
LEARNING CHEMOTHERAPY DRUG ACTION VIA UNIVERSAL PHYSICS-INFORMED NEURAL NETWORKS

A PREPRINT

Lena Podina

Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada
lpodina@uwaterloo.ca

Ali Ghodsi

Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada
ali.ghodsi@uwaterloo.ca

Mohammad Kohandel

Department of Applied Mathematics
University of Waterloo
Waterloo, ON, Canada
kohandel@uwaterloo.ca

April 15, 2024

1 Introduction

Quantitative systems pharmacology (QSP) is a commonly used approach of mathematically assessing drug pharmacokinetics and pharmacodynamics before they go to clinical trial [27]. Many drugs fail phase II and III clinical trials [28] due to inadequate understanding of the mechanism of action. However, via QSP models, which integrate biological, physiological and pharmacological data [24], it is possible to understand the underlying biological mechanisms and optimize drug dose and schedule. Additionally, QSP models can be used to predict drug toxicity or efficacy [19], as well as individual patient response to the drug.

Despite its benefits, QSP suffers from several limitations which make the models costly and time-consuming to develop. First, many QSP models are built by making simplifying assumptions about the underlying biology. In general, manual distillation of the literature (and potentially large amounts of data) is done in order to build these assumptions accurately, rather than building them directly or automatically in a data-driven way. Secondly, building a model fully (including parameter estimation) can be very time-consuming due to the model complexity and manual distillation of the literature [30]. QSP modelling is especially affected by these issues because a QSP model may comprise more than a dozen dynamical variables, and even more parameters [30]. Recently, machine learning (ML) has been used to improve and simplify the QSP model-building process. Decision trees, regression, and neural networks are some of the tools that have been used to predict drug toxicity and patient response in a data-driven way [5]. A major benefit of ML is the almost entirely automatic learning of patterns in the data by a model, and potentially robust generalization to unseen data [5]. However, purely ML-based approaches often need large amounts of data to make accurate predictions, and are often uninterpretable. This means that although the model may produce reasonable predictions, the mechanism by which it does so is opaque. For this reason, an ML model may be sometimes called a “black box”. In systems pharmacology, it is often important to know which features are the most important for prediction and why a certain outcome is expected. These findings shed light on underlying biological mechanisms, and inform drug development. This is also key to a sanity-check of the model prediction, and trust in the output.

Integrating ML and QSP is a promising research direction which has promise in binding together their strengths while compensating for their respective weaknesses. Machine learning can be used to automate the learning of complex patterns, or abstract away the function of less important mechanisms. QSP, on the other hand, can provide structure to the machine learning model and inform it with prior knowledge. An integration of these two methods means that some assumptions will be built in or enforced, but other interactions and mechanisms will be learned directly from

the data. Due to the additional structure imposed by assumptions, potentially less data will be needed to learn the unknown components. The components of the model that need potentially more manual distillation can be learned more automatically from the data. Finally, the interpretability of the ML components may be augmented due to the surrounding structure imposed by the QSP model.

Existing integrations of QSP and ML tackle problems such as parameter inference [23], inference of model structure [9, 22], dimension reduction [6], and creation of virtual patients [1]. A prominent method to identify parameters from data is physics-informed neural networks (PINNs). PINNs are a flexible framework and have been extended in various ways [10, 20, 29] for the purposes of model fitting and inference, which are key problems in systems pharmacology. Given a differential equation that models a particular biological process, PINNs use a neural network to fit a surrogate solution. The unknown parameters are treated as additional weights to be optimized. However, PINNs cannot be used to identify entire unknown components of the model. Inference of model structure, as described in [30], has followed several different directions, such as logic-based modeling [9], integration of QSP with genome-scale computational models [21], and universal differential equations (UDEs) [22]. UDE’s are a popular method to identify unknown components of differential equations. However, they are not robust to noise, as [18] recently showed, and the Universal Physics-Informed Neural Network (UPINN) method is more robust. In this paper, we apply the UPINN method to chemotherapy modelling, as a proof of concept of what the method can accomplish in the realm of pharmacokinetic and pharmacodynamic modelling. We have applied the method to both simulated and in-vitro data, and showed high accuracy of identification of the unknown components of differential equations.

In the remaining sections, we start with an overview of existing modeling approaches to chemotherapy modelling, physics-informed neural networks and QSP structure identification methods. Following that, we use the UPINN method to identify the hidden components of various ODEs that model chemotherapy dynamics. We show the performance of our method on both synthetic and in-vitro data. For the experiments on in-vitro data, we learn the time-dependent effect of a chemotherapeutic, doxorubicin, on cell growth (proliferation).

Our contributions in this work are:

- We integrate machine learning in the form of PINNs with QSP models in order to identify unknown drug actions within QSP models. To illustrate, we apply the Universal PINN method to identify hidden terms in QSP models in cases of both synthetic and in-vitro experimental data.
- Via simulations, we show that three different types of drug action (Log-Kill, Norton-Simon, E_{max}) can be identified from the chemotherapy concentration and number of cells over time. We also employ our approach to recover dose-dependent parameters from several sets of data simultaneously and interpolate these parameters between dosages. This could potentially replace the repeated application of a standard physics-informed neural network.
- We show that our approach can successfully identify the time-dependent net proliferation rate in cases of both synthetic and in-vitro experimental doxorubicin data.

2 Background

2.1 Physics-informed Neural Networks

Physics-informed neural networks [23] (PINNs) were developed by Raissi et al. for the purposes of solving forward and inverse problems of differential equations (partial (PDE) and ordinary (ODE)). For the forward problem, a neural network is used to provide a numerical solution to a PDE using data that satisfies the PDE. In this case, the data generating process is fully known. For the inverse problem, the differential equation is known except for certain parameters, which take on a real value. Using data, PINNs can be used to identify the parameters that best fit the data.

As mentioned previously, a drawback of PINNs is that they cannot be used to identify entire hidden terms of differential equations. In [18], a method called Universal PINNs (UPINNs) was developed to find the hidden terms in a data-driven way. was demonstrated on the Lotka-Volterra system of ODEs, on a model of cell apoptosis with bistability, and on Burger’s equation, a PDE. In this paper, we explore further the power of the method in identifying unknown terms from data, thereby avoiding modelling assumptions. We base our work on the models in [16] and [13] We show that in every combination of drug action, we are able to make an approximation of the drug action using a neural network. Furthermore, we can combine the UPINNs approach from [18] with the traditional PINN approach to identify parameters first, and then use these parameters to inform the identification of the hidden terms.

2.2 Chemotherapy modelling

Much of the effort of the modelling of chemotherapeutics has gone into optimizing the drug schedule. This means finding the dosing schedule that kills the most cancer cells, while limiting the toxicity to normal cells. In many studies, the modelling of the drug is performed using assumptions on how the drug acts on the cells, and afterwards, the parameters of the model may be fit from data. For example, in [25], a Norton-Simon model of drug action and Gompertzian growth was assumed, and then fit to data. After, the drug schedule can be perturbed and the best dosing schedule can be identified from model predictions. In [7], a reaction kinetics analogy is assumed for the model, and the parameters are fit to tumours in untreated mice. Fits are subsequently evaluated for treated mice. [3] used assumptions based on paclitaxel’s effect on cells in different phases of the cell cycle. Hence their model is compartment based, with differential equations governing the transition of cells from one phase to the next. In [11], tumour growth was modelled with logistic growth, but the effect of the chemotherapeutic (also paclitaxel) is modelled only through its effect on the carrying capacity. In [4], a novel fractional-order Gompertz model is introduced and determined to be a better fit for experimental mouse tumour data.

Optimizing the order and sequence of chemotherapy and surgery is also an important problem. In [13], a mathematical model of ovarian cancer was developed to determine whether chemotherapy, then surgery, or whether surgery then chemotherapy was a better treatment. The authors examined Gompertzian and logistic growth, and three different types of cell-kill methods: log-kill, Norton-Simon, and E_{max} . In all cases, the sequencing of chemotherapy followed by surgery led to better outcomes than surgery followed by chemotherapy. However, it is key to note that in order to make a conclusive recommendation, all reasonable possibilities and assumptions need to be examined. When data is available, these assumptions can be learned from data, as we endeavour to show by adapting the UPINN method to this problem.

3 Methods

We build on the methods in [18]. In this work, we apply UPINNs in three different scenarios. First, we apply it to learn the drug action of a cancer growth ODE, modifying the method for better performance. Using the method, we are able to obtain an approximation of the drug action in three ODEs with three different drug actions. Secondly, we apply the method to a situation where the parameters vary with dose. We learn the parameters as a function of the dosage, and interpolate between observed dosages. Finally, we learn the net proliferation rate of doxorubicin from in-vitro data, after validating the method on similar synthetic data.

In the following subsections, we first describe how to apply the method generally, and in the subsequent sections, we detail the specific applications.

3.1 General approach

Suppose an ODE in the following form, with m variables, where $u : t \rightarrow \mathbb{R}^m$ and $F, G : u, t \rightarrow \mathbb{R}^m$:

$$\frac{du}{dt} = F(u, t) + G(u, t) \quad (1)$$

Suppose furthermore that F can be evaluated for every possible input, but G is unknown and so cannot be. Then given a dataset of n points $\{t_i, u_i\}$ which satisfy the differential equation, we can use the UPINN method to approximate G at these points, and at points that are linear combinations of pairs (t_j, u_j) in the dataset. We do this by representing $G(u, t)$ with a neural network G_{NN} , taking u and t as input and returning a vector-valued output for each such combination. However, some of the outputs of G may be a priori known to be zero, and as such don’t need to be modeled by the neural network. A case of this can be seen in the following sections. Furthermore, although in (1), F and G are assumed to be added, this assumption need not hold in order for the method to learn G . Instead, there may be a different (known) function combining F and G on the right-hand side of (1) (e.g. $F \cdot G$) and if G is identifiable given the data, the UPINN method should still be able to learn G . However, more complex expressions tend to be more difficult for the method to learn.

As per the standard PINN approach, along with an approximation of G , we obtain an approximation of the differential equation solution u using a neural network U_{NN} . U_{NN} and G_{NN} are trained by minimizing the following loss:

$$\mathcal{L} = \mathcal{L}_{MSE} + \mathcal{L}_{ODE}$$

The MSE loss ensures that the output of the surrogate solution U_{NN} adheres to the data $\{t_i, u_i\}$:

$$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{i=1}^n (U_{NN}(t) - u_i)$$

As for the ODE loss, U_{NN} is autodifferentiated¹ with respect to t and evaluated at a set of collocation points $\{t_j\}$. This loss minimizes the difference between the derivative of $U_{NN}(t)$ and the right hand side of eq. 1. In effect, this loss component enforces the differential equation to hold.

$$\mathcal{L}_{ODE} = \frac{1}{K} \sum_{j=1}^K \left(\left. \frac{dU_{NN}(t)}{dt} \right|_{t_j} - (F(U_{NN}(t_j), t_j) + G_{NN}(U_{NN}(t_j), t_j)) \right)$$

G_{NN} is trained through the ODE loss component, and U_{NN} is trained through both loss components. A flowchart of the components and their interactions can be seen in Fig 1.

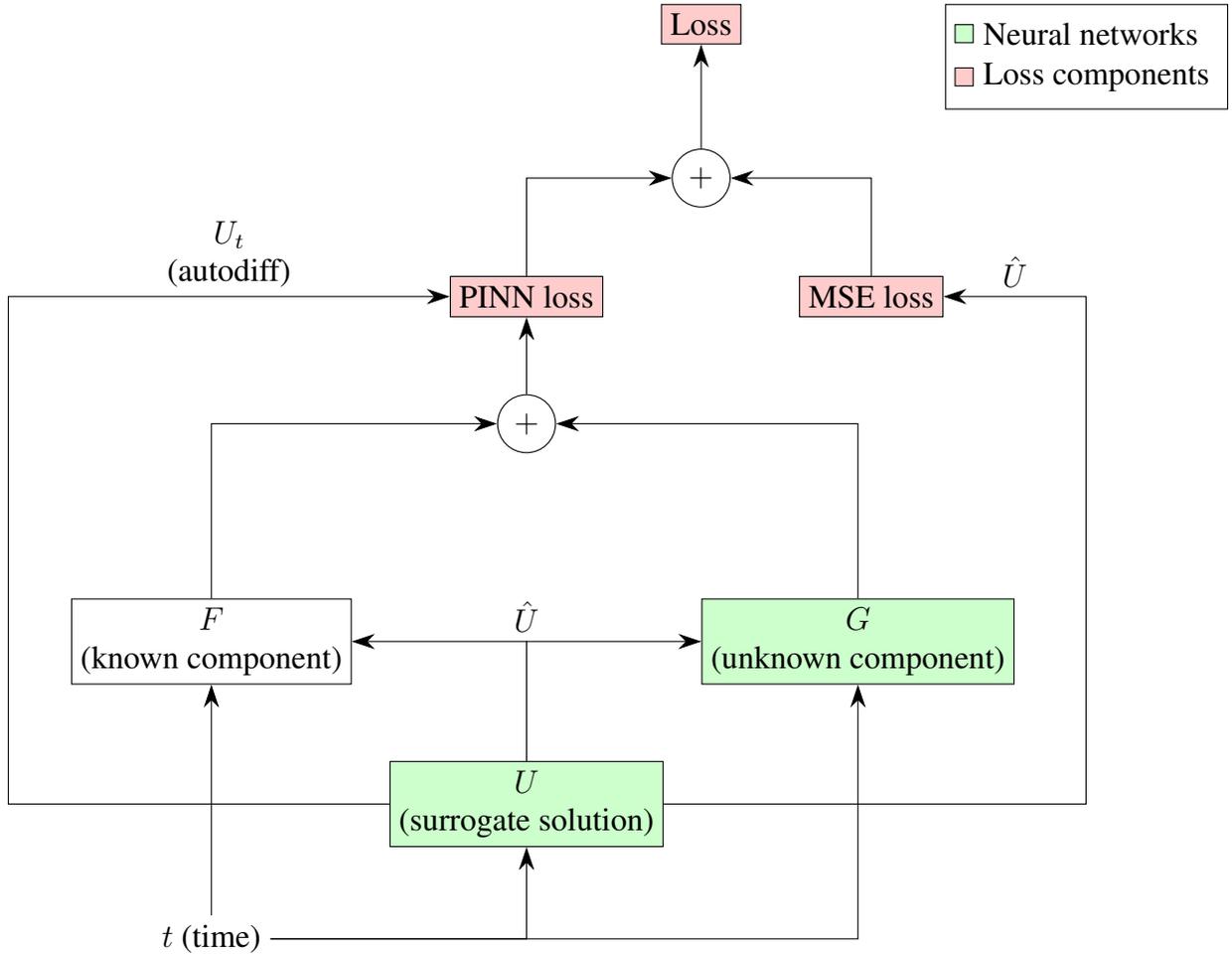


Figure 1: Overview of the structure of the UPINN method as applied to (1), which shows inputs and outputs of all known and unknown components, as well as losses. The surrogate solution U outputted by the UPINN takes time t as input. Both F (the known component of the differential equation) and G (the unknown component, to be fit by the UPINN) take in time and \hat{U} , the prediction of the neural network, as input. F and G , along with U_t (the autodifferentiated derivative of U_{NN} w.r.t. time) and is passed as input to the PINN loss. Then, the PINN loss computes the error between U_t and $F + G$. The MSE loss computes the error between the surrogate solution \hat{U} and the data.

¹A common operation in the training of neural networks, which in this case computes the derivative of the output of the neural network.

3.2 Application to the identification of a cell-kill strategy

We are interested in applying the above method to identify the kind of drug action that a specific chemotherapeutic has. Using an ODE, we simulate a situation where the chemotherapeutic is applied in a single dose to a culture of cells, and the cell numbers are measured at intervals after the application of the chemotherapeutic. We choose the following differential equation to simulate the concentration and cell numbers over time:

$$\frac{dN}{dt} = \beta N(1 - N) - C(t)G(N) \quad (2)$$

$$\frac{dC}{dt} = -\gamma NC \quad (3)$$

where $C(t)$ is the concentration of the drug (between 0 and 1), and $N(t)$ is normalized cell count, and as such takes values between 0 and 1. The first term governing the tumour dynamics is a growth term and is known. In general, we assume to have no information about $G(N)$, the drug action of the chemotherapeutic. However, for the purposes of testing the method, we choose G to have one of the three following forms, as per [13, 16]:

1. $a_1 N$ (Log-kill model)
2. $a_2 \beta N(1 - N)$ (Norton-Simon model)
3. $a_3 N / (N + \delta)$ (E_{max} model)

where a_j are constants.

We generate synthetic data $\{t_i, N_i, c_i\}$ satisfying the above differential equation, with initial condition $N(0) = 0.01$. C is assumed to be zero for $t \in [0, 12]$, and at day 12, the drug is instantaneously added to create a concentration of 1. From there the drug decays depending on the parameter $\gamma = 1.0$. For the E_{max} case, we have $\delta = 0.55$. β is assumed to be known and equal to 1.0, but can be fit using a regular PINN from untreated growth data. Finally, the constants $a_{1,2,3}$ are set as $a_1 = 2.8$, $a_2 = 11.0$, $a_3 = 2.4$. An example of this workflow is shown in the results.

We set up three neural networks as follows: let $G(t, N) = C(t)G(N)$ and we approximate $G(t, N)$ with a neural network G_{NN} with two inputs (t and N) and one output. Separately, we approximate $C(t)$ by C_{NN} with one input, t , and one output, $C(t)$. Finally, we have a neural network U_{NN} with one input (t) and one output (N), which approximates the solution $N(t)$.

The workflow proceeds in three steps. First, we train C_{NN} using only an MSE loss with respect to the $\{t_i, c_i\}$ data. The purpose of this network is to enable the interpolation of the concentration between observed timesteps. Then, we train $G_{NN}(t, N)$ using the data $\{t_i, N_i\}$ using exactly the method described in the previous subsection. Finally, having obtained predictions for a set of times $\{t_j\}$ from both C_{NN} and G_{NN} we divide the prediction of G_{NN} by the prediction of C_{NN} to obtain an approximation of $G(N)$ for every timestep t_j . Since we also have the solution U_{NN} for every timestep, we are able to get an approximation of $G(N)$ for any N within the bounds of the data. This function cannot extrapolate to unseen N , so in generating $\{t_i, N_i, c_i\}$ it is important that the widest range possible is covered by the values N_i .

This method enables us to create a black-box representation of drug actions. The approximation of $G(N)$ for different timesteps t_j may then be run through a symbolic regression algorithm to obtain a closed-form of the function. For example, AI Feynman [26] can suggest possible closed-form functions linking the t_j and $G(N)$, trading off the simplicity of the function with how well it fits the data. This is different than training a regression model or neural network because the symbolic regression algorithm searches a wide space of possible functions (e.g. including mathematical functions such as log or cos directly) rather than restricting the search space to polynomials or multi-layer perceptrons. Although the symbolic regression algorithm can find a closed form, which may shed some light on the underlying biological mechanisms of the hidden term, certain downstream applications for the drug action (e.g., finding an optimal drug schedule using reinforcement learning as in [8]) do not require a closed form.

The UPINN was implemented in PyTorch [17]. The neural network which represents the surrogate solution was a fully-connected network with 8 hidden layers with 20 hidden units each, tanh activation. All neural networks used in this work are structured similarly and use tanh activation. Each neural network which represents a hidden term was structured the same way. First, 5000 Adam [12] iterations were run, and then the Limited Memory BFGS optimization algorithm [14] was run until sufficient convergence. The MSE and PINN loss components were weighted equally at all times. The inputs to each neural network were scaled such that it would be between 0 and 1 for all inputs (both time and number of cells).

3.3 Application to parameter identification for several drug dosages

In some situations, the differential equations that govern a specific biological process has parameters that vary not in time, but as a function of chemotherapy dosage. One such example can be created by assuming that the parameters k_p, θ in logistic growth

$$\frac{dN}{dt} = k_p N \left(1 - \frac{N}{\theta} \right) \quad (4)$$

are in fact a function of the administered chemotherapy dosage, D . This changes the previous equation to

$$\frac{\partial N(t, D)}{\partial t} = k_p(D) N(t, D) \left(1 - \frac{N(t, D)}{\theta(D)} \right) \quad (5)$$

where $k_p(D)$ and $\theta(D)$ are now the dose-dependent growth rate and carrying capacity. An initial condition constraint $N(0, D) = N_0$ can be included as well. In this case, for a higher dosage of chemotherapy, the growth of the cells would remain logistic, but the growth rate and carrying capacity would decrease. Conversely, if the dosage is low then the growth rate and carrying capacity would be higher. We consider this a realistic scenario for modelling the effects of doxorubicin, due to a model employed in a recent work [15].

Rather than solving (5), it is simpler to use an ODE solver to solve (4) for different (k_p, θ) . This generates several time series datasets $\{t_i, n_i\}_D$ of the number of tumour cells n_i over time (t), one for each chemotherapy dosage condition D . It is possible to aggregate them into one dataset and fit $k_p(D)$ and $\theta(D)$ simultaneously to all available data. In our UPINNs setup, we simulate three different datasets using logistic growth, under low, medium and high dosage conditions. Three different dosages were considered in each experiment (low, medium and high), given as 15.0, 25.0, and 45.0 nM respectively to the model. Since this data is synthetic and the dosage is not used in the ODE directly, the numerical values of the drug concentration are somewhat arbitrary. 48 datapoints were used in total for each experiment, which means there were 16 datapoints per dosage. We added noise proportionally to the mean of the data, as per synthetic experiments in [18] at a noise level of 0.03. We aggregate the data for all dosages, and model the system via eq. 5. In our setup, we have three neural networks: one that models the surrogate solution, and two that model $k_p(D)$ and $\theta(D)$ respectively. The surrogate solution takes both the time and dosage as input. The remaining two neural networks have only one input (dosage) and one output each. As there are only three dosages for which k_p and θ need to be fit, the parameters for unobserved dosages can be interpolated by the model. We note that additional constraints on the interpolation can be included by adding another loss term.

It is worth noting that if standard PINNs were utilized to fit these dose-dependent parameters, a separate PINN would have to be fit for the dataset corresponding to each dose. This is a more significant computational burden, and also not scalable when more than a few dosages are available. It also does not allow interpolation between dosages. By modelling eq (2) using UPINNs, we are able to fit one surrogate solution N to all the datasets and obtain all the dose-dependent parameters simultaneously, provided they are identifiable.

The architecture of all neural networks was the same as in the previous section. Training proceeded very similarly, with 5000 Adam iterations at first. Then, the data loss was weighted with a value $\lambda = 0.001$ and Adam training was continued for 1000 iterations. Then, L-BFGS finished the optimization. This training method was used in order to ensure that the PINN loss was minimized sufficiently during training. Similar scaling was employed on the input.

3.4 Application to in-vitro experimental data

We now apply our method to identify the drug action of a real chemotherapeutic from in-vitro cell counts. For this purpose, we gathered the data used in [15]. In this series of experiments, four different cell lines (MDA-MB-468 (basal-like 1), SUM-149PT (basal-like 2), MDA-MB-231 (mesenchymal), and MDA-MB-453) were first allowed to grow undisturbed for at least three days, and then they were exposed to different concentrations of doxorubicin. The cells were exposed to doxorubicin for 6, 12 or 24 hours, after which the medium was changed. The concentrations of doxorubicin that were used by [15] are: 10nM, 20nM, 39nM, 78nM, 156nM, 312nM, 625nM, 1250nM, 2500nM.

In this work, the time-series cell count data is described by the following model:

$$\frac{dN_{TC}}{dt} = (k_p - k_d(t, D))N_{TC}(t) \left(1 - \frac{N_{TC}}{\theta(D)}\right) \quad (6)$$

$$k_d(t, D) = k_{d,A}(D) \quad (7)$$

$$k_d(t, D) = k_{d,B}(D)r(D)te^{1-r(D)t} \quad (8)$$

where $N_{TC}(t)$ is the number of cells over time, k_p is the constant growth rate under control conditions, $k_d(D, t)$ is the death rate, dependent on dosage and time, $\theta(D)$ is the dose-dependent carrying capacity, and $r(D)$ is also a function of dosage. These parameters also take on different values per cell line. In [15], eq 7 and eq 8 are fit separately, and the final prediction for each dosage is a weighted combination of the predictions of the two models. The authors have fit $k_{d,A}(D)$, $r(D)$, $k_{d,A}(D)$ as constant parameters per dosage using a nonlinear least squares approach. No time-dependent or dose-dependent functions are fit.

We note that if k_p is allowed to take on any value, then fitting $k(D, t)$ and $\theta(D)$ simultaneously using our method is not identifiable. Hence, we simplified this model to the two following forms:

$$\frac{dN_{TC}(t)}{dt} = F_D(N, t)N_{TC}(t) \quad (9)$$

$$\frac{dN_{TC}(t)}{dt} = G_D(t)N_{TC}(t) (1 - N_{TC}(t)) \quad (10)$$

The subscript D indicates that this function is different for each dosage. In our implementation, F_D takes only time as the input, although it could take both numbers of cells, N and time t as input. We suggest and work with these two formulations of the hidden term, because for different applications, more prior knowledge can be incorporated if it is known.

Equations 9 and 10 are certainly identifiable for all tuples of time, cell count and time derivative values, denoted by $\{t_i, N_i, dN_i\}$. This is because for every tuple $\{t_i, N_i, dN_i\}$ at each of the collocation points, there is a unique solution of either eqs. 9 and 10 for F_D or G_D (respectively) at each dose D . Note that the equations 9 and 10 are general enough to be applicable to several scenarios: simulated data from eq 7, simulated data from eq 8, and the in-vitro data. Hence, we learn $F_D(N, t)$ and $G_D(t)$ for all three of these cases, except $F_D(N, t)$ generated using eq 7. This is because $F_D(N, t)$ takes on a single constant value in that case.

The loss function was the same as in the previous sections, and the training proceeded similarly. The neural network architectures used were of 8 hidden layers of 128 units each, for each fully-connected neural network. Training was done such that Adam iterations were stopped at 1500 iterations, or upon reaching a loss of $2 \cdot 10^{-3}$, whichever comes first. Then, L-BFGS iterations were started. This conditional stopping of Adam iterations was done in order to bring the weights sufficiently close to the optimum before L-BFGS starts, but not so close that L-BFGS fails to perform any optimization steps.

4 Results

In the following three sections, we detail the performance of the identification of hidden terms in all three of the scenarios. We apply our method to learn the drug action (the model that best describes the cell dynamics after treatment) as a function of the cell count only, then we learn growth rate parameters as a function of the dosage, and finally we learn the net proliferation rate of doxorubicin as a function of time.

4.1 Identification of drug action (Log-kill, NS, E-max) from synthetic data

As described in the methods, we generate synthetic data for three different drug actions, and we evaluate how well the underlying true drug action can be recovered. We test our method on sparse noiseless data generated in two different ways, and noisy data. We also test a two-step procedure where we recover β first from the untreated data using a regular PINN, and then apply our method to find the drug action.

First, we test on sparse data with spacing that is uniform through time. We generate a time series dataset of 54 points for each drug action. Fig 2 shows the learned drug action for all three types of cell-kill strategies, when the spacing was uniform in time. In other words, it plots the learned $G(N)$ as a function of different N that appeared in the training data. We note that the mean squared error (MSE) between the true drug action and the predicted drug action, as reported in

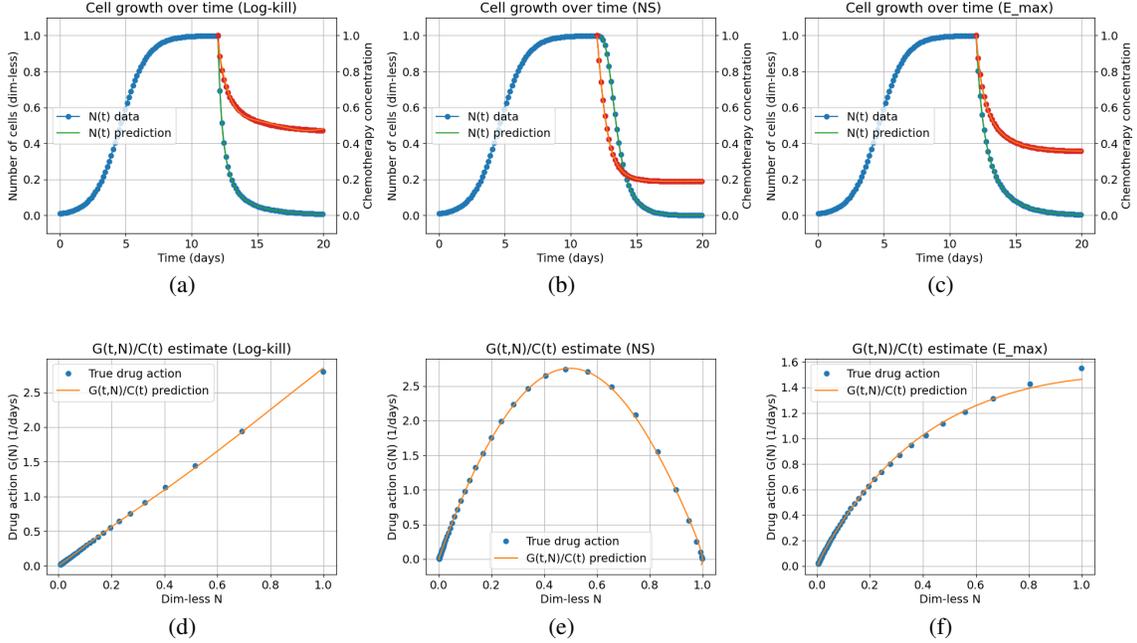


Figure 2: Datasets for the different drug actions (a, b, c) with their respective drug actions learned below them (d, e, f). Data was collected with a fixed time period (0.15 days) elapsing between each data point. The drug actions are the functions $G(N)$ in Eq (2), replaced by one of: log-kill, Norton-Simon and E_{max} .

Table 1 for all three types of drug action, are all on the order of 10^{-4} . A visual inspection shows that the predicted drug action matches the true drug action for almost all values of N .

For the second data collection strategy, the spacing of the datapoints was dynamically chosen based on the rate of change of cell populations, so that there are proportionally more points at higher-slope areas compared to low-slope areas. This was done by first solving each ODE with many timesteps (time-step size 0.001), and then filtering the resulting data. The settings for the filtering were such that 1 point was present for each cell decline of 0.05, and no more than 500 contiguous points were removed. This yielded datasets of 31 points for the log-kill model, and 29 points for the others. The results of running the UPINNs on these datasets can be seen in Fig 3. As evidenced by the mean-squared errors in Table 1, the learned drug actions for both the log-kill and Norton-Simon model are on the order of 10^{-4} and 10^{-5} , giving comparable or better MSEs than using spacing that is equal in time. Given that the adjusted spacing uses half as much data, but achieves errors on the same order of magnitude (10^{-4}) highlights the importance of data collection in high-slope areas. In practice, this might mean collecting data as frequently as possible when a high cell decline is to be expected, and collecting less frequently when the cell growth has reached a plateau.

Then, we test the method on noisy data, constructed by adding noise to both dynamically spaced (Fig 4) and equally spaced data (Fig 5). For each of the drug actions (log-kill, NS, and E_{max}), we find that the method has an MSE of 10^{-4} , 10^{-3} , and 10^{-5} respectively (Table 1) under noisy conditions when the spacing is equal in time, but not when the spacing is adjusted dynamically. When the spacing is adjusted dynamically then the errors are an order of magnitude higher (10^{-3} , 10^{-2} , and 10^{-4} respectively). The noise is added proportionally to the mean of the data, with the noise level being 0.03. The predicted drug action is visually quite a bit different from the true $G(N)$ for larger values of N when the spacing is adjusted. More investigation is needed into why the adjusted spacing has a detrimental effect when the data is noisy. In addition, it is worth noting that adding only noise to equally spaced data has a negligible effect on the MSE of all drug actions. The MSEs remain at most 10^{-4} for both noisy and noiseless equispaced data. This highlights the robustness of the method. A comparison of the MSE values of the learned drug actions for all three types of tests can be seen in Table 1, with the lowest values for each column bolded.

Finally, we apply a two-step process where first β is learned using a standard PINN from both noiseless and noisy data, and then this β estimate is employed during the estimation of $G(N)$. This showcases a way to apply UPINNs where no prior knowledge of the parameters or drug action is needed. The results of this test can be seen in Fig 6. Fig 6 (a) is the noiseless data, created using the same parameters as before, using the Norton-Simon model. For noiseless data, we obtain an estimate of $\beta = 0.999$ from the untreated data, which yields an MSE of 1×10^{-6} compared to the true $\beta = 1$,

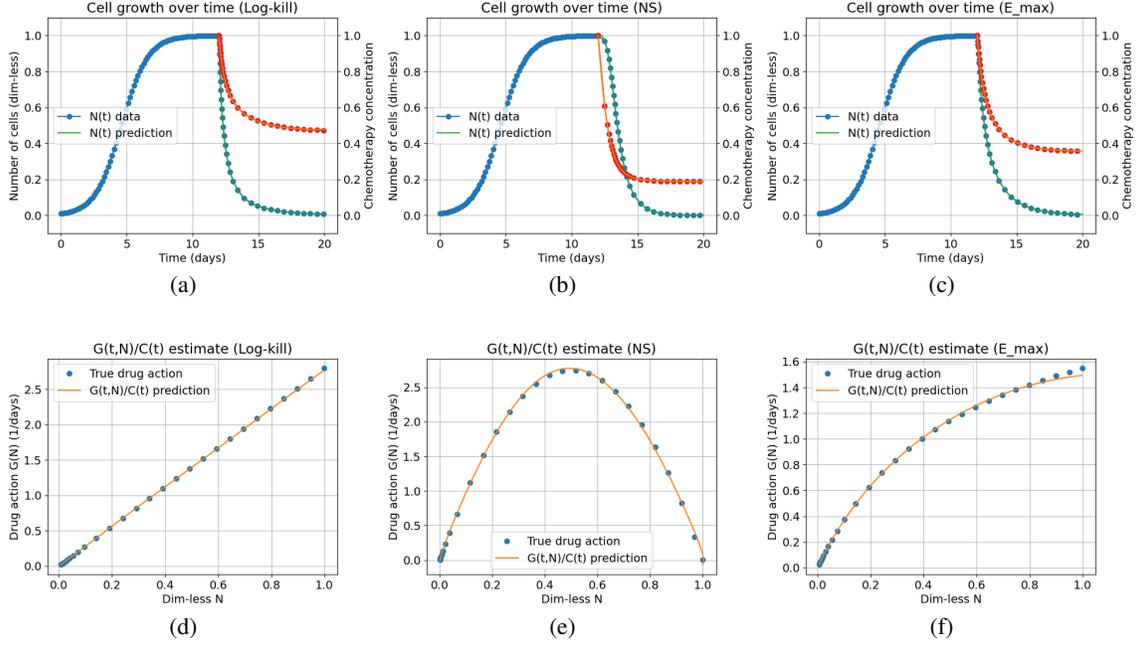


Figure 3: Datasets for the different drug actions (a, b, c) with their respective drug actions learned below them (d, e, f). The data is noiseless but collected such that there is at one datapoint for each 0.05-interval decrease in N . Although there are half as many points collected than in the equispaced case, the MSE between the true drug action and the predicted drug action is still on the order of 10^{-4} as shown by Table 1

Drug action	Log-kill	Norton-Simon	E_{max}
Type of data			
Equal spacing	1.00×10^{-4}	4.13×10^{-4}	1.85×10^{-4}
Adjusted spacing	6.30×10^{-5}	3.84×10^{-4}	2.00×10^{-4}
Equal spacing + noise	1.66×10^{-4}	4.17×10^{-3}	7.32×10^{-5}
Adjusted spacing + noise	4.29×10^{-3}	1.22×10^{-2}	1.96×10^{-4}

Table 1: Mean-squared error of the drug action predictions (compared to the ground truth drug action) for different types of data. Equal spacing means that the data was equally spaced in time (one datapoint per 0.15 days). Adjusted spacing means that the data was collected with proportionally more datapoints when the cell decline has a high rate. When noise is added, the noise level is 0.03, added proportionally to the mean of the data. The lowest value in each column is bolded.

with subsequent MSE of the drug action being 1.3×10^{-4} . For noisy data, we obtain an estimate of $\beta = 0.997$ (MSE of 9×10^{-6}), with subsequent MSE of the drug action being 4.5×10^{-3} . The performance is clearly worse when the data is noisy, but a visual inspection shows that we can recover the curve and β well for both noisy and noiseless data.

Overall, we show that the UPINN method applied to the discovery of chemotherapeutic drug actions is effective in recovering the drug action in three common cases. It recovers the drug action with a low MSE (10^{-3}) in cases of both sparse and noisy data, and can be combined with the standard PINN approach to yield MSEs on the same order of magnitude.

4.2 Inference and interpolation of dose-dependent logistic growth parameters

In this set of synthetic experiments, we show that we attain low MSE (10^{-3} to 10^{-8}) in recovering the true proliferation rate $k_p(D)$ and carrying capacity $\theta(D)$ for known dosages, and interpolate between the dosages. At the low dosage,

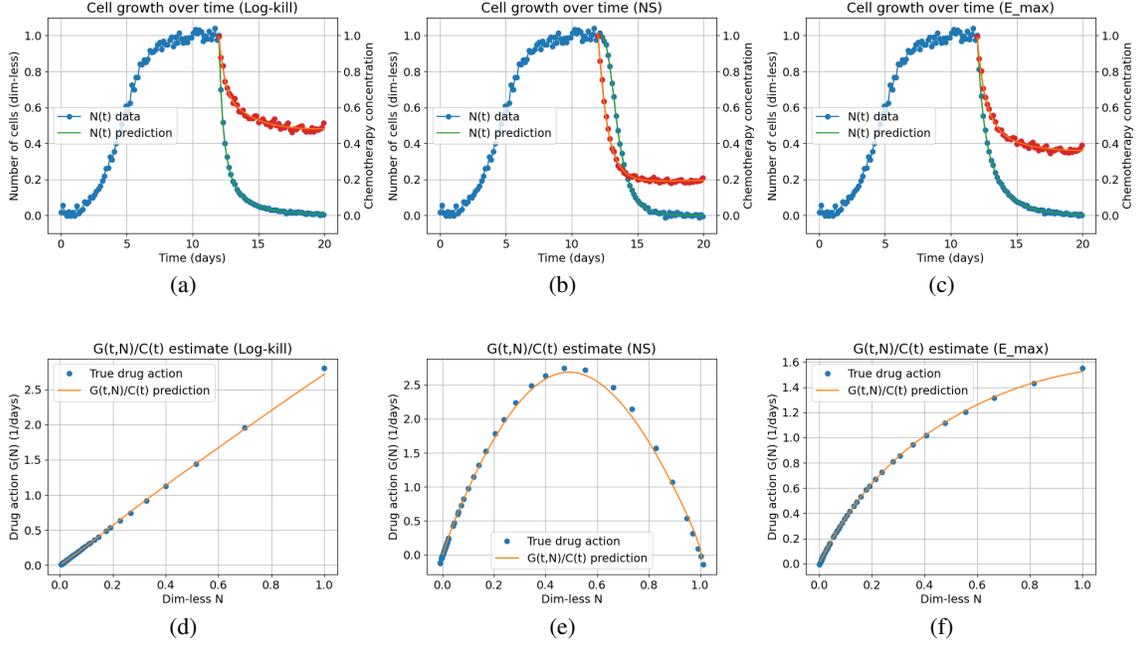


Figure 4: Datasets for the different drug actions (a, b, c) with their respective drug actions learned below them (c, d, f). The data has a noise level of 0.03 added proportionally to the mean of the variables.

the growth rate and carrying capacity are set to be the highest ($k_p = 0.03, 0.08 \text{ hr}^{-1}$, $\theta = 2.0, 4.0$), and at the highest dosage they are the lowest ($k_p = 0.01, 0.06 \text{ hr}^{-1}$, $\theta = 1.0, 3.0$). The set of values chosen for k_p reflects either low growth rates (0.01-0.03), high growth rates overall (0.06-0.08), or a large range of growth rates (0.01-0.08). Similar parameter choices were made for θ , where low growth rates correspond to θ ranges 1.0-2.0, large range of growth rates correspond to ranges 1.0-4.0, and high growth rates overall correspond to θ between 3.0-4.0. Table 2 and 3 show the accuracy in recovering the true parameters and fitting the data for both noisy and noiseless data respectively. The true parameters are recovered on the order of 10^{-3} to 10^{-6} even in cases of sparse and noisy data. The specific values of these parameters were selected to be similar to those fit by [15]. The resulting surrogate solution MSE for noiseless and noisy data is at most 10^{-3} .

$k_p(D)$	$\theta(D)$	MSE of θ	MSE of k_p	Model MSE
[0.03, 0.02, 0.01]	[2.0, 1.5, 1.0]	0.00012	6.3e-7	2.4e-05
[0.08, 0.07, 0.06]	[2.0, 1.5, 1.0]	3.678e-06	6.5e-05	0.0002
[0.08, 0.05, 0.01]	[2.0, 1.5, 1.0]	0.00014	0.0029	0.00154
[0.03, 0.02, 0.01]	[4.0, 3.5, 3.0]	0.000465	4.456e-08	2.8e-05
[0.08, 0.07, 0.06]	[4.0, 3.5, 3.0]	4.94e-05	6.69e-05	0.0004
[0.08, 0.05, 0.01]	[4.0, 3.5, 3.0]	0.00042	1.67e-05	0.006
[0.03, 0.02, 0.01]	[4.0, 2.5, 1.0]	0.0011	5.8e-07	1.9e-05
[0.08, 0.07, 0.06]	[4.0, 2.5, 1.0]	4.35e-05	7.01e-05	0.0002
[0.08, 0.05, 0.01]	[4.0, 2.5, 1.0]	0.0036	0.00089	0.0035

Table 2: Mean squared errors for each experiment (noiseless data). The row of θ and k_p indicates the values used for each dosage (low, medium, and high dosages respectively). Model MSE is the MSE between the data and the model.

Figure 7, (a)-(c) shows the results for row 4 of the noisy data, and Figure 7, (d)-(f) shows the results for row 7 of the noisy data. It can be seen that the interpolation is performed on a smooth continuous curve, the surrogate solution

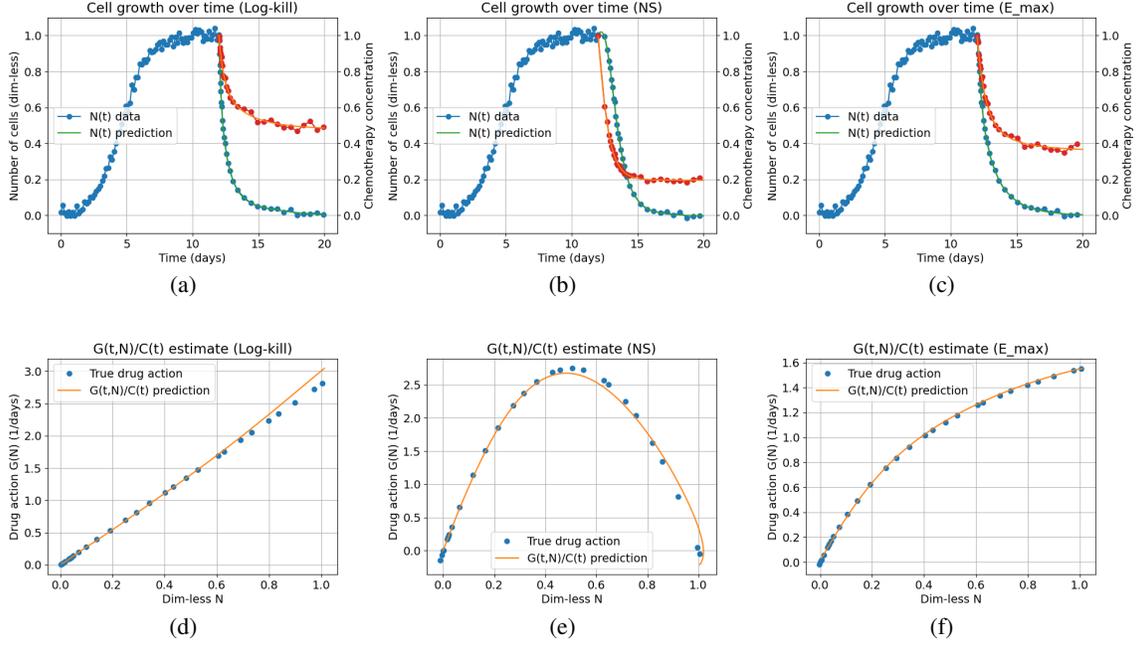


Figure 5: Datasets for the different drug actions (a, b, c) with their respective drug actions learned below them (c, d, f). The data is spaced so that times of high cell decline have proportionally more observations (one datapoint for each 0.05-interval decrease in N). It also has a noise level of 0.03 added proportionally to the mean of the variables.

$k_p(D)$	$\theta(D)$	MSE of θ	MSE of k_p	Model MSE
[0.03, 0.02, 0.01]	[2.0, 1.5, 1.0]	0.0006	3.72e-06	0.001
[0.08, 0.07, 0.06]	[2.0, 1.5, 1.0]	0.0001	7.97e-05	0.0017
[0.08, 0.05, 0.01]	[2.0, 1.5, 1.0]	0.00024	0.0029	0.0015
[0.03, 0.02, 0.01]	[4.0, 3.5, 3.0]	0.0029	9.0e-07	0.0044
[0.08, 0.07, 0.06]	[4.0, 3.5, 3.0]	0.00016	6.6e-05	0.0086
[0.08, 0.05, 0.01]	[4.0, 3.5, 3.0]	0.025	0.00094	0.0095
[0.03, 0.02, 0.01]	[4.0, 2.5, 1.0]	0.0011	2.7e-07	0.0033
[0.08, 0.07, 0.06]	[4.0, 2.5, 1.0]	0.00023	7.1e-05	0.0055
[0.08, 0.05, 0.01]	[4.0, 2.5, 1.0]	0.0058	0.0013	0.0092

Table 3: Mean squared errors for each experiment (noisy data). The row of θ and k_p indicates the values used for each dosage (low, medium, and high dosages respectively). The noisy data was created using a noise level of 0.03. Model MSE is the MSE between the data and the model.

is correct for each dosage, and the resulting equations, when solved using the estimated parameters, fit the data very well visually. In addition, the model MSE (the MSE between the data and the model) is on the order of 10^{-3} . It is worth noting that sometimes, the surrogate solution returns and the fit implied by the parameters does not agree. The surrogate solution may agree with the data but the inferred parameters, when substituted into the differential equation, do not show an accurate fit. This may happen because the PINN loss is not sufficiently minimized compared to the MSE loss. For this reason, for this set of experiments, we perform both checks to ensure that the results are reasonable. Overall these results show that the UPINN method may provide an alternative to fitting multiple PINNs, with the added capability of interpolating between observations.

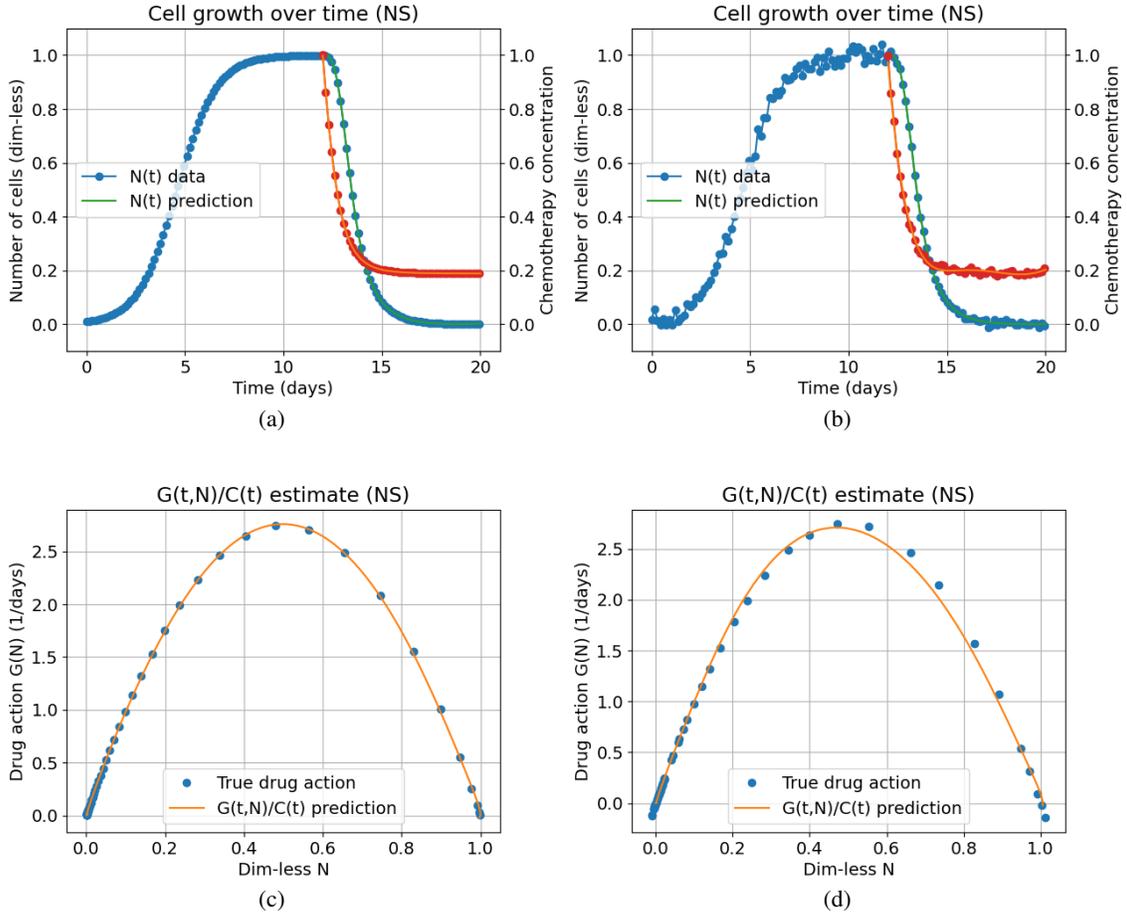


Figure 6: (a) Equispaced noiseless data with Norton-Simon drug action, (b) Learned drug action after the parameter β was fit from the first 12 days of noiseless data (final estimate: 0.999, MSE 1×10^{-6}), (c) Equispaced noisy (0.03) data with Norton-Simon drug action, (d) Learned drug action after the parameter β was fit from the first 12 days of noisy data (final estimate: 0.997, MSE 9×10^{-6})

4.3 Learning doxorubicin dynamics

In the next set of experiments, we aim to learn doxorubicin dynamics from in-vitro data, as gathered and modelled by [15]. Due to the absence of the ground truth net proliferation rate for in-vitro experimental data, we first validate the method using simulated data. We use [15]’s ODE to generate data from eq. 6 and one of 7 and 8. We then learn $F_D(N, t)$ and $G_D(t)$ as per the methods for eq. 8, and we learn $F_D(N, t)$ from data generated by eq. 7. Subsequently, we learn $F_D(N, t)$ and $G_D(t)$ from the in-vitro SUM-149 data.

For our in-silico analysis, we generated time series data $\{t_i, n_i\}$ (number of tumour cells over time) either via equation 7 or 8, for different sets of realistic parameters. Since $k_{d,A}(D)$, $k_{d,B}(D)$, $\theta(D)$, $r(D)$ take on scalar real values for a given dosage, it suffices to generate a dataset with one parameter combination for each “dosage”. We generated several such datasets. By comparing it to the ground truth net proliferation rate, we are able to validate the performance of the model. Parameter values for these data were chosen to be similar to the values obtained via fitting by [15]. k_p was selected to be 0.0354 hr^{-1} . This value was obtained by fitting a standard continuous PINN [23] to the control data provided by the authors of [15]. Finally, we added noise to the synthetic data, as per synthetic experiments in [18] at a noise level of 0.03. The data generated for each dosage was normalized such that both the time and cell counts range from 0 to 1. In other words, the carrying capacity information and the timescale information is lost at this stage, but can be recovered later after $F_D(N, t)$ and $G_D(t)$ are fit, by undoing the scaling. This initial scaling improves the performance of the UPINN method.

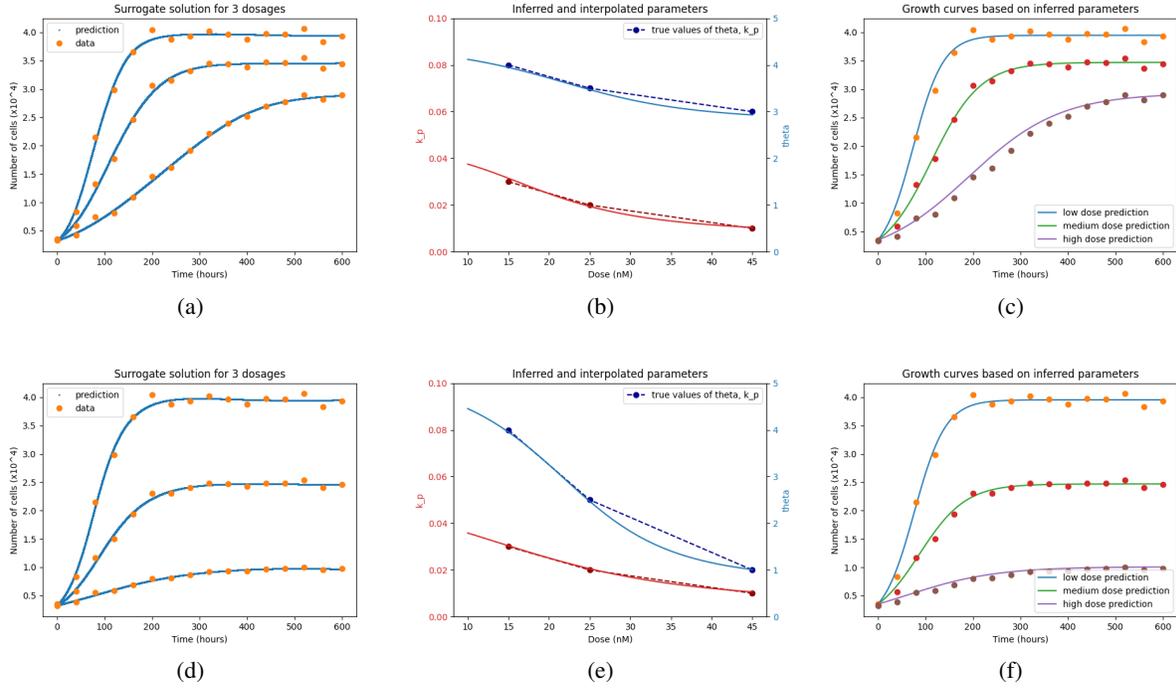


Figure 7: (a) Surrogate solution for row 4 of Table 3. (b) Fit and interpolated parameters per dosage. Blue is θ , and red is k_p . (c) Data was simulated using the inferred parameters for each of the three dosages. (d)-(f) Same as (a)-(c) but for row 7 of Table 3.

First, data from eq. 6 combined with 7 was generated, utilizing different parameter values to create datasets for different “dosages”. Table 4 summarizes the different parameters and noise levels tested. For the same dataset, the UPINN was used to fit $F_D(N, t)$ from eq 9 5 times, and the best, mean and standard deviation MSE is reported. The MSE is computed between the predicted cell count and the actual cell count. The predicted cell count was obtained by substituting the learned $F_D(N, t)$ into eq 9 and solving the equation numerically. Figure 8 shows a particular fit of $F_D(N, t)$ using data generated from $\theta = 1.0$, $k_{d,A} = 0.03 \text{ hr}^{-1}$. The method performs well in identifying $F_D(N, t)$ for different realistic values of the parameters, as evidenced by the MSE model error being at most 10^{-4} . Given that the standard deviation of the fits is on the order of 10^{-5} and 10^{-6} , we can conclude that for this equation, the UPINN method produces reproducible fits.

θ	$k_{d,A}$	noise	Best solution fit MSE	Mean MSE	St. Dev. of MSE
1.0	0.05	0.0	1.11e-05	4.17e-05	2.31e-05
1.0	0.03	0.0	6.03e-06	1.42e-05	1.48e-05
1.0	0.01	0.0	2.20e-07	3.24e-06	2.99e-06
1.0	0.05	0.03	5.83e-04	5.98e-04	2.41e-05
1.0	0.03	0.03	5.02e-04	5.11e-04	1.30e-05
1.0	0.01	0.03	2.15e-04	2.17e-04	1.71e-06

Table 4: Fit of $F_D(N, t)$ using data generated via eq. 7, with MSE computed between the inferred solution (using the learned hidden term) and the data. Each experiment was run 5 times; the error of the best fit is shown, along with the mean and standard deviation of all 5 runs.

Next, data from eq. 6 combined with 8 was generated. In this case, the net proliferation rate is time-dependent, whereas in eq. 7 it is constant. Table 5 summarizes the MSE of the learned $F_D(N, t)$ when the data was generated according to

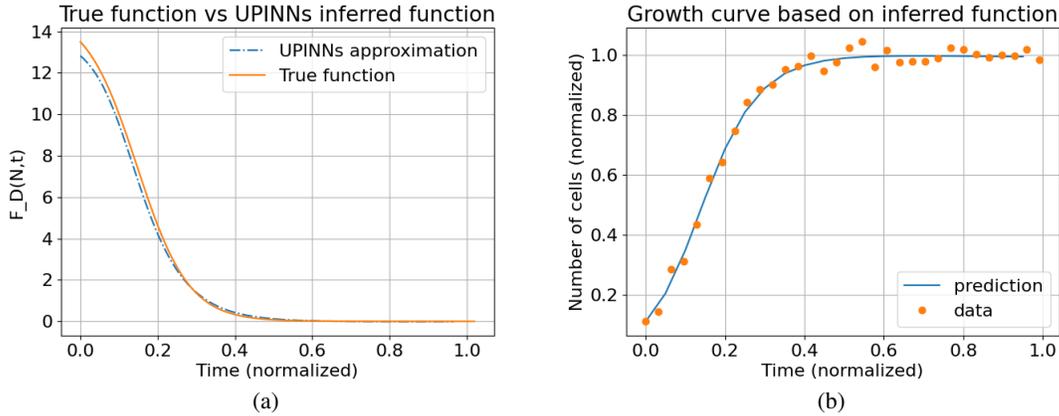


Figure 8: (a) Fit of $F_D(N, t)$ to data from eq 7 with parameters $k_{d,A} = 0.03$, $k_p = 0.0354$ (b) ODE solution generated using the inferred $F_D(N, t)$, along with the data for comparison. F_D is recovered well using the UPINN method, and the resulting model using the recovered F_D matches the data well.

eq 8. We can see that the model MSE is on the order of at most 10^{-4} for most parameter combinations. In this case, several different $(r_d, k_{d,B})$ combinations are chosen, and the best runs of 5 are chosen similarly. Since the standard deviation is at most 10^{-3} (unless one of the runs completely fails to learn the underlying function), we do not anticipate that adding more runs would change these MSEs significantly. The values chosen for these parameters were informed in part by [15] such that a variety of realistic growth curves were generated. Carrying capacity θ is set to 1 to avoid the additional step of rescaling the cell counts. Table 5 reports the results for both noisy and noiseless data. It can be seen that the very first row had a run that has an error on the order of 10^1 , but the MSE for the best model run is still 10^{-5} . This means that for this particular parameter combination, many of the model fits were not accurate, but due to the stochasticity of the neural network fitting process, it is still possible to obtain a model with MSE less than 10^{-3} within 5 trials. Figure 9 (a), (b) shows a fit of $F_D(N, t)$ to row 2 of Table 5. The solution shows a time-dependent net proliferation rate, which is learned with a model MSE of 10^{-6} . We also demonstrate that $G_D(t)$ can be learned well from equation (8). Figure 9 (c), (d) shows the fit of the same row 2, with parameters $r_d = 0.017$ and $k_{d,B} = 0.05$, but this time fitting $G_D(t)$. In this case, the solution matches the ground truth very closely except for $t \in [0.6, 1.0]$. At this point, the predictions start to diverge. This is likely because the function ceases to be identifiable in this time interval.

r_d	θ	$k_{d,B}$	noise	Best solution MSE	Mean MSE	St. Dev. of MSE
0.017	1.0	0.05	0.0	1.83e-05	23.0	46.0
0.017	1.0	0.03	0.0	4.38e-06	8.06e-05	6.14e-05
0.017	1.0	0.01	0.0	1.35e-05	2.59e-05	9.87e-06
0.017	1.0	0.05	0.03	2.28e-04	1.03e-02	6.56e-03
0.017	1.0	0.03	0.03	4.37e-04	4.99e-04	7.25e-05
0.017	1.0	0.01	0.03	5.00e-04	5.11e-04	8.82e-06

Table 5: $F_D(N, t)$ (eq 9) using data generated via eq. 8, with MSE computed between the inferred solution (using the learned hidden term) and the data. Each experiment was run 5 times; the error of the best fit is shown, along with the mean and standard deviation of all 5 runs.

Since the UPINN method is able to accurately recover the hidden terms $F_D(N, t)$ and $G_D(t)$ for both model 7 and 8, we apply the UPINN method to learn $F_D(N, t)$ and $G_D(t)$ for in-vitro data, where no ground truth is available. Figure 10 shows the learned time-dependent term $F_D(N, t)$ for the application of the doxorubicin chemotherapeutic to SUM-149 cells, at 312nM concentration, for an exposure time of 6hrs. The cells were then allowed to grow undisturbed. This curve is very similar in shape to the one shown in Fig. 9 (a). The peak occurs at approximately the same height as well. Figure 10 (c), (d) shows the time-dependent net proliferation, $G_D(t)$, learned for the same in-vitro dataset. We can see that this curve is not possible to construct using equation 7 or 8 alone, or even with a weighted average of these

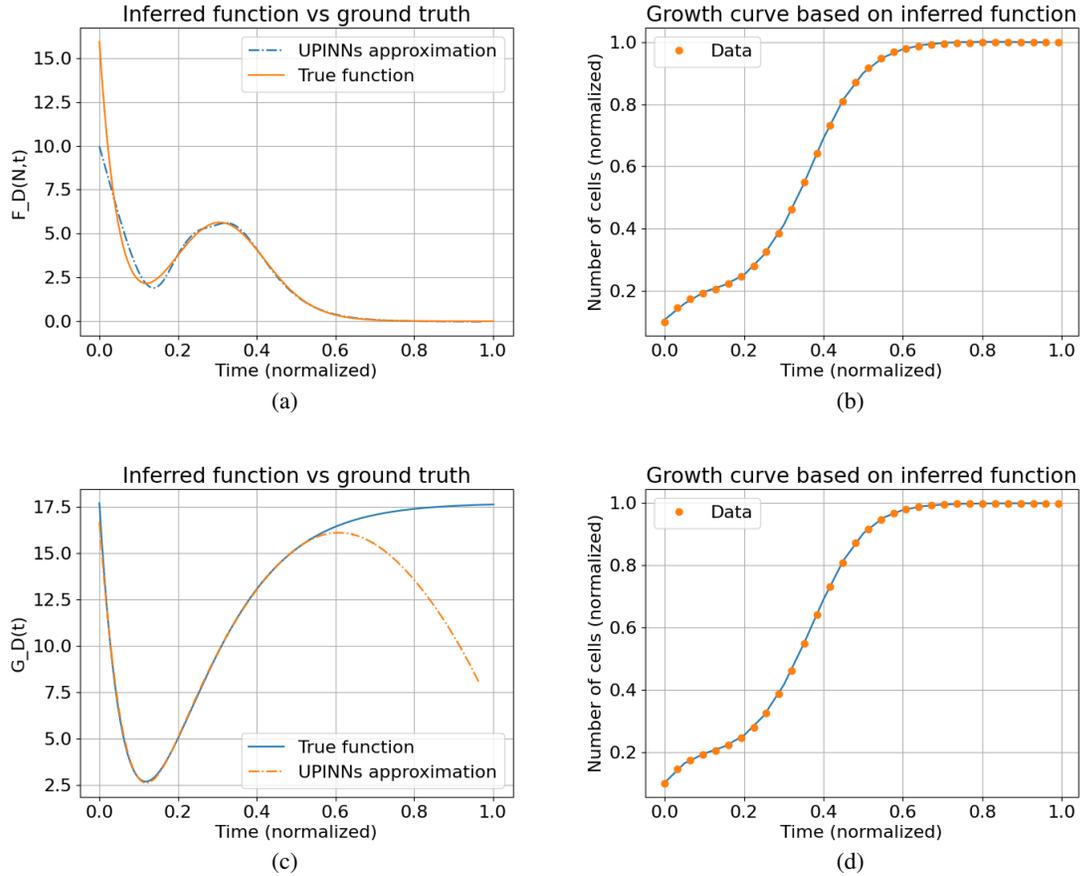


Figure 9: **(a)** Fit of $F_D(N, t)$ (eq (9)) to data from row 2 of Table 5 (eq. 8, parameters $r = 0.017, k_{d,B} = 0.03$) **(b)** ODE solution generated using the inferred $F_D(N, t)$, along with the data for comparison **(c), (d)** Same as (a), (b) but $G_D(t)$ is fit. F_D in (a) is recovered with a solution MSE of 10^{-6} , but G_D in (c) appears to not be fully identifiable from the data.

equations. This is because the function shown is very flat up until $t = 0.4$, but the derivative increases after this point. By contrast, a weighted average of 7 and 8 could not have a derivative of zero for any finite time interval given their formulations. Hence, fitting the UPINN to find $G_D(t)$ provides insights into the true net proliferation rate as a function of time. Initially, the net proliferation rate behaves more like 7, and subsequently more like 8. Table 6 shows the MSE between the solution using the learned function $F_D(N, t)$ and the data. The inferred solution fits the data well in most cases, as evidenced by the MSE between the inferred solution and the data being on the order of 10^{-4} for over half of the time-series datasets.

5 Discussion

In this work, we apply the UPINN method to learn the chemotherapy drug action in several different ODEs, applying the method to both synthetic and in-vitro experimental data. Rather than making assumptions about the drug action, we can learn it from data in order to identify how to best model the effect of the chemotherapeutic on cells over time. In addition, this learned drug action may allow us to learn more about the underlying biological mechanism. Finally, the learned drug action can then be used for downstream tasks such as drug treatment schedule optimization. Current limitations of this method is a lack of uncertainty quantification, which is especially important given the method’s stochastic nature and potential high-risk downstream applications, lack of integration with model identifiability workflows, and minimal quantification of differential equation setups where it is unlikely to perform well.

General future work in this direction would involve understanding the conditions under which this method performs well and when it is not able to learn the drug action effectively (e.g. noise level, shape and properties of the ground

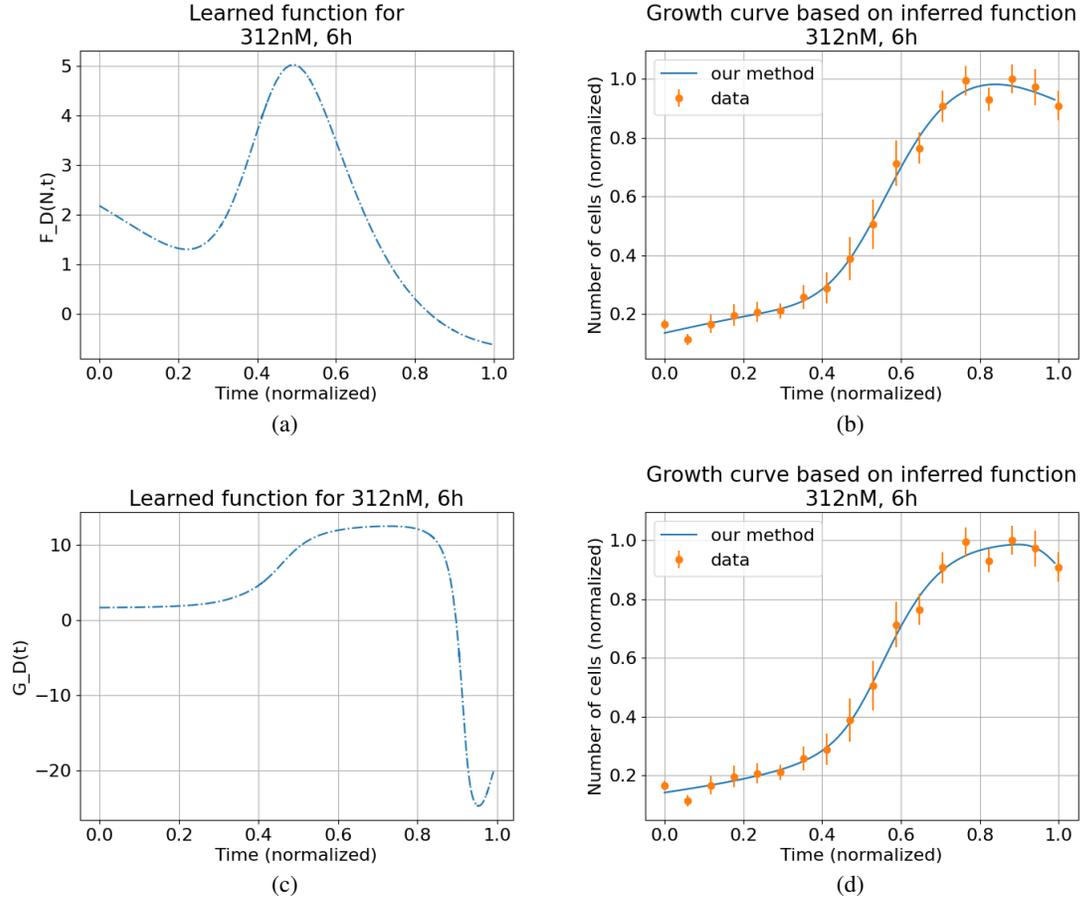


Figure 10: **(a)** Fit of $F_D(N, t)$ (eq (9)) to experimental data: 312nM, 6h exposure **(b)** ODE solution generated using the inferred $F_D(N, t)$, along with the data for comparison **(c)**, **(d)** Same as (a), (b) but $G_D(t)$ (eq (10)) is fit. The overall curve of (a) is similar in shape visually to Fig. 9 (a), indicating that the growth is captured to some extent by Model (6). Additionally, (c) shows a plateau in the function before an increase in cell killing, which cannot be captured by a linear combination of (7) and (8).

truth function). If there is more than one function to be learned, understanding the identifiability of the functions would provide more guidance on whether the method can successfully recover the ground truth. This could be a mathematical analysis similar to parameter identifiability, or it could be a more nuanced quantification of the uncertainty of the output of the neural networks, taking into account the abundance of data. Uncertainty quantification has been proposed for PINNs [29], but it would be especially helpful to quantify the uncertainty for UPINNs. Additionally, it would be beneficial if the uncertainty quantification method has validity guarantees, such as conformal prediction [2]. Finally, applying the method to a drug with a known model of drug induced death (e.g. Norton-Simon) would further validate the method’s performance. As an additional step after the functions have been learned, symbolic regression could be performed to find a closed form of the function.

As an extension to learning the dynamics of doxorubicin, the UPINN method can be easily adapted to learn the unknown net proliferation rate $G(t, D)$ as a function of dosage and time (and possibly cells) simultaneously. This would involve treating the number of cells as a function of both dosage and time as well. However, there are several considerations: due to the limited dosage data (only 9 measurements), interpolation between these measurements may not yield good results. However, it should be possible to fit $G(t, D)$ correctly for the available dosages. Secondly, there may be identifiability issues and there may not be a single unique G which satisfies the equation.

6 Conclusion

In this paper, we integrate machine learning in the form of the Universal Physics-Informed Neural Network (UPINN) method with QSP models in order to learn the drug action of chemotherapeutics accurately and with minimal computational expenses. We showcase the ability of the method to identify three different well-known drug actions, the Log-kill model, Norton-Simon, and E_{max} . The learned drug actions match the ground truth very well. This method can also be used to infer many parameter sets simultaneously rather than running a separate fitting procedure for each dataset. In addition, the method can interpolate between fitted parameters. Finally, we employ the method to learn the drug action of doxorubicin from time-series data.

7 Acknowledgements

We thank Matthew McKenna (Vanderbilt University), Thomas Yankeelov (The University of Texas at Austin), and Ernesto Lima (The University of Texas at Austin) for sharing their data and for informative discussions and comments on the first draft of the manuscript. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

- [1] RJ Allen, Theodore R Rieger, and Cynthia J Musante. Efficient generation and selection of virtual populations in quantitative systems pharmacology models. *CPT: pharmacometrics & systems pharmacology*, 5(3):140–146, 2016.
- [2] Anastasios N Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.
- [3] Britta Basse, Bruce C Baguley, Elaine S Marshall, Wayne R Joseph, Bruce van Brunt, Graeme Wake, and David JN Wall. Modelling cell death in human tumour cell lines exposed to the anticancer drug paclitaxel. *Journal of Mathematical Biology*, 49:329–357, 2004.
- [4] Larisse Bolton, Alain HJJ Clout, Schalk W Schoombie, and Jacobus P Slabbert. A proposed fractional-order gomPERTZ model and its application to tumour growth data. *Mathematical medicine and biology: a journal of the IMA*, 32(2):187–209, 2015.
- [5] Diogo M Camacho, Katherine M Collins, Rani K Powers, James C Costello, and James J Collins. Next-generation machine learning for biological networks. *Cell*, 173(7):1581–1592, 2018.
- [6] Abdallah Derbalah, Hesham Al-Sallami, Chihiro Hasegawa, Abhishek Gulati, and Stephen B Duffull. A framework for simplification of quantitative systems pharmacology models in clinical pharmacology. *British Journal of Clinical Pharmacology*, 88(4):1430–1440, 2022.
- [7] Dániel András Drexler, Tamás Ferenci, András Füredi, Gergely Szakács, and Levente Kovács. Experimental data-driven tumor modeling for chemotherapy. *IFAC-PapersOnLine*, 53(2):16245–16250, 2020.
- [8] Brydon Eastman, Michelle Przedborski, and Mohammad Kohandel. Reinforcement learning derived chemotherapeutic schedules for robust patient-specific therapy. *Scientific Reports*, 11(1):17882, 2021.
- [9] Federica Eduati, Patricia Jaaks, Jessica Wappler, Thorsten Cramer, Christoph A Merten, Mathew J Garnett, and Julio Saez-Rodriguez. Patient-specific logic models of signaling pathways from screenings on cancer biopsies to prioritize personalized combination therapies. *Molecular systems biology*, 16(2):e8664, 2020.
- [10] Ameya D Jagtap and George Em Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5), 2020.
- [11] Angela M Jarrett, Alay Shah, Meghan J Bloom, Matthew T McKenna, David A Hormuth, Thomas E Yankeelov, and Anna G Sorace. Experimentally-driven mathematical modeling to improve combination targeted and cytotoxic therapy for her2+ breast cancer. *Scientific reports*, 9(1):12830, 2019.
- [12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [13] M Kohandel, S Sivaloganathan, and A Oza. Mathematical modeling of ovarian cancer treatments: sequencing of surgery and chemotherapy. *Journal of theoretical biology*, 242(1):62–68, 2006.
- [14] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

Duration	Concentration	Best MSE	Mean MSE	St. Dev. of MSE
6h	10nM	0.00045	0.0022	0.0036
6h	20nM	0.00031	0.0011	0.0013
6h	39nM	7.57e-05	0.024	0.062
6h	78nM	0.00050	0.0013	0.00054
6h	156nM	0.00082	0.00084	8.04e-06
6h	312nM	0.00019	0.00086	0.00049
6h	625nM	0.00061	0.010	0.016
6h	1250nM	0.00038	0.0072	0.0065
6h	2500nM	0.0047	0.0062	0.00097
12h	10nM	0.00022	0.0011	0.0015
12h	20nM	0.00066	0.0021	0.0015
12h	39nM	0.00056	0.0011	0.00020
12h	78nM	0.00084	0.00086	7.50e-06
12h	156nM	0.00079	0.0021	0.0032
12h	312nM	0.00011	0.021	0.036
12h	625nM	0.00043	0.012	0.033
12h	1250nM	0.00086	0.0076	0.0031
12h	2500nM	0.0014	0.011	0.0098
24h	10nM	0.00050	0.00065	0.00013
24h	20nM	0.00067	0.0014	0.00097
24h	39nM	0.00092	0.0011	0.00010
24h	78nM	0.00043	0.00098	0.00024
24h	156nM	0.00062	0.011	0.019
24h	312nM	0.00031	0.0023	0.0040
24h	625nM	0.00033	0.0063	0.0038
24h	1250nM	0.0048	0.0048	6.66e-05
24h	2500nM	0.0045	0.0078	0.0013

Table 6: Fits of $F_D(N, t)$ to in-vitro SUM-149 time-series data from [15]. MSE is computed between the inferred solution (using the learned hidden term) and the in-vitro data. Each experiment was run 10 times; the error of the best run is shown, along with the mean and standard deviation of all 10 runs.

- [15] Matthew T McKenna, Jared A Weis, Stephanie L Barnes, Darren R Tyson, Michael I Miga, Vito Quaranta, and Thomas E Yankeelov. A predictive mathematical modeling approach for the study of doxorubicin treatment in triple negative breast cancer. *Scientific reports*, 7(1):5725, 2017.
- [16] John Carl Panetta and K Renee Fister. Optimal control applied to competing chemotherapeutic cell-kill strategies. *SIAM Journal on Applied Mathematics*, 63(6):1954–1971, 2003.
- [17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

- [18] Lena Podina, Brydon Eastman, and Mohammad Kohandel. A pinn approach to symbolic differential operator discovery with sparse data. *arXiv preprint arXiv:2212.04630*, 2022.
- [19] Sebastian Polak, Zofia Tylutki, Mark Holbrook, and Barbara Wiśniowska. Better prediction of the local concentration–effect relationship: the role of physiologically based pharmacokinetics and quantitative systems pharmacology and toxicology in the evolution of model-informed drug discovery and development. *Drug discovery today*, 24(7):1344–1354, 2019.
- [20] Michelle Przedborski, Munisha Smalley, Saravanan Thiyagarajan, Aaron Goldman, and Mohammad Kohandel. Systems biology informed neural networks (sbinn) predict response and novel combinations for pd-1 checkpoint blockade. *Communications Biology*, 4(1):877, 2021.
- [21] Bhanwar Lal Puniya, Rada Amin, Bailee Lichter, Robert Moore, Alex Ciurej, Sydney J Bennett, Ab Rauf Shah, Matteo Barberis, and Tomáš Helikar. Integrative computational approach identifies drug targets in cd4+ t-cell-mediated immune disorders. *NPJ systems biology and applications*, 7(1):4, 2021.
- [22] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.
- [23] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.
- [24] Peter K Sorger, Sandra RB Allerheiligen, Darrell R Abernethy, Russ B Altman, Kim LR Brouwer, Andrea Califano, David Z D’Argenio, Ravi Iyengar, William J Jusko, Richard Lalonde, et al. Quantitative and systems pharmacology in the post-genomic era: new approaches to discovering drugs and understanding therapeutic mechanisms. In *An NIH white paper by the QSP workshop group*, volume 48, pages 1–47. NIH Bethesda Bethesda, MD, 2011.
- [25] Tiffany A Traina, Ute Dugan, Brian Higgins, Kenneth Kolinsky, Maria Theodoulou, Clifford A Hudis, and Larry Norton. Optimizing chemotherapy dose and schedule by norton-simon mathematical modeling. *Breast disease*, 31(1):7–18, 2010.
- [26] Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.
- [27] Yaning Wang, Hao Zhu, Rajanikanth Madabushi, Qi Liu, Shiew-Mei Huang, and Issam Zineh. Model-informed drug development: current us regulatory practice and future considerations. *Clinical Pharmacology & Therapeutics*, 105(4):899–911, 2019.
- [28] Chi Heem Wong, Kien Wei Siah, and Andrew W Lo. Estimation of clinical trial success rates and related parameters. *Biostatistics*, 20(2):273–286, 2019.
- [29] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *Journal of Computational Physics*, 425:109913, 2021.
- [30] Tongli Zhang, Ioannis P Androulakis, Peter Bonate, Limei Cheng, Tomáš Helikar, Jaimit Parikh, Christopher Rackauckas, Kalyanasundaram Subramanian, and Carolyn R Cho. Two heads are better than one: current landscape of integrating qsp and machine learning: an isop qsp sig white paper by the working group on the integration of quantitative systems pharmacology and machine learning. *Journal of Pharmacokinetics and Pharmacodynamics*, 49(1):5–18, 2022.