WAKE-SLEEP BAYESIAN PROGRAM SYNTHESIS APPLICATIONS IN BIOINFORMATICS

BY

MADHAV KHIRWAR

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science in Computer Engineering
in the Grainger College of the
University of Illinois Urbana-Champaign, Fall 2021

Urbana, Illinois

Adviser:

Professor Rohit Bhargava

ABSTRACT

Program synthesis is the process of learning mappings between sets of inputs and outputs in a way that generalizes to new inputs. Contrary to deep learning in the gradient descent sense, the goal of program induction isn't to 'converge' to a correct solution by performing gradient descent on millions of parameters - rather it is to generate and search for discrete programs that are expressed as combinations of a library of known 'concepts' that will solve the given problem.

The goal of my thesis is to explore the portability of program induction onto the bioinformatics domain – specifically the problem of tumor grade prediction. Programs enumerated to predict tumor grade from a data set of colon cancer were up to 76% accurate when the library of primitives was limited to arithmetic, exponential and logarithmic operations. Further work will involve building in models for solving differential equations (another success was to induce Dreamcoder to discover the forward Euler method for solving PDEs), as well as building conceptual representations of n-dimensional spatial data such as images.

Subject Keywords: Program synthesis; Bayesian wake-sleep Learning; Bioinformatics; Cancer

To my family, for their love and support.

ACKNOWLEDGMENTS

I would like to extend a special thanks to Professor Rohit Bhargava for his guidance throughout the year and for encouraging me to explore novel areas of AI research that I likely would have never otherwise come across as a computer engineer. A thanks goes out to Kianoush Falahkheirkhah as well for providing me with a solid foundation for this project as well as being willing to guide me at all times and brainstorm solutions to every obstacle that came along during this work.

Lastly, I would like to thank my parents, my sister, and my friends for their steadfast support and for empowering me to seek knowledge.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 LITERATURE REVIEW 2
2.1 Deep Learning in Oncology 2
2.1.1 Successes of Deep Learning Approaches 2
2.1.2 Drawbacks of Deep Learning 3
2.2 Program Synthesis 4
2.2.1 Definition 4
2.2.2 Previous Work 5
2.3 Wake-Sleep Program Synthesis and Dreamcoder 5
2.3.1 Description 5
2.3.2 Motivation for Use in Bioinformatics 8

CHAPTER 3 EXPERIMENTAL METHODOLOGY AND RESULTS 9
3.1 Mathematical Modelling of Bio-chemical Systems 9
3.1.1 Differential Equations and Fick's Second Law 9
3.1.2 Discovering Mathematical Models for known Bio-chemical systems 11
3.2 Discovering Models of Cancer 14
3.2.1 Discovering Known Models of Cancer 14
3.2.2 Synthesizing Unknown Hypotheses to Predict Tumor Tolerance 16

CHAPTER 4 CONCLUSION 20

REFERENCES 21

CHAPTER 1

INTRODUCTION

State-of-the-art artificially intelligent (AI) systems that have been successful at accurately recognizing cancer and predicting its progression have relied entirely on architectures of deep neural networks (DNNs) as noted by Tran et al [1]. However, as noted in the same paper as well as by Rudin [2], a major challenge with implementing deep learning (DL) into clinical practices is the non-interpretable nature of these models' results. While workarounds to this problem have been proposed to quantify the influence each feature has on the output such as Grad-CAM[3] and PG-CAM [4], inferring generalizable logical rules from their results (even in visual form such as saliency maps) is left up to domain experts' own abilities to postulate hypotheses that explain the saliency of sets of features. DL-based AI systems are not inherently interpretable and need secondary 'post-hoc' models, which at best can attempt to explain the output of a DNN with a probability distribution for the saliency over the dimensions of the input.

As Rudin [2] notes, an alternative framework that outputs the fundamental connections between sets of features as opposed to the aforementioned saliency maps would have to be inherently interpretable. Program synthesis is one such framework because it is interpretable by construction. Since programs are represented as logically connected concepts that are bound by rules of the domain-specific language of the task, successful programs can be readily interpreted by domain experts. Such information is useful for domain experts in bioinformatics, where knowing exactly how an AI system, for instance, performed well at predicting tumor progression can allow for potential unforeseen relationships between the tumor and its surrounding microenvironment to come to light. Thus, the aim of this thesis is to explore the applicability of program synthesis to bioinformatics – specifically the problem of predicting tumor tolerance given a set of primitive features.

CHAPTER 2

LITERATURE REVIEW

2.1 Deep Learning in Oncology

Deep Learning leverages multiple layers of (deep) neural networks to extract non-linear interpolated features from high-dimensional data [5]. While covering all emergent DL methods that are applied to oncology is outside the scope of this thesis, at a high level, layers of artificial neurons within DNNs are connected sequentially (sometimes with recurrent or skipped connections [6]). During supervised training, these networks forward-propagate an input signal along with a ground truth signal, and upon quantifying the incorrectness or 'loss' of the output in comparison to the ground truth, then back-propagate the gradient of the loss signal through the network whilst updating the scalar weights and biases of each neuron, thereby guaranteeing convergence to a locally optimal solution via gradient descent.

2.1.1 Successes of Deep Learning Approaches

In oncology, Deep Learning systems that frame cancer prediction as a tissue classification problem – be that in the form of sample-wise classification or pixel-wise classification (semantic segmentation) – have had the highest success in recent years [1]. State-of-the-art DNN architectures for cancer prediction such as U-Net [7] have demonstrated up to 95% accuracy at pixel-level classification [8] and have been applied to domains such as predicting tumor gradation for prostrate [9, 10, 11], breast [12] and colon [13] cancers with up to 85% accuracy, predicting cell composition of tissues from gene expression information [14] with up to 90% accuracy, classifying molecular subtypes of lung cancer [15] and modelling prognosis and survivability [16] with up to 88% accuracy amongst others. Researchers at Seoul National

University Hospital and College of Medicine have proposed a novel system called Deep Learning-based Automatic Detection and demonstrated that it is able to analyse chest radiographs to detect abnormal growths and upon being compared to 18 physicians' detection abilities on the same set of images, outperforms 17 of the 18 physicians [17].

2.1.2 Drawbacks of Deep Learning

Despite these successes, DL models have not seen wide adoption into clinical environments because of their lack of explainability. As noted by Gulum et al [18] as well as DARPA [19], there is an emergent inverse relationship between the accuracy of models and their explainability.

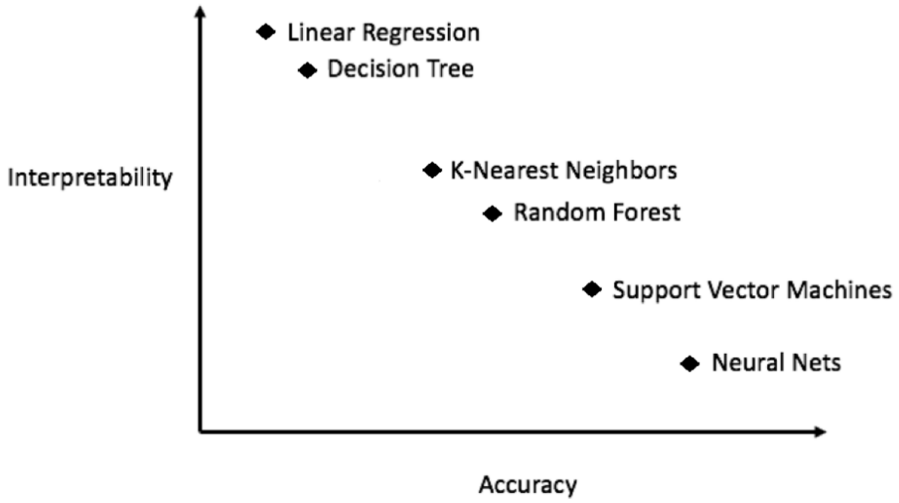

Figure 2.1: Depiction of Interpretability-Accuracy trade-off [18].

This is a drawback because explainability of models is essential in practice for a variety of reasons. There are legal and ethical concerns that arise from medical advice or prognoses being output by a 'black-box' system, such as the General Data Protection Regulation of the European Union that require the use of patient data to be readily explainable [20]. Explainability is also important to build a degree of trust between patients, doctors and clinicians as noted by Holzinger et al [21].

While there have been attempts to explain the decisions made by DL-based systems, these have been limited to highlighting interpolative relationships

between learned features as in GradCAM and PG-CAM [3, 4]. Even when these relationships are represented as saliency maps, inferring causal relationships between features is left up to domain experts – often leading to incorrect and misleading results [2]. This is because the secondary 'post-hoc' model that attempts to explain the outputs of a target 'black-box' model by highlighted impactful features does so by finding areas of the input signal that under deformation have the highest statistical correlation with deformation of the ground-truth signal. This approach can inherently only capture correlative relationships (at best) and cannot capture causal relationships.

2.2 Program Synthesis

Treating learning a new task as searching for a program that solves it rather than the purely statistical approach that neural networks take is closer to how AI was initially defined [22]. Purely statistical approaches lack some of the advantages of program synthesis – one being the ability to generalize strongly by extrapolating concepts across tasks instead of interpolating between features. This leads to program synthesis being much more sample-efficient as well. Another advantage is that this approach is interpretable by construction – there is no need for saliency maps or gradient-based localization because the program subsumes the domain specific language of the task which is specified by the human expert and the knowledge discovered by the inductive programming system can be reused across tasks. Lastly, any Turing-complete language represents solutions to all solvable computational tasks.

2.2.1 Definition

Program synthesis can be seen as search in a class of programs, referred to as the hypothesis space H. Programs are characterized by their atomic building blocks, which are hard-coded primitive operations (also called background knowledge or B), and search over H for a successful program is search over combinations (recursive or not) of these primitives that successfully solve all the given task examples, denoted by E. Formally, the core inductive programming goal is to find programs $p \in H$ such that $B \wedge H \models E$.

2.2.2 Previous Work

The initial stages of program synthesis were rule-based and employed classical search techniques where the task is well specified with assertions and boundary conditions on both input and output, and the program is derived from a formalized grammar [23]. Since the process of specifying these boundary conditions is often more resource intensive than simply writing the program itself, modern approaches tend to focus on paradigms of program synthesis where a system is provided with a set of examples with paired of input/output data and a set of primitive (background) functions. Emphasis in research in the modern era has been in developing efficient pruning methods for the search over hypothesis space. Thus, approaches such as DeepCoder [24] have implemented neural networks to learn mappings from input-output pairs to attributes using latent vector representation of these pairs and used these neural networks to recognize patterns in the training data and prioritize programs that take those patterns into account.

2.3 Wake-Sleep Program Synthesis and Dreamcoder

The combinatorically large nature of the hypothesis space as well as the need for a hand-tuned domain specific language have been major roadblocks in program synthesis proliferating across different domains of tasks, as inducing useful programs based solely on pruning search only helps to a certain extent.

2.3.1 Description

To more efficiently prune the search tree, Dreamcoder builds in the ability to acquire domain expertise by learning a domain-specific language as well as the ability to learn procedural skill – such that the more expertise the system has at a set of tasks, the better 'skilled' it is at searching for solutions to other similar tasks. Procedural skill is acquired similar to the Deepcoder strategy, by training a neural network in the use of a learned domain-specific language during program synthesis.

One of the ways in which this system is novel is that it uses a wake-sleep paradigm to infer both programs as well as a prior probability distribution over programs likely to solve tasks. The prior distribution is represented as a concept library L which in turn is used as the background for neural-network pruned search where the neural network predicts, conditioned on the task, a posterior probability distribution over candidate programs. The search occurs in three phases. The wake phase searches for solutions to a given set of tasks X by evaluating potential programs on their 'intuitive' correctness using a trained recognition model $Q(p|x)$ that ranks candidate programs with a probability of being useful $P(p|x)$ for each task $x \in X$ and attempts to find successful programs p_x such that:

$$P[pL] \propto P[x|p]P[p|L] \tag{2.1}$$

is maximized.

The first sleeping phase is called abstraction, where the goal is to grow the concept library by discovering abstractions that allow solutions to be expressed more compactly than purely in terms of primitives (along the lines of Occam's razor) and prioritize programs that have the shortest lengths (are shallower in the search).To best perform this compression, the description lengths of both the library as well as the programs enumerated during waking must be minimized. The library L is expanded such that:

$$P(L)\Pi_{p \text{ a refactored } p_x} P[x|p]P[p|L] \tag{2.2}$$

is maximized. This is achieved by refactoring commonly occurring branches of program search that are functionally equivalent and assimilating them into new primitives for the library.

The second sleep phase is called dreaming, and it is during this phase that the neural network that is the recognition model for selecting useful programs is trained. The training data for the neural network takes the form of (program, task) pairs and is drawn from two sources: one being the set of programs and tasks seen during the waking (program synthesis) phase and the other being 'fantasies' – programs from the library. The neural network's posterior probability distribution $Q(p|x)$ is trained to be approximately equivalent to $P[p|x, L]$ such that a task x is drawn either from replays

or fantasies.

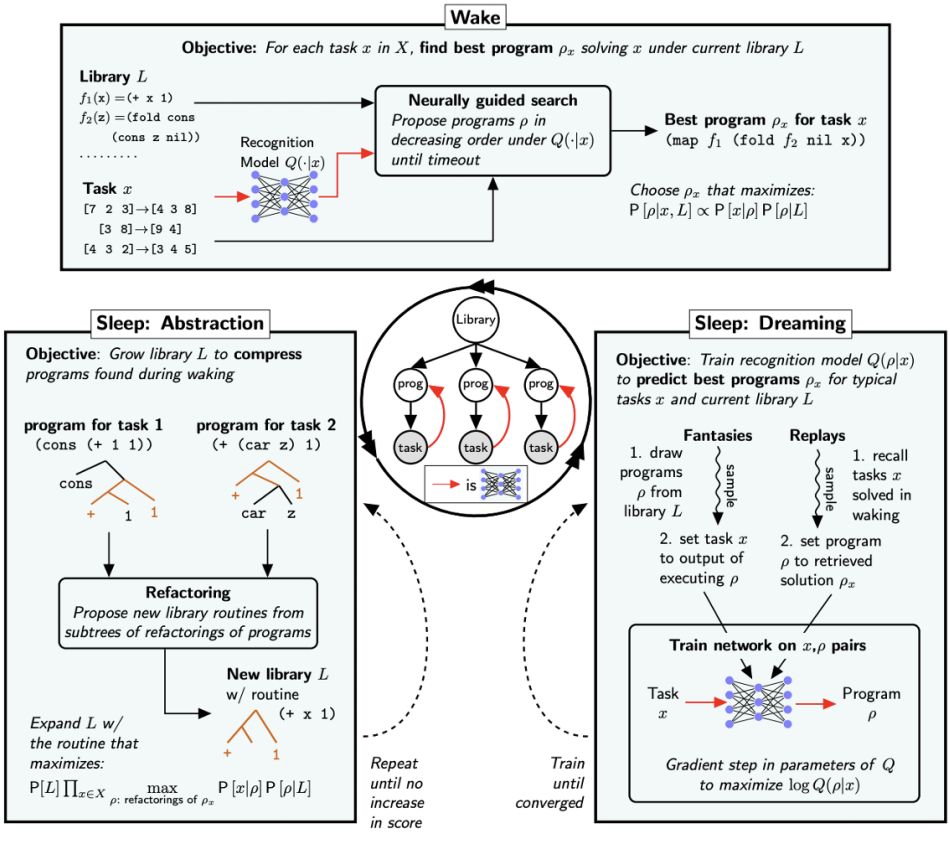

Figure 2.2: A diagram of the wake-sleep algorithm, Ellis et al [25].

High-level pseudocode for the algorithm is as follows:

Algorithm 1 Full DreamCoder algorithm

function DreamCoder(D, X):
Input: Initial library functions D, tasks X
Output: Infinite stream of libraries, recognition models, and beams
Hyperparameters: Batch size B, enumeration timeout T, maximum beam size M
$\theta \leftarrow$uniform distribution
$\mathcal{B}_x \leftarrow \varnothing, \forall x \in X$ ▷ Initialize beams to be empty
while true **do** ▷ Loop over epochs
    shuffle $\leftarrow$ random permutation of X ▷ Randomize minibatches
    while shuffle is not empty **do** ▷ Loop over minibatches
        batch $\leftarrow$ first B elements of shuffle ▷ Next minibatch of tasks
        shuffle $\leftarrow$ shuffle with first B elements removed
        $\forall x \in$ batch: $\mathcal{B}_x \leftarrow \mathcal{B}_x \cup \{\rho \mid \rho \in \text{enumerate}(\text{P}[\cdot|D,\theta], T) \text{ if } \text{P}[x|\rho] > 0\}$ ▷ Wake
        Train $Q(\cdot|\cdot)$ to minimize $\mathcal{L}^{\text{MAP}}$ across all $\{\mathcal{B}_x\}_{x \in X}$ ▷ Dream Sleep
        $\forall x \in$ batch: $\mathcal{B}_x \leftarrow \mathcal{B}_x \cup \{\rho \mid \rho \in \text{enumerate}(Q(\cdot|x), T) \text{ if } \text{P}[x|\rho] > 0\}$ ▷ Wake
        $\forall x \in$ batch: $\mathcal{B}_x \leftarrow$ top M elements of $\mathcal{B}_x$ as measured by $\text{P}[\cdot|x, D, \theta]$ ▷ Keep top M programs
        $D, \theta, \{\mathcal{B}_x\}_{x \in X} \leftarrow \text{ABSTRACTION}(D, \theta, \{\mathcal{B}_x\}_{x \in X})$ ▷ Abstraction Sleep
        yield (D, θ), Q, $\{\mathcal{B}_x\}_{x \in X}$ ▷ Yield the updated library, recognition model, and solutions found to tasks
    end while
end while

Figure 2.3: High level algorithm of Dreamcoder as described by Ellis et al [25]

2.3.2 Motivation for Use in Bioinformatics

One of the main applications of wake-sleep program synthesis described in this thesis is related to cancer. In the past various versions of program synthesis have been applied to discovering rules governing cancer progression (as by Nassif et al [26] and Bevilacqua et al [27]), as well as in areas such as DNA sequencing [28]. However, these approaches have had highly tailored and high-level primitives encoded before program synthesis and refactoring/breaking symmetries between programs in the library is not emphasized, and as a result the breadth of the search explodes exponentially. Using a system that bootstraps program synthesis from a well-refactored and expressive library of concepts and a neural network trained to prioritize useful-seeming programs greatly reduces the breadth of the search, recognizing and refactoring often-used primitives into a new program greatly reduces the depth of the search.

CHAPTER 3

EXPERIMENTAL METHODOLOGY AND RESULTS

This section outlines the application of wake-sleep program synthesis to two sets of tasks in the bioinformatics domain. By demonstrating the ability of and challenges faced by a modified version of Dreamcoder for each task a baseline is established for each of these tasks, this thesis proposes wake-sleep program synthesis as a strong candidate for future work in hypothesis discovery in bioinformatics.

3.1 Mathematical Modelling of Bio-chemical Systems

This subsection describes discovery of known biochemical models. The datasets are constructed with input/output tuples for each equation and in order to simulate the noise inherent in experimentally procured biochemical data, the output of each equation is randomly jittered by 10%. The parameters of the most successful searches for each of these equations are tabulated next to them.

3.1.1 Differential Equations and Fick's Second Law

Fick's Law for diffusion describes how diffusion causes the concentration to change with respect to time. This partial differential equation underpins models in fields such as neurons, biopolymers, diffusion across a membrane amongst others. Fick's second law is:

$$\frac{\partial \varphi}{\partial t} = D\frac{\partial^2 \varphi}{\partial x^2}(x,t) \tag{3.1}$$

Where φ is the concentration function, t is time and x is position. Since the default Dreamcoder system is unable to solve differential equations, this

feature had to be added. The first step was to make Dreamcoder solve an ordinary differential equation, for which the dataset was constructed using the Forward Euler method for approximating ODEs and was set as the target program for the system to discover. Since the ODE was in one dimension, the problem was formulated as a list processing problem where the ODE in question was of the form $\frac{dy}{dx} = f(x)$ and the dataset was constructed with the following formula and initial conditions:

$$y_{n+1} = y_n + f(y_n, x_n)(x_{n+1} - x_n), y_0 \text{ a random constant} \tag{3.2}$$

The following table describes the primitives (background programs) that made up the initial library of the program synthesis, the number of wake/sleep cycles it took for the correct program to be induced for each basic algebraic function:

Table 3.1: Discovery of Euler's method for Ordinary Differential Equations

Primitives	$+, -, \times$, car(Lisp), map(Lisp), index(Lisp)
Wake/Sleep Cycles	1
Datatypes	real number list, real number $\rightarrow$ real number list

Since increasing the number of primitives increases the combinatorial space, program synthesis with more primitives require more wake/sleep cycles to find a successful program. Then, the next step was to discover the forward Euler method for partial differential equations, with Fick's second Law in one spatial dimension as the target equation (3.1). To leverage parallelism, the set of tasks included the forward Euler method discovery task for ordinary differential equations. The dataset was constructed with the following formula and initial conditions:

$$\varphi_i^{n+1} = \varphi_i^n + \alpha \frac{\Delta t}{\Delta x^2}(\varphi_{i+1}^n - 2\varphi_i^n + \varphi_{i-1}^n), \varphi_0^0 \text{ a random constant} \tag{3.3}$$

Table 3.2: Discovery of Euler's method for Partial Differential Equations and Fick's second law

Primitives	$+, -, \times, 1, 0$, Euler's Method for ODEs
Wake/Sleep Cycles	2
Datatypes	real number, real number $\rightarrow$ real number

The following graph shows the progression of search for the forward Euler formula for a Fick's second law. The search for the forward Euler method for solving PDEs was parallelized with the search for the forward Euler's method for solving ODEs, and thus the ODE formula is discovered in the first cycle which leads to the PDE formula being discovered in the following cycle:

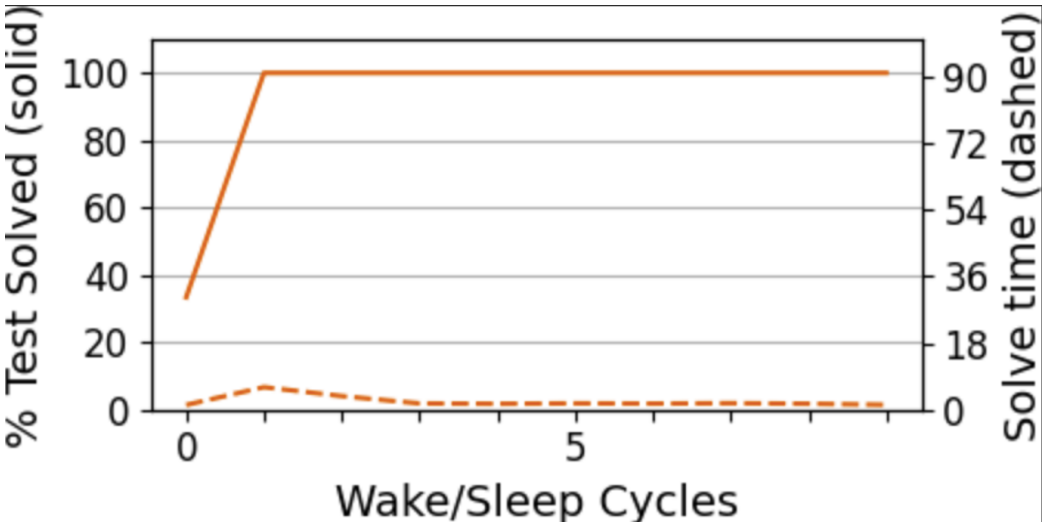

Figure 3.1: Graph showing the progress of the program synthesis with each wake/sleep cycle.

3.1.2 Discovering Mathematical Models for known Biochemical systems

To illustrate the portability of wake-sleep program synthesis as a paradigm for hypothesis discovery in biochemical datasets, this subsection details the program discovery process for known biochemical laws, as presented in [29]. The following subsection describes some equations as well as the process of their discovery:

Equation 1: Model for Red Blood Cell Production:

$$R_{n+1} = R_n(1 - f) + \gamma f R_{n-1} \tag{3.4}$$

Where γ is a reproduction constant, R_n is the number of red blood cells at a given time n, and f the number of cells produced at once.

Table 3.3: Discovery of Model for RBC production

Primitives	$1, 0, +, -, \times$, car(Lisp), map(Lisp), index(Lisp)
Wake/Sleep Cycles	1
Datatypes	real number list, real number $\rightarrow$ real number list

Equation 2: Hardy-Weinberg Equations relating frequency of 2 alleles in a population:

$$\begin{aligned} p^2 + 2pq + q^2 &= 1 \\ p + q &= 1 \end{aligned}$$

Where p and q are the frequencies of genotype AA and aa in a population respectively and pq is the frequency of Aa in that population.

Table 3.4: Discovery of Hardy-Weinberg Equations

Primitives	$+, -, \times, 1, 0$, power, real coefficient
Wake/Sleep Cycles	2
Datatypes	real number $\in [0, 1]$, real number $\in [0, 1]$ $\rightarrow$ real number $\in [0, 1]$

As can be seen in the following figure, in synthesizing a program that arrives at a correct solution to these two equations with constraints, the first program to be discovered is the relationship between p and q, following which the quadratic form equation is discovered as well.

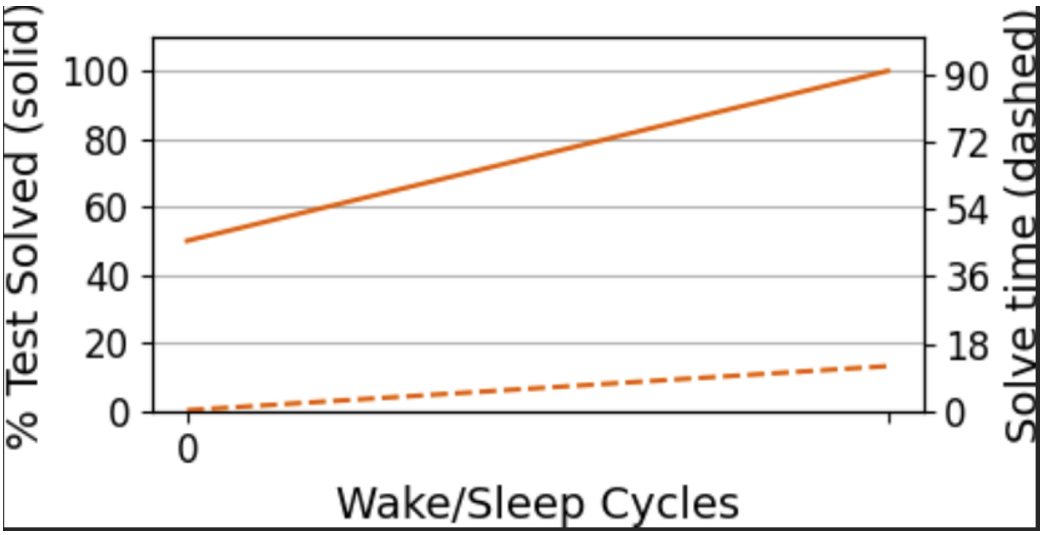

Figure 3.2: This graph shows the progression of search for the Hardy-Weinberg equations that describe an equilibrium.

Equation 3: Two-state model of protein folding kinetics, as described by Zwanzig [30]:

$$\begin{aligned} k_f P_U(t) - k_u P_F &= \frac{dP_N}{dt}(t) \\ P_N + P_U &= 1 \end{aligned}$$

Where P_N is the fraction of the protein in its native state and P_U is the fraction of the protein in its unfolded state, and k_f is the folding rate and k_u is the unfolding rate. Since this is an ordinary differential equation and a modified version of Dreamcoder is able to discover the form of Euler's method for an ODE, parallelizing search between the given equation and 3.2 along with the constraints of P_N and P_U allowed search for a correct solution to share a common library, with the latter equation effectively bootstrapping the former.

Table 3.5: Discovery of Model for protein folding kinetics

Primitives	$+, -, \times$, car(Lisp), map(Lisp), index(Lisp)
Wake/Sleep Cycles	1
Datatypes	real number list, real number $\rightarrow$ real number list

The following graph shows the progression of search for the two-state model of protein kinetics. Between wake/sleep cycles 7 and 8 the model came across a test case that did not fit its hypothesis, and generalized its hypothesis in order to fit all cases:

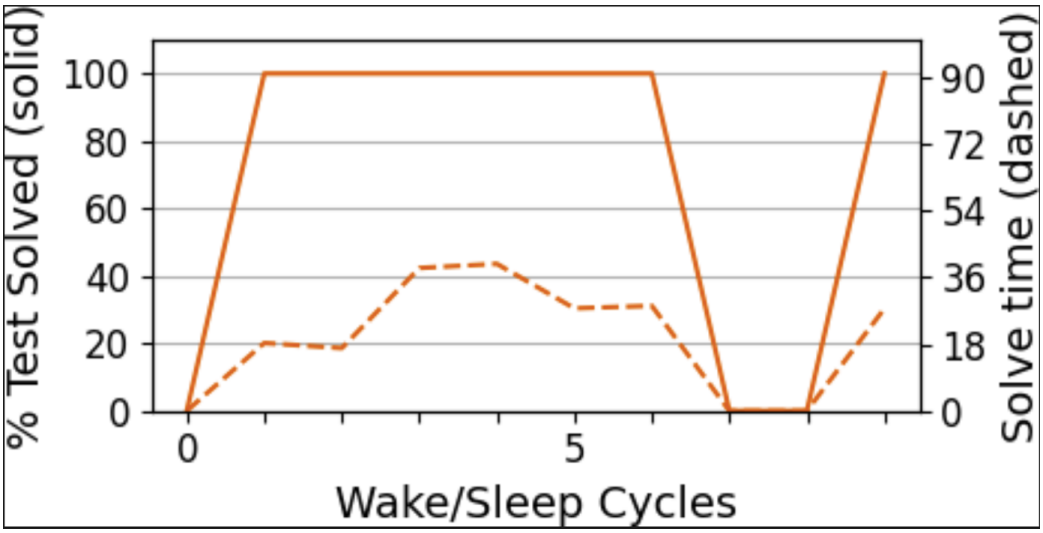

Figure 3.3: This graph shows the progression of search for a mathematical model to describe protein folding kinetics

3.2 Discovering Models of Cancer

This section discusses applications of wake/sleep program synthesis in discovering hypotheses that model cancer. In order to discover hypotheses that can describe tumor progression, the search was augmented by including primitives that are known to be key predictors in cancer progression to maximize the probability that the search would be biased in the direction of programs that are similar to known formula in the domain, because this approach resulted in the discovery of the most accurate hypotheses with 76.5%, 69.6% and 65.6% respectively.

3.2.1 Discovering Known Models of Cancer

Following the results described in section 3.1, the wake/sleep algorithm is applied to the task of discovering known mathematical models that govern

tumor growth as a sanity check as well as to establish a baseline of hypothesis synthesizing ability in oncology. The following tables describe two of these equations and the search process for them:

Equation 1:

$$\begin{aligned} \frac{dV}{dt}(t) &= V_N(k_p - k_l) \\ \text{where } V_N &= V_0 \times e^{\frac{(k_p - k_t)}{\Delta t}} \end{aligned}$$

Here, V_0 is the tumor volume at time 0, k_l and k_p are constants governing the rates of cell loss and cell production respectively.

Table 3.6: Discovery of Model for tumor growth progression

Primitives	$1, 0, +, -, \times$, power, car(Lisp), map(Lisp), index(Lisp)
Wake/Sleep Cycles	1
Datatypes	real number list, real number $\rightarrow$ real number list

Here the system is again able to use the high-level primitive provided by the forward Euler method for ODEs and is thus able to find a suitable program within the first wake/sleep cycle as seen in the following graph:

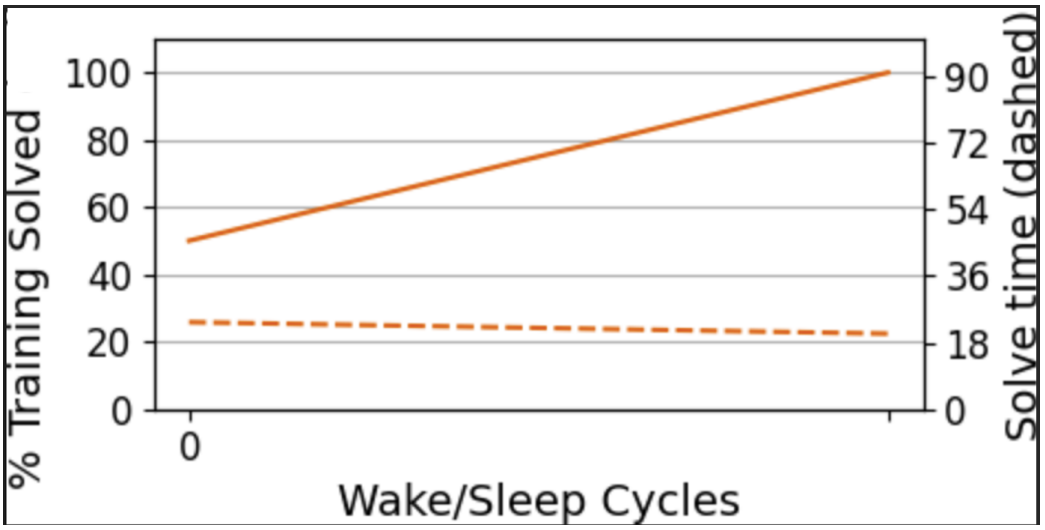

Figure 3.4: This graph shows the progression of search for the ODE that describes tumor growth in terms of its volume.

Equation 2:

$$p = 1 - (1 - (1 - (1 - \mu)^d)^k)^{Nm} \tag{3.5}$$

Where p is the probability of cancer, d is the number of cell divisions, Nm is the number of stem cells, k is the number of critical rate-limiting pathway driver mutations and μ is the mutation rate as described by Calabrese and Shibata [31].

Table 3.7: Discovery of model describing probability of cancer

Primitives	$+, -, \times, 1, 0$, power
Wake/Sleep Cycles	2
Datatypes	real number $\sim 10^{-6}$, real number $\sim 10^{-6}$, real number $\sim 10^{-6}$, real number $\in [0, 1]$ $\rightarrow$ real number $\in [0, 1]$

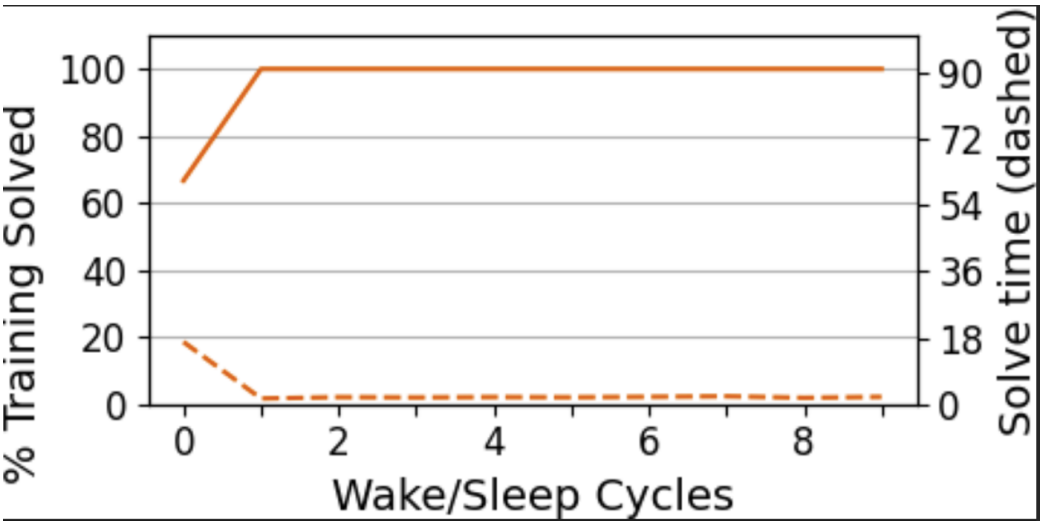

Figure 3.5: This graph shows the progression of search for the aforementioned probabilistic model of cancer.

3.2.2 Synthesizing Unknown Hypotheses to Predict Tumor Tolerance

The main dataset that was utilized for this thesis was a set of 269 images from HD-IR spectroscopic imaging data provided by Professor Rohit Bhargava's Laboratory at the Beckman Institute at the University of Illinios at Urbana Champaign. The images were segmented into 10 colon histological

components (HCs) as described by Tiwari et al [32]. For each HC, the authors used 2 features: d_{HC} which was defined as the distance between tumor cells and a non-tumor HC, and N_{HC} which was defined as the density of an HC in a predetermined radius R=600 microns. From the authors' tests, these features were only statistically significant with reactive stroma (RS) HC.

The following diagram from Tiwari et al. illustrates the features as they are measured on the hematoxylin and eosin (H&E) image:

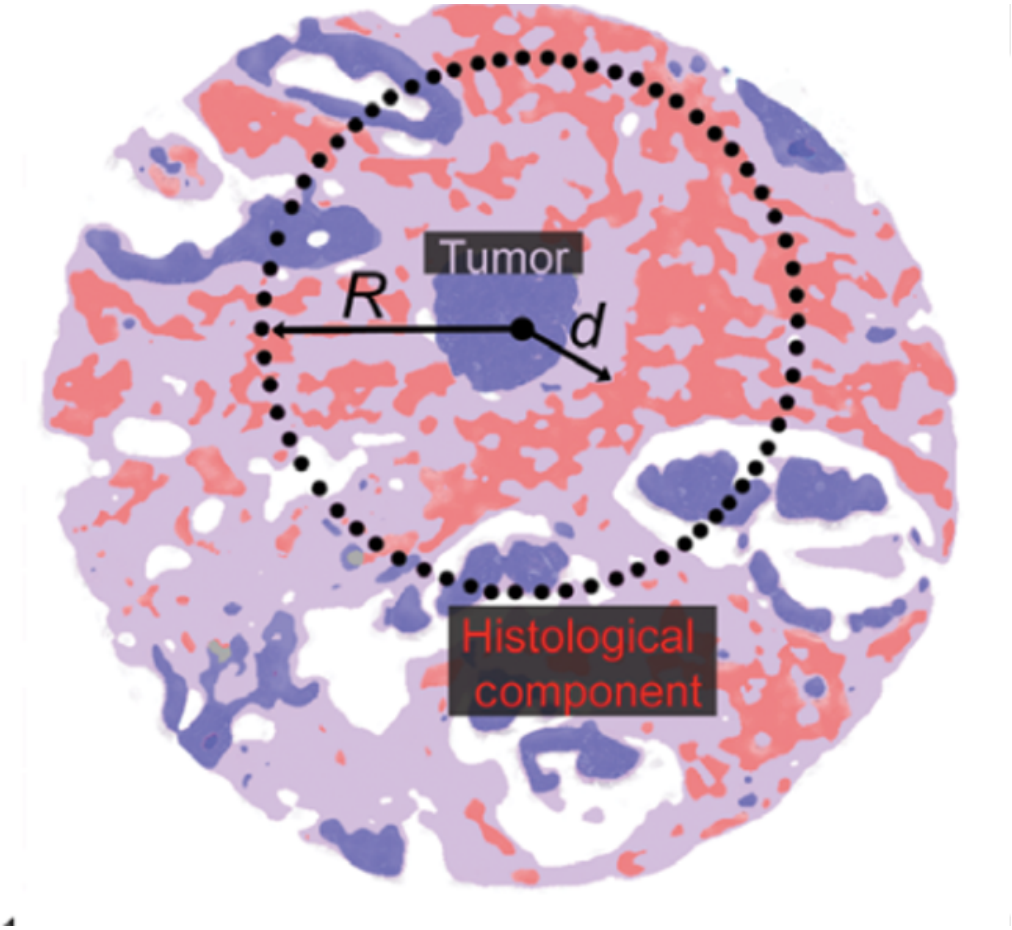

Figure 3.6: From figure 3A of Tiwari et al. [32]

These two features, namely d_{RS} and N_{RS} were selected as primitives for program discovery from these images. These features were obtained from segmented image that was part of the dataset by selecting 50 separate datapoints for each image by selecting 50 random tumor pixels from the segmentation map, and using Euclidean distance measurement to the nearest RS pixel to obtain d_{RS} and dividing the number of RS pixels by the number of total pixels in a radius R=600 microns to obtain N_{RS}. The following figure displays an example of H&E image of a tumor microenvironment along with a segmented image of the same.

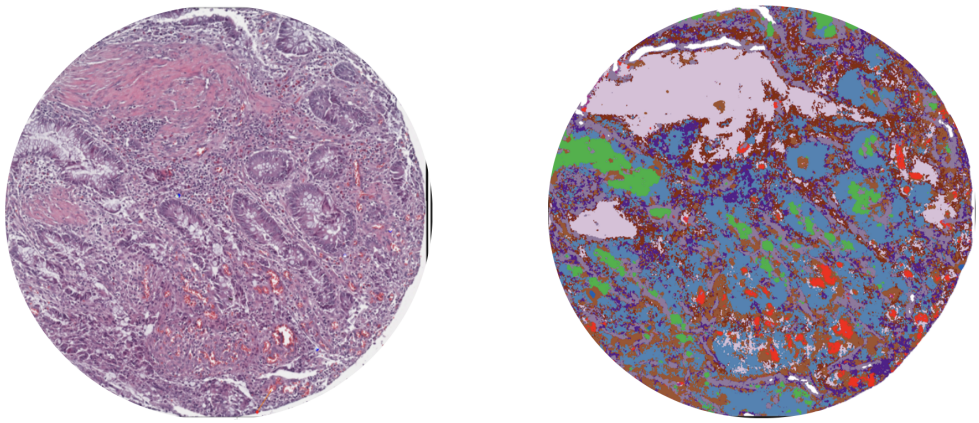

Figure 3.7: (left) H&E image of a tumor microenvironment. (right) Corresponding segmentation map from which features were obtained.

The dataset was posed as a program synthesis problem by making tuples (d_{RS}, N_{RS}) of the features as input, as well as True or False as output, depending on if the tumor was well or poorly tolerated respectively. The goal of the program synthesis was to discover a hypothesis that was able to model tumor tolerance based on the two features provided to it. Over multiple runs with different sets of primitives (systematically cycling through algebraic, Boolean, and advanced primitives such as Euler's formula for ODEs) as well as running different searches in parallel, the following table describes the runs with the top 3 best accuracies (defined as number of correct predictions on a test set withheld during training divided by total size of test set):

These results are over a baseline of 50% by randomly selecting True/False. Interestingly, despite being provided with a set of advanced primitives such as the ability to model ordinary differential equations, the highest accuracies were achieved by programs that were completely algebraic, with thresholding

Table 3.8: Discovery of model describing probability of cancer

Primitives	**Wake/Sleep Cycles**	**Hypotheses**	**Accuracy**
$+, -, \times, \div, \text{power}, \geq$	10	$d_{\text{RS}}^2 N_{\text{RS}} \geq 0$	76.57%
$1, 0, -1, +, -, \times, \geq$	5	$d_{\text{RS}} N_{\text{RS}} \geq 1$	69.64%
real numbers $\in [-1, 10], +, \times, \geq$	4	$d_{\text{RS}} \geq 12 N_{\text{RS}}$	65.67%

on certain real numbers or single-variable algebraic expressions. Of note is the necessary inclusion of a primitive that given a real number or set of real numbers, will output a Boolean. This was necessary as the problem is framed as a classification problem, since framing it as anything else (regression, for instance) would have made the search space much larger by virtue of the output space being the set of all real numbers rather than simply True/False, and done so without providing any relevant information.

CHAPTER 4

CONCLUSION

In this thesis, we introduced wake/sleep Bayesian program synthesis as a state-of-the-art general purpose program synthesis framework and demonstrated its ability to discover a wide range of mathematical and logical models in the bioinformatics domain without having to perform changes to the fundamental algorithm between changing domains. We demonstrated the ability of Dreamcoder to synthesise the Euler method for ordinary differential equations by extending Dreamcoder's list processing functionality, which opens the possibility for more complex and expressive models to be discovered. We also ported the system onto a tumor classification problem, and the results achieved are indicative of a degree of success – wake/sleep program synthesis will certainly not solve cancer but can help uncover correlations between features as well as use a set of limited features to induce a program that maximizes predictive power by discovering an 'interaction feature'.

Future work to enhance portability onto the bioinformatics domain will involve wake-sleep program synthesis on linear algebra primitives as well as on multi-dimensional tuples. This is because given an input image, the ability to perform matrix operations and capture locality from images will likely allow for spatial and geometric relations in images to be implicitly conceptualized, and by leveraging the feature extraction power of Convolutional Neural Nets and then using those as primitives for program synthesis we may be able to discover more expressive hypotheses that describe tumor tolerance with higher accuracy and give novel insight into fundamental processes in the tumor microenvironment.

REFERENCES

[1] K. Tran et al., "Deep learning in cancer diagnosis prognosis and treatment selection," *Genome Medicine*, 2021.

[2] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, 2019.

[3] S. Lee et al., "Robust Tumor Localization with Pyramid Grad-CAM," *CORR*, 2018.

[4] R. Selvaraju et al., "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," 2016.

[5] Y. LeCun et al., "Deep learning," *Nature*, 2015.

[6] K. He et al., "Deep Residual Learning for Image Recognition," *Computer Vision Foundation*, 2016.

[7] O. Ronneberger et al., "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[8] G. Jimenez and D. Racoceanu, "Deep Learning for Semantic Segmentation vs Classification in Computational Pathology: Application to Mitosis Analysis in Breast Cancer Grading," *Frontiers in Bioengineering and Biotechnology*, 2019.

[9] H. Ryu et al., "Automated Gleason Scoring and Tumor Quantification in Prostate Core Needle Biopsy Images Using Deep Neural Networks and Its Comparison with Pathologist-Based Assessment," *Cancers*, 2019.

[10] G. Nir et al., "Comparison of Artificial Intelligence Techniques to Evaluate Performance of a Classifier for Automatic Grading of Prostate Cancer From Digitized Histopathologic Images," *JAMA Netw Open*, 2019.

[11] P. Strom et al., "Artificial intelligence for diagnosis and grading of prostate cancer in biopsies: a population-based diagnostic study," *Lancet Oncology*, 2020.

[12] B. Ehteshami et al., "Using deep convolutional neural networks to identify and classify tumor-associated stroma in diagnostic breast biopsies," *Modern Pathology*, 2018.

[13] T. Vuoung et al., "Multi-task deep learning for colon cancer grading," *ICEIC Barcelona*, 2020.

[14] K. Menden et al., "Deep learning–based cell composition analysis from tissue expression profiles," *Science Advances*, 2020.

[15] K. Yu et al., "Classifying non-small cell lung cancer types and transcriptomic subtypes using convolutional neural networks," *Journal of the American Medical Informatics Association*, 2020.

[16] B. Jing et al., "A deep survival analysis method based on ranking," 2019.

[17] J. Nam et al., "Development and Validation of Deep Learning-based Automatic Detection Algorithm for Malignant Pulmonary Nodules on Chest Radiographs," *Radiology*, 2019.

[18] M. Gulum et al., "A Review of Explainable Deep Learning Cancer Detection Models in Medical Imaging," *Applied Sciences*, 2021.

[19] Defense Advanced Research Projects Agency, "Broad agency announcement explainable artificial intelligence," *DARPA-BAA-16-53*, 2016.

[20] E. Hoofnagle et al., *The European Union general data protection regulation: What it is and what it means.* Information Communications Technology Law, 2019.

[21] A. Holzinger et al., "Causability and explainability of artificial intelligence in medicine," *Wiley Interdiscip Rev Data Min Knowl Discov*, 2019.

[22] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Wiley, 2010.

[23] Z. Manna and R. Waldinger, "A deductive approach to program synthesis," *ACM Transactions on Programming Languages and Systems*, 1980.

[24] M. Balog et al., "Deepcoder: Learning to write programs," *CoRR*, vol. abs/1611.01989, 2016. [Online]. Available: http://arxiv.org/abs/1611.01989

[25] K. Ellis et al., "Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning," *Association for Computing Machinery*, p. 835–850, 2021. [Online]. Available: https://doi.org/10.1145/3453483.3454080

[26] H. Nassif et al., "Uncovering Age-Specific Invasive and DCIS Breast Cancer Rules Using Inductive Logic Programming," *IHI'10 - Proceedings of the 1st ACM International Health Informatics Symposium*, pp. 76–82, 11 2010.

[27] V. Bevilacqua et al., "Identification of tumor evolution patterns by means of inductive logic programming," *Genomics Proteomics Bioinformatics*, 2008.

[28] J. Fisher and S. Woodhouse, "Program synthesis meets deep learning for decoding regulatory network," *Current Opinion in Systems Biology*, 2017.

[29] C. Kuttler, "Mathematical models in biology," 2009.

[30] R. Zwanzig, "Simple model of protein folding kinetics," *Proceedings of the National Academy of Sciences of the United States of America*, 1995.

[31] P. Calabrese and D. Shibata, "A simple algebraic cancer equation: calculating how cancers may arise with normal mutation rates," *The American Journal of Pathology*, 2010.

[32] S. Tiwari et al., "INFrared-based Organizational Measurements of tumor and its Microenvironment to predict patient survival," *Science Advances*, 2021.