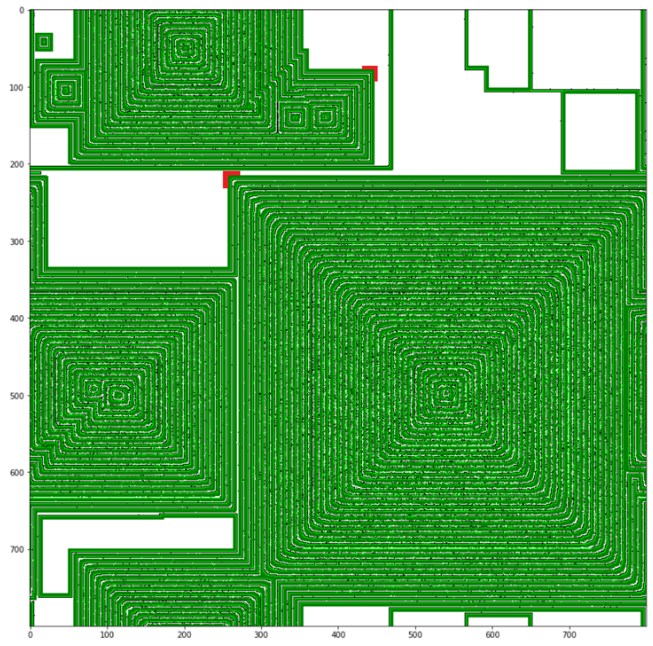


Figure 5: Corners (highlighted by red borders) that can be detected by image processing libraries.

aims to combat this shortcoming by adding a penalizing factor to the object function, which now reads $\|\mathbf{Xw} - \mathbf{y}\|_2^2 + \alpha\|w\|_1$, where $\|\cdot\|_1$ denotes the ℓ_1 norm. As a consequence, lasso will tend to prefer solutions with fewer nonzero entries in $\mathbf{w}$, effectively limiting the number of features we consider [23].

- **Decision Tree Regressor**: this model is the simplest of the four regressors. Essentially, given a data set, a DT regressor will begin to cut the data based on its ability to predict values correctly based on the cut. The data is then cut recursively until there are not enough data points to cut into smaller sections or until the model reaches its predetermined maximum depth [9, p.169-180].

3 Methods

From several thousand simulations of these models at various sizes (side lengths of 800×800, 1000×1000 and 1200×1200), image features were collected every five timesteps until $t = 200$. The implementations of these models are described in section 3.1, and the chosen image features are detailed in section 3.2. Once this step of data collection was complete, several aforementioned regression models were used to predict the final steps of each of the HIV models. The hyperparameter tuning (HT) of these regression models and the specific hyperparameters used are described in section 3.3. Figure 6 helps visualize this process.

3.1 Implementations of HIV CA Models

Instead of using the original implementations of the HIV CA models, the authors of this paper reimplemented them with five optimizations that would allow for the models to run faster and at larger sizes, based on previous research [7]. Listed below are the five optimizations:

1. *Just-in-time (JIT) Compilation*, which saves time by compiling the code as it runs, rather than Python's usual method of interpreting each line of code.

2. *Parallel Processing*, which allows the code to use more of the computer's cores at once for its calculations.

3. *Xoroshiro_128 Random Number Generation*, which is a multithreaded way to generate new randomized values. This is especially useful when determining if a cell will change to a certain state based on probability.

4. *Swapping memory addresses*, which saves computational power that would otherwise be spent copying data at a memory address to a new one. This is especially helpful when moving the replacing the old grid with the buffered grid.

5. *Addition-based rules*, which allow the count each cell state as a number rather than a class, which allows for Moore neighborhood checks to be boiled down to a simple mathematical equation.

To verify that these new optimizations did not alter the outcomes of the HIV models from the findings of the originals, a sample of ten simulation runs were taken to find average cell counts of the four classes (healthy, infected and detected, infected and not detected, and dead) at each timestep. Then the minimum number of replications needed to produce a confidence interval of 95% was calculated at each timestep. The maximum of these values found then became the minimum number of replications the new optimized model needed to run to achieve proper confidence that the optimizations had not affected the outcomes of the models. Once the confidence intervals were constructed, the cells counts were compared between the optimized models and original models. This allowed for verification of the emerging biology of the models. After verifying the biological aspect of the newly optimized CA models, full simulation runs with all 600 timesteps were put into short video files to verify that the optimizations did not change the image analysis aspect of the HIV modeling. With both points of verification passed, the authors of this paper are confident that the optimizations maintained the biological

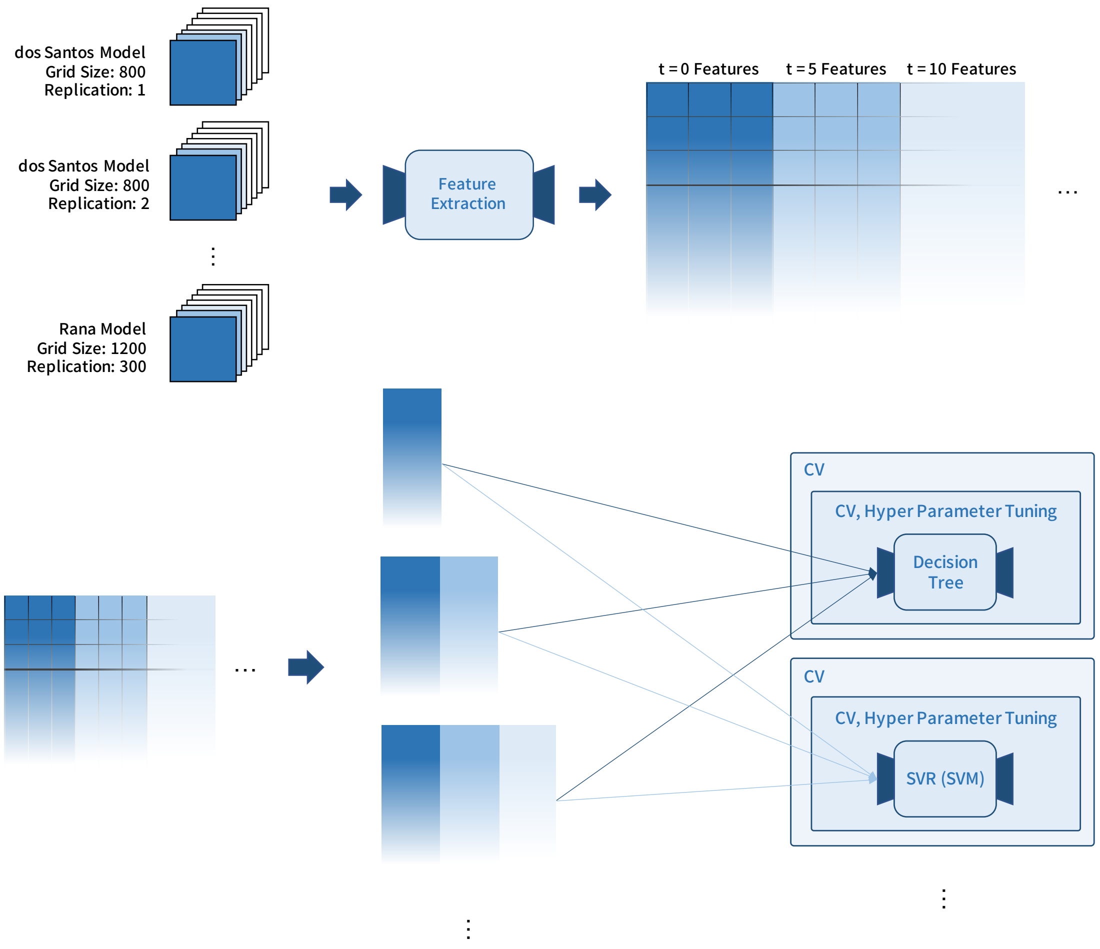


Figure 6: Summary of the Machine Learning process followed in this report. We repeatedly run all simulation models for various grid sizes and extract image features based on the saved simulation grids. These data are organized into feature matrices (spreadsheets) and are divided into sections by the time step from which the features were derived. We then build subsets of the original spreadsheet as shown in the diagram, and use them as training data for the four regressors. Nested CV is employed to prevent overfitting during hyperparameter tuning.

and visual integrity of the original CA models while dramatically increasing the speed at which they process.

3.2 Image Feature Collection

After running each HIV simulation model 300 times at the three desired grid lengths (800×800, 1000×1000 and 1200×1200 cells), all of the timestep data from all grid lengths was collected into one dataset for image feature analysis. (The different grid lengths were combined into a single dataset to prevent regressors overfitting to one particular size, and instead allow regressors to generalize to any desired grid length.) Special attention was given to any model with treatment, since the introduction of treatment completely changes the dynamics of the cells within the simulations. Even though current results are based on starting treatment at the time specified in the original models' papers, we also investigated the effect of administering treatments at $t = 2$, $t = 4$, and $t = 8$ [1, 18][2]. Using the SciKit-Image and OpenCV libraries, image features were identified at timesteps $t \in \{0, 5, \ldots, 195, 200\}$. This range of timesteps was selected to build a linear response curve that would give us an early indication of how the system was responding to different sets of timestep data. If we are able to prove the link between end states of CA models and image features

[2]The González model introduces treatment normally at $t = 300$. Since the authors of this paper only collect image features up until $t = 200$, the error was exceptionally high for this model because the CA dynamics when the features were collected differ wildly from those that exist at $t = 600$. Thus, the introduction of treatment at $t = 2$, $t = 4$, and $t = 8$ does not align with the original González model. The authors acknowledge this difference, but argue that a differing approach to treatment administration times is necessary for this paper to achieve its end goal, and a compromise must be made.

in the early stages, further research will more finely explore the extent of this relationship. For the purposes of this report, a timestep gap of five will provide a rough picture of the tendencies of the system. Once image features were identified, there came a challenge: *how does one turn the location of a feature into a numerical value*? For the purposes of this paper, this team focused on easily obtainable numerical values such as the number of occurrences of a feature or the average distance between said occurrences. Parameters for the functions from the OpenCV and SciKit-image libraries can be found in Figure 13.

- *Blobs*: once blobs were located in the image using the various libraries, a few values were extracted. From the OpenCV library, blobs features found using the `SimpleBlobDetector` included the number of blobs with an area of at least 10, the average distances between any two blobs with an area of at least 10, and the average size of blobs with an area of at least 10. From the Sci-Kit Image library, blob features included the number of blobs located by a standard `blob_log` function, the average size of these blobs, the variance in the X coordinates of located blobs, and the variance in the Y coordinate of the located blobs. The cutoff of 10 is chosen so that we will filter out cell-by-cell level information and focus on behaviors arising from ensembles, especially at the early stages of the simulation. An illustration of this idea is given in figure 7. When the figure has many bubbles of infected cells in the $t = 1$ plot, our cutoff is not overly restrictive as to wash out those blobs, and when there are only a few big blobs at $t = 15$, the cutoff is suitably large to ignore the pixel level infections. However, when treatment is turned on for the models that have it, the blobs become intermixed and many of them are small, as shown in figure 8. In this case, although the cutoff is serving to reduce the amount of trivial blobs, one might argue it is too aggressive. Further research will examine if using smaller cutoffs, especially for models with treatment, will result in more informative image features.

- *Contours*: after locating contours using the OpenCV library's `findContours` function, the following values were obtained: the number of contours with an area of at least 10, the average area of contours with an area of at least 10, the number of contours with an area of at least 100, the average area of contours with an area of at least 100, the number of contours with a perimeter of at least 10, the average area of contours with a perimeter of at least 10, the number of contours with a perimeter of at least 20, he average area of contours with a perimeter of at least 20, the maximum area of all contours, and the maximum perimeter of all contours. Similar to blobs, the area and perimeter values of 10 were chosen in an attempt to filter out noise from the data. The larger values of 100 for area and 20 for perimeter were chosen to see if the smaller contours made a difference, or if only larger shapes found within the image had bearing over its final state. An illustration of different sized parameter cutoffs is shown in figure 9. When the figure has many specks in the $t = 1$ plot, our cutoff is large enough to filter out the small contour shapes with a perimeter less than 10. A contour perimeter cutoff size of 20 is too large for the first couple timesteps and filters out all of the contour perimeter sizes. When observing the $t = 15$ timestep, the cutoff at 10 is large enough to filter out the trivial contours with parameters less than 10 similar to the blob cutoffs. The cutoff of 20 filters out for parameter sizes and is more restrictive that the cutoff at 10. When the simulation reaches $t = 100$, the number of contours with larger perimeters drop and the cutoff of 10 and 100 both filter out the specks of infected cells with parameter sizes less than 10. For all three timesteps, one might argue that there is not that much of a difference between a cutoff of 10 and a cutoff of 20 and that the second cutoff should be a larger value to test if there is a difference between being aggressive with the cutoff sizes and being relaxed with the cutoff sizes. The different cutoffs of contour areas is illustrated in figure 10. When the figure has many specks of infected cells as in the $t = 1$ plot, our cutoff is large enough to filter out the small contour shapes with an area less than 10 just like in the perimeter figure. Just like in the previous figure on contour perimeter sizes, the cutoff greater than 10 ended up filtering out all of the contour. When observing the $t = 15$ timestep, out cutoff of 10 is still able to filter out the trivial contours with an area less than 10. There are also a more medium sized contours with areas between 10 and 100 which are being filtered out by our second cutoff size of 100. This is too aggressive of a cutoff size when there is a small number of contours with areas greater than 10 such as in $t = 1$ and $t = 100$ but is not too aggressive when there is a greater number of contours with areas larger than 10 as in $t = 15$. At the $t = 100$ timestep, the graph is relatively flat past the contours with areas greater than 10, so one could argue that the size cutoff of 100 is too aggressive and a smaller value of the cutoff could be chosen.

- *Corners*: using the SciKit-Image library, the following values were obtained: the number of corners found by the `corner_harris` function and the distance between these corners. The Harris Corner Detector function essentially works to identify corners by creating a map of the image that associates a "corner score" with each pixel. Used in conjunction with the `corner_peaks` function, clusters of high scores are identified as corners within the image.

3.3 Regression

With the image features collected, the data is nearly ready for regression. *Regression* is the statistical practice of predicting some numerical measure of a new data point in a dataset, given the characteristics of said data point. The target measures for the HIV CA models are the percentages (as opposed to the counts, since multiple grid sizes were used) of each target class (healthy, dead, latently infected, and acute infected). From each of the simulation runs of all five HIV models, all 4 class percentages were calculated at the final timestep, $t = 600$, or 600 weeks after the onset of HIV in the human body. Once the composite data set of image features and target class percentages is created, regression can begin. Each regression task has a rather complex process, described in Figure ?. Essentially, the regressors choose a target numerical value to predict, and are fed increasingly more information; at first, they have access to the data from $t = 0$, then from both $t = 0$ and $t = 5$, and so on until the regressor has access to the data from all timesteps during which image information was collected. At the end of each regression task, an error rate is calculated using *root mean squared error* (RMSE), which, for a sequence of data $\{y_i\}_{i=1}^{N}$, is given by

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \bar{y})^2}{N}},$$

where $\bar{y}$ is the sample average. The RMSE is a positive value that dictates how far a regressors prediction is away from the target value. For instance, if the target value indicated that 7.3% of the cells at the end of a simulation run were dead, and a regressor predicted that 6.1% of the cells would be dead given information from timesteps 0 through 100, an RMSE value of 1.2. In context, this would mean that this particular regressor will predict, on average, a percentage of dead cells that is off from the true percentage by 1.2%.

The regressors are trained by attempting to lower the RMSE error without approaching an error of 0, with the end goal of generalizing predictions on new data. If an error of 0 were to happen, then the model would not adapt well to new data points, and would be *overfit* to the current dataset. Thus, a rigorous approach must be taken to lower error without overfitting. This process is a combination of *nested cross-fold validation* (CFV) and *hyperparameter tuning* (HT). When training a regression model on a dataset, there must be a subdivision within the dataset to test the model on. It cannot be tested on data it has already seen; rather, the test set of data is not seen by the model until it is being tested for accuracy. This, to properly validate the accuracy of a model, CFV is utilized. This process breaks down the dataset into k *folds* or "chunks". The model is first trained on the first $k-1$ folds, and tested on the kth fold. This process is repeated until all k folds have been utilized as the test set of data. The *nested* portion of this process comes from further subdividing the dataset one more time into a *validation set*. This validation set is a subset of the training set already previously identified. Why validate the model before testing it? This comes back to HT. Hyperparameter tuning is a process by which the various parameters of a model are tuned to have the lowest error. In order to find which given set of parameters obtains the lowest error, each regressor must be validated. Listed below are the various models used and the hyperparameter ranges that were tuned. Specific hyperparameter ranges can be found in Figure 13, if they deviated from the default values.

- **Support Vector Machine**: We chose to use SciKit-Learn's Support Vector regressor. This regressor used two kernels: the Radial Basis Function (RBF) and Polynomial kernels, as well as the `gamma`, `C`, `epsilon`, and `degree` parameters The authors used a range consistent with prior research [12], While, according to this report's findings, it is desirable that a larger range is used, the authors used a smaller range to save on computational resources.

- **Gaussian Process**: We chose to use SciKit-Learn's Gaussian Process regressor. We used the RBF kernel and varied the `alpha` parameter.

- **Lasso** We chose to use SciKit-Learn's basic Lasso regressor. Unlike previous regressors, this regressor does not have different kernels. Instead, the only hyperparameter adjusted in this research is `alpha`.

- **Decision Tree** We chose SciKit-Learn's Decision Tree regressor. Parameters `max_depth` and `min_samples_split` were tested.

Note that we chose to use these four regressors because of their pervasive usage within the machine learning academic sphere. A convolutional neural network (CNN) did not make sense in this context given the complexity of the model and lack of data. Additionally, in light of the novelty of this approach, simpler models were preferred for the authors' and reader's ability to comprehend the results and their implications better than the "black-box" approach of CNNs.

Once all parameter combinations have been validated within the inner fold (the fold the CFV with the validation sets), then the regressor with the best parameter combination moves to the outer fold to calculate an average error and standard deviation from this average error using the *k* folds of test datasets. Once again, this complex process is described in figure 6.

Finally, after CFV and HT, a *factorial analysis* (FA) is performed. This process involves two steps. First, each of the features is either included in a test data set or removed. RMSE scores on a Decision Tree are calculated to determine how each factor affects the outcome of the RMSE. After that, once all of these RMSE scores have been calculated, interactions between the features are computed and all RMSE scores are analyzed to see which feature or feature set had the most effect on the outcome. For this FA, the authors bundled each of the similar features together into 7 categories: blob features seen by the `SimpleBlobDetector`, contours features of area 10, contour features of area 100, contour features of perimeter 10, contour features of perimeter 20, blob features located by the `blob_log` function, and corner features. Since the resulting FA has a total of 2^7 (128) combinations, the results were further filtered into any feature or feature set that contributed 5% or more to the outcome of the RMSE. The results from this rigorous process are provided in section 4 and discussed in section 5.

4 Results

The objective of this section is to show how successful the different regressors are at predicting the final state of each CA model based on the features extracted. In the following sections, we first examine the feature extraction algorithms used and then examine the plots generated after running the four regressors.

4.1 Feature Extraction

After each CA model was simulated and a grid was saved for every time step, features were extracted based on what was observed. As the number of infected cells spread, this affected the number and type of features detected during the feature extraction process. There were 20 features extracted such as the number of blobs with certain areas, corners and contours. When the $t = 1$, there are a couple blobs detected which represent the first infected cells. Then, as the number of infected cells increased, the number of features detected increased in tandem. When observing a grid after 40-time steps, in all the models except the Rana model, a ”ripple effect” was observed. This ripple effect refers to the process of an infected cell infecting its neighbors, which then in turn infect their own neighbors, and so on. This effect is similar to when a rock is thrown into a lake which creates a ripple across the surface of the water. When the original cell dies, another ripple is created. A new healthy cell then replaces the dead cell which then becomes infected again. This process continues and creates several ripples. For every time step afterwards, the ripple effect spreads until a steady state is reached; the number of each type of cell remains relatively constant. As the ripples spread, it leads to more blobs, corners, and contours being detected.

4.2 Regression Plots

Once the features were extracted, they were run through different regressors to see if the last time step could be predicted. After running the regressors, the data that was generated was compiled into plots of the RMSE, which were then examined to see how each regression algorithm performed over time. Each cell count was converted into a percentage before being fed into each regression algorithm.[3] Thus, the RMSE data indicates by what percentage a regressor's prediction deviated from the expected percentage of a target cell class. The information on the contribution of each feature to the overall RMSE can be found in Table1.

4.2.1 Parameter Tuning

After performing a nested CV as described in section 3 on our methods, it is observed during each outer fold, the best parameters chosen by the inner fold stayed roughly the same, except for a few situations where randomness in data selection resulted in abnormal training data. This behavior agrees with our expectation. Particularly illuminating however are the parameters for the decision tree regressor, since the parameters chosen (`max_depth` and `min_samples_split`) have rather straightforward implications on our data. For most models and classes, `max_depth` remained low (at 4), indicating that our regressor might not have enough informative features to base predictions on.

4.2.2 Support Vector Machine

The first regressor investigated is the support vector machine shown in figure 11. For the dos Santos graph,

[3]Note that, though these graphs are grouped by regressor, they do not share the same scale. Each of the RMSE graphs has a radically different magnitude of error, from 10^{-4} all the way to 10^{-1}. This drastic difference between scales is not conducive even to a logarithmic scale graph, as all four individual classes compress into an unreadable format. Thus, when reading these graphs, take care to notice their scale, as each model tends to stay within its own scale. This is explored more in the discussion, which proposes the idea that results are heavily model-dependent.

the RMSE for healthy cell plot started around 0.02 and stayed the same for every timestep. This is the same for the dead and infected cells. From examining the plot further, it can also be seen that the RMSE for the acute infected cell plots has the highest variance while dead and latent infected cell plots have the smallest variance. For the González model, the RMSE plots also stayed about the same throughout the simulation. The RMSE variance for the acute infected cells and the healthy cells has a lot of overlap and the latent infected cell plots has a lot of overlap with the dead cell plots. The graph for the Rana graph follows a similar trend to the last two and the Precharattana graph mostly follows a similar trend to the other graphs except for the start and middle of the simulation. During the first couple time steps, the variance decreases, increases again in the middle of the simulation and then decreases again. The Moonchai graph showed the most change than any other graph, mainly for the latent infected and healthy cell plots. The two plots follow a decaying sinusoidal pattern as the time step increased. Another thing to note is that the Moonchai and Precharattana graphs had the smallest RMSE with under 0.0008 and 0.0006 respectively while González had the largest RMSE at 0.15.

4.2.3 Gaussian Process

The second regressor examined is the Gaussian Process regressor shown in figure 18 which has the same range for the RMSE as the SV regressor. One notable exception is in the first couple time steps of the Moonchai graph. At the start of the simulation, the calculated RMSE for the latent infected cells is around 0.0005 and decreases for the next couple timesteps. As the simulation continues and the number of latent infected cells increases, the calculated RMSE at time step 25 jumps up to 0.0010 and has a larger variance. This coincides with with the rapid expansion of infected cells after the latent period is over. The RMSE for the latent infected cells and the acute infected cells is different than the SV regressor in that the latent infected cell RMSE plot is higher than the RMSE plot for the acute infected cells. This is only true for the time steps after 30 because before this the RMSE for Moonchai's latent infected cells was higher than 0.0006. The RMSE for the Precharattana and González graphs is similar to the SV regressor. The only difference in the González graph of this regressor is that there is a slight drop in the start of the graph until time step 30 when it goes back to the starting value. The Rana graph has a maximum RMSE of about 0.006 and stays at around the same RMSE through the entire graph. There is one drop in the RMSE at time step 15 but jumps back up to the previous RMSE at time step 25. This is interesting because this is the one of the only rund where the RMSE ended better than it started. Except for the Moonchai plot, the RMSE is greater for the healthy and acute infected cell plots and lower for the latent infected and dead cell plots.

4.2.4 Lasso

In the Lasso regressor shown in figure 13, the ranges for the graphs are roughly equivalent to the last two regressors except for the Moonchai graph for the Gaussian process. The González graph has the highest RMSE and the Moonchai and Precharattana graphs have the smallest RMSE. In the Precharattana model, the RMSE for the latent infected cells is 0 throughout the whole simulation. The Rana graph with this regressor has the most fluctuations even though it ends and begins with the about the same RMSE and variance. The graph shows that the regressor was the least accurate at predicting the outcome around the 25th timestep but got gradually better afterwards. The Rana graph RMSE was improving at the beginning of the simulation but once the medication treatments were implemented, the Rana graph RMSE got progressively worse. The latent infected and the dead cell plots were two cell types that had the smallest RMSE in all the graphs while the acute infected cells and the healthy cells had the largest RMSE. In the dos Santos and the Rana graphs, the healthy cells had the largest RMSE while in the Precharratana, Moonchai and González graphs, the acute infected cell plots had the highest RMSE.

4.2.5 Decision Tree

The last regressor examined is the decision tree regressor shown in figure 14 with the RMSE being plotted for each time step in each model. When looking at the figure, González has the largest RMSE just like the other graphs. The plots with the smallest RMSE plots are Moonchai and Precharattana. When looking at the five graphs, it can be seen that the latent infected cells plots have the smallest RMSE and variance than other three cell types. The RMSE for each graph stays about same the with some fluctuation throughout the simulation while the variance either increased or decreased for each graph. The Rana graph has a large fluctuation in the beginning but decreases and levels off at about the 20th time step. The dos Santos graph has the opposite effect where it starts with a smaller RMSE and increases for the next couple time steps. It has another spike in its RMSE at around time step 105 and 110.

4.3 Special Treatment Introductions

For the two models with ART treatments, the authors wanted to analyze the effects of early on-set treatments on the spread of HIV, as discussed in the methods section. This subsection will focus on comparing the original results with each of the various treatment introduction times.

4.3.1 Treatment at 2 Weeks

As seen in Figure 15 , the Rana model has no particularly interesting patterns seen in the RMSE over time for any of the models, except perhaps the Decision Tree, during which the Rana model oscillated irregularly across a small range of values much more than the other models, though the difference in RMSE in these variations is not of great significance, as the RMSE hovers around a value of between 0.0001 and 0.0010. The González model, on the other hand, showed a similar pattern to Rana for only the SV regressor. However, all of the other regressors showed their own characteristic differences. Notably, the Tree seems to be the only of the models that had a consistently lowering RMSE, which would suggest that it is following a pattern of increased performance with increased access to data. The Lasso regressor follows that pattern until around timestep 125, at which point it starts to rise again: not to the original RMSE values at the $0th$ timestep, but still higher than 100 weeks. The GP regressor also appears to be rather consistent except for a drastic dip in RMSE scores before t=25.

4.3.2 Treatment at 4 Weeks

As seen in Figure 16 , the Rana model once again maintains its similar pattern of steady RMSE scores across the board, with slight variations along the way. The Gonzĺez model, however, shows great promise. While most of the models so far have held relatively steady for most of the regressors, the González model had increased performance with increased data almost across the board for three of the four regressors. The Decision Tree had a consistent decline in RMSE, minus a slight bump upwards at t=20. The SV regressor had a slower, but still markedly negative curve to its RMSE graph. The Lasso also showed a continual decline in RMSE. In fact, the only regressor that did not show this consistent decline is the GP regressor, which maintained the pattern of dipping the RMSE at t=25 and then returning to consistent values.

4.3.3 Treatment at 8 Weeks

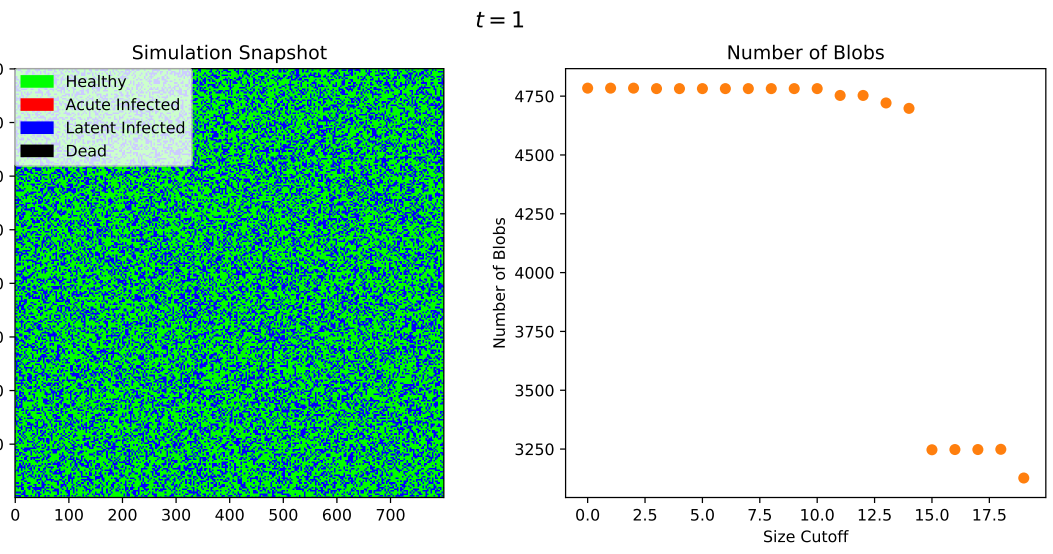
Unfortunately, as explored in Limitation 3 in section 5.3, these were unable to finish in time, despite given almost 5 full days of work. One of the regressors, the González 1200x1200 model took far longer to complete, and in fact did not complete in time, than the other models. Thus, a complete analysis for this data must wait until all the data has been completely processed: a sad fact that cannot be met before this paper's deadline.

5 Discussion

HIV virtual laboratories have existed for quite a long time, as has using cellular automata to simulate a small group of cells. The problem with simulating disease dynamics in an individualized manner for a patient is how computationally expensive it may become. Many patients do not have the access to the resources necessary to run these simulations. With a large grid size, many time steps, and variability requiring up to thousands of replications to guarantee 95% confidence, even a minor increase in cell count or grid size can have a more pronounced effect on the computational requirements of the simulation. There is a balance to be struck between lowering these requirements and providing results that are accurate enough to be confidently used in the decision-making process for treatment. The process detailed within this paper describes an approach that attempts to find this balance using feature extraction and regression.

5.1 Present Work

The first step in this procedure was to identify five different CA models, each with their own ruleset to determine cell state-changes, and seed each model. The models were each optimized if they did not contain previously researched optimization methods such as JIT, parallel processing, and PRNG. Each model was run 300 times for each of three grid sizes (800x800, 1000x1000, and 1200x1200), with data being collected and viewed as greyscale images with the SciKit-Image and OpenCV libraries. Feature extraction was performed on desired time steps up to 200. Then Support Vector, Gaussian Processes, Lasso, and Decision Tree regressors were each used to get the error rate with RMSE. To avoid overfitting the new model to the data, nested cross-fold validation and hyperparameter tuning were employed.



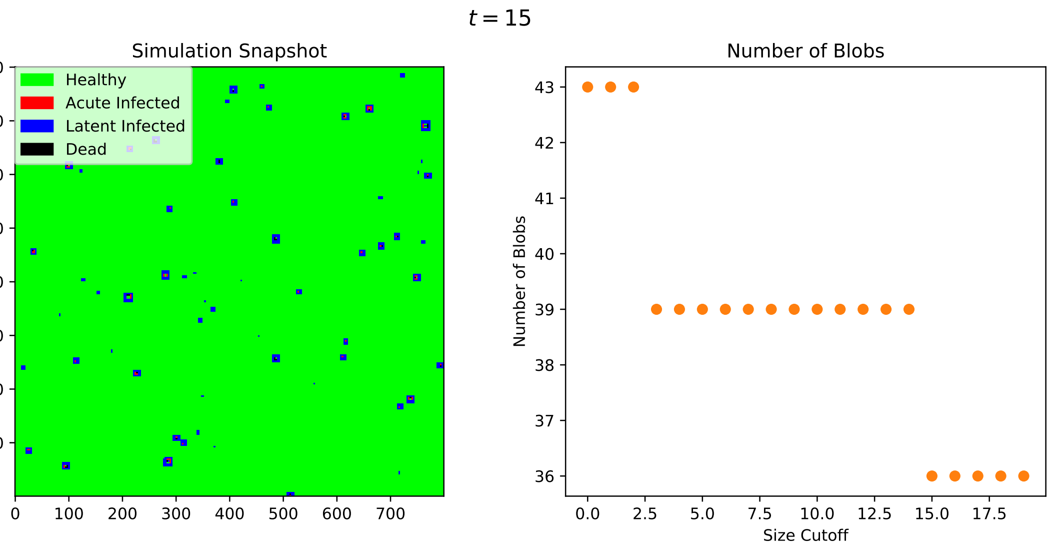


Figure 7: Snapshot of the simulation (left) and the number of blobs detected from that snapshot as a function of cutoff size by the OpenCV Blob Detector at $t = 1$ and $t = 15$ of the Rana model.

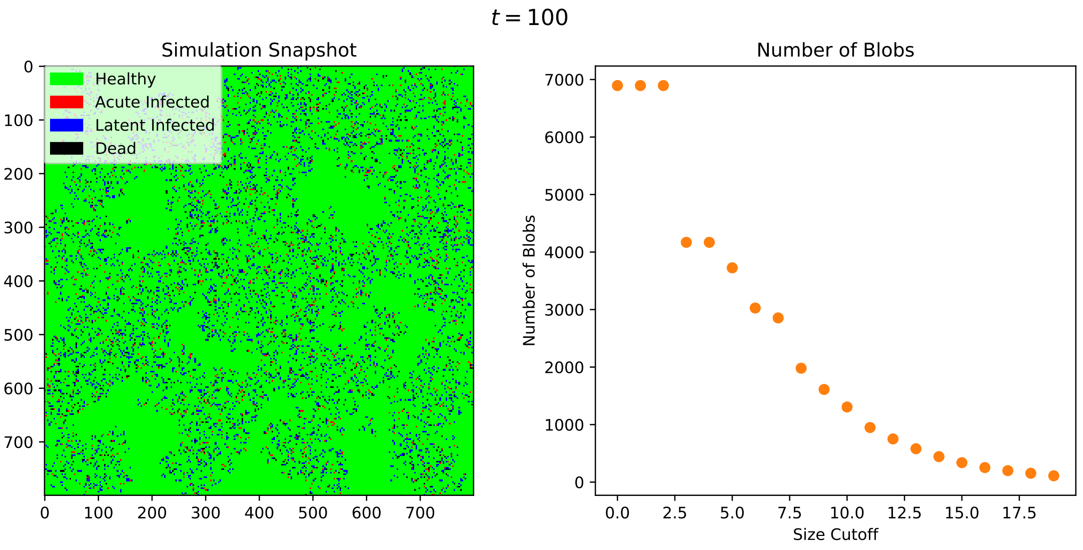


Figure 8: Snapshot of the simulation (left) of the Rana model and the number of blobs detected from that snapshot as a function of cutoff size by the OpenCV Blob Detector at $t = 500$.

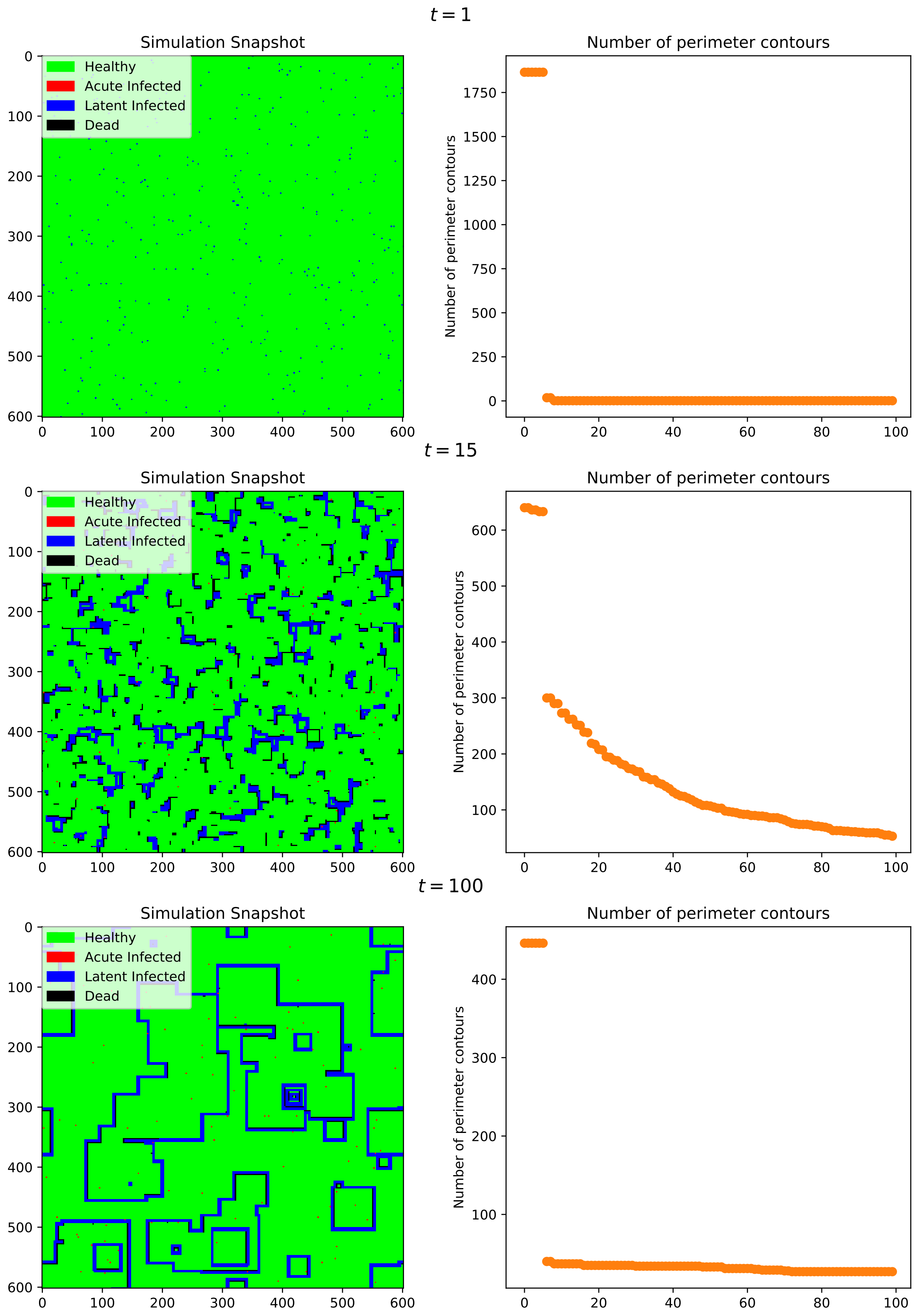


Figure 9: Snapshot of the simulation (left) and the number of contours detected from the snapshot with a perimeter greater than the cutoff at $t = 1$, $t = 15$ and $t = 100$ of the Precharattana Model.

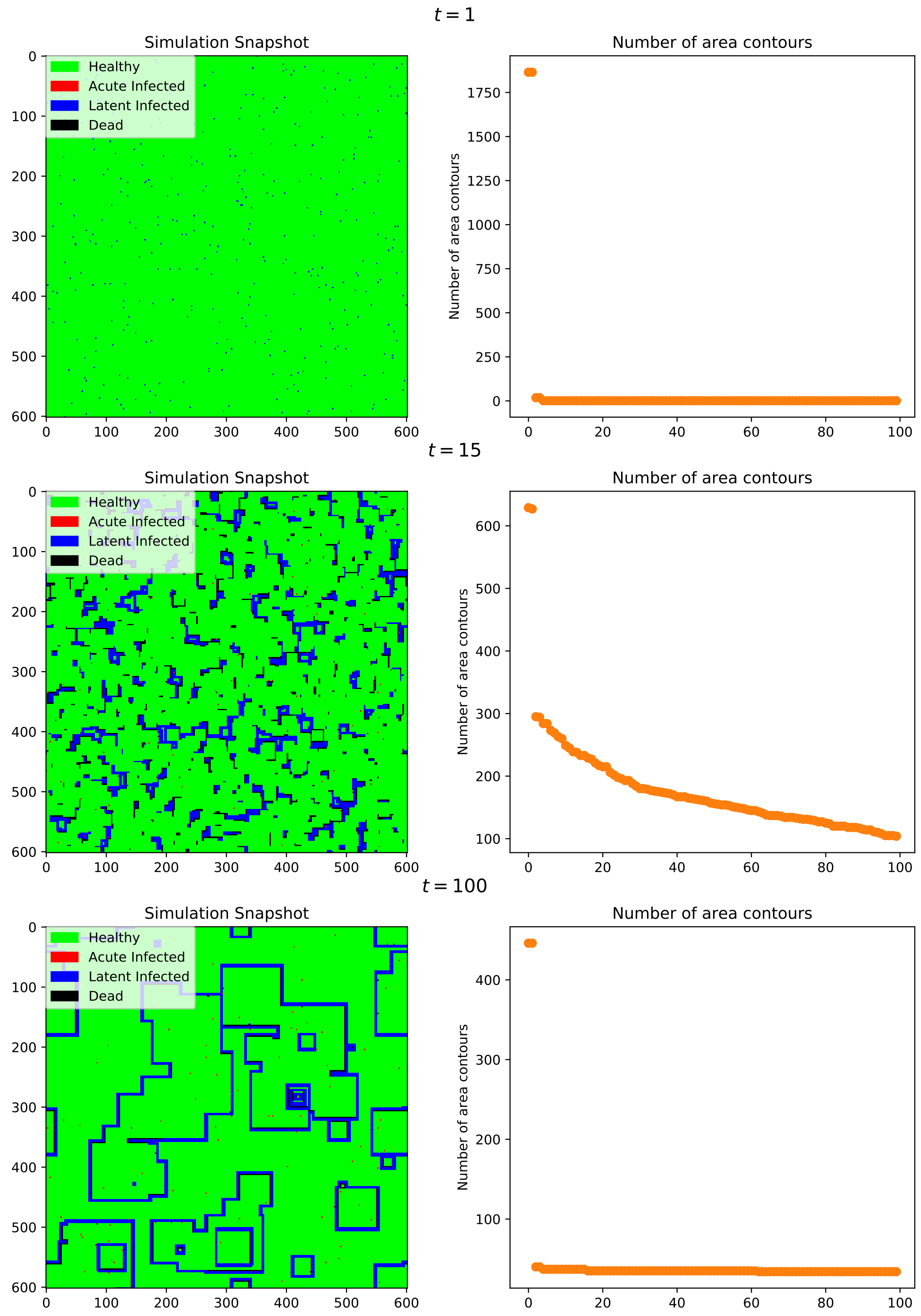


Figure 10: Snapshot of the simulation (left) and the number of contours detected from the snapshot with an area greater than the cutoff at $t = 1$, $t = 15$ and $t = 100$ of the Precharattana Model.

Table 1: Factorial analysis of the individual contributions of a feature or set of features

Model	Cell Type	Feature	Contribution
Rana	H	blob_log	31.89%
	A_1	blob_log	27.51%
	A_2	blob_log	29.15%
	D	blob_log	29.36%
Gonzalez	H	None above 5%	
	A_1	None above 5%	
	A_2	None above 5%	
	D	contours w/ perim 10	7.60%
		contours w/ perim 20	5.56%
Precharattana	H	None above 5%	
	A_1	None above 5%	
	A_2	None above 5%	
	D	corners	5.57%
Moonchai	H	None above 5%	
	A_1	None above 5%	
	A_2	None above 5%	
	D	None above 5%	
dos Santos	H	contours w/ area 100	9.51%
		contours w/ area 10, contours w/ area 100, contours w/ 20 (all combined)	5.06%
		blob_log	12.55%
	A_1	blobs by SimpleBlobFinder, contours w/ area 100, contours w/ perim 10, contours w/ perim 20 (all combined)	5.14%
		contours w/ perim 10	8.40%
		blob_log	15.38%
	A_2	blobs by SimpleBlobFinder, contours w/ area 10, contours w/ area 100, contours w/ perim 20, corners (all combined)	5.45%
		contours w/ area 10, contours w/ area 100, contours w/ perim 10, corners (all combined)	5.08%
	D	contours w/ area 100, blog_log (all combined)	5.05%
		contours w/ area 10, contours w/ perim 20 (all combined)	6.68%

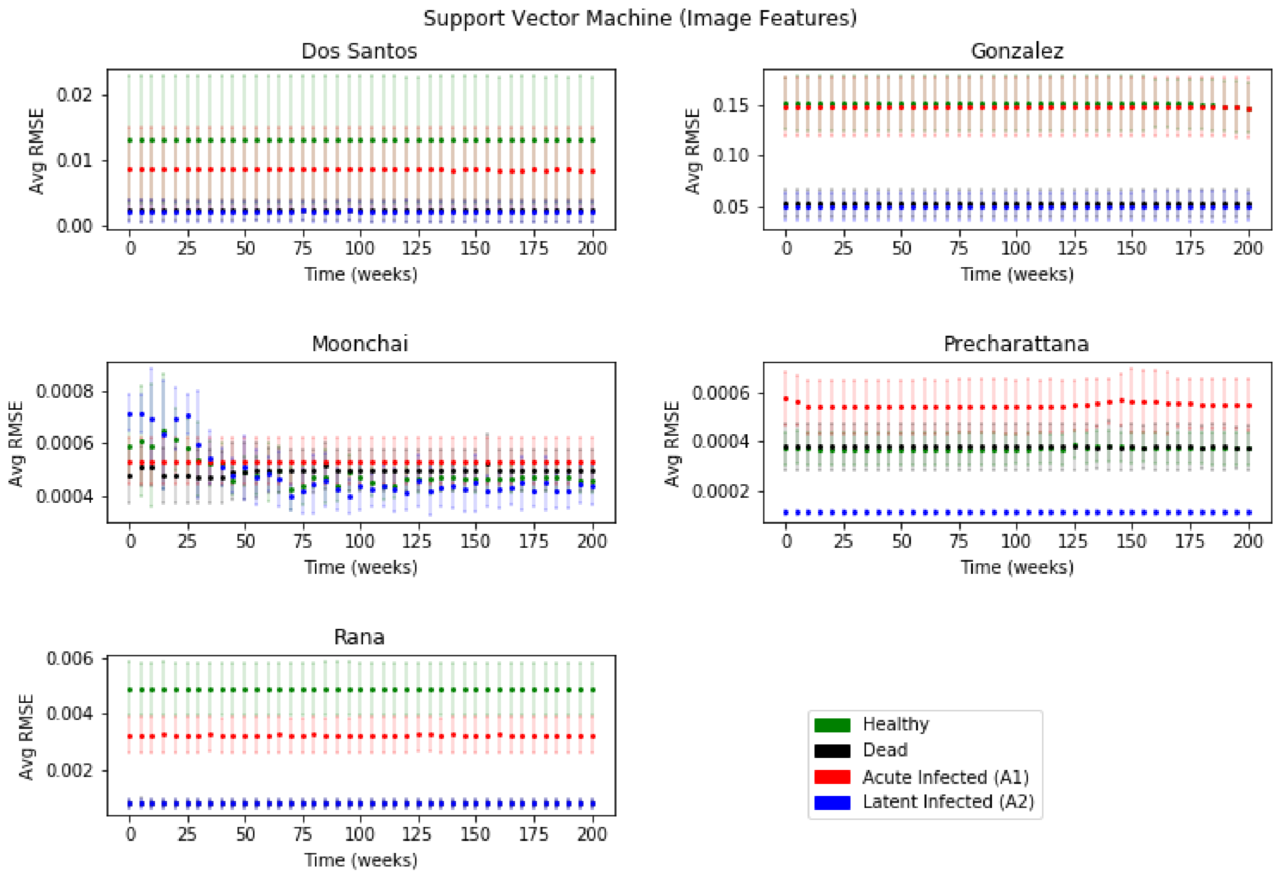


Figure 11: Average RMSE of the support vector machine regressor for data from each CA model after a 10-fold cross validation. The y axis plots the percentages in decimal, i.e. 0.01 means 1% (of total cells count). The x axis plots the maximum t up to which image features were considered by the regressor.

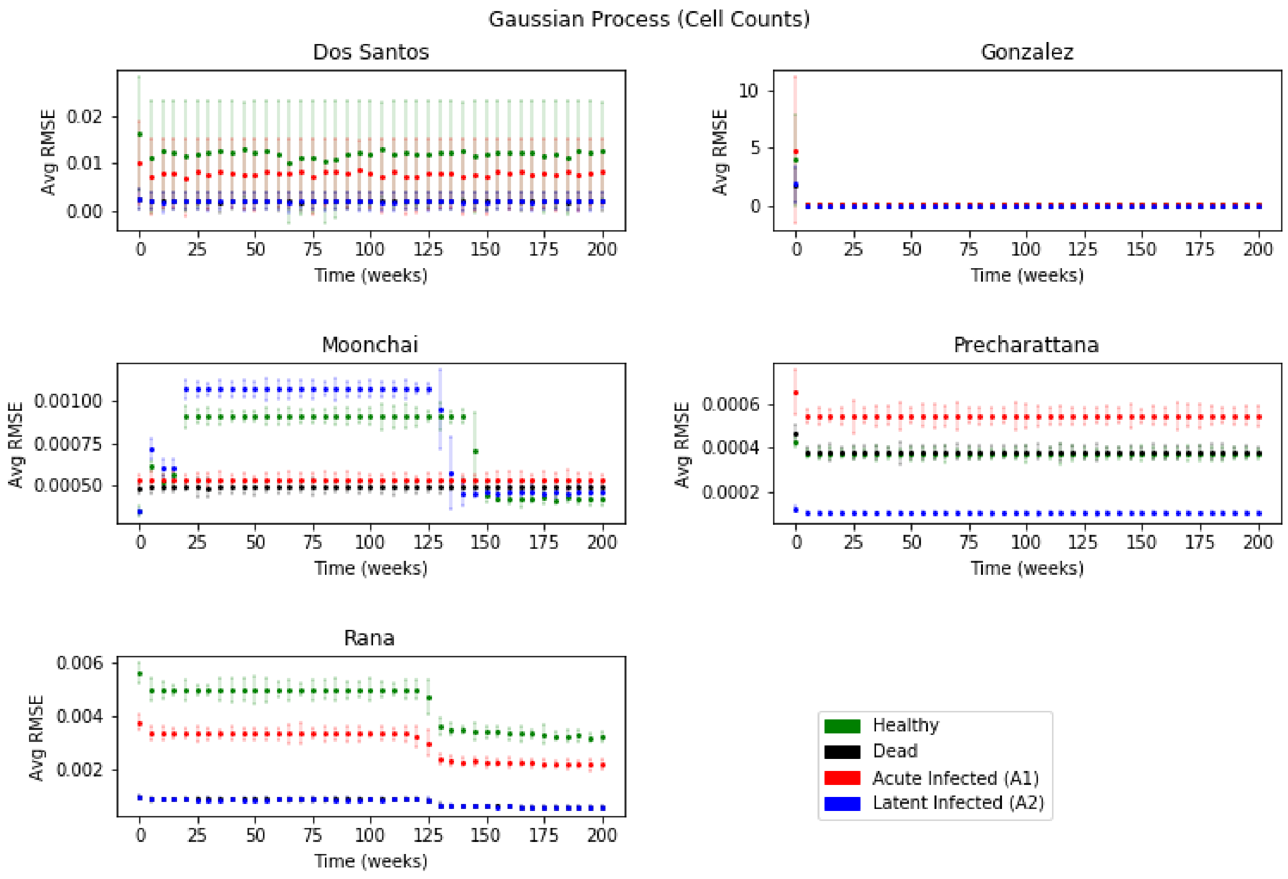


Figure 12: Average RMSE of the gaussian process regressor for data from each CA model after a 10-fold cross validation. The y axis plots the percentages in decimal, i.e. 0.01 means 1% (of total cells count). The x axis plots the maximum t up to which cell counts were considered by the regressor.

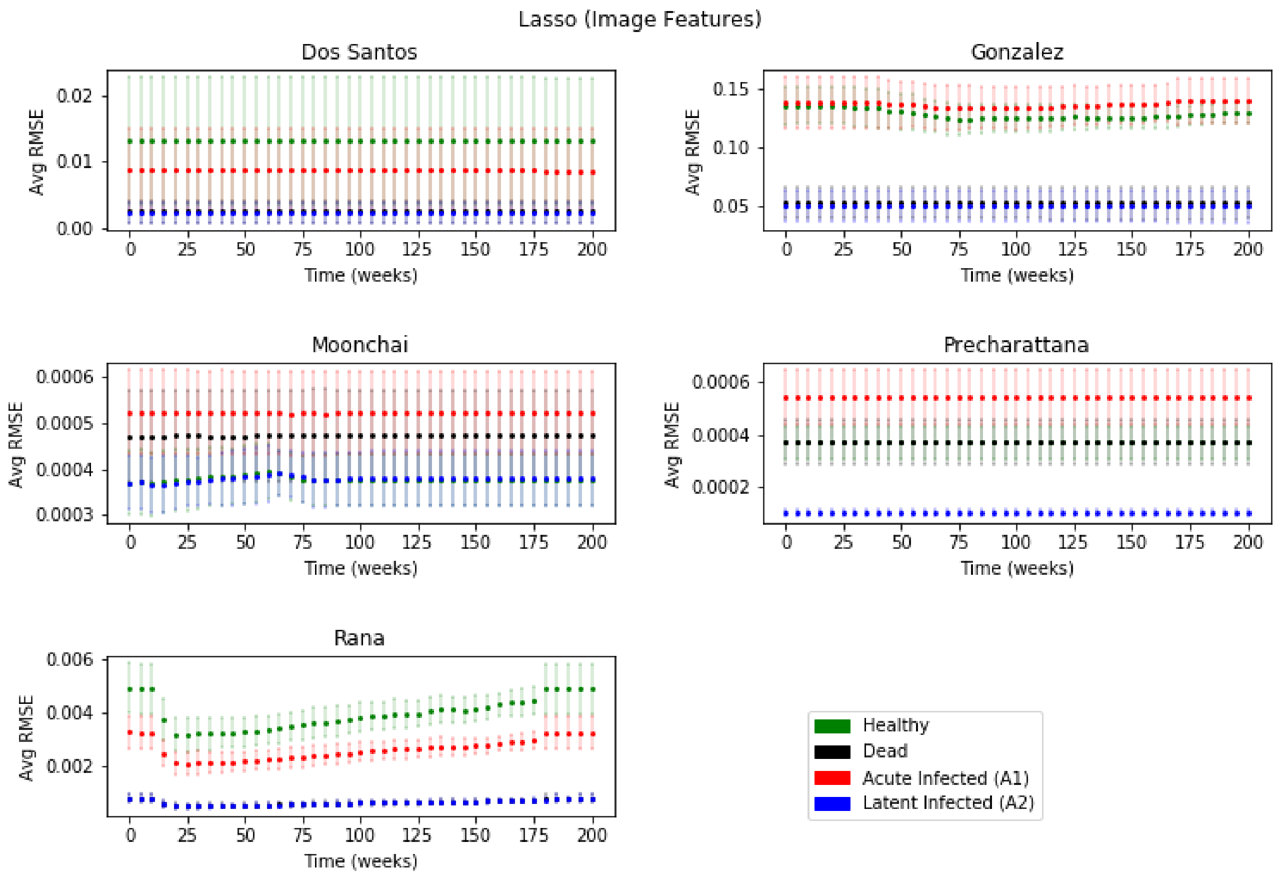


Figure 13: Average RMSE of the lasso regressor for data from each CA model after a 10-fold cross validation. The y axis plots the percentages in decimal, i.e. 0.01 means 1% (of total cells count). The x axis plots the maximum t up to which image features were considered by the regressor.

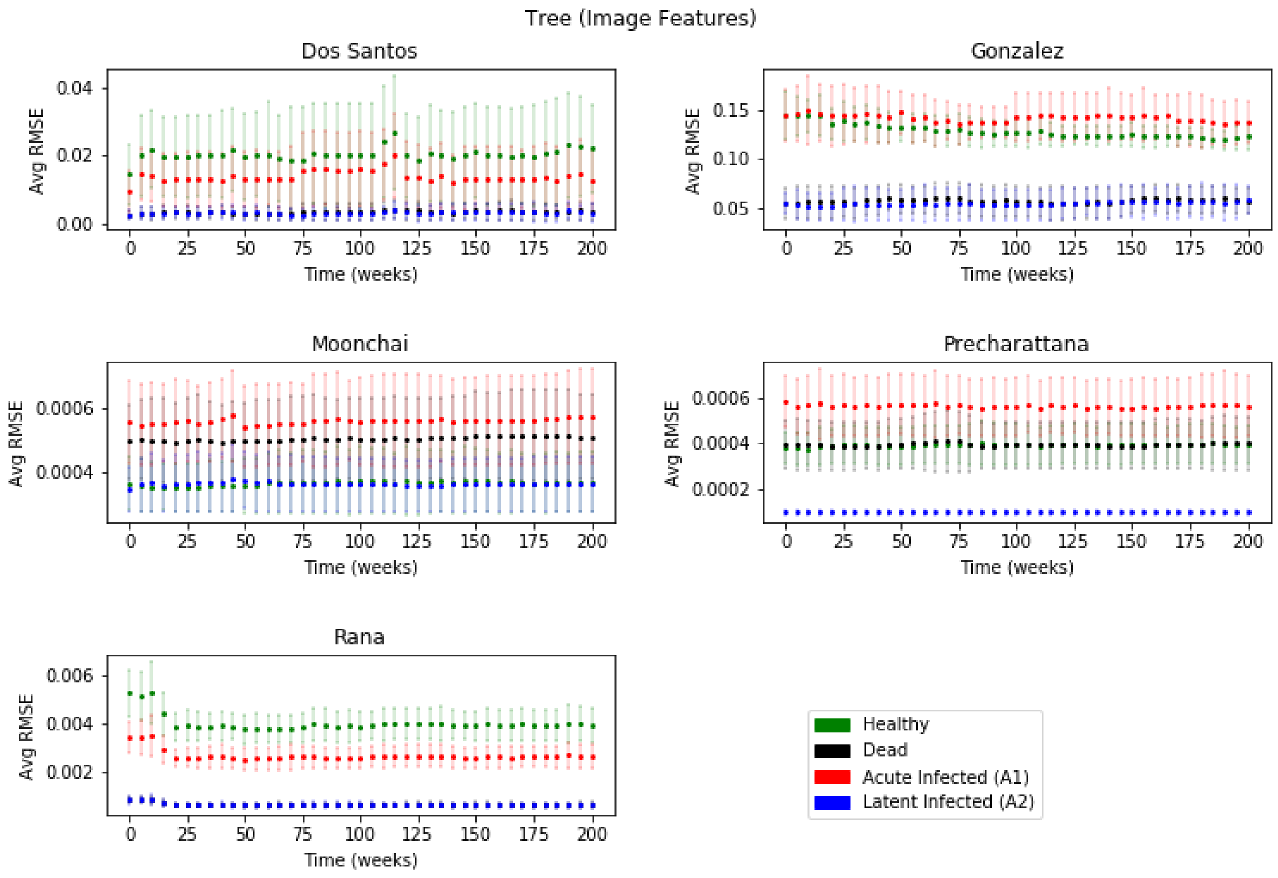


Figure 14: Average RMSE of the decision tree regressor for data from each CA model after a 10-fold cross validation. The y axis plots the percentages in decimal, i.e. 0.01 means 1% (of total cells count). The x axis plots the maximum t up to which image features were considered by the regressor.

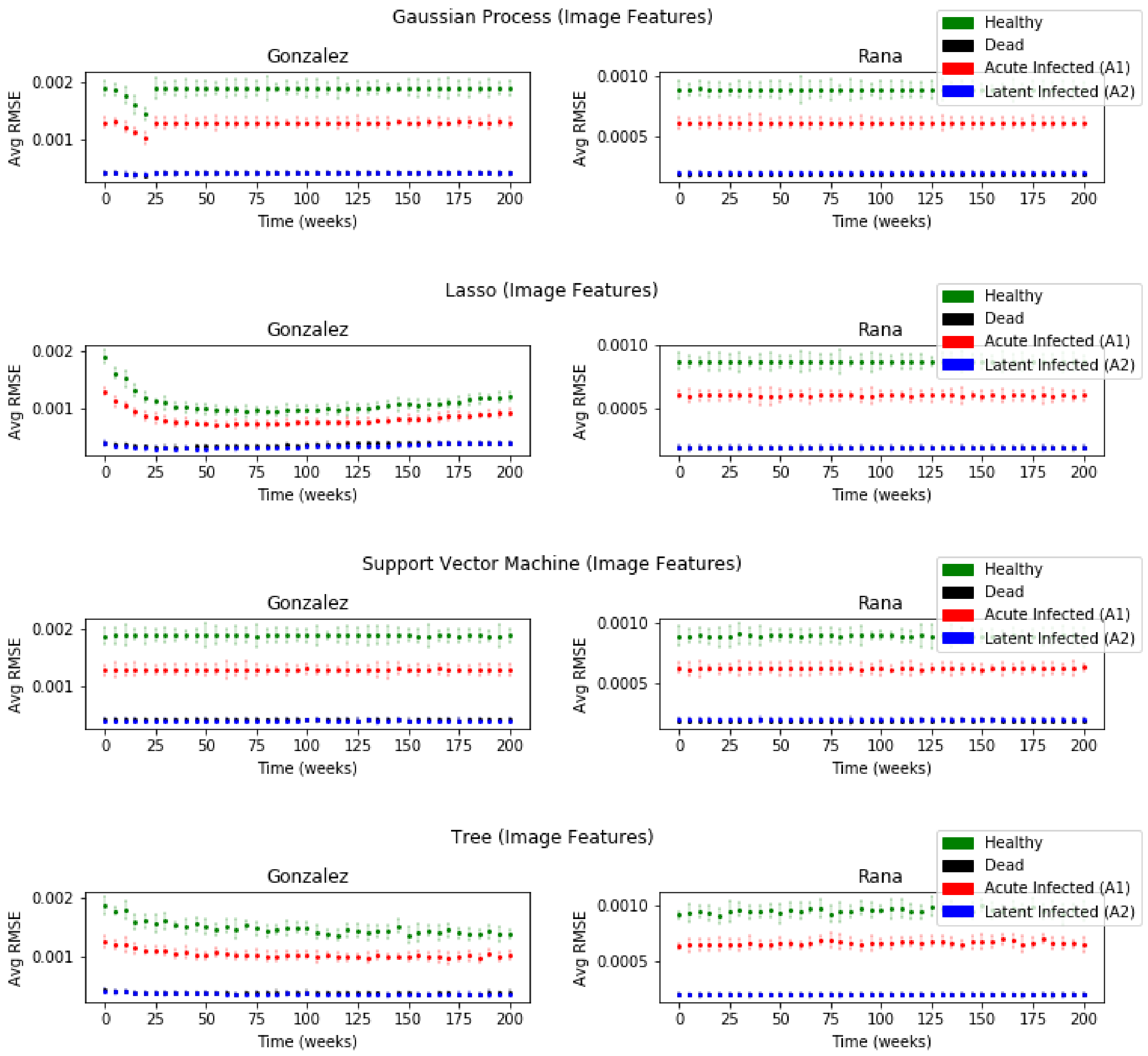


Figure 15: RMSE of all 4 regressors for data from the González and Rana models where treatment was introduced after 2 weeks after a 10-fold cross validation. The y axis plots the percentages in decimal, i.e. 0.01 means 1% (of total cells count). The x axis plots the maximum t up to which image features were considered by the regressor.

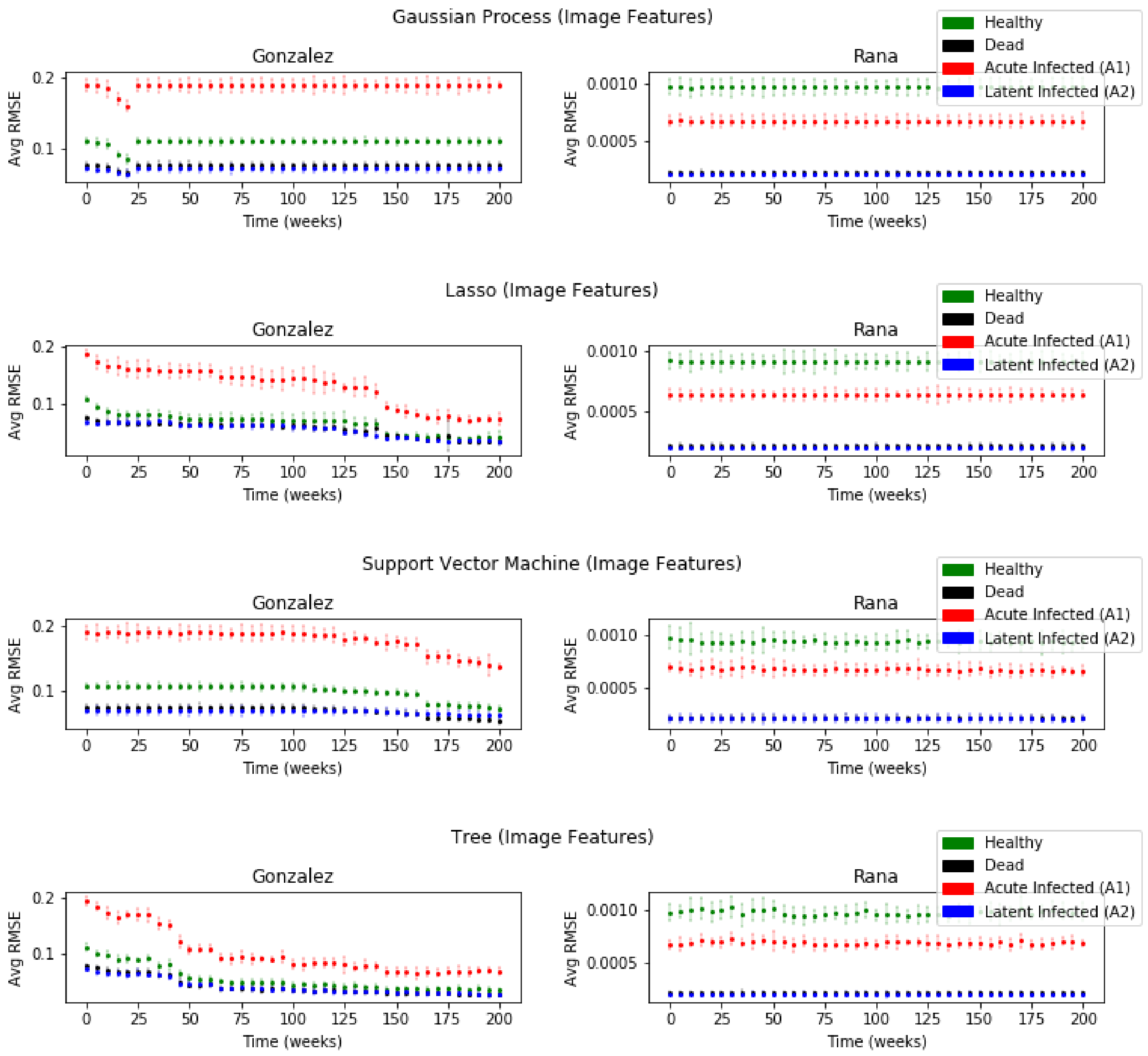


Figure 16: RMSE of all 4 regressors for data from the González and Rana models where treatment was introduced at 4 weeks after a 10-fold cross validation. The y axis plots the percentages in decimal, i.e. 0.01 means 1% (of total cells count). The x axis plots the maximum t up to which image features were considered by the regressor.

Table 2: Table of RMSE ranges for different regressors and CA simulation models.

Model	Regressor	State	RMSE_RANGE
RANA	GAUSSIAN PROCESS	CELL_H	0.000801
RANA	GAUSSIAN PROCESS	CELL_D	0.000131
RANA	GAUSSIAN PROCESS	CELL_A1	0.00053
RANA	GAUSSIAN PROCESS	CELL_A2	0.000133
RANA	LASSO	CELL_H	0.001764
RANA	LASSO	CELL_D	0.000288
RANA	LASSO	CELL_A1	0.001173
RANA	LASSO	CELL_A2	0.000293
RANA	TREE	CELL_H	0.001503
RANA	TREE	CELL_D	0.000249
RANA	TREE	CELL_A1	0.000981
RANA	TREE	CELL_A2	0.000229
RANA	SVR	CELL_H	2.4e-05
RANA	SVR	CELL_D	2e-05
RANA	SVR	CELL_A1	3.2e-05
RANA	SVR	CELL_A2	1e-06
DOSSANTOS	GAUSSIAN PROCESS	CELL_H	1e-06
DOSSANTOS	GAUSSIAN PROCESS	CELL_D	0.0
DOSSANTOS	GAUSSIAN PROCESS	CELL_A1	0.0
DOSSANTOS	GAUSSIAN PROCESS	CELL_A2	0.0
DOSSANTOS	LASSO	CELL_H	0.000115
DOSSANTOS	LASSO	CELL_D	0.0
DOSSANTOS	LASSO	CELL_A1	8.5e-05
DOSSANTOS	LASSO	CELL_A2	0.0
DOSSANTOS	TREE	CELL_H	0.012068
DOSSANTOS	TREE	CELL_D	0.001441
DOSSANTOS	TREE	CELL_A1	0.010515
DOSSANTOS	TREE	CELL_A2	0.001336
DOSSANTOS	SVR	CELL_H	0.000106
DOSSANTOS	SVR	CELL_D	4.2e-05
DOSSANTOS	SVR	CELL_A1	7.3e-05
DOSSANTOS	SVR	CELL_A2	6.2e-05
PRECHARATTANA	GAUSSIAN PROCESS	CELL_H	0.0
PRECHARATTANA	GAUSSIAN PROCESS	CELL_D	0.0
PRECHARATTANA	GAUSSIAN PROCESS	CELL_A1	0.0
PRECHARATTANA	GAUSSIAN PROCESS	CELL_A2	0.0
PRECHARATTANA	LASSO	CELL_H	0.0
PRECHARATTANA	LASSO	CELL_D	0.0
PRECHARATTANA	LASSO	CELL_A1	0.0
PRECHARATTANA	LASSO	CELL_A2	0.0
PRECHARATTANA	TREE	CELL_H	2.7e-05
PRECHARATTANA	TREE	CELL_D	2.2e-05
PRECHARATTANA	TREE	CELL_A1	2.7e-05
PRECHARATTANA	TREE	CELL_A2	0.0
PRECHARATTANA	SVR	CELL_H	2.1e-05
PRECHARATTANA	SVR	CELL_D	1e-05
PRECHARATTANA	SVR	CELL_A1	3.6e-05
PRECHARATTANA	SVR	CELL_A2	0.0
GONZALEZ	GAUSSIAN PROCESS	CELL_H	0.00439
GONZALEZ	GAUSSIAN PROCESS	CELL_D	0.0

Table 2: Table of RMSE ranges for different regressors and CA simulation models.

Model	Regressor	State	RMSE_RANGE
GONZALEZ	GAUSSIAN PROCESS	CELL_A1	0.000763
GONZALEZ	GAUSSIAN PROCESS	CELL_A2	0.0
GONZALEZ	LASSO	CELL_H	0.011753
GONZALEZ	LASSO	CELL_D	0.000361
GONZALEZ	LASSO	CELL_A1	0.006386
GONZALEZ	LASSO	CELL_A2	0.00052
GONZALEZ	TREE	CELL_H	0.025423
GONZALEZ	TREE	CELL_D	0.006107
GONZALEZ	TREE	CELL_A1	0.014754
GONZALEZ	TREE	CELL_A2	0.007756
GONZALEZ	SVR	CELL_H	0.004509
GONZALEZ	SVR	CELL_D	0.000107
GONZALEZ	SVR	CELL_A1	0.001531
GONZALEZ	SVR	CELL_A2	0.000418
MOONCHAI	GAUSSIAN PROCESS	CELL_H	0.000511
MOONCHAI	GAUSSIAN PROCESS	CELL_D	0.0
MOONCHAI	GAUSSIAN PROCESS	CELL_A1	0.0
MOONCHAI	GAUSSIAN PROCESS	CELL_A2	0.000683
MOONCHAI	LASSO	CELL_H	2.7e-05
MOONCHAI	LASSO	CELL_D	1e-06
MOONCHAI	LASSO	CELL_A1	4e-06
MOONCHAI	LASSO	CELL_A2	2.6e-05
MOONCHAI	TREE	CELL_H	2.3e-05
MOONCHAI	TREE	CELL_D	2.3e-05
MOONCHAI	TREE	CELL_A1	3.2e-05
MOONCHAI	TREE	CELL_A2	3.3e-05
MOONCHAI	SVR	CELL_H	0.00022
MOONCHAI	SVR	CELL_D	5.4e-05
MOONCHAI	SVR	CELL_A1	0.0
MOONCHAI	SVR	CELL_A2	0.000323

5.2 Implications of Results

Based on the results there is evidence that the predictions are viable to a degree, but there are some concerns. The RMSE values for each plot vary only slightly throughout the timesteps and for each regressor. The range of the average RMSE values can be observed in Table 2. This raises questions, as the inclusion of more data should correlate with a decrease in RMSE, which is often not observed. The stable RMSE graphs could be caused by the particular method by which these CA models are updated and could indicate that image features are not as powerful as the authors hoped. These problems are explored in the limitations section below. Besides these potential problems with the results, there are a few RMSE graphs that are very promising. RMSE for Moonchai and Precharattana were 0.0008 and 0.0006 for the state vector machine. González RMSE struggled in both the SV and Tree regressors, sitting at around 15%. Moonchai and Precharattana routinely have the lowest RMSE regardless of regressor used, while González tends to have the highest. Despite generally low RMSE, there is little dependence on the RMSE when compared with $t\_$-Bound. There were some spikes, but the trend for each RMSE generally stays the same. This was unexpected and suggests that the features used were not the most informative. This implies that a better set of features would benefit the model. Some class labels were consistently low and their validity is questionable at best. The only regressor-CA model combination that achieved expected results was the Rana model with the Tree regressor. This suggests that the resulting ability of a regressor to predict the end state of the model may be entirely dependent on which model is being analyzed; essentially, a new model might perform even better than the given models in this research, but it could also perform worse, especially if the model has a divergent end state (potentially a model with two end states: one where a treatment that is able to contain all HIV infection, and another where the HIV completes its progression to AIDS).

Additionally, the factorial analysis revealed that blob features detected by both libraries are particularly effective, especially in the context of the Rana model, where they contributed to around a third of the RMSE score. For the González model, many of the cell classes did not have a dominating feature, suggesting that the image features were either equally unsuccessful or equally contributed to the model's success. Based on the RMSE graphs and the limitations discussed in the next section, it is more likely that the features were rather unsuccessful at predicting the end state of the model. This conclusion also applies to the Moonchai model, which did not have a driving feature for any of the target classes, and the Precharattana model, which only had a minor driving factor of corner features for the dead cells. The dos Santos model seemed to use the most number of feature effectively, utilizing a combination of various blob and contour features. Rarely though, did it utilize the corner features. After looking at all five models and all four target classes, the blob features and contour features seem more effective than the corner features. Within blob and contour features, it appears that focusing on the larger structure of the model with features that have higher areas and perimeters seemed to have a greater effect on the RMSE than small features. This is understandable, as for many models, smaller blobs can be detected in large numbers given the almost randomized appearance of the colors of the cells. Thus, smaller blobs would have much less significance than the larger structure of a blob or contour with an area of 100 or more.

The introduction of treatments at different stages revealed an interesting phenomenon. In the Rana model, the RMSE values were consistently lower for all target classes than the original Rana model, by a magnitude of almost 10 times a difference. This suggests that the model is much more consistent when introducing the treatment at earlier points in time, possibly hinting at the treatment's effectiveness. However, as described earlier, the consistent RMSE scores are still troubling. In the González model, more concerns were raised. Despite this CA model originally introducing treatment at 300 weeks, the RMSE scores for the treatment at t=2 and t=4 weeks did not change. They both stuck in ranges of 10% - 20% RMSE scores. This is concerning, as the authors' previous hypothesis for the comparatively large RMSE scores for the González model were that the dynamics of the model when the image features were collected differed from those at the end. This potentially highlights an additional problem with performing image feature regression on the González model: the image features might be detected within the structure of a model image, but they do not appear to be a good method to determine the end state of the model with accuracy. This is likely due to the model's complexity once more. However, though the new treatment values did not change the RMSE scores themselves, they changed the tendencies of the RMSE, especially for treatment introduced at t=4. Of all of the CA models viewed, this is the only model to see consistent declines in RMSE with more inforamtion. Though this model still maintains the highest RMSE scores, it might be indicative of a different phenomenon: that potentially, the Gonzĺez model is the only CA model for which image features actually correlate with regression, perhaps for its ruleset, and that this regressability can be improved with a more correlated feature set. This is by far the most promising result simply because it is the only set of RMSE graphs that properly correlate with the increase of information, which should be the case for all of these CA model-regressor combinations should be doing, though it is not.

5.3 Limitations

Despite measures being taken to improve upon the original simulation process, there were a few limitations with this proposed process. First, there are additional features that may not have been explored. In this paper most features were focused on finding blobs and their characteristics. Some features relied on corners or edges, but these were explored at a lower-level of complexity. Of the features that were collected, some of the complex information about these features were not investigated, such as the orientation of corners or feature clustering. Additionally, the greyscale method we used to approach this problem is potentially a misguided approach. The images could have been treated as boolean masks. In this approach, each class of cell (Healthy, Dead, Infected A1, A2) would be treated as an entirely discrete and unrelated class to the others, and multiple grids of boolean values would be formed. For example, a boolean mask for the healthy cells would include `True` if a particular cell were healthy, and `False` otherwise, making no discrimination between non-healthy cells. This differs from treating images as greyscale because of the relationship that two similar colors or two pixels in close proximity have within a greyscale image. In a series of boolean masks, such a relationship does not exist. It is arguable that, given the nature of these CA models, the color relationship between different classes of cells is not equivalent to that of an actual greyscale image, and that boolean masking is a better approach.

Second, another limitation comes in the form of treatment initiation. Specifically, for the González model the treatment process did not start until timestep 300. Features were only extracted for timesteps up to 200. At first, this model had as large as 15% error after regression, markedly higher than the other models, since the González model did not have a chance to deploy its treatment and garner independent and accurate results. Predicting the end state based on states where treatment had not taken effect simply is not accurate. The authors then decided to ignore the normal treatment procedure within not only the González model, but all of the models we used. Previous research shows that treatment for patients typically happens within a month after diagnosis, while patients possessing other conditions may initiate antiretroviral therapy after only two weeks [1]. As noted in footnote 2, the authors chose to initiate treatment on the models which included it at 2, 4, and 8 weeks given these previous conclusions. Choosing this fixed time to start treatment on these models, however, imposes yet another limitation. The initiation times of treatment can potentially be affected by how well a patient feels. Patients that have CD4+ counts of above 500 cells per microliter tend to refuse treatment because they feel well [5]. A promising fix for this would be to start the treatment once the number of healthy cells falls behind an optimal threshold.

This method does suggest an improvement over the initial simulations, but is still not ideal. As previously mentioned, there are combinations of features that have not been explored in this procedure. Pooling together a more robust set of diverse features including those outlined above along with edge and corner-focused features could improve results. Using a larger span of values to test the hyperparameters may indicate that some of these parameters are viable to include in the regression. During model validation, implementing bins could allow for use of stratified k-means to improve the model validation scores.

Third, some of models, despite optimization, struggled to scale to larger sizes when attempting to add the various treatment procedures at 2, 4, and 8 weeks. In particular, the $1200x1200$ González model struggled to implement treatment at 8 weeks. To understand why this model did not finish where all others had, a run time versus model timestep graph for each type of feature was calculated, and can be seen in Figure 17 . The treatments for t=2 have little variation in the amount of time they take to calculate, which would indicate that the model is relatively stable in its state. This is likely because the HIV infection has not had enough time to fully infect the body, and therefore the dynamics of the model are relatively stagnant compared to later times such as t=4 and t=8. At these two later times, the model is more "active" as more cells are involved in the simulation. Thus, more complex structures appear and the image feature collection models must work harder to address, analyze, and identify image features. For González in particular, factorial analysis revealed that it relies more heavily on contours than other features to reach its regression conclusions. Thus, given the substantial spikes in run time that contours with a treatment introduced at t=8 take on the González model, it would then follow that the larger version of this model would take longer than other models to perform. Essentially, each CA model has its own dynamics which lead certain image features to drive regression more than others. Certain models might struggle to scale up if given a rather complex ruleset and high variance within the model at a given timestep, which would cause the most important features or feature sets to become a bottleneck.

Fourth, another limitation could come in the form of how the CA models are updated. The synchronous method of updating the CA model could be leading to perfectly regular patterns. As seen in previous research

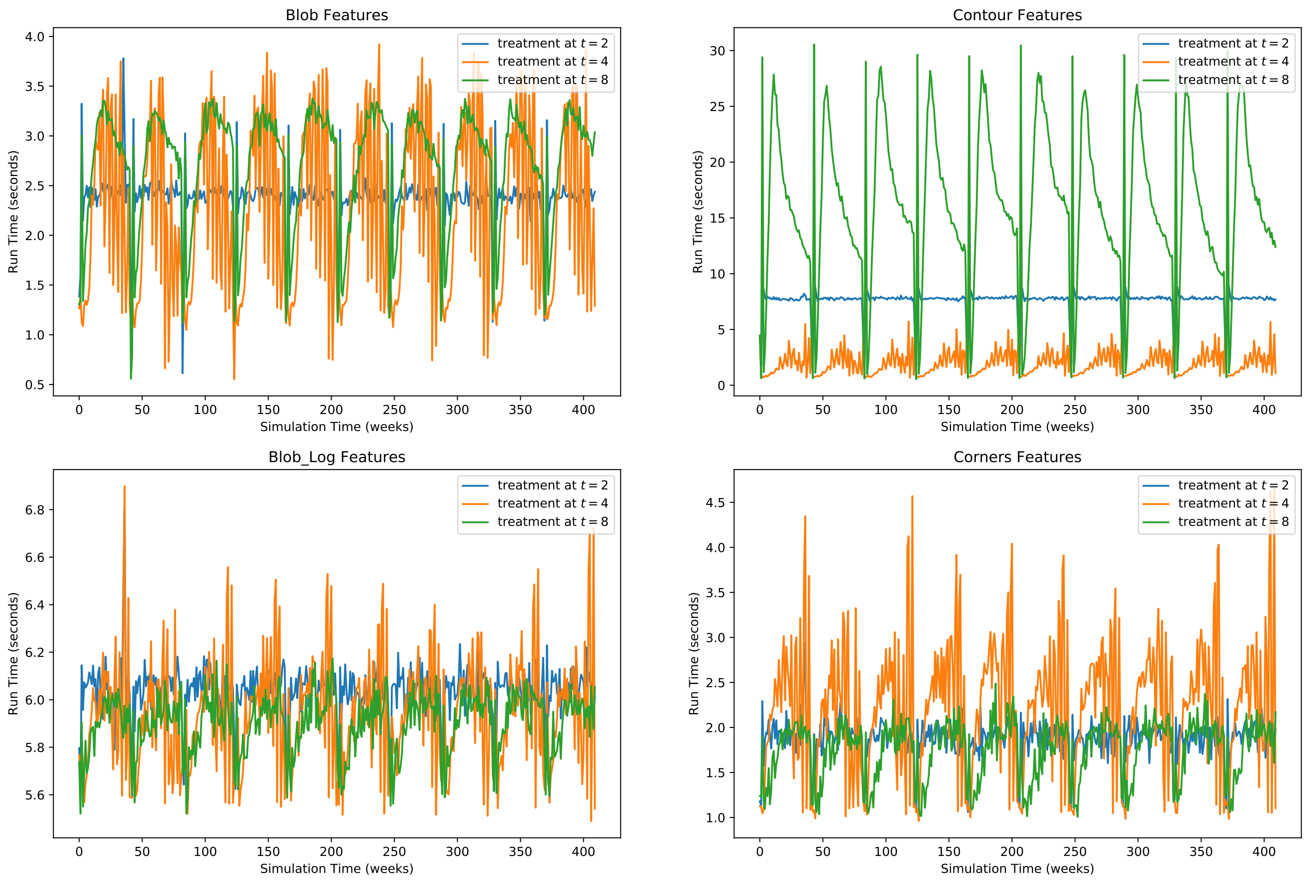


Figure 17: Run time vs model timestep graphs for the González 1200x1200 CA model. This shows why the González model in particular did not meet the specified time requirements that all other models met with ease. Special notice must be given to the variance of each model at a given timestep and complexity of the ruleset as the models are scaled up to larger sizes and more complex rulesets.

by Schönfisch and de Roos, the method by which a CA model is updated can drastically impact the outcomes of the a simulation model[27], and can lead to contradictory conclusions as seen in the case of Huberman and Glance's prisoner dilemma model of 1990[13]. In Huberman and Glance's model, a coexistence between defecting and compliant prisoners continued indefinitely. Schönfisch and de Roos found that, with asynchronous updating of Huberman and Glance's model lead to an entirely different conclusion: a completely defecting population of prisoners. Thus, these regular patterns might be artifacts of the synchronous updating known as *anisotropic fronts*[26]. A remedy to anisotropic fronts is to update the grid asynchronously in a randomized fashion [19].

Fifth and finally, these stable RMSE rates could be indicative of the overall utility, or lack thereof, of image features in predicting model end states. To compare, the authors performed similarly structured regression tasks utilizing only the cell counts of each class of cell at each given timestep in place of image features. The resulting data (Found in Figures 12 ,19 , 20 , and 21) indicates that image features may not be as informative as they were thought to be initially. In all cases, there was only a marginal shift (if any) in the trend and values of the average RMSE score. There are some additional avenues to explore before definitively deciding that these image features are not suitable in predicting end-states. Primarily, it would be revealing to attempt feature extraction on a set of non-convergent models. As it stands now, however, image features had little to no bearing on the outcome of the RMSE.

6 Conclusion

We investigated the possibility of predicting the prevalence of HIV 10 years after initial infections, based on image features extracted from simulated snapshots of the host's lymphatic tissue at various stages of the infection

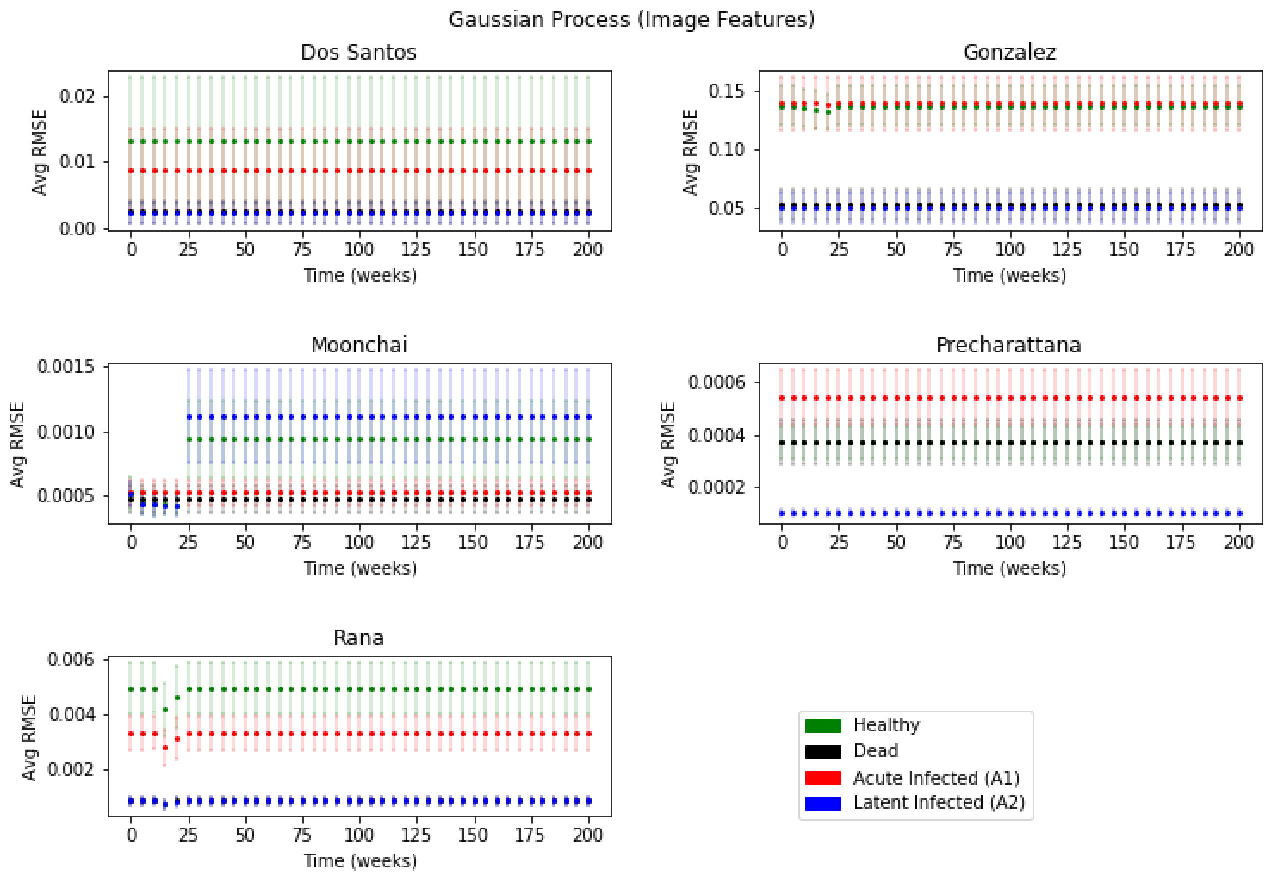


Figure 18: Average RMSE of the gaussian process regressor for data from each CA model after a 10-fold cross validation. The y axis plots the percentages in decimal, i.e. 0.01 means 1% (of total cells count). The x axis plots the maximum t up to which image features were considered by the regressor.

using different CA models. Given the comparison drawn between using only cell counts of the various cell types and our attempts to use image features, little to no difference was found. As a result, we cannot conclude that the image features were of any particular utility. There are additional avenues left to explore before this approach can be rejected, such as the usage of boolean masks, eliminating anistropic fronts, and engaging with new image features. Of all the models tests, the only affirmative connection drawn between image features and lowered RMSE scores was in the a specific treatment version of the González model. This provides some hope for this approach, as we can additionally conclude that the scope of utility for these image feature regressors is highly dependent on the CA model and its specific dynamics and ruleset. Therefore, future work should comprehensively examine image features that are more closely tied to the underlying biological processes or generated with a smaller cutoff, as well as improve how the CA models update in the hopes of finding a more widely applicable approach to using image features to predict the end states of the model.

Libraries Used

The Python libraries used in this project are listed below and cited in references.

- Scikit-Learn [23].
- Scikit-Image [28].
- OpenCV [2].
- Matplotlib [14].
- Numba [16].
- Numpy [17].

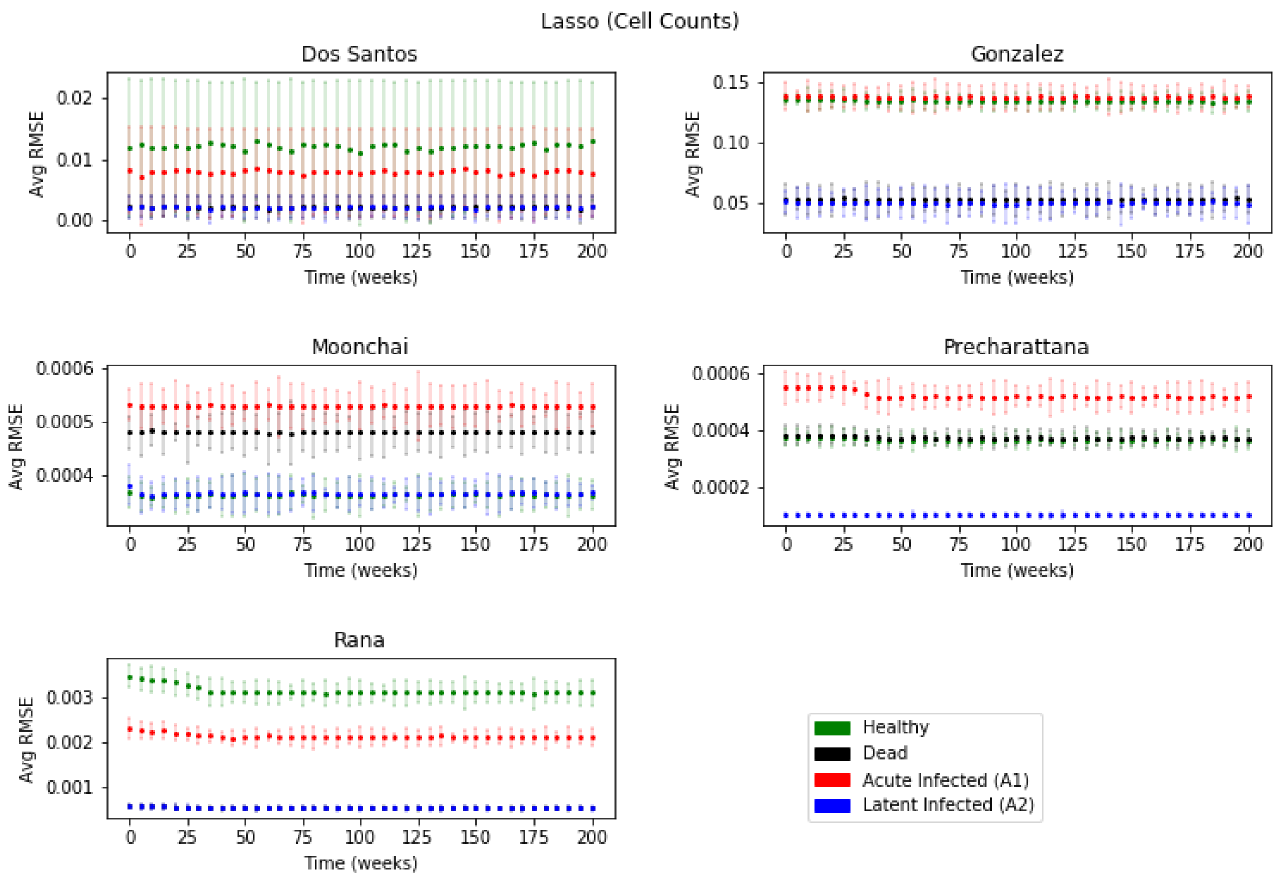


Figure 19: Average RMSE of the lasso regressor for data from each CA model after a 10-fold cross validation. The y axis plots the percentages in decimal, i.e. 0.01 means 1% (of total cells count). The x axis plots the maximum t up to which cell counts were considered by the regressor.

References

[1] François-Xavier Blanc, Thim Sok, Didier Laureillard, Laurence Borand, Claire Rekacewicz, Eric Nerrienet, Yoann Madec, Olivier Marcy, Sarin Chan, Narom Prak, and et al. Earlier versus later start of antiretroviral therapy in hiv-infected adults with tuberculosis. *The New England Journal of Medicine*, 365(16):1471–1481, Oct 2011.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] CDC. HIV/AIDS. `https://www.who.int/news-room/fact-sheets/detail/hiv-aids`.

[4] CDC. CDC - About HIV/AIDS. `https://www.cdc.gov/hiv/basics/whatishiv.html`, Dec 2019.

[5] Jan Fehr, Dunja Nicca, Jean-Christophe Goffard, David Haerry, Michael Schlag, Vasileios Papastamopoulos, Andy Hoepelman, Athanasius Skoutelis, Ruth Diazaraque, and Bruno Ledergerber. Reasons for not starting antiretroviral therapy in hiv-1-infected individuals: a changing landscape. *Infection*, 44(4):521–529, Mar 2016.

[6] Andrew Fisher, Bhisma Adhikari, Chao Zhai, and Joshua Morgan. Predicting the resource needs and outcomes of computationally intensive biological simulations. 2019.

[7] Philippe J. Giabbanelli, Jared A. Kohrt, and Joshua A. Devita. Optimizating Discrete Simulations of the Spread of HIV-1 to Handle Billions of Cells on a Workstation. Unpublished.

[8] Ramon Gonzalez, Sergio Coutinho, Rita Zorzenon dos Santos, and Pedro Figueirêdo. Dynamics of the hiv infection under antiretroviral therapy: A cellular automata approach. *Phys-*

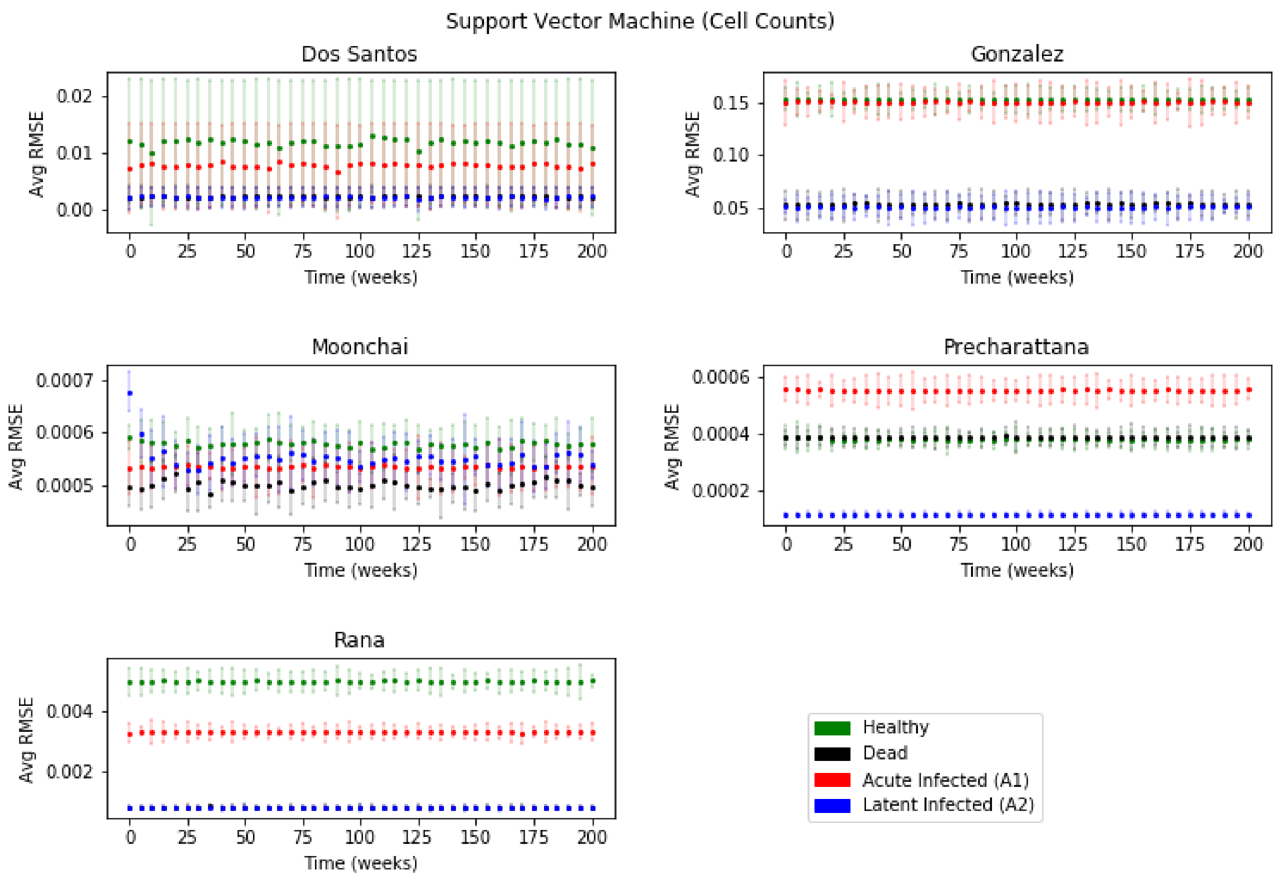


Figure 20: Average RMSE of the support vector machine regressor for data from each CA model after a 10-fold cross validation. The y axis plots the percentages in decimal, i.e. 0.01 means 1% (of total cells count). The x axis plots the maximum t up to which cell counts were considered by the regressor.

ica A Statistical Mechanics and its Applications, 392:4701–4716, 10 2013.

[9] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. O'reilly Media, 2017.

[10] Jochen Görtler, Rebecca Kehlbeck, and Oliver Deussen. A visual exploration of gaussian processes. *Distill*, 4(4):e17, Apr 2019.

[11] Mark D Halling-Brown, David S Moss, and Adrian J Shepherd. Towards a lightweight generic computational grid framework for biological research. *BMC Bioinformatics*, 9:407, Oct 2008.

[12] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification, 2003.

[13] B. A. Huberman and N. S. Glance. Evolutionary games and computer simulations. 90:7716–7718, Aug 1993.

[14] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[15] J. Kougias, Ch. F.and Schulte. Simulating the immune response to the HIV-1 virus with cellular automata. *Journal of Statistical Physics*, 60(1):263–273, Jul 1990.

[16] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, New York, NY, USA, 2015. Association for Computing Machinery.

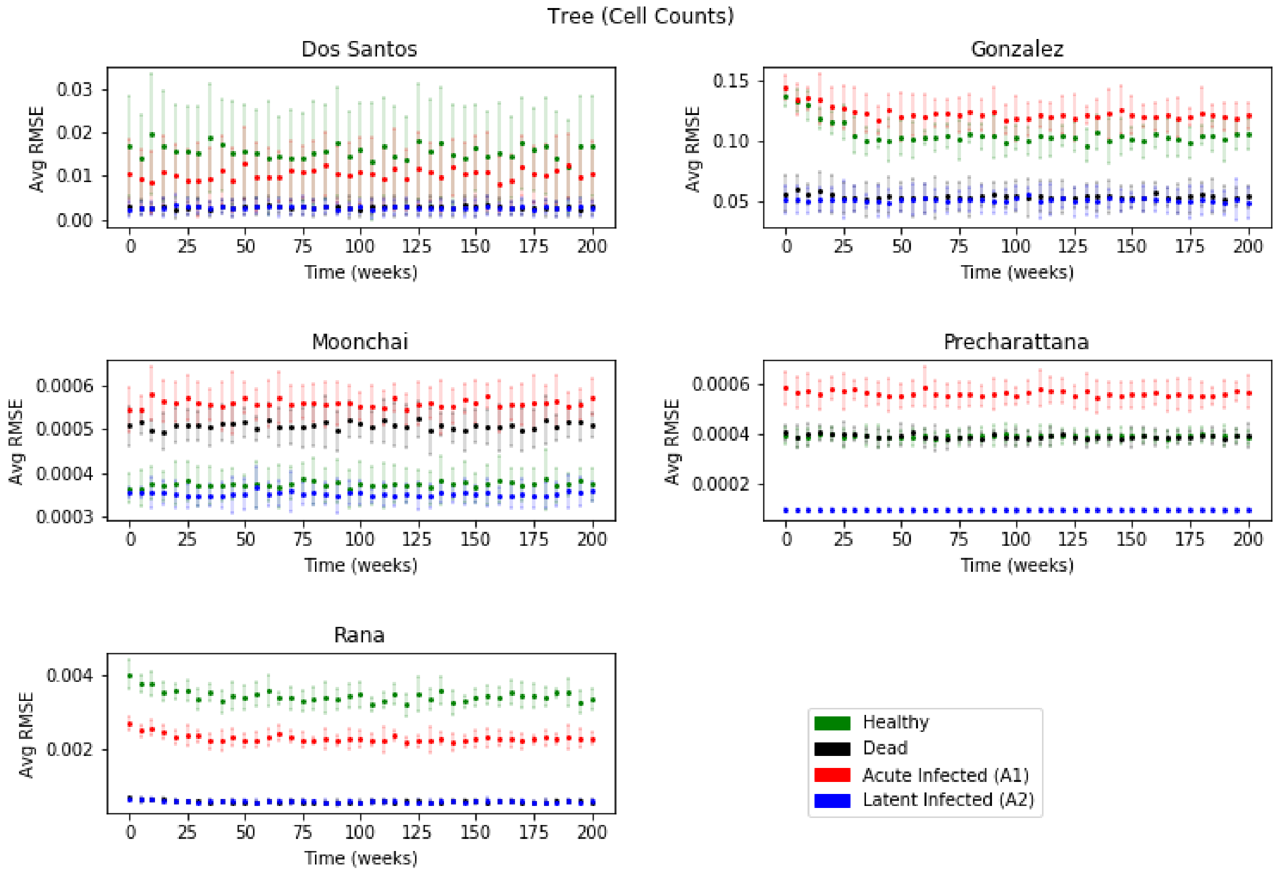


Figure 21: Average RMSE of the decision tree regressor for data from each CA model after a 10-fold cross validation. The y axis plots the percentages in decimal, i.e. 0.01 means 1% (of total cells count). The x axis plots the maximum t up to which cell counts were considered by the regressor.

[17] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, New York, NY, USA, 2015. Association for Computing Machinery.

[18] Brendan Maughan-Brown, Philip Smith, Caroline Kuo, Abigail Harrison, Mark N. Lurie, Linda-Gail Bekker, and Omar Galárraga. Readiness for antiretroviral therapy: Implications for linking hiv-infected individuals to care and treatment. *AIDS and Behavior*, 22(3):691–700, Jul 2017.

[19] John Metzcar, Yafei Wang, Randy Heiland, and Paul Macklin. A review of cell-based computational modeling in cancer biology. *JCO Clinical Cancer Informatics*, (3):1–13, Feb 2019.

[20] Sompop Moonchai and Yongwimon Lenbury. Investigating combined drug and plasma apheresis therapy of hiv infection by double compartment cellular automata simulation. *International Journal of Computer Theory and Engineering*, 8:190–197, 06 2016.

[21] NIH. Acquired immunodeficiency syndrome (AIDS). `https://aidsinfo.nih.gov/understanding-hiv-aids/glossary/3/acquired-immunodeficiency-syndrome`, 2019.

[22] U.S. Department of Health and Human Services. HIV treatment overview. `https://www.hiv.gov/hiv-basics/staying-in-hiv-care/hiv-treatment/hiv-treatment-overview`, Apr 2019.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, B. Michel, V.and Thirion, O. Grisel, M. Blondel, R. Prettenhofer, P.and Weiss, V. Dubourg, J. Vanderplas, D. Passos, A. andCournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learn-

ing in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] Monamorn Precharattana, Wannapong Triampo, C. Modchang, and Yongwimon Lenbury. Investigation of spatial pattern formation involving CD4+ T cells in HIV/AIDS dynamics by a stochastic cellular automata model. *International Journal of Mathematics and Computers in Simulation*, 4:135–143, 01 2010.

[25] Ela Rana, Philippe J. Giabbanelli, Naga H. Balabhadrapathruni, Xiaoyu Li, and Vijay K. Mago. Exploring the relationship between adherence to treatment and viral load through a new discrete simulation model of hiv infectivity. In *Proceedings of the 3rd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM PADS '15, page 145–156, New York, NY, USA, 2015. Association for Computing Machinery.

[26] Birgitt Schönfisch. Propagation of fronts in cellular automata. *Physica D: Nonlinear Phenomena*, 80(4):433–450, Feb 1995.

[27] Birgitt Schönfisch and André de Roos. Synchronous and asynchronous updating in cellular automata. *Biosystems*, 51(3):123–143, Sep 1999.

[28] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[29] Rita Maria Zorzenon dos Santos and Sérgio Coutinho. Dynamics of hiv infection: A cellular automata approach. *Phys. Rev. Lett.*, 87:168102, Sep 2001.