Unravelling cell fate decisions through single cell methods and mathematical models

Mircea, M.

6

Gene regulatory network inference with physics informed neural networks

One of the main goals of developmental biology is to reveal the gene regulatory networks (GRNs) underlying the robust differentiation of multipotent progenitors into precisely specified cell types. Most existing methods to infer GRNs from experimental data have limited predictive power as the inferred GRNs merely reflect gene expression similarity or correlation. Here, we demonstrate, how physics-informed neural networks (PINNs) can be used to infer the parameters of GRNs that provide mechanistic understanding of biological processes. Specifically we study GRNs that exhibit bifurcation behavior and can therefore model cell differentiation. We show that PINNs outperform regular feed-forward neural networks on the parameter inference task and analyze two relevant experimental scenarios: 1. a system with cell communication for which gene expression trajectories are available and 2. snapshot measurements of a cell population in which cell communication is absent. Our analysis will inform the design of future experiments to be analyzed with PINNs and provides a starting point to explore this powerful class of neural network models further.

Maria Mircea, Diego Garlaschelli and Stefan Semrau. Gene regulatory network inference with physics informed neural networks. Manuscript in preparation.

6.1 Introduction

Since the advent of single-cell molecular profiling, developmental biology has been inundated with high-dimensional data we are still learning to make sense of. Various machine learning methods have been used to find patterns in single-cell data, such as cell types or differentiation paths [1, 2]. Notwithstanding the great success of these methods, it remains difficult to infer mechanistic insights or quantitative, predictive models from single-cell data. Yet, one of the main goals of developmental biology is to understand the gene regulatory mechanisms underlying the robust differentiation of precisely defined cell types from multipotent progenitors [3] (see Chapter 1).

A common approach to the predictive mathematical modeling of differentiation uses the framework of dynamical systems theory [4–7]. In the context of differentiation, the dynamical system governs the abundance of gene products in the cell and stable attractor states are interpreted as cell types. Under certain conditions, the system can be represented by a quasi-potential [8]. This potential is the mathematical equivalent of Waddington's landscape [9], a seminal, qualitative model of differentiation in which the valleys in the landscape correspond to different cell types. In most models, the dynamical system has the structure of a network, in which the nodes are gene products, typically transcription factors, and the edges indicate interactions between them. Ideally, such a gene regulatory network (GRN) model should be able to predict the outcome of a differentiation process, given the initial cell state and external cues. Simulations using small GRNs with 2 – 5 nodes indeed exhibit bifurcations resembling actual differentiation processes [10–14].

As the parameter space grows quickly with the the size of a GRN, it can be tedious to find regimes with relevant behavior. A large body of work has therefore been devoted to inferring GRNs from measurements, typically transcriptomics or proteomics data sets. Most recently, single-cell data has been leveraged to that end [15]. Many inference methods use measures of similarity or correlation between genes and prior biological knowledge, most often about protein-protein binding affinities or the targets of transcription factors [16–19]. These methods can infer the existence of correlative or even causal relationships, especially if chromatin accessibility is taken into consideration [20]. However, they are typically unable to infer interaction strengths and are thereby lacking in predictive power. In fact, if only single-cell snapshot data is used and there is no prior biological knowledge, there are fundamental limits to GRN inference [21]. One should therefore 1. use time-resolved data that ideally contains information about the trajectories of individual cells and 2. constrain the inference problem with assumptions about the GRN. Seminal work using a Boolean network approach [22] or, more recently, catastrophe theory and approximate Bayesian computation [23], have successfully inferred predictive GRN models from time resolved data.

Another class of machine learning tools that have become extremely important in many fields are neural networks (NNs). These have been highly successful in pattern recognition and classification tasks [24] and are used extensively to interpret single-cell omics data [25, 26]. Naturally, NNs have also been used to infer GRNs from measurements [27, 28]. However, existing NN methods require GRNs obtained by other means as training data, which might limit the fidelity of the inferred GRNs. The optimal NN method would only use gene expression as training data while allowing us to implement prior knowledge or assumptions about the GRN to make the inference problem feasible. The recently developed

physics-informed neural networks (PINNs) [29–31] enable us to do just that. PINNs can solve a broad range of differential equations and also infer undetermined paramters. They have been applied successfully to various systems biology tasks [32–34].
In this chapter, we explore in how far PINNs can be used to infer GRNs. In our case, the differential equations to be solved by the PINN are defined by GRN topology and the mathematical expressions describing the interactions between the genes. Given gene expression measurements as training data, PINNs should be able to infer gene interactions. Fig. 6.1 shows a schematic of the inference procedure. Once the parameters have been learned, the dynamical system can then be used to make predictions. PINNs should thus allow us to gain mechanistic insights from measured expression data. As a proof of concept, we simulate data based on a GRN model recently introduced by Stanoev et al. [35]. At its core this GRN has two mutually inhibiting genes, u and v, which can be seen as the master transcription factors governing two alternative cell fates. This network motif has been studied intensively since it is one of the simplest motifs that exhibit bifurcations [7, 10, 14, 36]. Historically, this network motif was proposed decades ago to describe the mutual inhibition of gap genes [37]. The genetic toggle switch has been studied extensively in theory and demonstrated experimentally in organisms as simple as bacteria [38]. Kaneko et al. linked cell autonomous regulation with cell-cell interactions to show that cell division can drive differentiation and the formation of spatial patterns [39, 40]. The basic idea is the following: The idea is the following: Depending on the network parameters, a single stable attractor, interpreted as a multilineage primed (mlp) state can split into two stable attractors, which correspond to two different lineages. Such bifurcations successively create the large diversity of cell types in an adult organism, starting from the one-cell embryo [41]. In addition to the cell intrinsic mutual inhibition of the master transcription factors u and v, the model by Stanoev et al. also implements cell communication, which plays an essential role in development [42, 43] (see Chapter 3-5). In the Stanoev model, cells communicate via a diffusible signaling molecule s. This molecule is assumed to be activated by u and in turn inhibits u in both an autocrine and paracrine manner. The differential equations defining this GRN are shown in Fig. 6.2a. With this system, Stanoev et al. showed that population size can effectively serve as a control parameter that can bring the system from a homogeneous, progenitor state to a heterogeneous, differentiated state. This mechanism is an interesting alternative to the previously suggested noise-driven fate decisions [44, 45]. In certain parameter regimes, the GRN also creates regular, Turing-like spatial patterns.
In this chapter, we first explore the qualitative behavior of the GRN introduced by Stanoev et al. Subsequently we demonstrate that a naive feed-forward NN architecture and training based on simulations in a limited parameter regime are insufficient for robust GRN inference. We then explore how accurately PINNs can infer GRN parameters. Surprisingly, we find that it is not necessary to use all variables of the GRN for training. In other words, the measurement of a subset of genes across time can be sufficient for GRN inference. Lastly, we investigate a simpler system without cell communication. We study in how far GRN inference is still possible, if only snapshot data at discrete time points is available. This scenario is highly relevant as it describes the typical single-cell profiling experiment. This chapter thus provides a thorough assessment of PINNs for the purpose of GRN inference.

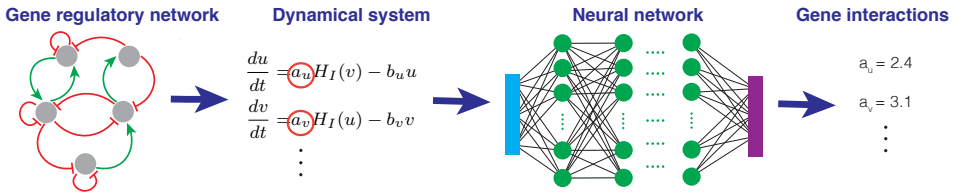

Figure 6.1: Gene regulatory network (GRN) inference with neural networks (NNs) First, a particular topology of the GRN is assumed. Together with the functional form of the interactions, the GRN topology defines a set of differential equations with undetermined parameter values. Next, a NN is trained on experimental or simulated time-series data. The parameters learned during training set the strength of interactions between genes. The fully determined dynamical system can then be used for predictions.

6.2 Results

6.2.1 Cell communication drives bifurcations in a GRN model of differentiation

We first set out to recapitulate the qualitative behavior of the GRN reported by Stanoev et al. to find interesting parameter regimes and establish a ground truth for GRN inference. We placed between 1 and 15 cells on a regular square grid and allowed cell communication between nearest neighbors, unless otherwise indicated by edges in 6.2b). The corresponding system of differential equations (shown in Fig. 6.2a) was solved by numerical integration. In the two-cell configuration, the parameter a_u, which sets the inhibition of u by v is a control parameter that can elicit a bifurcation (Fig. 6.2c). For low values of a_u there is one stable state. In this state, both cells have identical concentrations of u and v. Following Stanoev et al., we interpret a homogeneous state, where cells have identical, intermediate expression of both u and v as the mlp state. At a particular, critical value of a_u, two additional stable states appear through saddle node bifurcations. In one of these states, cell 1 has a high level of u but a low level of v, while cell 2 has a low level of u but a high level of v. In the other stable state, the expression patterns of cell 1 and 2 are reversed. These two states are thus considered differentiated states. Due to the mutual inhibition between u and v we will always find anti-correlation between the two genes, outside of the mlp state. Increasing a_u further, at a second critical point, the mlp state becomes unstable through a subcritical pitchfork bifurcation. Between a_u values of 2.33 and 2.69 the two differentiated states are the only stable states of the system. In summary, when controlled by a_u the system goes from the mlp state for low values of a_u via a small interval with three stable states (mlp, two differentiated states) to a wider interval with the two differentiated states as the only stable states. In other words, for differentiation to occur, a certain level of mutual inhibition between u and v is necessary. At even higher values of a_u additional bifurcations occur and the system returns to a single stable state. We will not explore this behavior at high a_u any further here, since it is unphysiological: In the real biological system, differentiated states are likely stabilized by other means (such as epigenetic marks), which would prevent the reversal to a single stable state.

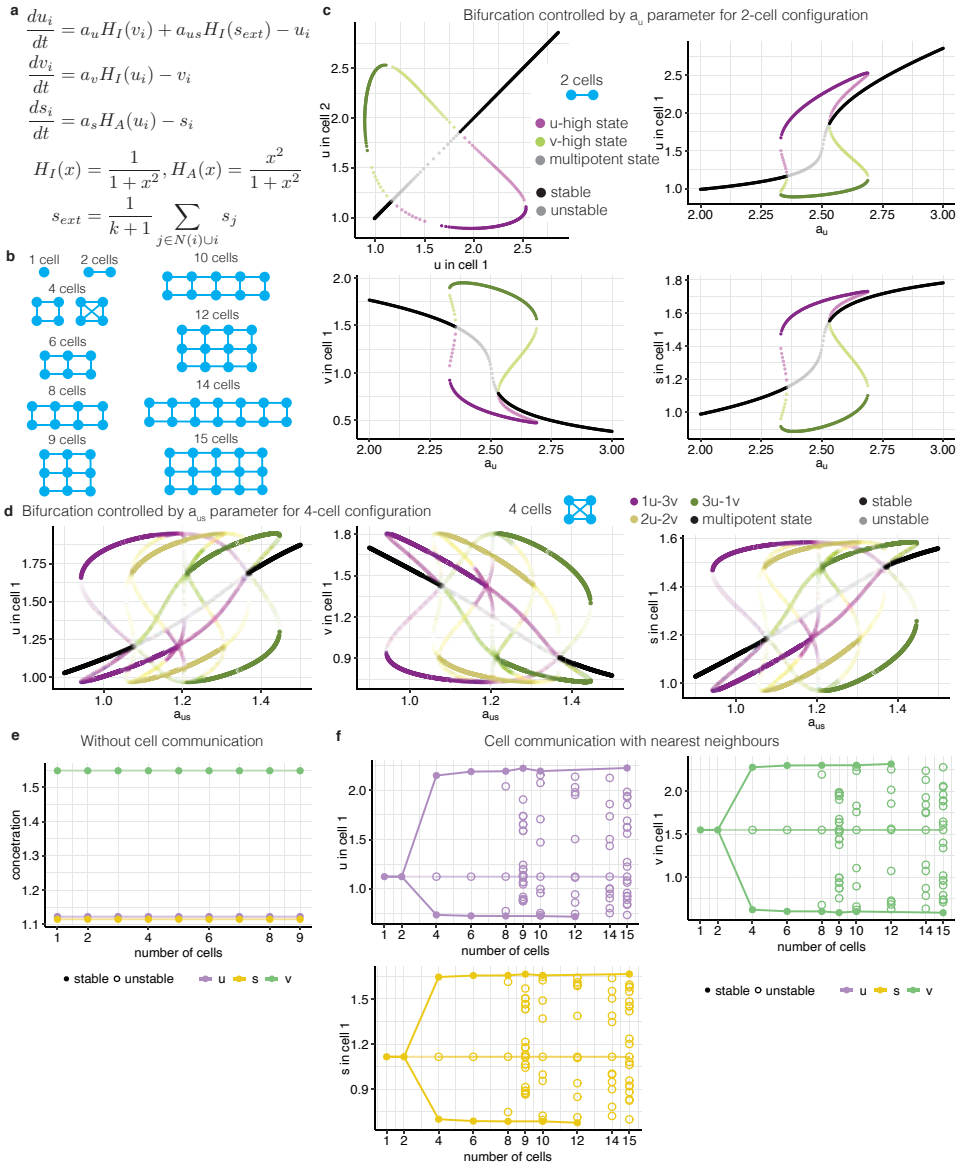

Figure 6.2: Cell communication drives bifurcations in a GRN model of differentiation. (Caption on the next page.)

For larger number of cells, the behavior of the system becomes more complex. We studied in detail a population of 4 cells where each cell was allowed to communicate with all other cells (Fig. 6.2d). We varied the strength of the intercellular communication, which is parameterized by a_{us}, and found that it can serve as a control parameter, similar to a_u. Starting from low values of a_{us} for which the mlp state is the only stable state, a sequence of bifurcations leads to the appearance and disappearance of several additional stable states. The first stable states to appear have one cell with high levels of u (low levels of v) and 3 cells with high levels of v (low levels of u). With increasing values of a_{us} the balance shifts to more cells with high u. At the extreme there is an interval of a_{us} with only two stable states in which 3 cells have a high level of u and one cell has a high level of v. Interestingly, the differentiated states are never homogeneous for the set of parameters used here.
Instead of modulating the strength of cell communication by tuning a_{us}, the system can also be driven out of the mlp state by increasing cell number, which would happen naturally through cell division. Without cell communication the system remains in the mlp state, irrespective of cell number (Fig. 6.2). With cell communication, a symmetry-breaking event occurs and two new stable states appear starting from 4 communicating cells, for the particular parameter set used (Fig. 6.2f). For 1 or 2 cells, only the mlp state is stable, and there are no other steady states. From 4 cells on, this state becomes unstable. As demonstrated previously by Stanoev et al. differentiation can occur simply after a certain number of cell divisions without changing the topology of the GRN or imposing a change of its parameters by external cues. Interestingly, more steady states appear in the system with increasing cell numbers. In the following, we will use simulations based on the 4-cell configuration with communication between all cells as ground truth training data for GRN inference with NNs.

Figure 6.2: Cell communication drives bifurcations in a GRN model of differentiation. (Figure on the previous page.) **a** System of differential equations corresponding to the GRN model by Stanoev et al. The mutual inhibition of the two master transcription factors u and v as well as the inhibition of u by the signaling molecule s are modeled with repressive Hill functions H_I. The cell autonomous activation of signalling molecule s by u is modeled with an activating Hill function H_A. i is the cell index. s_{ext} is the level of s averaged over cell i and its neighbors (typically nearest neighbors, unless otherwise indicated by the edges in panel b). The degradation rate for u, v and s is assumed to be identical, and time was rescaled with the inverse degradation rate, so that the rate does not appear explicitly in the equations. **b** Studied configurations of cells. Edges indicate cell communication. **c** Results for the 2-cell configuration. Several bifurcations are driven by the parameter a_u, which sets the strength of the inhibition of u by v. **d** Results for the 4-cell configuration with communication between all cells. Bifurcations are controlled by the parameter a_{us} which determines the strength of inter-cellular communication. Colors distinguish stable states with different ratios of u- and v-high cells. **e,f** Steady states (both stable and unstable) for the cell configurations shown in panel b without cell communication (panel **e**) or with cell communication (panel **f**). The following parameters were used: $a_u = 2.4,\ a_v = 3.5,\ a_s = 2, a_{us} = 1$.

6.2.2 Feedforward NN regression is unsuitable for GRN parameter inference

NNs have showed impressive performance in a large variety of supervised learning tasks [46]. The power of NNs usually relies on the existence of a large amount of high quality training data. Our first, naive idea was therefore to simulate expression trajectories, based on the dynamical system discussed above (see Fig. 6.2a), with randomly sampled parameters and use these trajectories to train a feedforward NN regression model (Fig. 6.3a). The input layer of this NN consists of the trajectories of u, v and s for n cells and k time points. Training samples are therefore vectors of length $3 \cdot n \cdot k$. The output nodes correspond to the 4 parameters of the GRN, a_u, a_v, a_s and a_{us}. Input and output layer were connected by several, fully-connected hidden layers.

To test this approach we used a configuration of 4 cells, with communication between all cells (as in Fig. 6.2d), and simulated 1000 trajectories with 25 time points for all variables in all cells. Parameters were sampled uniformly from intervals chosen such that trajectories from both the mlp as well as the differentiated regime were created (Fig. 6.3b). Initial states were also chosen randomly within reasonable intervals (see Methods). With this setup, the NN seemed to converge quickly and training was stopped after 1000 epochs (Fig. 6.3c). To create the test data we simulated 50 sets of 20 trajectories where the parameters were identical for each trajectory in a set, but the initial states were chosen randomly. Comparison of the parameter values used to simulate the trajectories (ground truth values) with the parameter values inferred by the NN model (Fig. 6.3d) revealed good accuracy of the model. Large systematic biases were absent for most parameter values. The random initial conditions contributed to the observed spread around the true values, which might limit the precision of the model.

At first glance, the simple feedforward architecture seemed to perform well. We next wanted to test, how important it is that the training data covers the different regimes of the dynamical system. When we trained the model with trajectories from the bistable, differentiated regime we observed that the model performed poorly for test trajectories outside of that regime (Fig. 6.3e). As the model is agnostic to the differential equations governing the dynamical system, it was unable to extrapolate beyond the parameter ranges it was trained on. In other words, if trained on a particular regime of the dynamical system, the NN model learns the behavior of that regime and does not generalize well. It would therefore be crucial to cover a large enough area of parameter space with the training data. Importantly, we were only able to identify the correct parameter ranges, because the system is relatively simple, allowing us to obtain a detailed understanding of its qualitative behavior (see Fig. 6.2). In an experimental setup, the relevant parameter ranges are usually unknown and it is typically hard to tune individual parameters. The naive feedforward NN regression model is therefore unsuitable for inferring GRN parameters from experimental data.

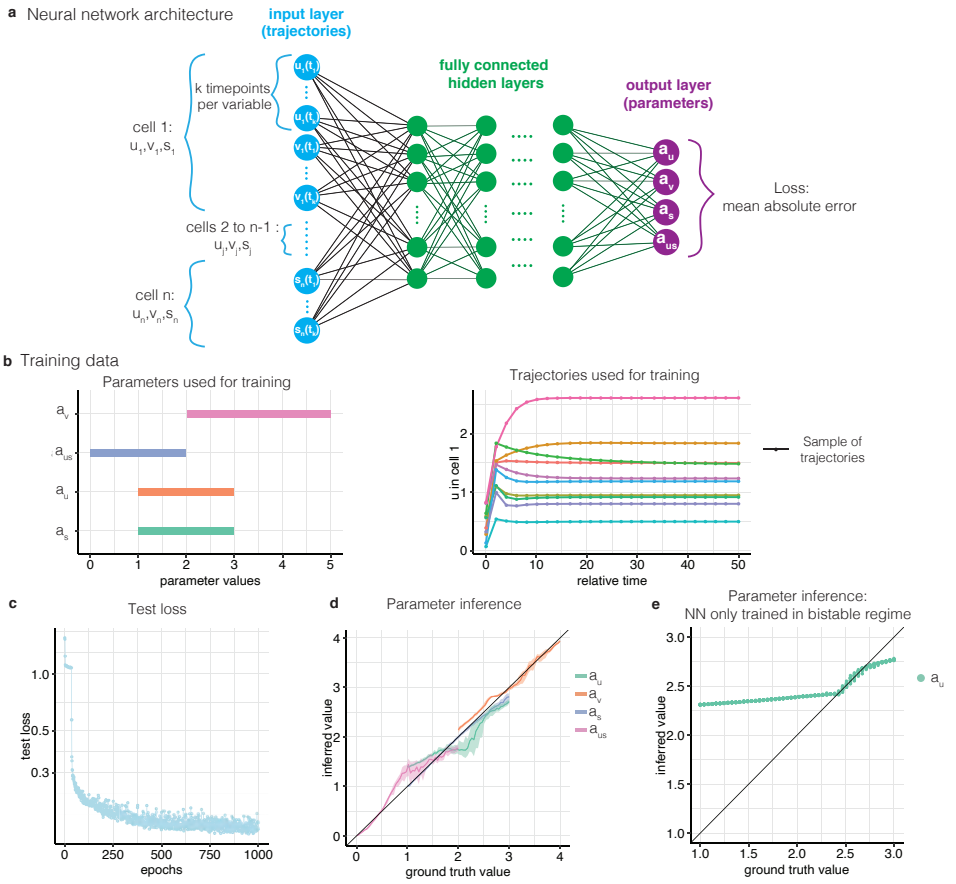

Figure 6.3: Feedforward NN regression is unsuitable for GRN parameter inference a Architecture of the feedforward NN. The mean absolute error was used for optimization. **b** Training data. Left: Parameter ranges used for creating simulated trajectories. Right: 10 example trajectories. **c** Test loss during training of the NN. **d,e** Ground truth parameter values (used for simulating the trajectories) versus parameter values inferred by the NN. In **d**, training trajectories covered the mlp as well as the bistable, differentiated regime. In **e** the training trajectories came exclusively from the bistable regime.

6.2.3 PHYSICS INFORMED NEURAL NETWORKS CAN INFER GRN PARAMETERS FROM PARTIAL AND NOISY DATA.

To ameliorate the reliance of NNs on large mounts of training data that represent all regimes of the dynamical system, we have to constrain the inference problem in a meaningful way. Ideally, the NN model should be aware of and respect the underlying differential equations. Physics-informed NNs (PINNs) leverage automated differentiation to solve a broad class of differential equations [29, 31]. The input layer of a PINN is composed of the independent variables of the differential equations (such as, for example, space and time for many applications in physics).

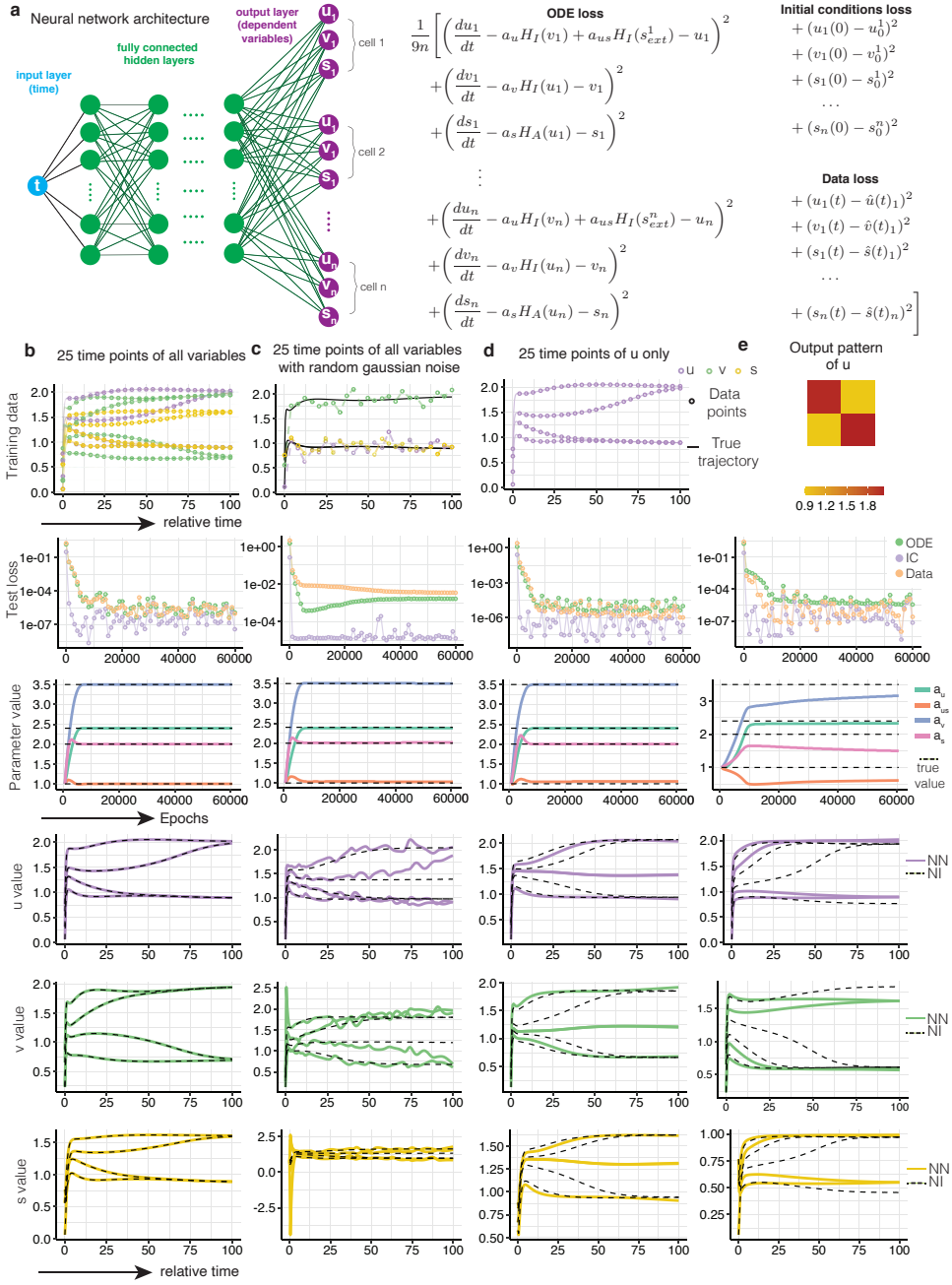

Figure 6.4: Physics informed neural network to infer gene regulatory networks. (Caption on the next page.)

The PINN is then trained such that the output layer approximates a solution to the differential equations for arbitrary points (in space and time) given as input. Fulfillment of the (ordinary) differential equations as well as initial and boundary conditions is ensured by appropriate loss terms, called ODE, IC and BC loss, respectively. During training, residual points on which the loss terms are evaluated are chosen randomly or in a way that adapts to the particular differential equations [30]. For this 'forward problem' of finding a solution of fully determined differential equations, no training data is necessary. PINNs can also infer undetermined parameters of the differential equations ('inverse problem'), which does require measured or simulated training data and a corresponding loss term ('data loss') that penalizes deviation of the solution from that data. The loss terms that ensure fulfillment of the differential equations strongly constrain the output space of the NN and thereby reduce the variance of the parameter inference. The issue of poor generalizability we observed with feedforward NN regression (Fig. 6.3e) should therefore be absent in PINNs.

To explore whether PINNs can successfully infer GRN parameters ('inverse problem'), we implemented the architecture shown in Fig. 6.4a with the DeepXDE package [30]. The input layer of the NN consists of only one node, which corresponds to time, and the output layer contains all dependent variables (u,v and s in all cells). As above, we used the 4-cell configuration with communication between all cells as a proof-of-concept. To generate training data we simulated trajectories with identical parameters but randomly drawn initial states. To explore the limitations of the PINN, we added noise and/or subset the data (Fig. 6.4b-e). Starting from noise-free trajectories with 25 time points per variable (Fig. 6.4b), we added Gaussian noise, since measurements are likely noisy due to biological and technical variability (Fig. 6.4c). We also explored training the PINN with a subset of variables as it is typically difficult to obtain measurements of all relevant dependent variables in experiments (Fig. 6.4d). Lastly, we studied training the model on the first and last time points only. For the set of parameters used here, the system has closely approached a stable steady state with two u-high and two v-high cells (Fig. 6.4e, top) by the last time point. This scenario is relevant for measurements with only one or a few time points or if the system is practically always in a stable steady state. In Fig. 6.4b-e we give examples of model behavior for different training scenarios. A systematic exploration and quantification of model performance is presented in Fig. 6.5.

Figure 6.4: Physics informed neural network to infer gene regulatory networks. (Figure on the previous page.) a Architecture of the PINN. The input to the network is time and the output consists of all dependent variables of the dynamical system. The PINN is optimized via a loss function that considers the differential equations (ODE loss), the initial conditions (IC loss) and training data (data loss). **b, c, d, e** The *first row* shows examples of training scenarios. A GRN with 4 cells that all communicate with each other was used. **b** Training on noise-free trajectories of all dependent variables with 25 fixed time points. **c** Training on trajectories shown on the left with added Gaussian noise. Only the trajectories in one cell are shown. **d** Training on noise-free trajectories of u only. **e** Only the first and last time point of the u trajectories in all 4 cells were used for training. The *second row* shows the resulting test losses. Colours indicate the different loss terms. *Row 4* shows the inferred parameters and *rows 3 - 6* show the approximated trajectories for the four scenarios. In the trajectory plots solid lines are trajectories approximated by the PINN and dashed lines are trajectories calculated by numerical integration using the inferred parameters.

When using complete trajectories for training, the PINN converges robustly after a few epochs and all three loss terms have similar convergence rates (Fig. 6.4b, second row). The inferred parameters are close to the ground truth parameters (Fig. 6.4b, third row) and the trajectories approximated by the PINN coincide with the trajectories calculated by numerical integration using the inferred parameters (Fig. 6.4b, rows 4-6). In contrast to feedforward NN regression (Fig. 6.3), which required many training samples, the PINN needs only one set of trajectories for accurate GRN inference. As to be expected, noise reduced the performance of the model, likely due to over-fitting, which can be seen for the inferred trajectories in Fig. 6.4c. Model performance was also compromised when only one dependent variable was used for training (Fig. 6.4d). Providing only the initial and final time point presented the biggest challenge for the PINN (Fig. 6.4e): The trajectories approximated by the PINN show large discrepancies with the trajectories calculated by numerical integration using the same (inferred) parameters. Hence, the trajectories approximated by the PINN are not a proper solution of the differential equations. Surprisingly, the inferred parameters were still roughly correct.

For a more systematic and quantitative assessment of model performance we tested 84 different conditions and considered: 1. the mean squared error between trajectories approximated by the PINN and trajectories found through the numerical integration using the inferred parameters, 2. the test loss and 3. the relative error of the inferred parameters (Fig. 6.5). For each condition we averaged over 10 runs with identical GRN parameters but randomly drawn initial states. First, we focused on the training scenarios that utilized all time points (Fig. 6.5b-d). As to be expected, increasing levels of noise reduced model performance (Fig. 6.5b).

In an attempt to mitigate over-fitting to the noisy training data, we introduced weights for the three loss terms and gave the ODE loss a 1000-times higher weight. Weighting improved trajectory approximation, but did not have a strong influence on parameter inference. Removing dependent variables from the training set had a strong and systematic effect on parameter inference and trajectory approximation was similarly affected when no weights were used (Fig. 6.5c). Weighting strongly improved trajectory approximation when only one dependent variable was used for training. Importantly, the relative errors of the parameter values depended on the set of variables used for training (Fig. 6.5d). For example, when only u and v were used, the parameters a_u and a_v were inferred more accurately than the parameters a_{us} and a_s. Conversely, when only u and s were used, a_{us} and a_s had a smaller error than the other parameters. Learning from only the first and last time point of the training data was overall a harder task for the PINN (Fig. 6.5e-g), but we observed similar trends for the dependence of model performance on noise (Fig. 6.5f) or the number of dependent variables used for training (Fig. 6.5g). Surprisingly, parameter inference from two time points was almost as accurate as when the whole trajectories were used for training, while the PINN's approximation of the trajectories was compromised. In summary, the PINN was able to infer GRN parameters even when only partial data was supplied for training.

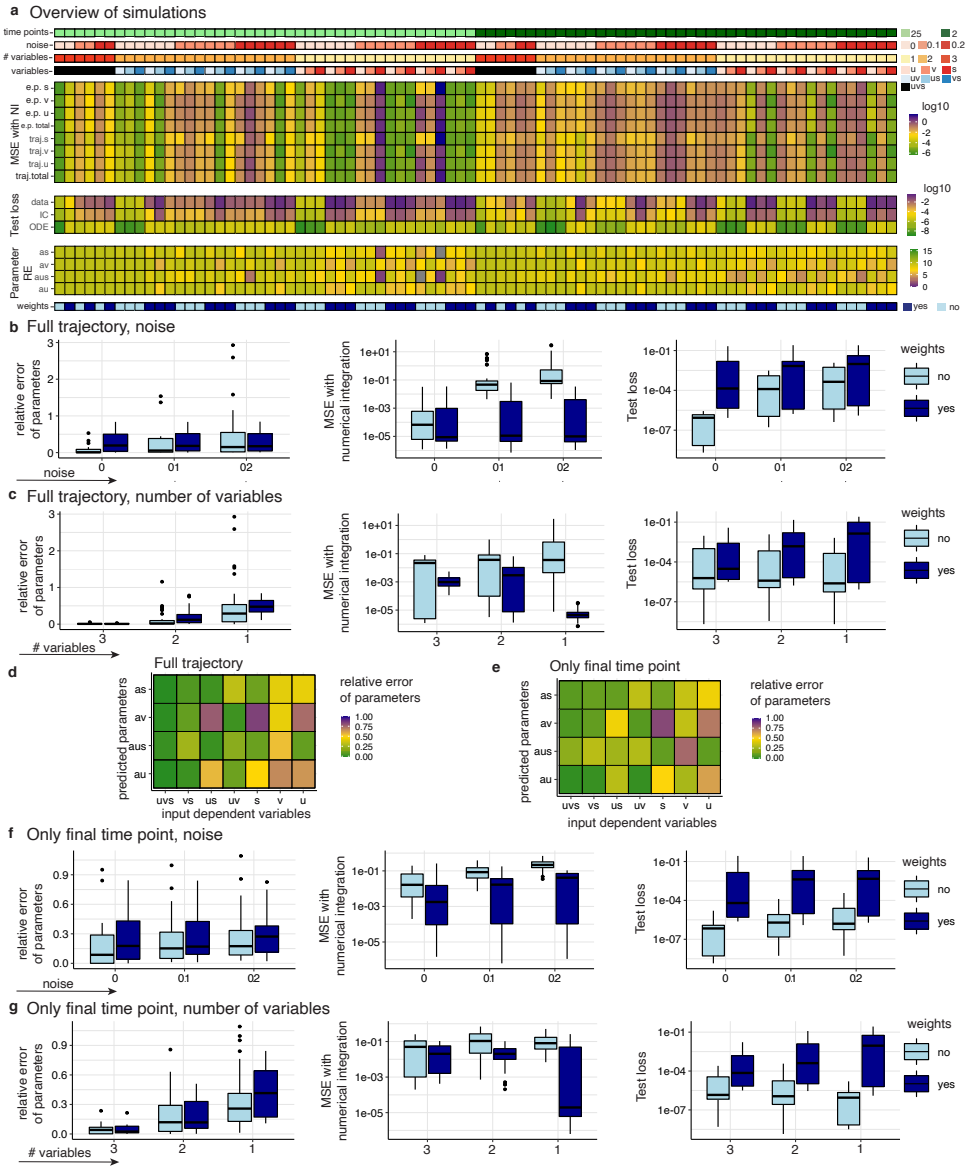

Figure 6.5: PINNs can infer GRN parameters from partial and noisy data. a Overview of all simulation results. The following parameters of the training data set were varied: the number of time points (either 25 per variable or only the first and last time point), the amount of Gaussian noise (standard deviation of 0, 0.1 or 0.2 with a mean of 0) and the dependent variables ($[u, v, s], [u, v], [u, s], [v, s], [u], [v], [s]$). When weights were given to the loss terms the ODE loss was weighted with a factor 1000. e.p.: end point, traj: full trajectory. **b-d** Dependence of PINN performance on noise level **b**, number **c** and identity **d** of dependent variables used for training when the complete trajectories were used. **e-g** Same performance comparisons as in **b-d** but the PINN was trained only on the initial and final time point of the trajectories.

6.2.4 PINNs can infer GRN parameters from snapshot data in the absence of cell communication.

Most high-throughput single-cell profiling assays are destructive, which prevents the measurement of single-cell trajectories. These assays therefore only provide "snapshots" of the system dynamics. Additionally, in conventional single-cell omics experiments, any information about the spatial arrangement of the cells is lost. Therefore we wanted to explore, how a PINN would perform when trained with snapshot data that lacks spatial resolution. As any parameter related to cell communication is unlikely to be estimated well in such a scenario, we considered a simpler dynamical system of two mutually inhibiting genes, u and v, without cell communication [10] (Fig. 6.6a). The parameters I_u and I_v modulate the inhibition of u or v, respectively, by the other gene. For a particular set of parameters, the dynamical system is bistable and leads to the cell autonomous differentiation into either a u-high or a v-high state. Cells will be attracted to one of these stable steady states depending on their initial state. As trajectories cannot be obtained from destructive snapshot measurements it is more natural to model the system at the level of a population of cells and consider a bivariate probability density of u and v. In the absence of noise, the dynamics of the probability density is completely determined by the conservation of probability (Fig. 6.6a), which is therefore the differential equation that must be fulfilled by the PINN.

Fig. 6.6b shows the architecture of the PINN together with the loss terms. The input layer is now composed of three nodes, corresponding to time, u and v. The only output node is the probability density at the point $[u,v,t]$ given as input. As before, the loss considers the governing differential equations, initial conditions and the training data. For training we simulated 1000 trajectories with initial values of u and v randomly drawn from a bivariate normal distribution centered around 1.5. As mentioned above, the parameters of the GRN and the distribution of the initial states are chosen such that trajectories tend to one of two stable steady states (Fig. 6.6c). The simulated trajectory positions were binned for each time point and the probability densities were approximated by the relative frequencies (Fig. 6.6d). As intended, the initial probability density, a normal distribution, developed into a bimodal distribution, reflecting the existence of two stable steady states with anti-correlated expression of u and v. Using these simulations we trained the PINN, leaving the parameters I_u and I_v undetermined. The PINN converged quickly (Fig. 6.6e) and the parameters were inferred with reasonable precision (Fig. 6.6f). The probability densities approximated by the PINN show qualitatively the same dynamics as the densities used for training (Fig. 6.6g). However, the PINN approximation of the probability density had a few negative values and was not always properly normalized, which could likely be improved by additional constraints. All in all, the PINN was able to infer GRN parameters from snapshot data for a model without cell communication.

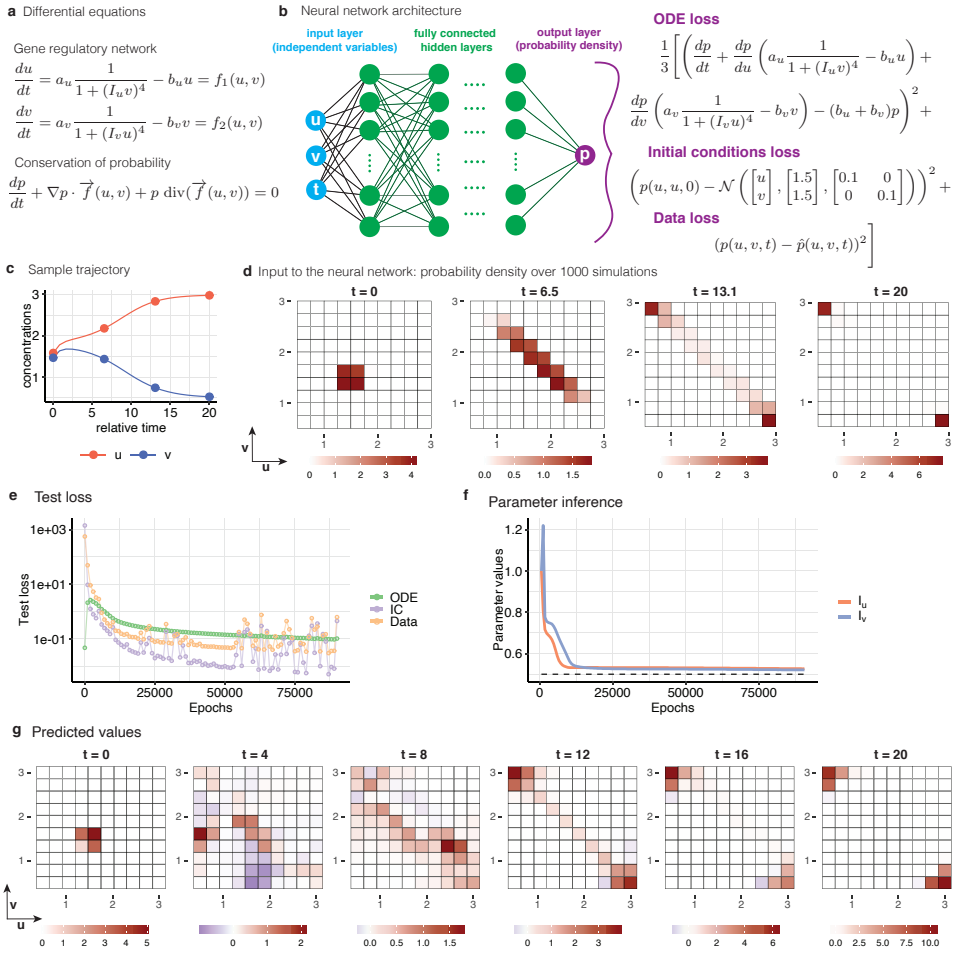

Figure 6.6: PINNs can infer GRN parameters from snapshot data in the absence of cell communication. **a** Top: Differential equations describing two, mutually inhibiting genes u and v. Bottom: Differential equation describing the conservation of probability. **b** Architecture of the PINN. The input nodes correspond to u , v and time t. The output is the probability density. The PINN is optimized via a loss function that considers the differential equations, the initial conditions and the training data. **c** Example trajectory for the differential equations shown in panel a. **d** Probability density of the values u and v at four different time points, generated by repeated simulation of trajectories as in c. This simulation was used as training data. **e** Test loss of PINN training using the data shown in d. **f** Convergence of the parameters I_u and I_v during PINN training. **g** Probability densities approximated by the PINN at 6 different time points.

6.3 Discussion

The inference of GRNs from noisy and usually incomplete measurements is a long-standing challenge which inspired the development of many different approaches. In this chapter, we studied the performance of PINNs in this context.

PINNs are general tools that approximate the solutions to a broad class of differential

equations. To apply them to GRN inference requires expressing the dynamical system defined by the GRN as a set of differential equations. To that end, specific expressions that model the gene interactions have to be assumed. Here, we used Hill functions with fixed Hill coefficients for both activation and inhibition. We selected a subset of relevant parameters to be learned by the PINN, but it might be interesting to leave more parameters undetermined, especially the Hill coefficients. Most importantly, we used the same network topology for simulation and training the PINN: The same genes were connected with the same type of interaction (either activating or inhibiting). In principle, one could base the training on a fully connected network and model each interaction as the sum of activating and inhibiting expressions. Such a setup would leave network topology unconstrained and it would be interesting to explore, if a PINN could infer it from the data. In this context, it might be useful to add a regularization term to the loss function such that only the strongest interactions are selected and the inferred GRN is sparse.

As a proof of concept we studied a minimal GRN with two mutually inhibiting genes. Such a GRN exhibits a bifurcation that models the differentiation of a multipotent progenitor into one of several differentiated cell types. While several pairs of master transcription factors that govern such bifurcations have been identified in experiments, real GRNs contain other relevant genes. It would therefore be useful to determine, how much and what kind of experimental data would be necessary to infer a much larger GRN with a PINN.

In this chapter we first studied an experimental scenario in which the trajectories of individual cells are measured and demonstrated that a PINN outperforms a simple feedforward NN regression model. While the feedforward NN requires many training samples that must cover all dynamical regimes of the GRN, the PINN efficiently infers GRN parameters from a single sample, at the cost of making assumptions or using prior knowledge about the GRN. The PINN was able to infer parameters even if only a subset of dependent variables was used for training. The relative errors of the inferred parameter values depended on the identity of the used variables, which is important to keep in mind for optimal experiment design. As the training of the PINN becomes computationally more costly with increasing number of cells, it would be interesting to explore, whether using measurements of a subset of cells for PINN training is sufficient for GRN inference. Possibly, that would require a kind of mean field approximation of the cells that are not used for training. Surprisingly, parameter inference was still possible when we only used the initial and final time point of the trajectories for training. However, in this case, the approximate trajectories provided by the PINN did not fulfill the differential equations as they deviated from trajectories calculated by numerical integration using the inferred parameters. It seems that the loss terms related to additional time points support the ODE loss in ensuring fulfillment of the differential equation. Inferring the GRN from the final state, which is in this case essentially a spatial pattern of differentiated cell types, would be very useful not only to study morphogenesis but also to inform synthetic biology applications. Recently, NNs were used to implement a cellular automaton that models morphogenesis [47]. Impressively, it was shown that providing a desired spatial pattern as training data is sufficient to train the NN such that the automaton robustly develops into that spatial pattern. While this is certainly a conceptually important feat, cellular automata are only rough approximations of real biological dynamics and it would be preferable to achieve a similar performance with GRNs. We speculate that constraining the final state to be a globally stable steady

state, potentially by using a Lyapunov function [48], might help in that respect.
In the final section of this chapter we studied a scenario in which only snapshot data of cell populations are available, which is the case for single-cell RNA-sequencing experiments [49]. As spatial context is not available in this scenario, we described the system at the population level, with probability densities of gene abundances. We showed that a PINN was able to infer a simple GRN without cell communication. While parameter inference was successful, the probability density approximated by the PINN sometimes had negative values and was not always properly normalized. Positive values could be enforced by adding an additional term to the loss function or using a rectified linear unit (ReLU) as activation function in the output layer. To ensure proper normalization one could reformulate the differential equations such that the dependent variable is a normalized function [50]. Another option is to discretize the domains of u and v and add a constraint on the sum of the resulting discrete densities via an additional loss term [51]. As there was no noise in the dynamical system used here, the relevant differential equation was simply given by the conservation of probability. In the presence of noise, one would have to use the Fokker-Planck equation which the PINN should be able to fulfill without problems.
Next to the two experimental scenarios studied in this chapter, there are a few others, which are currently very popular and would therefore be worthwhile to explore in future work. Many snapshot measurements of highly dynamical systems, such as developing tissues, in fact contain dynamical information: Pseudotime methods have been used to establish developmental progression from snapshot data [52, 53]. It would be interesting to work out, how pseudotime information could be leveraged for GRN inference with PINNs. Spatially resolved omics modalities are also being used extensively at the moment. To infer GRNs with cell communication from such data will be an interesting challenge.
In summary, we have established that PINNs can be used for the accurate inference of GRNs. PINNs thus present an exciting, new way to obtain mechanistic insights from single-cell data. We hope that our work will stimulate colleagues from mathematics, physics and biology to collaborate on the many fascinating problems presented by single-cell developmental biology.

6.4 Materials and Methods

Python (V 3.9) was used for all computations. NNs were implemented with *tensorflow* (V.2.6). All figures were generated with R (V 4.1).

6.4.1 Inference of a GRN with cell communication from trajectories

Differential equations

The set of coupled differential equations in Fig. 6.2a were adopted from [35]. For each cell, there are two mutually inhibiting genes, u and v. Additionally, a signalling molecule s that is stimulated by u, inhibits u in an autocrine and paracrine way. In the model considered here, all signalling molecules of neighboring cells contribute equally. The dynamics of each cell $i \in \{1, ..., N\}$ is governed by this system of differential equations

$$\begin{aligned} \frac{du_i}{dt} &= a_u H_I(v_i) + a_{us} H_I(s_{ext}) - u_i \\ \frac{dv_i}{dt} &= a_v H_I(u_i) - v_i \\ \frac{ds_i}{dt} &= a_s H_A(u_i) - s_i, \end{aligned} \tag{6.1}$$

where s_{ext} denotes signalling molecule abundance averaged over the neighbors: $s_{ext} = \frac{1}{k+1} \sum_{j \in N(i) \cup i} s_j$ with k the number of neighbours. H_I and H_A are the inhibiting and activating Hill functions, respectively, defined here with a fixed Hill coefficient of 2:

$$H_I(x) = \frac{1}{1+x^2}, \quad H_A(x) = \frac{x^2}{1+x^2} \tag{6.2}$$

This results in a set of coupled differential equations with $3N$ equations and $3N$ variables. The parameters a_u, a_v, a_s and a_{us} are the same for each cell. Time was rescaled by an inverse degradation rate which was assumed to be identical for all genes. An additional parameter λ, used by Stanoev et al. to control the speed of the temporal evolution, was set to 1.

We distributed the cells on a regular grid, generated with *python-igraph* (V 0.9), and cell communication was typically restricted to nearest neighbors, unless otherwise indicated by edges in the graph.

Steady states

Steady states were found with a multi-start optimization algorithm using *scipy*. The stability was calculated based on the Jacobian matrix evaluated around the steady state. Bifurcation analysis was performed by repeating the optimization algorithm for each value of the control parameter (either a_u or a_{us}). For the 2-cell and 4-cell configurations, 500 initial values were chosen uniformly in the interval $[0, 3]$ for each of the $3N$ variables. In the study of the effect of cell number (Fig. 6.2e,f) 100 initial values were used.

Steady states were either mlp, where each cell was an mlp, or differentiated, consisting of u-high and v-high cells. The mlp steady state was identified based on comparison with gene expression in the 1-cell configuration with the same parameters. If the relative error

(comparing to the 1-cell mlp state) for all dependent variables was below 0.1, a steady state was annotated as mlp. To identify u-high and v-high cells in a differentiated steady state, the ratio of u and v in individual cells was compared to the ratio of u and v in the mlp steady state for the same parameters. If the ratio of u and v in a cell was larger than in the mlp steady state, the cell was consider to be a 'u-high' cell, if the ratio was smaller than in the mlp steady state, the cell was considered to be 'v-high'.

Data simulation

Data points were generated by numerical integration (NI) of the differential equations for a given set of parameters. We used scipy (V 1.7) with the explicit Runge-Kutta method for integration. The initial conditions were chosen randomly from a uniform distribution in the interval $[0,1]$. Numerical integration was performed on 100 equidistant time points in the interval $[0,T]$. When the entire trajectory was used for training, a subset of 25 equidistant time points was used. When only the initial and final state were used as input, we considered the values at $t = 0$ and $t = T$.

6.4.2 Feed-forward neural network

The feed-forward NN architecture is depicted in Fig. 6.3a. The NN input takes 25 time points for each variable. The GRN consisted of 4 cells, which all communicate with each other, resulting in 12 independent variables in total. Thus, the input layer of the NN consists of 300 nodes. We found that a 20% dropout rate in the input layer prevented over-fitting during training. The NN has 4 fully-connected hidden layers with 32 nodes each and Rectified Linear Unit (ReLu) activation functions. The output layer has 4 nodes for the results shown in Fig. 6.3d, corresponding to the parameters a_u,a_v,a_s and a_{us}, and one node for the result shown in Fig. 6.3e, corresponding to the parameter a_u. The loss function was defined using the mean absolute error, which slightly outperformed the mean squared error. For optimization the Adam optimizer was used.
For training and testing, trajectories were generated by numerical integration using randomly drawn initial states from the interval $[0,1]$. For Fig. 6.3d, 1000 training trajectories were generated with parameters chosen randomly from uniform distributions over the following intervals: $a_u \in [1,3]$, $a_v \in [2,5]$, $a_s \in [1,3]$ and $a_{us} \in [0,2]$. To generate trajectories for testing in Fig. 6.3d, parameters were chosen from the same parameter ranges. For each run, 3 out of the 4 parameters were kept fixed ($a_u = 2.4$, $a_v = 3.5, a_u = 2, a_{us} = 1$), while the remaining parameter was varied. For each parameter, 50 values were taken from the above intervals equidistantly and each set of parameters was used 20 times with different initial values for the numerical integration.
For Fig. 6.3e, 1000 training trajectories were generated with three parameters fixed ($a_v = 3.5, a_u = 2, a_{us} = 1$) and a_u sampled from the interval $[2.4, 2.7]$, where the dynamical system has multiple steady states in the 4-cell configuration. The trajectories used for testing were generated with the same fixed parameters, but 50 values of a_u were equidistantly drawn from the interval $[1,3]$. Each parameter value was used 20 times with different initial conditions for the numerical integration.

6.4.3 Physics Informed Neural Network

All PINNs were implemented using *DeepXDE* (V 0.14) [30]. The network architecture is shown in Fig. 6.4a. The input layer consists of only one node, which corresponds to the time t. The hidden layers are 4 fully connected layers with 40 nodes each. The output layer has $3N$ nodes, where N is the number of cells, corresponding to the $3N$ dependent variables of the dynamical system. In Fig. 6.5, a configuration of 4 cells that all communicate with each other was implemented, which results in an output layer of size 12. *tanh* was used as the activation function.

The loss function consists of three terms. The first penalizes deviation from the differential equations:

$$\left(\frac{du_1}{dt} - a_u H_I(v_1) + a_{us} H_I(s^1_{ext}) - u_1\right)^2 + \left(\frac{dv_1}{dt} - a_v H_I(u_1) - v_1\right)^2 + \left(\frac{ds_1}{dt} - a_s H_A(u_1) - s_1\right)^2 +$$

$$\ldots$$

$$\left(\frac{du_n}{dt} - a_u H_I(v_n) + a_{us} H_I(s^n_{ext}) - u_n\right)^2 + \left(\frac{dv_n}{dt} - a_v H_I(u_n) - v_n\right)^2 + \left(\frac{ds_n}{dt} - a_s H_A(u_n) - s_n\right)^2 \tag{6.3}$$

The second loss term considers the initial conditions, using the mean squared error. The last term in the loss function includes the training data, also using the mean squared error. For time points within the trajectory we defined boundary conditions using the *PointSet* object from the *DeepXDE* package. This object allows the user to supply measured data at any point in the input domain and the coresponding loss term will be added to the loss function. In order to specify the final time point in the *DeepXDE* framework, we used the Dirichlet boundary condition object. In this way, we could set values for the time domain at the initial point $t = 0$ and the final point $t = T$.

To create the results shown in Fig. 6.5, PINNs were trained using 84 different scenarios, 10 times each, with randomly selected initial conditions for the differential equations. Results were averaged over the 10 simulations. The following properties of the training data and PINN were varied:

(a) number of time points used for training: 25 (full trajectory) or 2 (initial and final state)
(b) noise level: no noise; Gaussian with mean 0, standard deviation 0.1; Gaussian with mean 0, standard deviation 0.2. Negative values resulting from addition of noise addition were set to 0.
(c) number of dependent variables used for training: 1,2 or 3
(d) identity of dependent variables used for training: $[u, v, s], [u, v], [u, s], [v, s], u, v, s$
(d) weights: no weights or ODE loss weighted with factor 1000

Neural network validation

We used three measures to quantify the performance of the PINN. First, we computed the relative error between the inferred parameters and the true parameters. Second, we calculated the mean squared error between the PINN approximation of the trajectories and trajectories obtained by numerical integration using parameters and initial conditions inferred by the PINN. Lastly, we considered the test loss.

6.4.4 Inference of a GRN without cell communication from snapshot data

Differential equations

Two mutually inhibiting genes, u and v, were modeled with expressions used to describe lateral inhibition [10]. The differential equations are given by:

$$\begin{aligned}\frac{du}{dt} &= a_u \frac{1}{1+(I_u v)^4} - b_u u = f_1(u,v)\\ \frac{dv}{dt} &= a_v \frac{1}{1+(I_v u)^4} - b_v v = f_2(u,v)\end{aligned} \tag{6.4}$$

We used a set of parameters for which this dynamical system is bistable: $a_u = a_v = 1.5$, $I_u = I_v = 0.5$ and $b_u = b_v = 0.5$. To model inhibition we used an inhibiting Hill function with Hill coefficient 4.

We describe the system at the population level with the joint probability density for the abundance of u and v. The time evolution of this probability density is governed by the conservation of probability:

$$\frac{\partial p}{\partial t} + \nabla p \cdot f(u,v) + p \,\mathrm{div}(f(u,v)) = 0, \tag{6.5}$$

where $f(u,v)$ is defined as

$$f(u,v) = \begin{bmatrix} f_1(u,v) \\ f_2(u,v) \end{bmatrix} \tag{6.6}$$

Plugging in $f(u,v)$ gives the following differential equations:

$$\frac{dp}{dt} + \frac{dp}{du}\left(a_u \frac{1}{1+(I_u v)^4} - b_u u\right) + \frac{dp}{dv}\left(a_v \frac{1}{1+(I_v u)^4} - b_v v\right) - (b_u + b_v)p = 0 \tag{6.7}$$

Data simulation

Training data was created by numerical integration as described in section 6.4.1. The initial conditions were sampled from a normal distribution with mean 1.5 and standard deviation 0.1. Trajectories with 50 time points in the interval $[0,20]$ were obtained. 4 equidistant time points were used for further computations. In order to create a probability density function, 1000 trajectories were generated. The probability density was then estimated by the relative frequencies calculated for 100 bins with u and v in the interval $[0.5,3]$, for each time point. The values for u and v in each bin were taken as the bin's midpoint.

Physics Informed Neural Network

A PINN with the architecture shown in Fig. 6.6a was implemented with *DeepXDE*. The PINN takes three independent variables as input (u, v and t). The hidden layers are 4 fully connected layers with 40 nodes each and *tanh* activation functions. The output layer has only one node, which corresponds to the probability density $p(u,v,t)$. Initial conditions were defined at $t = 0$ as a bivariate normal distribution with mean $\begin{bmatrix}1.5\\1.5\end{bmatrix}$ and

variance $\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$. Simulated data was added, as described previously, with the *PointSet* object in *DeepXDE*. As before, the loss function was composed of 3 terms that consider the differential equations, the initial conditions and the training data, using the mean squared error. For training, 500 points were chosen randomly from the joint domain of u and v to define the initial condition loss, 100 points from the simulated data were chosen for the data loss and 1000 points from the joint domain of u, v and t were chosen randomly to define the differential equation loss. We weighted the data loss with a factor of 1000. The parameters a_u, a_v, b_u and b_v were fixed and the parameters I_u and I_v were inferred by the PINN.
To visualize the PINN approximation of the probability density, 6 time points $(0, 4, 8, 12, 16, 20)$ were selected and the approximated density was plotted on the u, v-grid that was used during training.

Funding M. M. and S.S. were supported by the Netherlands Organisation for Scientific Research (NWO/OCW,www.nwo.nl), as part of the Frontiers of Nanoscience (NanoFront) program. The computational work was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative.

Disclosure of potential conflict of interest The authors indicated no potential conflicts of interest.

References

[1] L. Yu, Y. Cao, J. Y. Yang, and P. Yang. Benchmarking clustering algorithms on estimating the number of cell types from single-cell RNA-sequencing data. *Genome Biology*, 23(1):1–21, dec 2022.

[2] W. Saelens, R. Cannoodt, H. Todorov, and Y. Saeys. A comparison of single-cell trajectory inference methods. *Nature Biotechnology*, 37(5):547–554, may 2019.

[3] M. Mircea and S. Semrau. How a cell decides its own fate: a single-cell view of molecular mechanisms and dynamics of cell-type specification. *Biochemical Society Transactions*, dec 2021.

[4] A. Guillemin and M. P. H. Stumpf. Non-equilibrium statistical physics, transitory epigenetic landscapes, and cell fate decision dynamics. *Mathematical Biosciences and Engineering*, 17(6):7916–7930, nov 2020.

[5] P. Greulich, R. Smith, and B. D. MacArthur. The physics of cell fate. In *Phenotypic Switching*, pages 189–206. Elsevier, jan 2020.

[6] L. Xu and J. Wang. Quantifying Waddington landscapes, paths, and kinetics of cell fate decision making of differentiation/development. In *Phenotypic Switching*, pages 157–187. Elsevier, jan 2020.

[7] S. Huang. The molecular and mathematical basis of Waddington's epigenetic landscape: A framework for post-Darwinian biology? *BioEssays*, 34(2):149–157, feb 2012.

[8] J. X. Zhou, D. S. Aliyu, E. Aurell, and S. Huang. Quasi-potential landscape in complex multi-stable systems. *Journal of The Royal Society Interface*, 9(77):3539–3553, dec 2012.

[9] C. Waddington. *The strategy of the genes : a discussion of some aspects of theoretical biology / by C.H. Waddington. | Wellcome Collection.* Allen and Unwin, London, 1975.

[10] J. E. Ferrell. Bistability, Bifurcations, and Waddington's Epigenetic Landscape. *Current Biology*, 22(11):R458–R466, jun 2012.

[11] N. Saiz et al. Growth factor-mediated coupling between lineage size and cell fate choice underlies robustness of mammalian development. *eLife*, 9:1–38, jul 2020.

[12] D. Raina et al. Cell-cell communication through FGF4 generates and maintains robust proportions of differentiated cell fates in embryonic stem cells, feb 2020.

[13] M. K. Franke and A. L. Maclean. A single-cell resolved cell-cell communication model explains lineage commitment in hematopoiesis. *bioRxiv*, page 2021.03.31.437948, apr 2021.

[14] S. Huang, Y. P. Guo, G. May, and T. Enver. Bifurcation dynamics in lineage-commitment in bipotent progenitor cells. *Developmental Biology*, 305(2):695–713, may 2007.

[15] A. Pratapa et al. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nature Methods*, 17(2):147–154, feb 2020.

[16] V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, and P. Geurts. Inferring regulatory networks from expression data using tree-based methods. *PLoS ONE*, 5(9):12776, 2010.

[17] S. Aibar et al. SCENIC: Single-cell regulatory network inference and clustering. *Nature Methods*, 14(11):1083–1086, oct 2017.

[18] N. Papili Gao, S. M. Ud-Dean, O. Gandrillon, and R. Gunawan. SINCERITIES: Inferring gene regulatory networks from time-stamped single cell transcriptional expression profiles. *Bioinformatics*, 34(2):258–266, jan 2018.

[19] X. Qiu et al. Inferring Causal Gene Regulatory Networks from Coupled Single-Cell Expression Dynamics Using Scribe. *Cell Systems*, 10(3):265–274.e11, mar 2020.

[20] K. Kamimoto, C. M. Hoffmann, and S. A. Morris. CellOracle: Dissecting cell identity via network inference and in silico gene perturbation, feb 2020.

[21] C. Weinreb et al. Fundamental limits on dynamic inference from single-cell snapshots. *Proceedings of the National Academy of Sciences of the United States of America*, 115(10):E2467–E2476, mar 2018.

[22] S. J. Dunn et al. Defining an essential transcription factor program for naïve pluripotency. *Science*, 344(6188):1156–1160, jun 2014.

[23] M. Sáez et al. Statistically derived geometrical landscapes capture principles of decision-making dynamics during cell fate transitions. *Cell Systems*, sep 2021.

[24] O. I. Abiodun et al. Comprehensive Review of Artificial Neural Network Applications to Pattern Recognition. *IEEE Access*, 7:158820–158846, 2019.

[25] C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle. Deep learning for computational biology. *Molecular Systems Biology*, 12(7):878, jul 2016.

[26] G. Eraslan et al. Single-cell RNA-seq denoising using a deep count autoencoder. *Nature Communications*, 10(1), 2019.

[27] Y. Yang, Q. Fang, and H. B. Shen. Predicting gene regulatory interactions based on spatial gene expression data and deep learning. *PLOS Computational Biology*, 15(9):e1007324, 2019.

[28] J. Wang et al. Inductive inference of gene regulatory network using supervised and semi-supervised graph neural networks. *Computational and Structural Biotechnology Journal*, 18:3335–3343, jan 2020.

[29] M. Raissi, P. Perdikaris, and G. 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, feb 2019.

[30] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, jul 2019.

[31] G. E. Karniadakis et al. Physics-informed machine learning. *Nature Reviews Physics 2021 3:6*, 3(6):422–440, may 2021.

[32] J. H. Lagergren et al. Biologically-informed neural networks guide mechanistic modeling from sparse experimental data. *PLOS Computational Biology*, 16(12):e1008462, dec 2020.

[33] A. Yazdani, L. Lu, M. Raissi, and G. E. Karniadakis. Systems biology informed deep learning for inferring parameters and hidden dynamics. *PLOS Computational Biology*, 16(11):e1007575, nov 2020.

[34] M. AlQuraishi and P. K. Sorger. Differentiable biology: using deep learning for biophysics-based and data-driven modeling of molecular mechanisms. *Nature Methods 2021 18:10*, 18(10):1169–1180, oct 2021.

[35] A. Stanoev, C. Schröter, and A. Koseska. Robustness and timing of cellular differentiation through population-based symmetry breaking. *Development*, 148(3):dev197608, feb 2021.

[36] S. Bessonnard et al. Gata6, Nanog and Erk signaling control cell fate in the inner cell mass through a tristable regulatory network. *Development (Cambridge)*, 141(19):3637–3648, oct 2014.

[37] J. Reinitz and D. H. Sharp. Mechanism of eve stripe formation. *Mechanisms of Development*, 49(1-2):133–158, jan 1995.

[38] T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in Escherichia coli. *Nature 2000 403:6767*, 403(6767):339–342, jan 2000.

[39] C. Furusawa and K. Kaneko. Emergence of rules in cell society: differentiation, hierarchy, and stability. *Bulletin of mathematical biology*, 60(4):659–687, 1998.

[40] Y. Goto and K. Kaneko. Minimal model for stem-cell differentiation. *Physical Review E*, 88(3):032718, sep 2013.

[41] B. Xia and I. Yanai. A periodic table of cell types. *Development (Cambridge)*, 146(12), jun 2019.

[42] E. Giacomelli et al. Human-iPSC-Derived Cardiac Stromal Cells Enhance Maturation in 3D Cardiac Microtissues and Reveal Non-cardiomyocyte Contributions to Heart Disease. *Cell Stem Cell*, 26(6):862–879.e11, jun 2020.

[43] N. M. Bérenger-Currias et al. A gastruloid model of the interaction between embryonic and extra-embryonic cell types:. *Journal of Tissue Engineering*, 13, jun 2022.

[44] M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain. Stochastic gene expression in a single cell. *Science*, 297(5584):1183–1186, aug 2002.

[45] H. Safdari et al. Noise-driven cell differentiation and the emergence of spatiotemporal patterns. *PLOS ONE*, 15(4):e0232060, apr 2020.

[46] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of Machine Learning Research*, 2010.

[47] A. Mordvintsev, E. Randazzo, E. Niklasson, and M. Levin. Growing Neural Cellular Automata. *Distill*, 5(2):e23, feb 2020.

[48] L. Haustenne et al. On stability analysis of genetic regulatory networks represented by delay-differential equations. *IFAC-PapersOnLine*, 48(1):453–457, jan 2015.

[49] A. A. Kolodziejczyk et al. The Technology and Biology of Single-Cell RNA Sequencing, may 2015.

[50] W. I. T. Uy and M. D. Grigoriu. Neural network representation of the probability density function of diffusion processes. *Chaos*, 30(9), jan 2020.

[51] Y. Xu et al. Solving Fokker-Planck equation using deep learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(1):013133, jan 2020.

[52] L. Haghverdi et al. Diffusion pseudotime robustly reconstructs lineage branching. *Nature Methods 2016 13:10*, 13(10):845–848, aug 2016.

[53] Z. Ji and H. Ji. TSCAN: Pseudo-time reconstruction and evaluation in single-cell RNA-seq analysis. *Nucleic Acids Research*, 44(13):e117, jul 2016.