LETTERS IN BIOMATHEMATICS
An International Journal

Intercollegiate Biomathematics Alliance

RESEARCH ARTICLE

OPEN ACCESS

Data-Driven Approaches for Predicting Spread of Infectious Diseases Through DINNs: Disease Informed Neural Networks

Sagi Shaier[a], Maziar Raissi[a], Padmanabhan Seshaiyer[b]

[a]University of Colorado, Boulder, USA; [b]George Mason University, Fairfax, Virginia, USA

ABSTRACT

In this work, we present an approach called Disease Informed Neural Networks (DINNs) that can be employed to effectively predict the spread of infectious diseases. We build on the application of physics informed neural network (PINNs) to SIR compartmental models and expand it to a scaffolded family of mathematical models describing various infectious diseases. We show how the neural networks are capable of learning how diseases spread, forecasting their progression, and finding their unique parameters (e.g., death rate). To demonstrate the robustness and efficacy of DINNs, we apply the approach to eleven highly infectious diseases that have been modeled in increasing levels of complexity. Our computational experiments suggest that DINNs is a reliable candidate to effectively learn the dynamics of their spread and forecast their progression into the future from available real-world data. Code and data can be found here: https://github.com/Shaier/DINN

ARTICLE HISTORY

Received February 3, 2022
Accepted August 3, 2022

KEYWORDS

Compartmental Models, Epidemiology, Neural Networks, Deep Learning

1 Introduction

Understanding the early transmission dynamics of infection diseases has never been more important in history as of today. The outbreak of severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) that led to several million confirmed cases across the globe has challenged us to re-envision how we model, analysis and simulate infectious diseases and evaluate the effectiveness of non-pharmaceutical control measures as important mechanisms for assessing the potential for sustained transmission to occur in new areas.

An important contribution to the mathematical theory of epidemics was developed by Kermack-McKendrick epidemic model of 1927 (Kermack and McKendrick, 1927). This was considered one of the earliest attempts to formulate a simple mathematical model to predict the spread of an infectious disease where the population being studies is divided into compartments namely a susceptible class S, an infective class I, and a removed class R.

This simple SIR epidemic model can be illustrated in compartments as in Figure 1. Not only was it capable of generating realistic single-epidemic outbreaks but also provided important theoretical epidemiological insights. In Figure 1, it is assumed

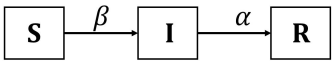


Figure 1: Compartmental Model for SIR model.

that each class resides within exactly one compartment and can move from one compartment to another. The dynamics of the three sub-populations $S(t)$, $I(t)$ and $R(t)$ may be described by the following SIR model given by first order coupled ordinary differential equations (ODE) (Hethcote, 2009; Brauer et al., 2012; Martcheva, 2015; Brauer, 2017):

$$\frac{dS}{dt} = -\beta S I, \qquad \frac{dI}{dt} = \beta S I - \alpha I, \qquad \frac{dR}{dt} = \alpha I. \tag{1}$$

CONTACT Sagi Shaier Sagi.Shaier@Colorado.edu

Note that this closed system does not allow any births/deaths. This SIR model in system (1) is fully specified by prescribing the *transmission* rate β and *recovery* rate α along with a set of initial conditions $S(0)$, $I(0)$ and $R(0)$. The total population N at time $t = 0$ is given by $N = S(0) + I(0) + R(0)$. Adding all the equations in system (1), we notice that $N'(t) = 0$ and therefore $N(t)$ is a constant and equal to its initial value. One can further assume $R(0) = 0$ since no one has yet had a chance to recover or die. Thus a choice of $I(0) = I_0$ is enough to define the system at $t = 0$ since then $S_0 = N - I_0$.

Following the influenza pandemic, several countries and leading organizations increased funding and attention to finding cures for infectious diseases in the form of vaccines and medicines. Along with these policy implementations, newer modified SIR models for mathematical epidemiology continued to evolve, particularly for those diseases that are categorized as re-emerging infections (Castillo-Chavez et al., 2002), those that are spread through sexual transmission such as HIV (Castillo-Chavez, 2013; Luo et al., 2016), those that are spread through vectors such as mosquitoes such as Malaria or Dengue (Li, 2011; Chowell et al., 2007), those that can spread through both sexual and vector transmissions such as Zika (Padmanabhan et al., 2017; Padmanabhan and Seshaiyer, 2017), and those that can be spread by viruses, including SARS and MERS (Dye and Gay, 2003; Alshakhoury et al., 2017). Diseases were also categorized according to the rate at which they spread, for example, super-spreader diseases. This point is especially relevant to COVID-19 (Ohajunwa et al., 2020; Ohajunwa and Seshaiyer, 2021), categorized as a super-spreader based on the disproportionately fast rate and large (and growing) number of infected persons.

Along with the development of mathematical modeling, there have been a variety of approaches that have been introduced to estimate the parameters such as the transmission, infection, quarantine and recovery using real data. These include nonparametric estimation (Smirnova et al., 2019), optimal control (Neilan and Lenhart, 2010), Bayesian frameworks (Coelho et al., 2011; Akman et al., 2016), partical swarm optimization (Akman et al., 2018), inverse methods, least-squares approach, agent-based modeling, using final size calculations (Bell, 1990; Pollicott et al., 2012; Yong et al., 2015a; Martcheva, 2015). Also, researchers have employed a variety of statistical approaches including maximum-likelihood, Bayesian inference and Poisson regression methods (Huang et al., 2006; Longini Jr et al., 1988; Hadeler, 2011; O'Dea et al., 2014; Capaldi et al., 2012). Some of this work also showed that the precision of the estimate increased with the number of outbreaks used for estimation (O'Dea et al., 2014). To determine the relative importance of model parameters to disease transmission and prevalence, there has also been work around *sensitivity analysis* of the parameters using techniques such as Latin Hypercube Sampling and Partial Rank Correlation Coefficients analysis with the associated mathematical models (Blower and Dowlatabadi, 1994; McKay et al., 2000; Chitnis et al., 2008). While there have been significant advances in estimating parameters, there is still a great need to develop efficient, reliable and fast computational techniques.

The dominant algorithm associated with the advancements in artificial intelligence ranging from computer vision (Goodfellow et al., 2014; Krizhevsky et al., 2012; Redmon et al., 2016; Tan et al., 2020) to natural language processing (Devlin et al., 2019; Vaswani et al., 2017) has been the neural networks (NN). A main reason for it is its behavior as a universal function approximator (Hornik et al., 1989). However, this field is largely relying on huge amounts of data and computational resources. Recent approaches (Raissi et al., 2019) have been shown to be successful in combining the best of both fields. That is, using neural networks to model nonlinear systems, but reducing the required data and by constraining the model's search space with known knowledge such as a system of differential equations.

Along with this, there have also been several works recently showing how differential equations can be learned from data. For example, Ling et al. (2016) used a deep neural network to model the Reynolds stress anisotropy tensor, E et al. (2017) solved for parabolic PDEs and backward stochastic differential equations using reinforcement learning, and Hagge et al. (2017) solved ODEs using a recurrent neural network. Additionally, Raissi and Karniadakis (2018); Raissi et al. (2019) developed physics informed models and used neural networks to estimate the solutions of such equations. Using this, recently such physics informed neural network approaches were applied for the first time to estimating parameters accurately for SIR model applied to a benchmark application (Raissi et al., 2019). The Physics Informed Neural Network approaches have also been recently used for studying the dynamics of COVID-19 involving human-human and human-pathogen interaction (Nguyen et al., 2022).

Building on this, a unified approach called DINNs: Disease Informed Neural Networks is introduced in this work and systematically applied to some increasingly complex governing system of differential equations describing various prominent infectious diseases over the last hundred years. These systems vary in their complexity, ranging from a system of three to nine coupled equations and from a few parameters to over a dozen. For illustration of the application of DINNs, we introduce its application to COVID, Anthrax, HIV, Zika, Smallpox, Tuberculosis, Pneumonia, Ebola, Dengue, Polio, and Measles. Our contribution in this work is three fold. First, we extend the recent physics informed neural networks (PINNs) approach to a large family of infectious diseases. Second, we perform an extensive analysis of the capabilities and shortcomings of DINNs on diseases. Lastly, we show the ease at which one can use DINNs to effectively learn the dynamics of the disease and forecast its progression a month into the future from real-life data.

The paper is structured as follows. In Section 2 we review the necessary background information. Section 3 introduces DINNs and presents our technical approach. Section 4 shows the application of DINNS to some of the benchmark models through computational experiments. Lastly, we conclude with a summary in Section 5.

2 Background Models and Methods

A grand challenge in mathematical biology and epidemiology, with great opportunities for researchers working on infectious disease modeling, is to develop a coherent framework that enables them to blend differential equations such as the system (1) with the vast data sets now available.

Noting that $\frac{dS}{dt} < 0$ for all t, the susceptible population $S(t)$ is monotonically decreasing and always declining independently of the initial condition S(0). Also, we have $\lim_{t\to\infty} S(t) = S_\infty$. This quantity is refered to as *final size* of the epidemic (Brauer et al., 2012). Also, when $S = \frac{\alpha}{\beta}$ the second equation in system (1), $\frac{dI}{dt} = 0$ which indicates that $I(t)$ has a stationary point at some maximum time. On the other hand, the number of infected individuals may be monotonically decreasing to zero, or may have non-monotone behavior by first increasing to some maximum level, and then decreasing to zero. One may note that the spread starts to increase if $\frac{dI}{dt} > 0$. This yields the following necessary and sufficient condition for an initial increase in the number of infectives given by $\frac{\beta S(0)}{\alpha} > 1$. Hence if $S_0 < \frac{\alpha}{\beta}$, the infection dies out and there is no epidemic. The last equation in system (1) also indicates that the recovered individuals also have monotone behavior, independent of $R(0) = R_0$. Since $\frac{dR}{dt} > 0$ for all t, the number of recovered is always increasing monotonically. Since we know that this number is constrained by the total population N, we also have $\lim_{t\to\infty} R(t) = R_\infty$. Since the total population $N = S_\infty + R_\infty = S_0 + R_0$, one can derive (Brauer et al., 2012)

$$S_\infty = S_0 e^{-\frac{\beta}{\alpha}(S_0 + I_0 - S_\infty)} \tag{2}$$

as well as the the maximum number of infected individuals I_{max} reached in the epidemic which occurs at $S = \frac{\alpha}{\beta}$:

$$I_{max} = -\frac{\alpha}{\beta} + \frac{\alpha}{\beta}\ln\left(\frac{\alpha}{\beta}\right) + I_0 + S_0 - \frac{\alpha}{\beta}\ln S_0. \tag{3}$$

2.1 Approaches for estimating rates

There are multiple approaches that can be used to estimate parameters α and β in system (1). Assuming the epidemic was initiated by one infected individual infecting n other individuals a day later, a crude approximation could be to use $\frac{dS}{dt} \approx -n$ per individual per day. Given S_0 and I_0, one can then estimate the initial transmission rate to be

$$\beta = -\frac{\frac{dS}{dt}}{S_0 I_0} = \frac{n}{S_0 I_0}.$$

If the infected individuals are isolated within d days of becoming sick, one may estimate that $\frac{1}{d}$ of the infected population was removed each day, or $\alpha = \frac{1}{d}$ per day. This then yields the ratio of $\frac{\alpha}{\beta}$. Using these values one can then plot the dynamics of the model predictions compared to the data using the system (1). Moreover one can use equations (2) and (3) to determine S_∞ and I_{max}.

Another approach to determine the rates that can be employed is by noticing the population that seem to have escaped the epidemic which could serve as the S_∞. Then, using equation (2) with the given dataset one can determine the ratio:

$$\frac{\beta}{\alpha} = \frac{\ln\left(\frac{S_0}{S_\infty}\right)}{N - S_\infty}. \tag{4}$$

Then assuming as before that the infected were quarantined for about d days as infectious individuals, one can find the recovery rate to be $\alpha = \frac{1}{d}$ which can then used to estimate β using (4).

It must be pointed out that this data set consists of a closed population. It must be also noted that all these models assumed that the recovery rate α can be computed heuristically. However, there are also methods in the literature that can help to estimate the parameters including S_0, α and β in system (1) by minimizing the deviation between the SIR model out and a given data set. One such software implementation is Berkeley Madonna (Macey et al., 2000) which has been shown to fit the data using the fitting parameter as α (Yong et al., 2015b).

Yet another approach for parameter estimation is an optimization algorithm that employs a least-squares minimization approach to estimate optimal parameters. Specifically, one can employ an unconstrained nonlinear optimization algorithm such as the Nelder-Mead algorithm which searches for a local minimum using a regression approach. This direct search method attempts to minimize a function of real variables using only function evaluations without any derivatives (Nsoesie et al., 2013). The minimized objective function is represented by differences in the daily infected counts from observed data and the computer simulated data.

Clearly, from these estimation approaches outlined so far, there can be variations in the ability of the dynamics of the computed values of infected population to track the true data for the various combination of parameters. As noted, these parameters

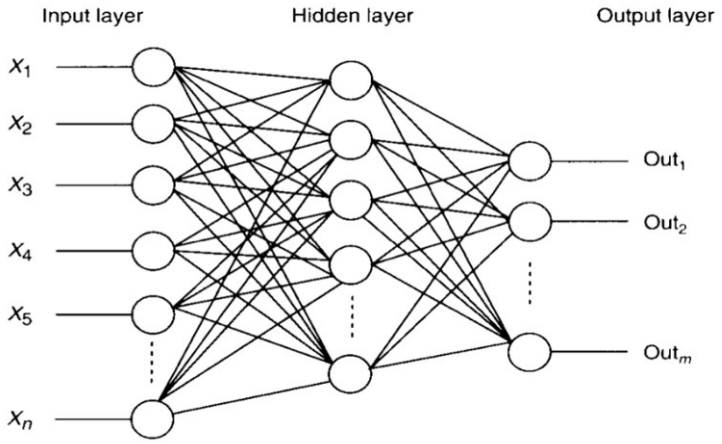


Figure 2: An illustration of a neural network.

are often calculated through heuristic methods in some of these algorithms and may not be optimal. Also, all the methods that have been discussed so far assumed the prior knowledge of initial number of each of the human sub-populations including the Susceptible S_0, Infected I_0 and Recovered R_0.

In this work, we introduce deep learning as an alternative and powerful approach, that employs *neural networks* which is a system of decisions modeled after the human brain (LeCun et al., 2015). Consider the illustration shown in Figure 2. The first layer of perceptrons first weigh and bias the input which can be observed values of infected data. The next layer then will make more complex decisions based off those inputs, until the final decision layer is reached which generates the outputs which can correspond to the values of parameters such as β and α. In this research, we implement a physics informed neural network based approach which makes decisions based on appropriate activation functions depending on the computed *bias* (b) and *weights* (w). The network then seeks to minimize the mean squared error of the regression with respect to the weights and biases by utilizing gradient descent type methods used in conjunction with software such as tensorflow. While there is currently a lot of enthusiasm about "big data", useful data in infectious diseases is usually "small" and expensive to acquire. In this work, we will describe how one can apply such physics informed neural network based deep learning approaches specifically to infectious diseases using DINNs and apply it to a real-world example to estimate optimal parameters, namely the transmission and recovery rates, in the SIR model.

3 Disease Informed Neural Networks

In this section, we present the DINNs methodology (sample architecture can be seen in figure 3). Subsection 3.1 briefly discusses background information for neural networks. Subsection 3.2 provides an overview of the DINNs approach and outlines the algorithm, associated loss functions, and training information.

3.1 Neural networks architechture

Briefly speaking, neural network is an attempt to mimic the way the human brain operates. The general fully connected model is organized into layers of nodes (i.e. neurons) where each node in a single layer is connected to every node in the following layer (except for the output layer), and each connection has a particular weight. The idea is that deeper layers capture richer structures (Eldan and Shamir, 2016). A neuron takes the sum of weighted inputs from each incoming connection (plus a bias term), applies an activation function (i.e nonlinearity), and passes the output to all the neurons in the next layer. Mathematically, each neuron's output looks as follows

$$\text{output} = \sigma\left(\sum_{i=1}^{n} x_i w_i + b\right)$$

where n represents the number of incoming connections, x_i the value of each incoming neuron, w_i the weight on each connection, b is a bias term, and σ is referred to as the activation function.

A schematic representation of the resulting *disease informed neural networks* is given in Figure 3. Note that for simplicity of illustration figure 3 depicts a network that comprises of 2 hidden layers with 5 neurons in the first hidden layer and 3 in the second. Networks with this kind of many-layer structure—two or more hidden layers—are called *deep neural networks*. These neurons in the network may be thought of as holding numbers that are calculated by a special *activation function* that depends

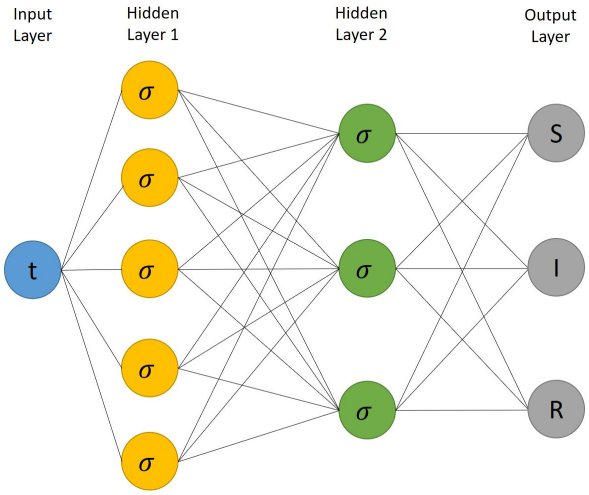


Figure 3: A Simple DINNs Architecture (input: time, has a size of 1 and the output can vary in size (S,I,R)).

on suitable *weights* and *biases* corresponding to each connection between neurons in each layer. With prior knowledge of such an activation function, the problem boils down to identifying the weights and biases that correspond to computed values of infected data that is close to the observed values. The three sub-populations are approximated by the deep neural network with calculus on computation graphs using a backpropogation algorithm (Hecht-Nielsen, 1992; Schmidhuber, 2015; Goodfellow et al., 2016).

Inspired by recent developments in *physics-informed deep learning* (Raissi and Karniadakis, 2018; Raissi et al., 2019), we propose to leverage the hidden physics of infectious diseases (1) and infer the latent quantities of interest (i.e., S, I, and R) by approximating them using deep neural networks. This choice is motivated by modern techniques for solving forward and inverse problems associated with differential equations, where the unknown solution is approximated either by a neural network or a Gaussian process. Following these approaches, we approximate the latent function

$$t \longmapsto (S, I, R)$$

by a deep neural network and obtain the following DINNs corresponding to equation (1) and the total population $N = S+I+R$, i.e.,

$$\begin{aligned} E_1 &:= \tfrac{dS}{dt} + \beta S I, \\ E_2 &:= \tfrac{dI}{dt} - \beta S I + \alpha I, \\ E_3 &:= \tfrac{dR}{dt} - \alpha I. \end{aligned} \tag{5}$$

We acquire the required derivatives to compute the residual networks E_1, E_2 and E_3 (disease informed) by applying the chain rule for differentiating compositions of functions using automatic differentiation (Baydin et al., 2018). In our computations, we employed a densely connected neural network, which takes the input variable t and outputs S, I, and R.

It is worth highlighting that parameters α and β of the differential equations turn into parameters of the resulting disease informed neural networks E_1, E_2 and E_3. The total loss function is composed of the regression loss corresponding to S, I and R and the loss imposed by the differential equations system (5). Moreover, the gradients of the loss function are back-propogated through the entire network to train the parameters using a gradient-based optimization algorithm. As will be explained next, we will assume that the only observables are noisy data that we will use in conjunction with the neural networks for S, I, R to estimate parameters α, β and γ by minimizing the sum of squared errors loss function. The idea employed builds on Physics Informed Neural Networks that can embed the knowledge of any physical law that govern a given data-set in the learning process (Raissi and Karniadakis, 2018; Raissi et al., 2019).

3.2 DINNs for parameter estimation

The predictive capability of any algorithm is measured partially by its robustness to unknown data. A dataset for known parameters can be simulated by solving a system of equations in a forward fashion and potentially adding some noise. If that is provided to any parameter estimation algorithm, the efficacy of the algorithm can be determined by how well it is able to predict the true values for a wide range of starting guesses.

For simplicity, we generated data by solving the systems of disease ODEs using LSODA algorithm (Hindmarsh and Petzold, 2005), the initial conditions, and the true parameters corresponding to each disease (e.g. death rate) from the literature. This limited dataset (50 − 100 points) corresponds to the SIR compartments. To make our neural networks disease informed, once the data was obtained we introduced it to our neural network without any prior knowledge of the transmission and recovery parameters. It is worth noting that in this formulation there are no training, validation, and test datasets, such as in most common neural networks training. Instead, the model is trained from data of how the disease is spread over time. The model then learned the system, and predicted the parameters that generated them. Since in many of these systems there exist a large set of parameters that can generate them, we restricted our parameters to be in a certain range around the true value. That is, to show that our model can in fact identify the systems and one set of parameters that match the literature they came from. However, our method is incredibly flexible in the sense that adding, modifying, or removing such restrictions can be done with one simple line of code. Additionally, we used nearly a years worth of real data aggregated over every US state and accurately predicted a month into the future of COVID transmission. Next we employ Literate programming style that is intended to facilitate presenting parts of written code in the form of a narrative (Knuth, 1984). DINNs takes the form

```
def net_sir(time_array):
    SIR = neural_network(time_array)
    return SIR

def net_f(time_array):
    dSdt = torch.grad(S, time_array)
    dIdt = torch.grad(I, time_array)
    dRdt = torch.grad(R, time_array)

    f1 = dSdt - (-beta SI)
    f2 = dIdt - (beta SI - alpha I)
    f3 = dRdt - (alpha I)
    return f1, f2, f3, S, I, R
```

The input of the neural network `net_sir` is a batch of time steps (e.g. $[0, 1, 2, ..., 100]$), and the output is a tensor (e.g. [S,I,R]) that represents what the network believes the disease's compartments look like at each time step. Here, `net_f` bounds the NN by forcing it to match the environment's conditions (e.g. f_1, f_2, f_3). These f_i corresponds to the E_i that was described earlier.

The parameters of the neural network `net_sir` and the network `net_f` can be learned by minimizing the mean squared error loss given by

$$\text{MSE} = \text{MSE}_{\texttt{net\_sir}} + \text{MSE}_{\texttt{net\_f}}$$

where

$$\text{MSE}_{\texttt{net\_sir}} = \frac{1}{N_{\texttt{net\_sir}}}\left[\sum_{i=1}^{N_{\texttt{net\_sir}}} \left|\texttt{net\_sir}(\text{time\_array}^i) - \text{SIR}^i\right|^2\right]$$

$$\text{MSE}_{\texttt{net\_f}} = \frac{1}{N_{\texttt{net\_f}}}\left[\sum_{i=1}^{N_{\texttt{net\_f}}} \left|\texttt{net\_f}^i\right|^2\right]$$

That is, minimizing the loss

$$\begin{aligned}\text{loss} = \text{mean}\big((\text{S}_{\text{actual}} - \text{S}_{\text{predict}})^2\big) + \text{mean}\big((\text{I}_{\text{actual}} - \text{I}_{\text{predict}})^2\big) \\ + \text{mean}\big((\text{R}_{\text{actual}} - \text{R}_{\text{predict}})^2\big) \\ + \text{mean}\big((\text{f}_1)^2\big) + \text{mean}\big((\text{f}_2)^2\big) + \text{mean}\big((\text{f}_3)^2\big)\end{aligned}$$

Here, "actual" and "predict" refer to the actual data that the model was provided with and the prediction the model computed, respectively. DINNs also leverages the automatic differentiation that neural networks are trained on to compute the partial derivatives of each S,I,R with respect to time. The neural networks themselves will consist of multiple fully connected layers with a multiple neurons each depending on the complexity of the system and *rectified linear activation function* (ReLU) activation in between.

4 Computational Experiments with DINNs

Most mathematical models describing the spread of a disease employ classical compartments, such as the Susceptible-Infected-Recovered (SIR) or Susceptible-Exposed-Infected-Recovered (SEIR) structure described as an ordinary differential equation system (Brauer and Castillo-Chávez, 2001). Over the past several months there have been a variety of compartmental models that have been introduced as modified SEIR models to study various aspects of COVID-19 including containment strategies (Maier and Brockmann, 2020), social distancing (Matrajt and Leung, 2020) and the impact of non-pharmaceutical interventions and the social behavior of the population (Ohajunwa et al., 2020; Ohajunwa and Seshaiyer, 2021). Along with these there have been a lot of work on modified SIR models as well including the SIRD model (Fernández-Villaverde and Jones, 2020; Anastassopoulou et al., 2019; Sen and Sen, 2021; Chatterjee et al., 2021). Next, to investigate the performance of DINNs, we apply DINNs on a simple SIRD model describing COVID-19 dynamics (Anastassopoulou et al., 2019).

4.1 Applying DINNS to an SIRD model applied to COVID-19

Consider following differential equation system describing the SIRD system where α is the transmission rate, β is the recovery rate, γ is the death rate from the infected individuals (Anastassopoulou et al., 2019), and N represents the total population:

$$\frac{dS}{dt} = -\frac{\alpha}{N}SI, \qquad \frac{dI}{dt} = \frac{\alpha}{N}SI - \beta I - \gamma I, \qquad \frac{dR}{dt} = \beta I, \qquad \frac{dD}{dt} = \gamma I. \tag{6}$$

The neural networks we considered, are fairly simple, consisting of 8 fully connected layers with either 20 or 64 neurons each depending on the complexity of the system and *rectified linear activation function* (ReLU) activation in between. Since the data is relatively small, our batch size contained the entire time array. The networks were trained on Intel(R) Xeon(R) CPU @ 2.30GHz, and depending on the complexity of the system the training time ranged from 30 minutes to 58 hours, which could be accelerated on GPUs and TPUs. That is, to learn both a system and its unknown parameters. However if the parameters are known, the training time to solely learn the system can be as short as 3 minutes. We used Adam optimizer (Kingma and Ba, 2014), and PyTorch's CyclicLR as our learning rate scheduler, with mode = "`exp_range`", `min_lr` ranging from 1×10^{-6} to 1×10^{-9} depending on the complexity of the system, `max_lr` $= 1 \times 10^{-3}$, gamma=0.85, and `step_size_up`=1000. In the next sections we will refer to "`min_lr`" simply as "learning rate". It is important to note that some diseases' systems were much more difficult for DINNs to learn (e.g. Anthrax considered later) and further training exploration such as larger/smaller learning rate, longer training, etc. may be needed to achieve better performance.

4.1.1 Influence of ranges in parameter estimation

Given that most models may include a large set of parameters, it is important to consider ranges for each of them. Hence, we restricted our parameters to be in a certain range to show that our model can learn the set that was used in the literature. First, we experimented with various parameter ranges to identify the influence they had on the model. In the following we used a 4 layer neural network with 20 neurons each, 1×10^{-6} learning rate, 100 data points, and the models were trained for 700,000 iterations (taking roughly 30 minutes). In our experiments we report two kinds of relative MSE loss errors. The first, "Error NN", is the error on the neural network's predicted system. The second, "Error learnable parameters", is the error on the system that was generated from the learnable parameters. That is, using LSODA algorithm to generate the system given the neural networks' parameters (e.g. α).

As an example, if the actual parameter's value was 0.1, a 0% search range would simply be $(0.1, 0.1)$, a 100% range would be $(0.1+1\times0.1, -0.1-1\times0.1) = (0.2, -0.2)$. Further ranges are multiplications of those: 1000% = $(2, -2)$, 10000% = $(20, -20)$, and so on. That is, each unknown parameter is initialized with a random initial value in between its corresponding search range. Table 1 (left) shows the parameters, their actual value, the range DINNs was searching in, and the parameters values that were found by DINNs. The right part of the table shows the error of the neural network and the LSODA generation of the system from the parameters. That is, it shows the effect that the search range had on how well the neural networks' learned the parameters. As seen from table 1 and figures 4–8, at least in the case of the COVID-19 system (6), DINNs managed to find extremely close set of parameters in any range we tested. Specifically, in figures 4–8, the panel on the left shows the effect that the parameter search range had on the neural networks' outputs and the right panel results show the effect that the search ranges had on how well the neural networks' learned the parameters. Additionally, the systems were almost learned perfectly, though, there was some variation in the relative error between experiments. It is worth noting that DINN might be able to learn the system quite well while also having some discrepancies in the learned parameters. Several reasons can explain this such as having a disease system that is relatively simple to learn and a too complex deep learning network, or that DINN found another set of parameters that can explain the data.

Table 1: Parameter predictions and relative MSE loss errors for various ranges.

Param.	Actual Value	Range	Param. Found	% Relative Error		(S,I,D,R)	
0% Search Range (exact parameters were given)							
α	0.191	(-0.382, 0.382)	0.191	0			
β	0.05	(-0.1, 0.1)	0.05	0		Error NN	(0.004, 0.025, 0.003, 0.003)
γ	0.0294	(-0.0588, 0.0588)	0.0294	0		Error Learnable Parameters	(0,0,0,0)
100% Search Range							
α		(-0.382, 0.382)	0.1942	1.62			
β		(-0.1, 0.1)	0.0510	2		Error NN	(0.002, 0.013, 0.001, 0.001)
γ		(-0.0588, 0.0588)	0.0294	0		Error Learnable Parameters	(0.027,0.113,0.020,0.024)
1000% Search Range							
α		(-3.82, 3.82)	0.1932	1.15			
β		(-1, 1)	0.0501	0.2		Error NN	(0.002, 0.012, 0.001, 0.001)
γ		(-0.588, 0.588)	0.0297	1.02		Error Learnable Parameters	(0.023,0.091,0.020,0.016)
10000% Search Range							
α		(-38.2, 38.2)	0.1951	2.14			
β		(-1, 1)	0.0506	1.2		Error NN	(0.002, 0.012, 0.002, 0.002)
γ		(-5.88, 5.88)	0.0296	0.68		Error Learnable Parameters	(0.041,0.164,0.029,0.031)
100000% Search Range							
α		(-382, 382)	0.1936	1.36			
β		(-10, 10)	0.0509	1.8		Error NN	(0.002, 0.012, 0.001, 0.001)
γ		(-58.8, 58.8)	0.0296	0.681		Error Learnable Parameters	(0.023,0.091,0.020,0.016)

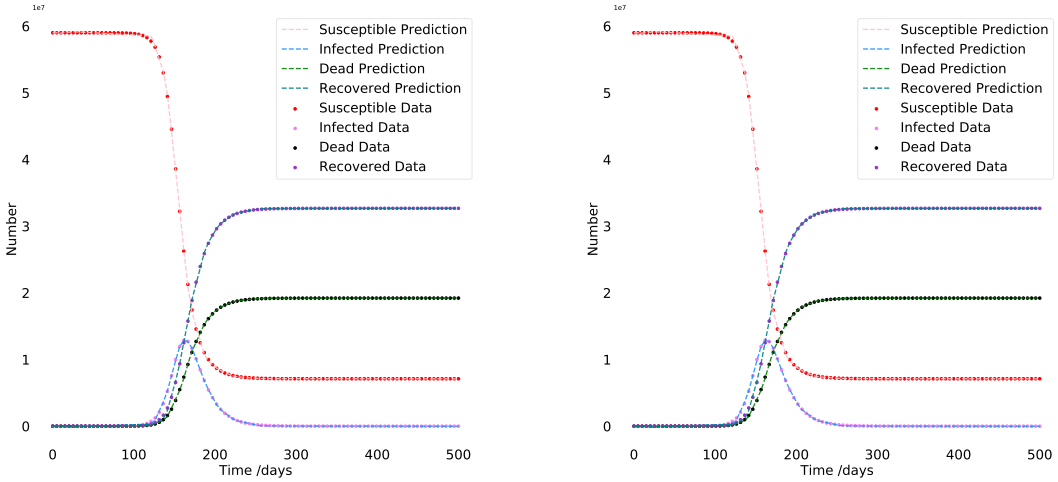


Figure 4: 0% search range for NN Output (Left panel) vs LSODA generation from Learnable Parameters (Right Panel).

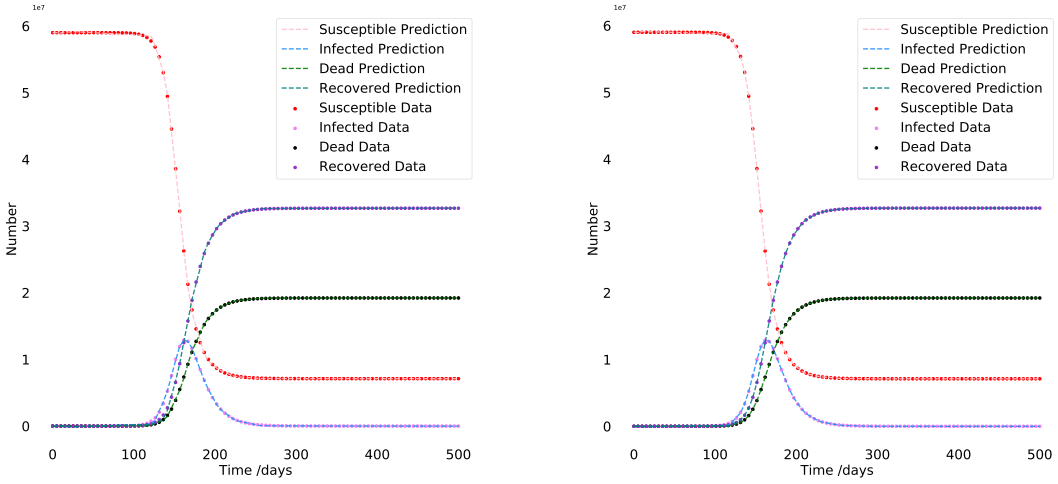


Figure 5: 100% search range for NN Output (Left panel) vs LSODA generation from Learnable Parameters (Right Panel).

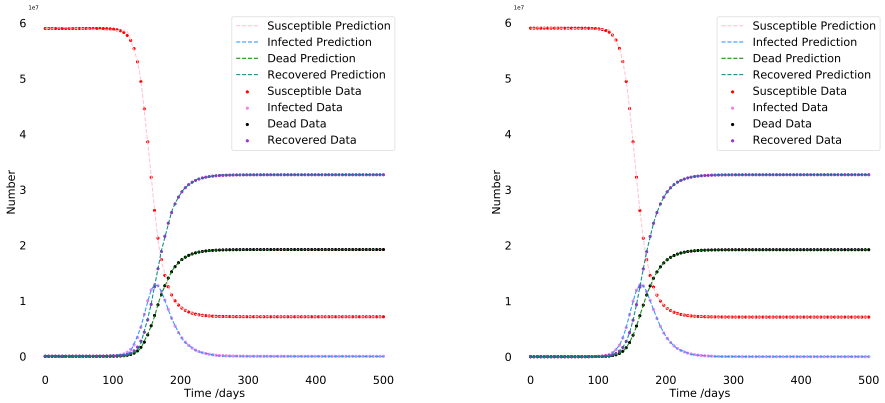


Figure 6: 1000% search range for NN Output (Left) vs LSODA generation from Learnable Parameters (Right).

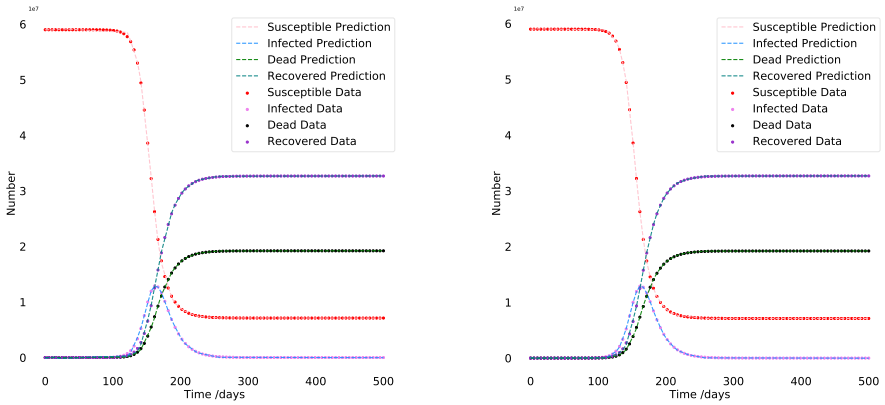


Figure 7: 10000% search range for NN Output (Left) vs LSODA generation from Learnable Parameters (Right).

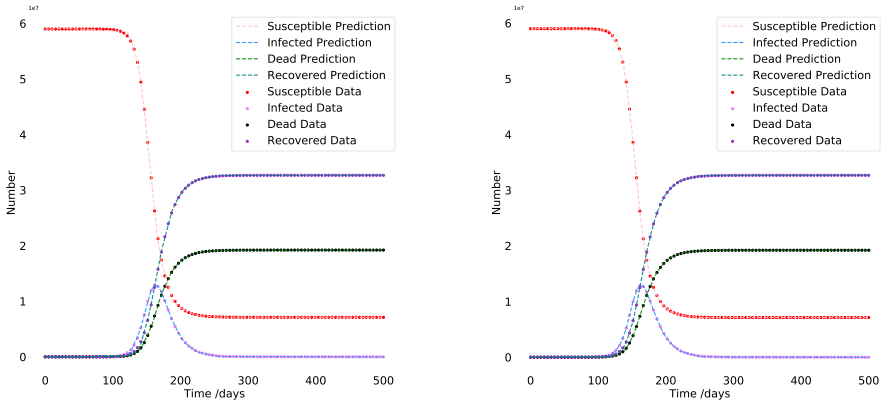


Figure 8: 100000% search range for NN Output (Left) vs LSODA generation from Learnable Parameters (Right).

Table 2: Parameter Values for various Uncorrelated Gaussian Noises.

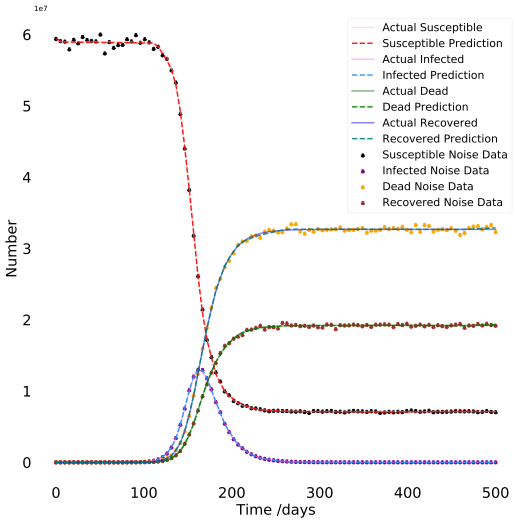
Param.	Actual Value	Param. Found	% Relative Error		(S,I,D,R)	
1% Uncorrelated Gaussian Noise						
α	0.191	0.1938	1.46			
β	0.05	0.05	0		Error NN	(0.003, 0.014, 0.002, 0.004)
γ	0.0294	0.0294	0		Error Learnable Parameters	(0.034, 0.133, 0.025, 0.025)
5% Uncorrelated Gaussian Noise						
α		0.1927	0.89			
β		0.0513	2.6		Error NN	(0.029, 0.026, 0.015, 0.019)
γ		0.0275	6.46		Error Learnable Parameters	(0.03, 0.11, 0.05, 0.049)
10% Uncorrelated Gaussian Noise						
α		0.1927	0.89			
β		0.0489	2.2		Error NN	(0.061, 0.051, 0.039, 0.026)
γ		0.0335	13.94		Error Learnable Parameters	(0.021, 0.059, 0.084, 0.069)
20% Uncorrelated Gaussian Noise						
α		0.189	1.04			
β		0.0564	12.8		Error NN	(0.072, 0.143, 0.059, 0.035)
γ		0.0274	6.8		Error Learnable Parameters	(0.089, 0.288, 0.16, 0.056)

4.1.2 Influence of noise

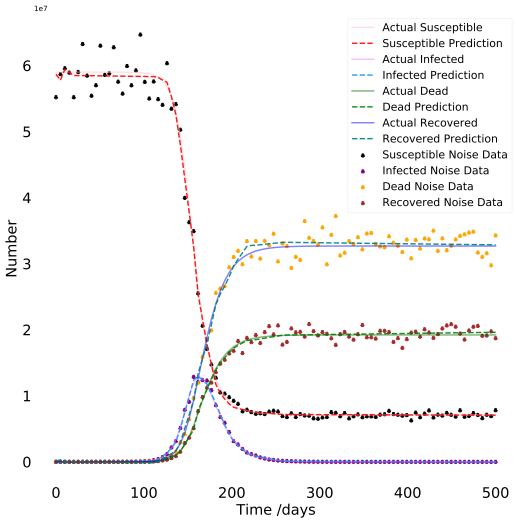
Next, to show the robustness of DINNs, we generated various amounts of uncorrelated Gaussian noise. The models were trained for 1.4 million iterations (roughly 1 hour), using parameter ranges of 1000% variation and similar learning parameters (e.g., learning rate) as the previous section. We used a 4 layer neural network with 20 neurons each, and 100 data points. The experiments showed that even with a very high amount of noise such as 20%, DINNs achieves accurate results with maximum relative error of 0.143 on learning the system. That being said, the exact parameters were harder to learn in that amount of noise. It appears that the models may need further training to stabilize the parameters, as there were some variations in the amount of noise versus the accuracy. Figure 9 shows DINN's predictions on 1%, 5%, 10% and 20% uncorrelated gaussian noise respectively. Table 2 summarizes the estimated optimal parameters for these varying noises.

4.1.3 Influence of data variability

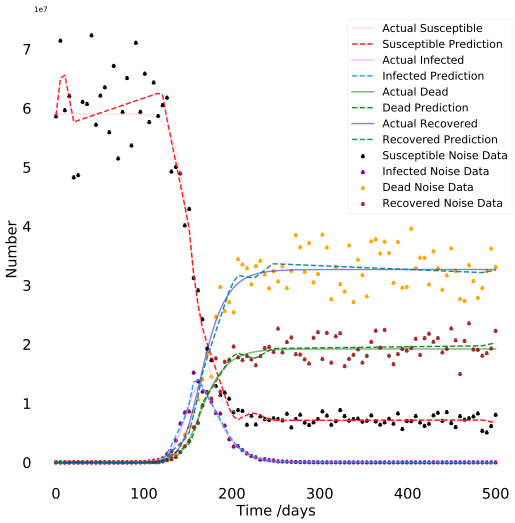
Next, we trained our models with various amounts of data: 10, 20, 100, and 1000 points (See Figure 10). The models were trained for 700,000 iterations, consisting of 4 layers with 20 neurons each, and 1×10^{-6} learning rate. Our analysis shows that there was a big increase in the parameters accuracy from 10 points to 20 points. The model that was trained on 1000 data points performed the best compared to the others. Note that even with 20 data points the model learns the system incredibly well (See



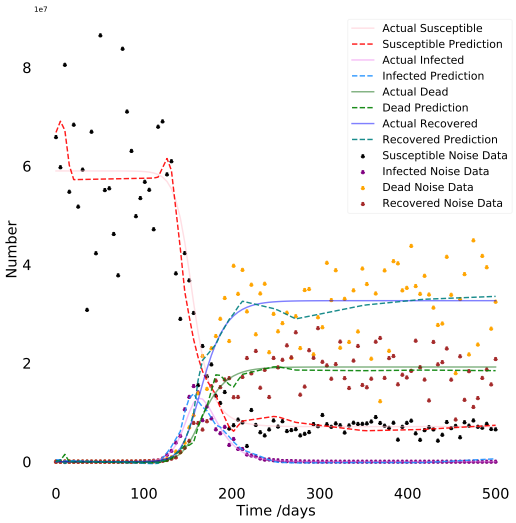
(a) 1% – Neural Network's System.



(b) 5% – Neural Network's System.



(c) 10% – Neural Network's System.



(d) 20% – Neural Network's System.

Figure 9: DINNs performance with varying Uncorrelated Gaussian Noise.

Table 3: Performance of DINNs for 20 data points.

Param.	Actual Value	Param. Found	% Relative Error		(S,I,D,R)	
α		0.1907	0.15			
β		0.0491	1.8		Error NN	($7e^{-4}$, 0.009, 6e−4, $6e^{-4}$)
γ		0.0338	14.96		Error Learnable Parameters	(0.055, 0.182, 0.077, 0.088)

Table 4: Influence of Architecture Variation with (S, I, D, R) error from neural network output.

	Neurons Per Layer		
Layers	10	20	64
2	(0.030, 0.109, 0.027, 0.027)	(0.024, 0.15, 0.022, 0.022)	(0.005, 0.038, 0.004, 0.004)
4	(0.002, 0.027, 0.004, 0.004)	(0.001, 0.007, 0.001, $8e^{-4}$)	($8e^{-4}$, 0.004, $7e^{-4}$, $7e^{-4}$)
8	(0.001, 0.008, 0.002, 0.002)	($4e^{-4}$, 0.002, $5e^{-4}$, $4e^{-4}$)	($3e^{-4}$, 0.001, $2e^{-4}$, $1e^{-4}$)
12	(0.001, 0.008, 0.002, 0.002)	($5e^{-4}$, 0.002, $8e^{-4}$, $6e^{-4}$)	($2e^{-4}$, 0.001, $2e^{-4}$, $2e^{-4}$)

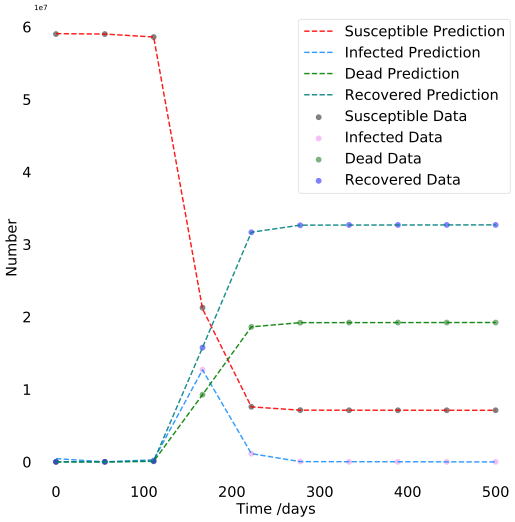
Table 3). The left-hand side of the table shows the parameters and values found after training. The right-hand side as before shows the two errors: "Error NN" is the relative MSE loss error from the system that the neural network output (what DINNs believes the systems' dynamics look like), and "Error Learnable Parameters" is the relative MSE loss error from the LSODA generated system using the parameters found values. DINNs was also compared against a traditional least-squares approach using Gauss-Newton and the Nelder-mead method with variable data points. Both of these techniques require an initial guess for the parameters which was chosen to be $(0.1, 0.1, 0.1)$. Additionally, a search range which the algorithms could search for parameters within was also included to be $(0, 2)$. The results of the two traditional approaches are illustrated in Figure 11 and Figure 12 respectively, which show that these traditional approaches only start to perform comparable to DINNs when there are more data points. While Nelder-Mead was a little better than Gauss-Newton method, both could not perform as well as DINNs for a minimal dataset. More experiments were also conducted by increasing the search range for the parameters, but both the Gauss-Newton and Nelder-Mead performed worse.

4.1.4 Influence of neural network architectures

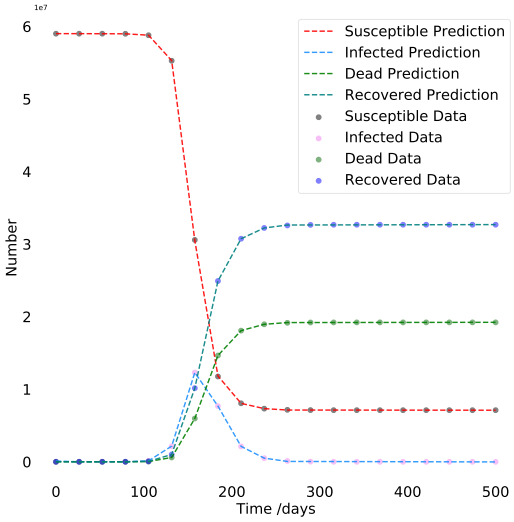
In the next computational experiment, we examined the effect that wider or deeper neural network architecture has on DINNs. The models were trained on 100 data points, using parameter ranges of 1000%, a learning rate of 1×10^{-6}, and 700,000 iterations. Tables 4 and 5 show a clear decrease in error as one increases the amount of neurons per layer. Specifically, Table 4 itemizes the (S,I,D,R) error from the neural network's output. For the Neural network architecture variations (depth and width), relative MSE errors were reported on the predicted NN system. Table 5 itemizes similar findings for LSODA generation of the learning parameters. There also seem to be a clear decrease in error as the number of layers increase. However, the error seem to stabilize around 8 layers, with very minor performance increase in 12 layers.

Table 5: Influence of Architecture Variation with (S, I, D, R) error from LSODA.

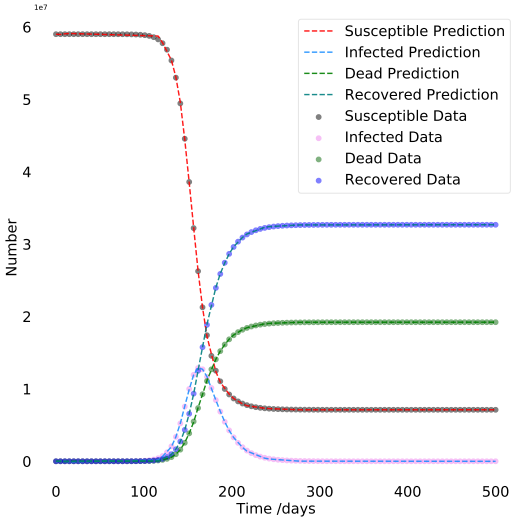
	Neurons Per Layer		
Layers	10	20	64
2	(0.132, 0.519, 0.088, 0.111)	(0.106, 0.423, 0.083, 0.077)	(0.001, 0.009, 0.019, 0.011)
4	(0.038, 0.148, 0.026, 0.029)	(0.064, 0.256, 0.045, 0.050)	(0.009, 0.044, 0.010, 0.008)
8	(0.036, 0.138, 0.033, 0.024)	(0.027, 0.107, 0.018, 0.022)	(0.057, 0.234, 0.045, 0.043)
12	(0.036, 0.138, 0.033, 0.024)	(0.022, 0.091, 0.015, 0.019)	(0.017, 0.076, 0.017, 0.017)



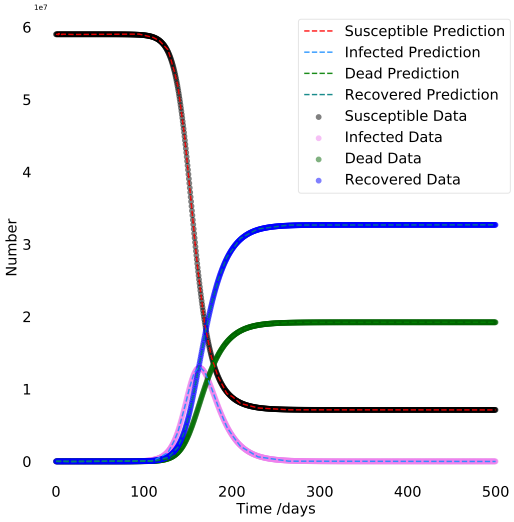
(a) 10 points – Neural Network's System.



(b) 20 points – Neural Network's System.



(c) 100 points – Neural Network's System.



(d) 1000 points – Neural Network's System.

Figure 10: DINNs performance for increasing Data Points: 10 (top left), 20 (top right), 100 (bottom left), 1000 (bottom right).

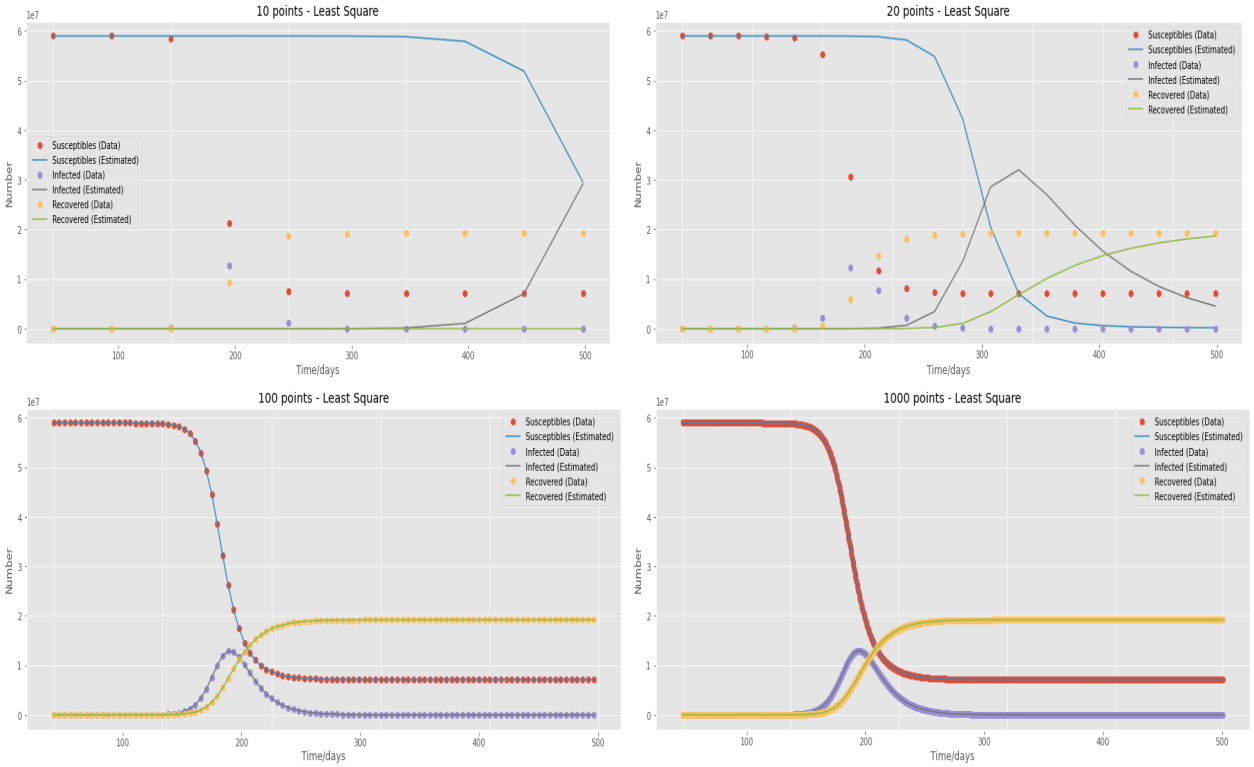


Figure 11: Least-squares performance for increasing Points: 10 (top left), 20 (top right), 100 (bottom left), 1000 (bottom right).

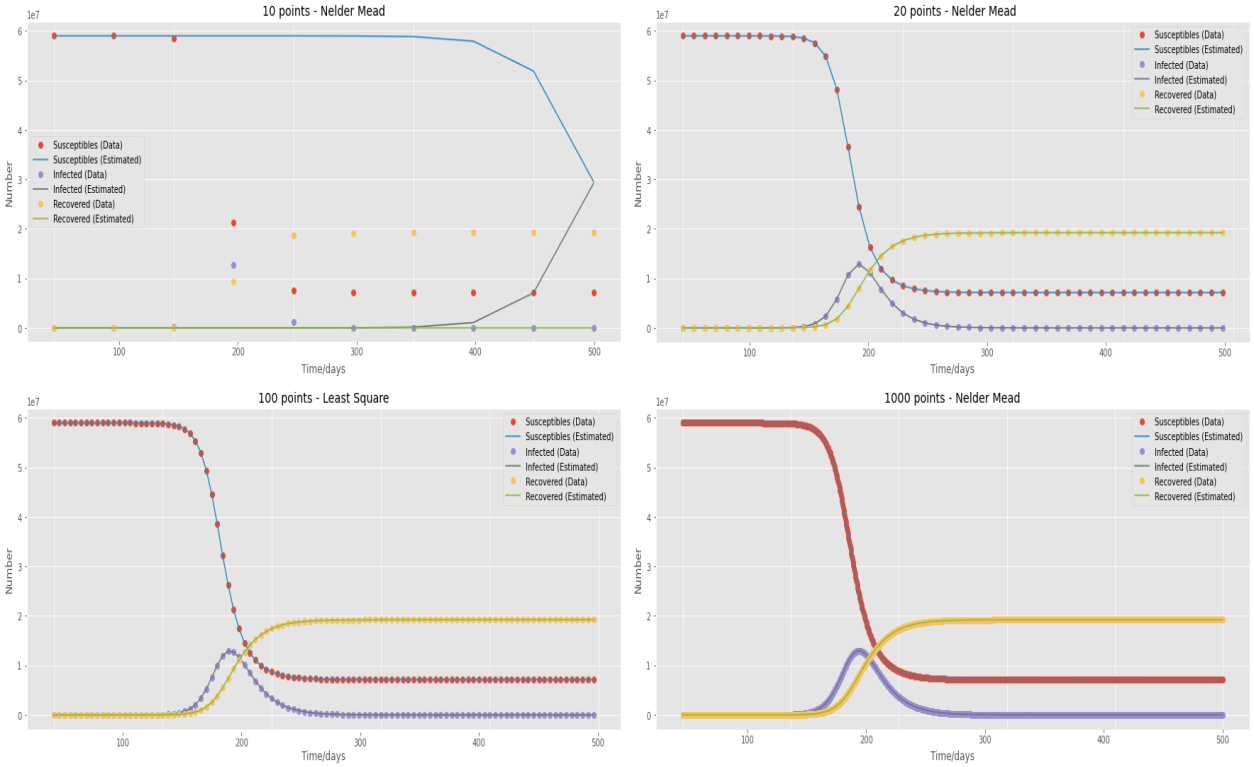


Figure 12: Nelder-Mead performance for increasing Points: 10 (top left), 20 (top right), 100 (bottom left), 1000 (bottom right).

Table 6: Learning Rate & Step Size vs Training Time.

	Step Size Up		
Learning Rate	100	1000	10000
1×10^{-5}	2min 31s	2min 57s	3min 16s
1×10^{-6}	21min 11s	20min 59s	18min 43s
1×10^{-8}	>8hrs	>8hrs	>8hrs

4.1.5 Influence of learning rates

We found that quickly increasing the learning rates and then quickly decreasing it to a steady value allows the network to learn well. One such learning rate schedule is PyTorch's CyclicLR learning rate scheduler. To show the importance of learning rate in the amount of needed training time, we trained DINNs with several values: $1 \times 10^{-5}, 1 \times 10^{-6}, 1 \times 10^{-8}$ as well as different step size for each one: 100, 1000, 10000. We used 4 layers with 20 neurons each, and 100 data points. The time was measured from the moment the network started training and until the loss was smaller than 4×10^{-4}, which usually corresponds to learning the system almost perfectly. As can be seen from the results (Table 6) both the minimum learning rate and the step size play an important role in learning the system. Reducing the learning rate to a small value too quickly may result in hours of training time instead of minutes. As an afterthought, this might be the reason why most of the systems were taking so long to train (>10 hrs), while the COVID system took <25 minutes.

4.1.6 Application of DINNs to real data

Finally, to verify that DINNs is in fact as reliable as it appears, we used 310 days (04-12-2020 to 02-16-2021) of real US data from Dong et al. (2020). We trained a neural network that learned the cumulative cases of susceptible, infected, dead, and recovered, and predicted the cases for a future month. Specifically, out of those 310 days we gave the network 280 days worth of data and asked it to predict each compartment's progression a month (30 days) into the future. The network received 31 data points (1 per 10 days), was trained for 100k epochs (roughly 5 minutes), had 8 layers with 20 neurons each, a 1000% parameters variation, and 1×10^{-5} learning rate.

Our results suggest that the learnable parameters found were quite different from the parameters in the literature ($\alpha = 0.0176$ instead of 0.191, $\beta = 0.0046$ instead of 0.05, and $\gamma = 0.0001$ instead of 0.0294). This may imply that either the data was different from the initial data distribution used in the literature (Anastassopoulou et al., 2019), or that as other authors mentioned these are time-varying parameters rather than constant ones. As seen from figure 13, the cumulative cases had less data variation and were fairly easy to learn. Additionally, it appears as DINNs managed to accurately predict the future month on each compartment.

4.1.7 Influence of missing data

So far we assumed that we have all the data for each compartment. However, this is often not the case. For example, there is a lot of data that went unreported during COVID-19. To test the reliability of DINNs, we tested the method on the SIRD model again which was trained on 100 data points, were given the known parameters from the literature, and were only given the initial conditions for the missing data. The model was trained with 1×10^{-6} learning rate for 1 million iterations (roughly 1 hour). Our results show that DINNs can in fact learn systems even when given partial data. However, it is important to note that the missing data compartments should be in at least one other compartment in order to get good results. For example, when we tried to remove the COVID recovered compartment (i.e., R), DINNs learned S, I, and D nearly perfectly. However, it did not do very well on R. That is because R is not in any of the other equations. The neural networks' systems outputs and their losses for COVID model was (0.003, **0.078**, 0.003, 0.003). The prediction using these values is shown in Figure 14.

4.2 Application of DINNs to other infectious diseases

In this sections, we apply DINNs to multiple infectious diseases. Note that we chose smaller ranges for the following diseases for demonstrating that DINN can in fact identify the systems and one set of parameters that match the literature they came from, as in many of these systems there exist a large set of parameters that can generate them. However, one can easily expand the ranges as done in a previous section. Similarly to the previous sections we report relative errors, except when the true value of the parameter is zero, which then we use the absolute error.

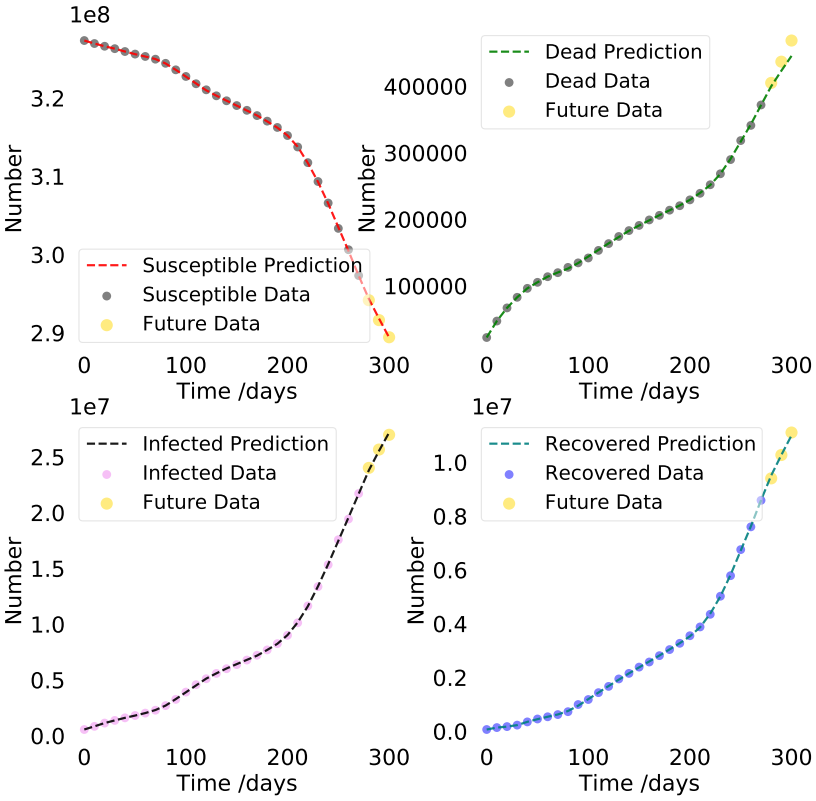


Figure 13: DINN's output on COVID real-life cumulative cases over 310 days.

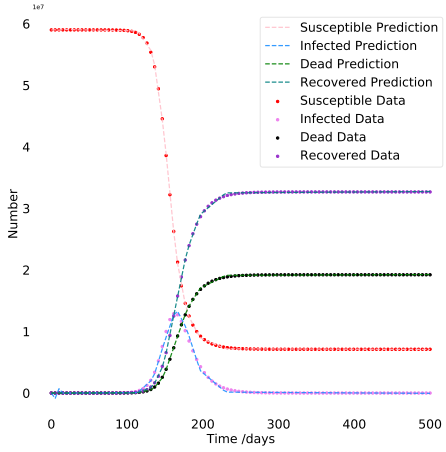


Figure 14: Performance of DINNs on Missing data for COVID.

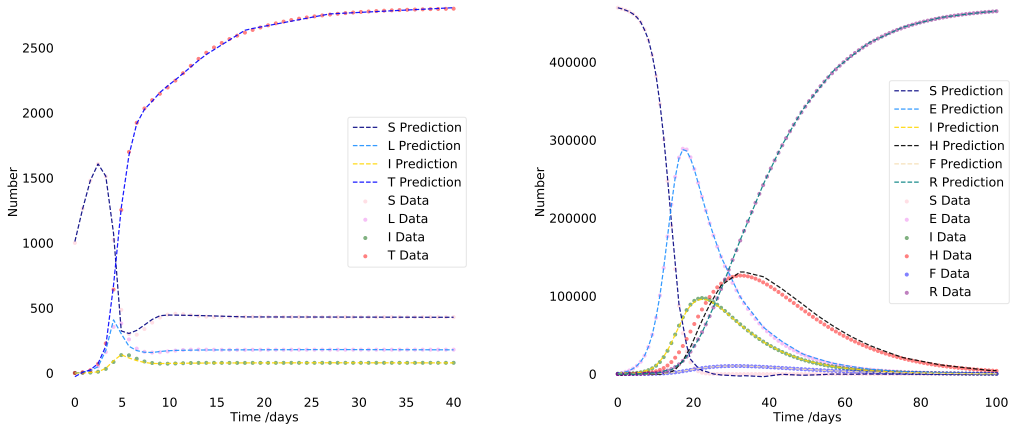


Figure 15: Performance of DINNs on Missing data for COVID Tuberculosis (left) and Ebola (right).

First, we employ DINNs to two systems modeling diseases with missing data as in the last section. These include the following:

- A Tuberculosis SLIT model (Castillo-Chavez and Feng, 1997) with missing data on Latent compartment L and infected individuals I given by

$$\frac{dS}{dt} = \delta - \frac{\beta cSI}{N} - \mu S, \qquad \frac{dL}{dt} = \frac{\beta cSI}{N} - (\mu + k + r_1)L + \frac{\beta' cT}{N},$$
$$\frac{dI}{dt} = kL - (\mu + d)I - r_2 I, \qquad \frac{dT}{dt} = r_1 L + r_2 I - \frac{\beta' cT}{N} - \mu T.$$

- An Ebola SEIHFR model (Legrand et al., 2007) with missing data on hospitalized cases H given by

$$\frac{dS}{dt} = -\frac{1}{N}(\beta_1 SI + \beta_h SH + \beta_f SF), \qquad \frac{dE}{dt} = \frac{1}{N}(\beta_1 SI + \beta_h SH + \beta_f SF) - \alpha E,$$
$$\frac{dI}{dt} = \alpha E - (\gamma_h \theta_1 + \gamma_i(1-\theta_1)(1-\delta_1) + \gamma_d(1-\theta_1)\delta_1)I, \qquad \frac{dH}{dt} = \gamma_h \theta_1 I - \big(\gamma_d h \delta_2 + \gamma_i h(1-\delta_2)\big)H,$$
$$\frac{dF}{dt} = \gamma_d(1-\theta_1)\delta_1 I + \gamma_d h \delta_2 H - \gamma_f F, \qquad \frac{dR}{dt} = \gamma_i(1-\theta_1)(1-\delta_1)I + \gamma_i h(1-\delta_2)H + \gamma_f F.$$

The models were also trained on 100 data points, were given the known parameters from the literature, and were only given the initial conditions for the missing data. The tuberculosis model was trained with 1×10^{-5} learning rate for 100k iterations. The Ebola model was trained with 1×10^{-6} learning rate for 800,000 iterations. The neural networks' systems outputs and their losses for the Tuberculosis model was (0.041, **0.086**, **0.051**, 0.004) and the Ebola model was (0.013, 0.011, 0.014, **0.103**, 0.007, 0.001). Figure 14 illustrate the outputs for the respective models.

4.2.1 A summary of DINNs applied to eleven diseases

Expanding on the relatively simple SIRD model for COVID that was used for simplicity to demonstrate the capability of DINNs, here we apply the method to ten other highly infectious diseases, namely Anthrax, HIV, Zika, Smallpox, Tuberculosis, Pneumonia, Ebola, Dengue, Polio, and Measles. These diseases vary in their complexity, ranging from a system of three to nine ordinary differential equations, and from a few parameters to over a dozen. Table 7 provides a summary of our analysis. Specifically, it itemizes for each disease its best, worst, and median parameter estimate error. In the subsequent subsections, for each of the diseases described by a system of differential equations, we identify the relative error for the disease from LSODA generation of the learnable parameters, a table representing parameter values (actual and computed values) with their range and percentage relative error, and a graph of the prediction from the data.

Table 7: Summary of the analysis for eleven diseases.

Disease	Best	Worse	Median
COVID	0.2	1.151	1.02
Anthrax	0.5754	6.0459	2.4492
HIV	0.007515	3.811689	0.829756
Zika	0.0588	5.8748	0.7261
Smallpox	0.0882	10.8598	4.9239
Tuberculosis	0.5424	11.0583	3.8952
Pneumonia	0.0005	11.6847	1.6372
Ebola	0.2565	9.6403	1.1244
Dengue	0.2696	9.7723	0.8796
Polio	0	0.4168	0.3587
Measles	2.9999	12.704	3.1453

4.2.2 COVID-19

The DINNs COVID system considered was given by

$$\frac{dS}{dt} = -\frac{\alpha}{N}SI, \qquad \frac{dI}{dt} = \frac{\alpha}{N}SI - \beta I - \gamma I, \qquad \frac{dD}{dt} = \gamma I, \qquad \frac{dR}{dt} = \beta I.$$

The model used 8 layers with 20 neurons per layer, 1×10^{-6} min learning rate, and was trained for 400k iterations (about 20 minutes). Figure 16 and table 8 show our results. The relative error corresponding to the SIDR system was (0.022, 0.082, 0.022, 0.014).

4.2.3 HIV

The DINN HIV model had 8 layers with 20 neurons per layer, 1×10^{-8} min learning rate, and was trained for 25mil iterations (about 22 hours). Figure 17 and table 9 show our results. The relative error corresponding to the system was (0.008, 0.002, 0.003).

System:

$$\frac{dT}{dt} = s - \mu_T T + rT\left(1 - \frac{T+I}{T_{max}} - k_1 VT\right) \qquad \frac{dI}{dt} = k_1' VT - \mu_I I \qquad \frac{dV}{dt} = N\mu_b I - k_1 VT - \mu_V V$$

4.2.4 Smallpox

The DINN Smallpox model had 8 layers with 20 neurons per layer, $1e^{-7}$ min learning rate, and was trained for 12mil iterations (about 14 hours). Figure 18 and table 10 show our results. The relative error corresponding to the system was (0.033, 0.053, 0.045, 0.060, 0.014, 0.036, 0.027, 0.021).

System:

$$\begin{aligned}
\frac{dS}{dt} &= \chi_1(1-{}_1)Ci - \beta(\phi+\rho-\phi\rho)SI & \frac{dEn}{dt} &= \beta\phi(1-\rho)SI - \alpha En \\
\frac{dEi}{dt} &= \beta\phi\rho SI - \left(\chi_{12} + \alpha(1-{}_2)\right)Ei & \frac{dCi}{dt} &= \beta\rho(1-\phi)SI - \chi_1 Ci \\
\frac{dI}{dt} &= \alpha(1-\theta)En - (\theta+\gamma)I & \frac{dQ}{dt} &= \alpha(1-{}_2)Ei + \theta(\alpha En + I) - \chi_2 Q \\
\frac{dU}{dt} &= \gamma I + \chi_2 Q & \frac{dV}{dt} &= \chi_1({}_2 Ei + {}_1 Ci)
\end{aligned}$$

4.2.5 Tuberculosis

The DINN Tuberculosis model had 8 layers with 20 neurons per layer, $1e^{-7}$ min learning rate, and was trained for 10mil iterations (about 12 hours). Figure 19 and table 11 show our results. The relative error corresponding to the system was (0.030, 0.034,

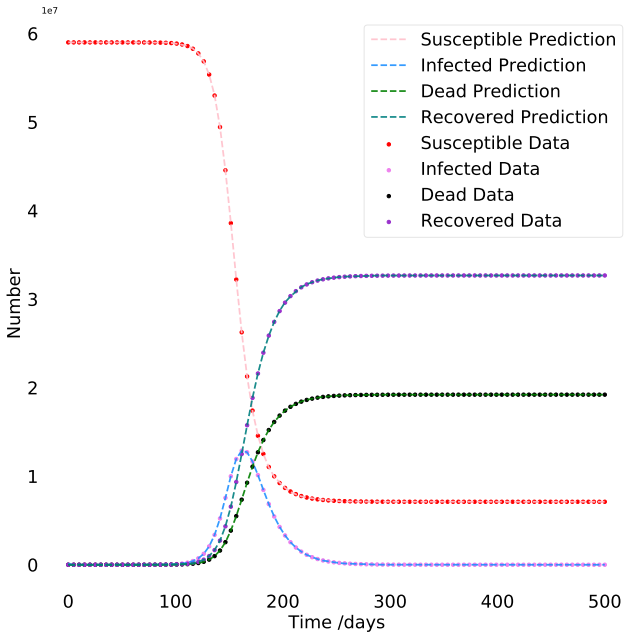


Figure 16: COVID: Neural Network Output.

Table 8: COVID: Parameter Estimation.

Parameter	Actual Value	Range	Parameter Found	% Relative Error
α	0.191	(-1,1)	0.1932	1.151
β	0.05	(-1,1)	0.0501	0.2
γ	0.0294	(-1,1)	0.0297	1.02

Table 9: HIV: Parameter Estimation.

Parameter	Actual Value	Range	Parameter Found	% Relative Error
s	10	(9.9,10.1)	10.000751	0.007515
μ_T	0.02	(0.018,0.022)	0.020762	3.811689
μ_I	0.26	(0.255,0.265)	0.261271	0.488758
μ_b	0.24	(0.235,0.245)	0.241747	0.727760
μ_V	2.4	(2.5,2.3)	2.419914	0.829756
r	0.03	(0.029,0.031)	0.030605	2.015910
N	250	(247.5,252.5)	249.703094	0.118762
T_{max}	1500	(1485,1515)	1506.543823	0.436255
k_1	$2.4 \cdot 10e^{-5}$	$(2.3 \cdot 10e^{-5}, 2.6 \cdot 10e^{-5})$	0.000246	2.447948
k_1'	$2 \cdot 10e^{-5}$	$(1.9 \cdot 10e^{-5}, 2.1 \cdot 10e^{-5})$	0.000203	1.599052

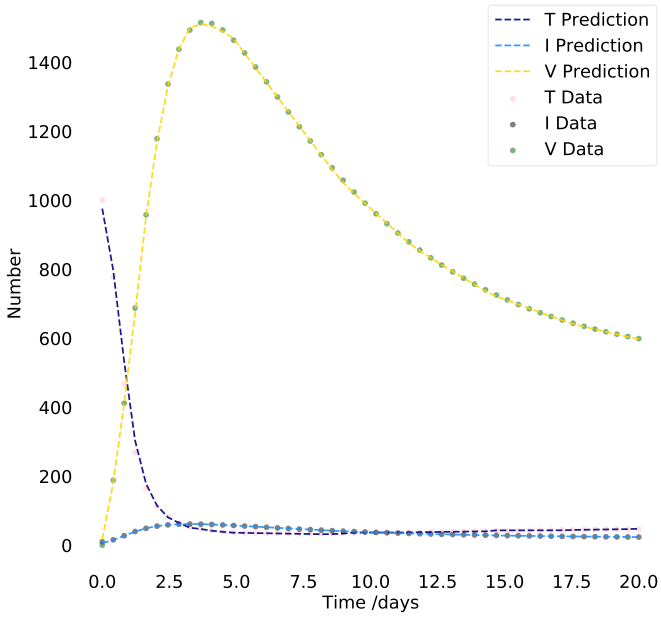


Figure 17: HIV: Neural Network Output.

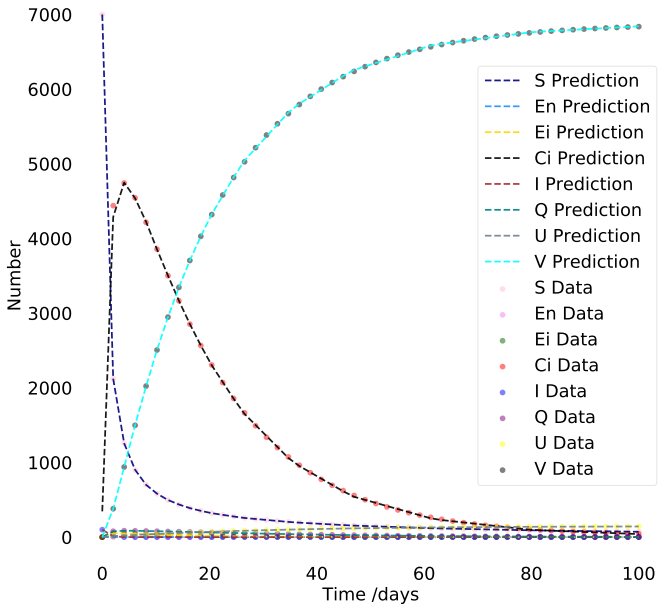


Figure 18: Smallpox: Neural Network Output.

0.034, 0.008).

System:

$$\frac{dS}{dt} = \delta - \frac{\beta cSI}{N} - \mu S \qquad \frac{dL}{dt} = \frac{\beta cSI}{N} - (\mu + k + r_1)L + \frac{\beta' cT}{N}$$
$$\frac{dI}{dt} = kL - (\mu + d)I - r_2 I \qquad \frac{dT}{dt} = r_1 L + r_2 I - \frac{\beta' cT}{N} - \mu T$$

4.2.6 Pneumonia

The DINN Pneumonia model had 8 layers with 64 neurons per layer, 1×10^{-7} min learning rate, and was trained for 25mil iterations (about 41 hours). Figure 20 and tables 12 show our results. The relative error corresponding to the system was (0.020, 0.039, 0.034, 0.019, 0.023).

System:

$$\frac{dS}{dt} = (1-p)\pi + \phi V + \delta R - (\mu + \lambda + \theta)S \qquad \frac{dV}{dt} = p\pi + \theta S - (\mu + \lambda + \phi)V$$
$$\frac{dC}{dt} = \rho\lambda S + \rho\lambda V + (1-q)\eta I - (\mu + \beta + \chi)C \qquad \frac{dI}{dt} = (1-\rho)\lambda S + (1-\rho)\lambda V + \chi C - (\mu + \alpha + \eta)I$$
$$\frac{dR}{dt} = \beta C + q\eta I - (\mu + \delta)R$$

4.2.7 Ebola

Next, we consider the Ebola model considered before. The DINN Ebola model had 8 layers with 20 neurons per layer, $1e^{-7}$ min learning rate, and was trained for 20mil iterations (about 33 hours). Figure 21 and table 13 show our results. The relative error corresponding to the SIDR system was (0.023, 0.050, 0.044, 0.062, 0.049, 0.005)

4.2.8 Dengue

The DINN Dengue model had 8 layers with 20 neurons per layer, $1e^{-7}$ min learning rate, and was trained for 35mil iterations (about 58 hours). Figure 22 and table 14 show our results.The relative error is (0.003, 0.012, 0.030, 0.054, 0.001, 0.001, 0.002).

System:

$$\frac{dSh}{dt} = \pi_h - \lambda_h Sh - \mu_h Sh \qquad \frac{dEh}{dt} = \lambda_h Sh - (\sigma_h \mu_h)Eh \qquad \frac{dIh}{dt} = \sigma_h Eh - (\tau_h + \mu_h + \delta_h)Ih$$
$$\frac{dRh}{dt} = \tau_h Ih - \mu_h Rh \qquad \frac{dSv}{dt} = \pi_v - \delta_v Sv - \mu_v Sv \qquad \frac{dEv}{dt} = \delta_v Sv - (\sigma_v + \mu_v)Ev$$
$$\frac{dIv}{dt} = \sigma_v Ev - (\mu_v + \delta_v)Iv$$

4.2.9 Anthrax

The DINN Anthrax model had 8 layers with 64 neurons per layer, $1e^{-8}$ min learning rate, and was trained for 55mil iterations (about 91 hours). Figure 23 and tables 15, 16 show our results.

System:

$$\frac{dS}{dt} = r(S+I)\left(1 - \frac{S+I}{K}\right) - \eta_a AS - \eta_c SC - \eta_i \frac{SI}{S+I} - \mu S + \tau I \qquad \frac{dA}{dt} = -\sigma A + \beta C$$
$$\frac{dI}{dt} = \eta_a AS + \eta_c SC + \left(\eta_i \frac{SI}{S+I} - (\gamma + \mu + \tau)\right)I \qquad \frac{dC}{dt} = (\gamma + \mu)I - \delta(S+I)C - \kappa C$$

Table 10: Smallpox: Parameter Estimation.

Parameter	Actual Value	Range	Parameter Found	% Relative Error
χ_1	0.06	(0.054,0.066)	0.0554	7.7222
χ_2	0.04	(0.036,0.044)	0.0380	4.9239
$_1$	0.975	(0.86,1.04)	0.9839	0.9089
$_2$	0.3	(0.27,0.33)	0.2841	5.2848
ρ	0.975	(0.86,1.04)	0.9759	0.0882
θ	0.95	(0.86,1.04)	0.9050	4.7371
α	0.068	(0.061,0.075)	0.0626	8.5490
γ	0.11	(0.10,0.12)	0.1034	10.8598

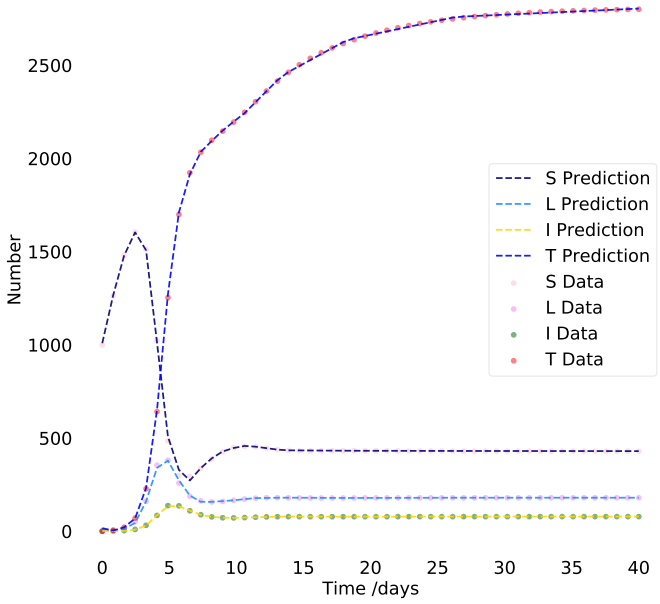


Figure 19: Tuberculosis: Neural Network Output.

Table 11: Tuberculosis: Parameter Estimation.

Parameter	Actual Value	Range	Parameter Found	% Relative Error
δ	500	(480,520)	509.4698	1.8587
β	13	(9,15)	12.5441	3.6341
c	1	(-1,3)	1.0405	3.8952
μ	0.143	(0.1,0.3)	0.1474	3.0142
k	0.5	(0,1)	0.5396	7.3433
r_1	2	(1,3)	1.9892	0.5424
r_2	1	(-1,3)	1.1243	11.0583
β'	13	(9,15)	13.7384	5.3746
d	0	(-0.4,0.4)	-0.0421	0.0421

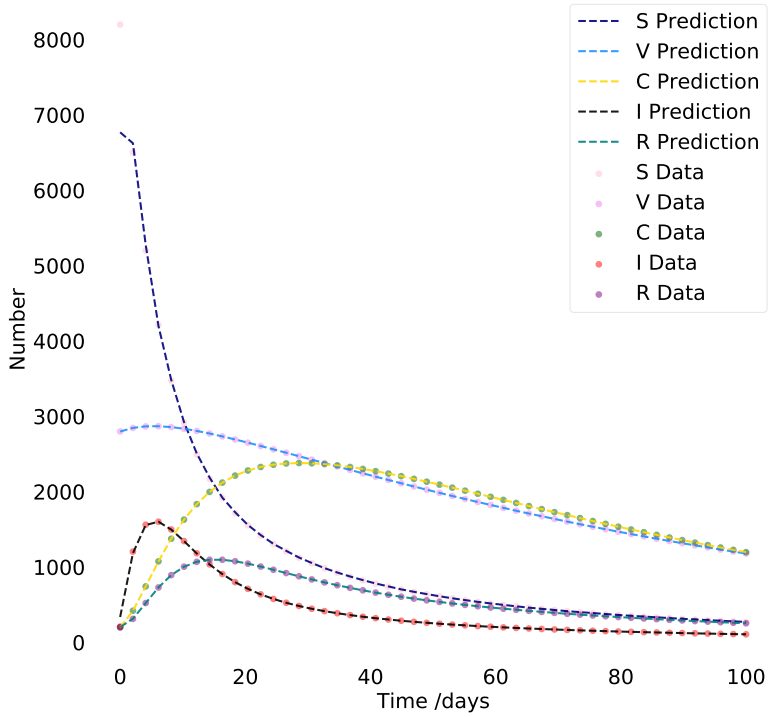


Figure 20: Pneumonia: Neural Network Output.

Table 12: Pneumonia: Parameter Estimation.

Parameter	Actual Value	Range	Parameter Found	% Relative Error
π	0.01	(0.0099,0.011)	0.0098	2.0032
λ	0.1	(0.099,0.11)	0.0990	0.9622
k	0.5	(0.49,0.51)	0.5025	0.5083
	0.002	(0.001,0.003)	0.0022	11.6847
τ	0.89	(0.87,0.91)	0.8912	0.1309
ϕ	0.0025	(0.0023,0.0027)	0.0027	7.4859
χ	0.001	(0.0009,0.0011)	0.0011	6.7374
p	0.2	(0.19, 0.21)	0.2033	1.6372
θ	0.008	(0.0075,0.0085)	0.0084	4.8891
μ	0.01	(0.009,0.011)	0.0092	8.4471
α	0.057	(0.056,0.058)	0.0570	0.0005
ρ	0.05	(0.049,0.051)	0.0508	1.5242
β	0.0115	(0.0105,0.0125)	0.0122	5.8243
η	0.2	(0.19,0.21)	0.2023	1.1407
q	0.5	(0.49,0.51)	0.4960	0.8003
δ	0.1	(0.09,0.11)	0.1038	3.7502

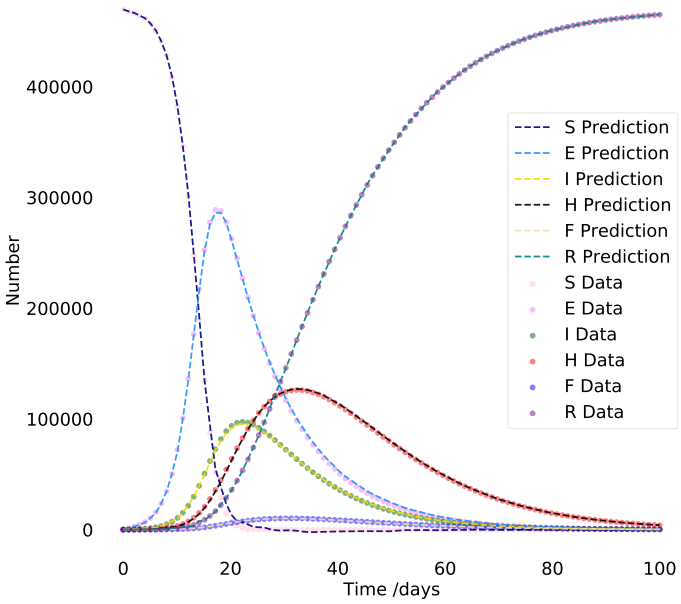


Figure 21: Ebola: Neural Network Output.

Table 13: Ebola: Parameter Estimation.

Parameter	Actual Value	Range	Parameter Found	% Relative Error
β_1	3.532	(3.5,3.56)	3.5589	0.7622
β_h	0.012	(0.011,0.013)	0.0129	7.8143
β_f	0.462	(0.455,0.465)	0.4638	0.3976
α	1/12	(0.072,0.088)	0.0866	3.9320
γ_h	1/4.2	(0.22,0.28)	0.2471	3.7853
θ_1	0.65	(0.643,0.657)	0.6523	0.3477
γ_i	0.1	(0.099,0.11)	0.0904	9.6403
δ_1	0.47	(0.465,0.475)	0.4712	0.2565
γ_d	1/8	(0.118,0.122)	0.1205	3.6124
δ_2	0.42	(0.415,0.425)	0.4247	1.1244
γ_f	0.5	(0.45,0.55)	0.5196	3.9246
γ_{ih}	0.082	(0.081,0.083)	0.0811	1.0932
γ_{dh}	0.07	(0.069,0.071)	0.0710	0.7563

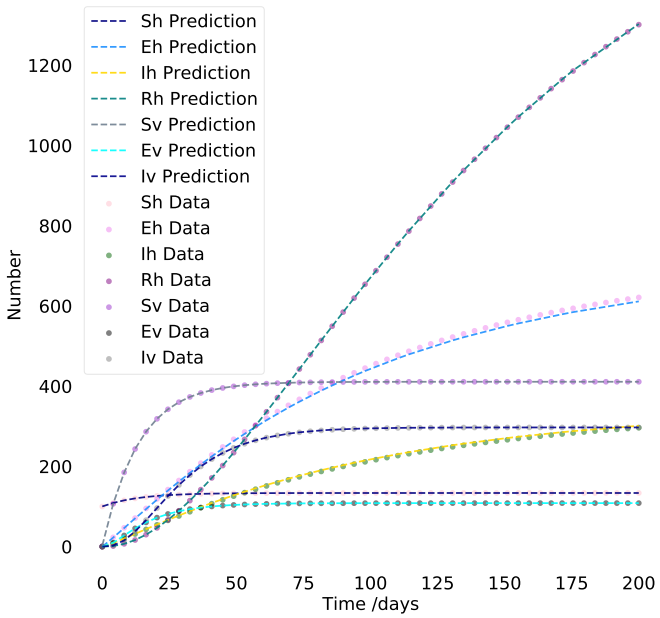


Figure 22: Dengue: Neural Network Output.

Table 14: Dengue: parameters, their values, the parameters search range that DINN was trained on, the parameters found after training, and the relative error percentage.

Parameter	Actual Value	Range	Parameter Found	% Relative Error
π_h	10	(9.9,10.1)	9.9317	0.6832
π_v	30	(29.7,30.3)	29.8542	0.4859
λ_h	0.055	(0.054,0.056)	0.0552	0.2696
λ_v	0.05	(0.049,0.051)	0.0506	1.2876
δ_h	0.99	(0.9,1.1)	0.9643	2.5967
δ_v	0.057	(0.056,0.058)	0.0567	0.5294
μ_h	0.0195	(0.0194,0.0196)	0.0194	0.3835
μ_v	0.016	(0.015,0.017)	0.0159	0.8796
σ_h	0.53	(0.52,0.54)	0.5372	1.3567
σ_v	0.2	(0.19,0.21)	0.1989	0.5483
τ_h	0.1	(0.05,0.15)	0.0902	9.7723

Table 15: Anthrax: relative error from LSODA generation of the learnable parameters.

(S, I, A, C) Error
(0.052, 0.144, 0.171, 0.171)

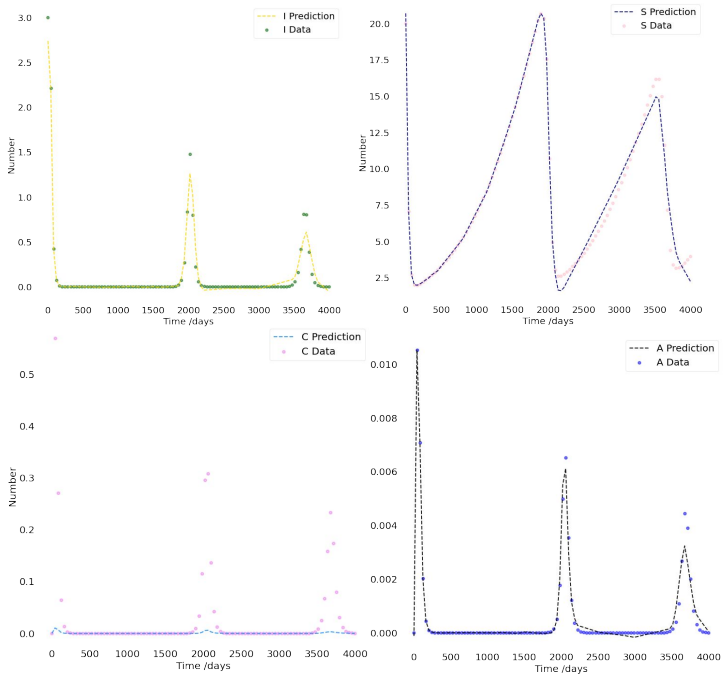


Figure 23: Anthrax: Neural Network Output.

Table 16: Anthrax: parameters, their values, the parameters search range that DINN was trained on, the parameters found after training, and the relative error percentage.

Parameter	Actual Value	Range	Parameter Found	% Relative Error
r	1/300	(0.003,0.0036)	0.0034	1.2043
μ	1/600	(0.0014,0.0018)	0.0017	0.5754
κ	0.1	(0.99,0.11)	0.1025	2.5423
η_a	0.5	(0.49,0.51)	0.5035	0.7022
η_c	0.1	(0.09,0.11)	0.1024	2.4492
η_i	0.01	(0.09,0.011)	0.0106	6.0459
τ	0.1	(0.09,0.11)	0.0976	2.4492
γ	1/7	(0.13,0.15)	0.1444	1.0542
δ	1/64	(0.03,0.07)	0.0512	2.3508
K	100	(98,102)	100.6391	0.6391
β	0.02	(0.0018,0.0022)	0.0021	6.5466
σ	0.1	(0.09,0.11)	0.1051	5.1029

4.2.10 Polio

The DINN Polio model had 8 layers with 64 neurons per layer, $1e^{-8}$ min learning rate, and was trained for 40mil iterations (about 66 hours). Figure 24 and tables 17, 18 show our results.

System:

$$\frac{dSc}{dt} = \mu N - \left(\alpha + \mu + \frac{\beta_{cc}}{Nc}Ic + \frac{\beta_{ca}}{Nc}Ia\right)Sc \qquad \frac{dSa}{dt} = \alpha Sc - \left(\mu + \frac{\beta_{aa}}{Na}Ia + \frac{\beta_{ac}}{Na}Ic\right)Sa$$

$$\frac{dIc}{dt} = \left(\frac{\beta_{cc}}{Nc}Ic + \frac{\beta_{ca}}{Nc}Ia\right)Sc - (\gamma_c + \alpha + \mu)Ic \qquad \frac{dIa}{dt} = \left(\frac{\beta_{ac}}{Na}Ic + \frac{\beta_{aa}}{Na}Ia\right)Sa - (\gamma_a + \mu)Ia + \alpha Ic$$

$$\frac{dRc}{dt} = \gamma_c Ic - \mu Rc - \alpha Rc \qquad \frac{dRa}{dt} = \gamma_a Ia - \mu Ra + \alpha Rc$$

4.2.11 Measles

The DINN Measles model had 8 layers with 64 neurons per layer, $1e^{-7}$ min learning rate, and was trained for 17mil iterations (about 28 hours). Figure 25 and tables 19, 20 show our results.

System:

$$\frac{dS}{dt} = \mu(N - S) - \frac{\beta SI}{N} \qquad \frac{dE}{dt} = \frac{\beta SI}{N} - (\mu\sigma)E \qquad \frac{dI}{dt} = \sigma E - (\mu + \gamma)I$$

4.2.12 Zika

The DINN Zika model had 8 layers with 64 neurons per layer, $1e^{-9}$ min learning rate, and was trained for 8mil iterations (about 13 hours). The following image has a selection of the compartments to reduce scatter in visualization. Figure 26 and tables 21, 22 show our results.

System:

$$\frac{dSh}{dt} = -ab\frac{Iv}{Nh}Sh - \beta\frac{\kappa Eh + Ih_1 + \tau Ih_2}{Nh}Sh \qquad \frac{dEh}{dt} = \theta\left(-ab\frac{Iv}{Nh}Sh - \beta\frac{\kappa Eh + Ih_1 + \tau Ih_2}{Nh}Sh\right) - V_h Eh$$

$$\frac{dIh_1}{dt} = V_h Eh - \gamma_{h1} Ih_1 \qquad \frac{dAh}{dt} = (1-\theta)\left(ab\frac{Iv}{Nh}Sh - \beta\frac{\kappa Eh + Ih_1 + \tau Ih_2}{Nh}Sh\right) - \gamma_h Ah$$

$$\frac{dIh_2}{dt} = \gamma_{h1} Ih_1 - \gamma_{h2} Ih_2 \qquad \frac{dRh}{dt} = \gamma_{h2} Ih_2 + \gamma_h Ah$$

$$\frac{dSv}{dt} = \mu_v Nv - ac\frac{\eta Eh + Ih_1}{Nh}Sv - \mu_v Sv \qquad \frac{dEv}{dt} = ac\frac{\eta Eh + Ih_1}{Nh} - (V_v + \mu_v)Ev$$

$$\frac{dIv}{dt} = V_v Ev - \mu_v Iv$$

5 Discussion and Conclusion

In this work, we have introduced Disease Informed Neural Networks (DINNs) which is a neural network approach capable of learning a number of diseases, how they spread, forecasting their progression, and finding unique parameters that are used in models to describe the disease dynamics. Specifically, for a benchmark problem we were able to study the influence in ranges of parameter estimation, noise, data variability, NN architechture, learning rates and missing data on the performance of DINNs. Our results from this work suggest that DINNs is a robust and reliable candidate that can be used as an inverse approach to characterize and learn parameters used in compartmental models for understanding dynamics of infectious diseases. To compare the performance of the proposed DINNs, we also wrote the parameter estimation in R and MATLAB that employed powerful nonlinear optimization methods such as Nelder-Mead, Gauss Newton and gradient decent methods. In all the types of simulations, we noticed DINNs outperformed and was more robust to initial parameter guesses. Especially, all the other methods failed to achieve the optimal solution if the initial guesses were far from the actual values for these optimization based methods compared to DINNs which worked extremely well.

Table 17: Polio: relative error from LSODA generation of the learnable parameters.

(Sc, Sa, Ic, Ia, Rc, Ra) Error
(0.001, 0.001, 0.017, 0.021, 0.004, 0.001)

Table 18: Polio: parameters, their values, the parameters search range that DINN was trained on, the parameters found after training, and the relative error percentage.

Parameter	Actual Value	Range	Parameter Found	% Relative Error
μ	0.02	(0.018,0.022)	0.0200	0.0200
α	0.5	(0.495,0.505)	0.5018	0.36
γ_a	18	(17.9,18.1)	18.0246	0.4168
γ_c	36	(35.8,36.2)	36.0701	0.3587
β_{aa}	40	(39,41)	40.2510	0.6275
β_{cc}	90	(89,91)	90.6050	0.6722
β_{ac}	0	(-0.001,0.001)	0.0002	0.0002
β_{ca}	0	(-0.001,0.001)	0.0004	0.0004

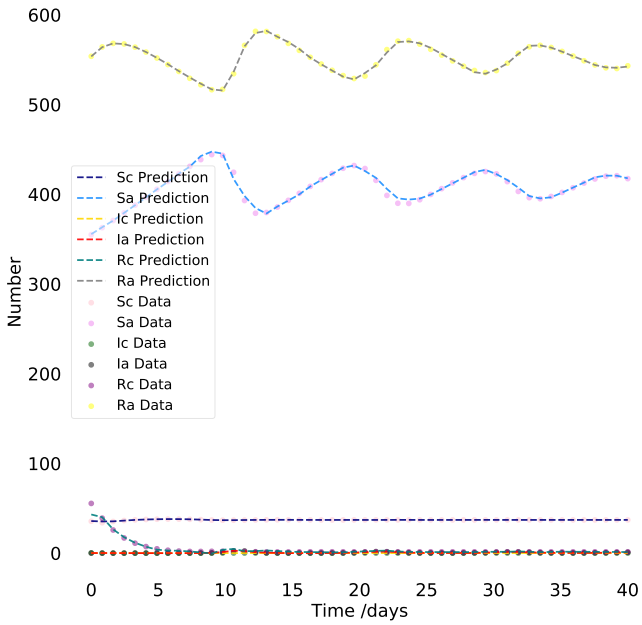


Figure 24: Polio: Neural Network Output.

Table 19: Measles: relative error from LSODA generation of the learnable parameters.

(S, E, I) Error
(0.017, 0.058, 0.059)

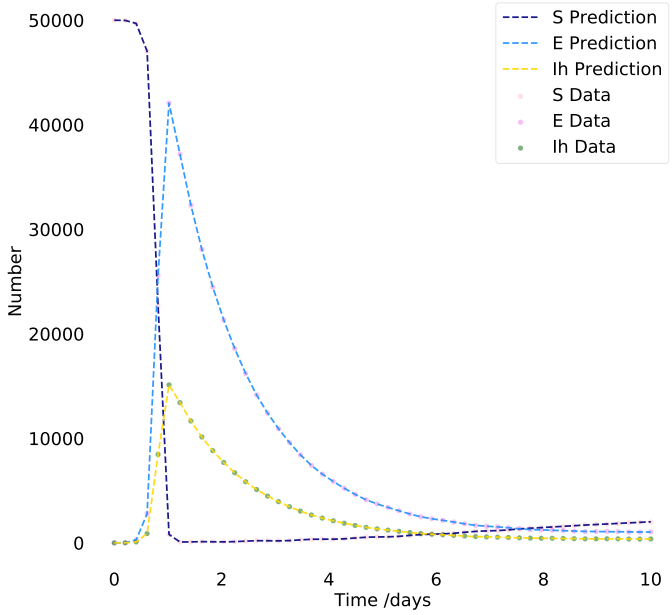


Figure 25: Measles: Neural Network Output.

Table 20: Measles: parameters, their values, the parameters search range that DINN was trained on, the parameters found after training, and the relative error percentage.

Parameter	Actual Value	Range	Parameter Found	% Relative Error
μ	0.02	(0.01,0.03)	0.0225	12.704
β_1	0.28	(0.27,0.37)	0.2700	3.5704
γ	100	(97,103)	97.0001	2.9999
σ	35.84	(33,37)	34.7127	3.1453

Table 21: Zika: relative error from LSODA generation of the learnable parameters.

$(Sh, Ih_1, Ih_2, Ah, Rh, Sv, Ev, Iv, I)$ Error
$(2.215e^{-06}, 0.017, 0.014, 0.003, 0.024, 0.091, 0.005, 0.012, 0.018, 0.018)$

Table 22: Zika: parameters, their values, the parameters search range that DINN was trained on, the parameters found after training, and the relative error percentage.

Parameter	Actual Value	Range	Parameter Found	% Relative Error
a	0.5	(0.49,0.51)	0.4997	0.0588
b	0.4	(0.39,0.41)	0.4033	0.8297
c	0.5	(0.49,0.51)	0.5015	0.3086
η	0.1	(0.09,0.11)	0.0999	0.0687
β	0.05	(0.0495,0.0505)	0.0498	0.4098
κ	0.6	(0.594,0.606)	0.6033	0.5486
τ	0.3	(0.27,0.33)	0.2902	3.2565
θ	18	(0.17.8,18.2)	17.9669	0.1838
m	5	(4.5,5.5)	5.2937	5.8748
V_h	1/5	(0.198,0.202)	0.1996	0.1798
V_v	10	(9.9,10.1)	10.0170	0.1700
γ_{h1}	1/5	(0.18,0.22)	0.1991	0.4651
γ_{h2}	1/64	(0.045,0.055)	0.0504	0.7261
γ_h	1/7	(0.139,0.141)	0.1406	1.5967
μ_v	1/14	(0.063,0.077)	0.0723	1.1806

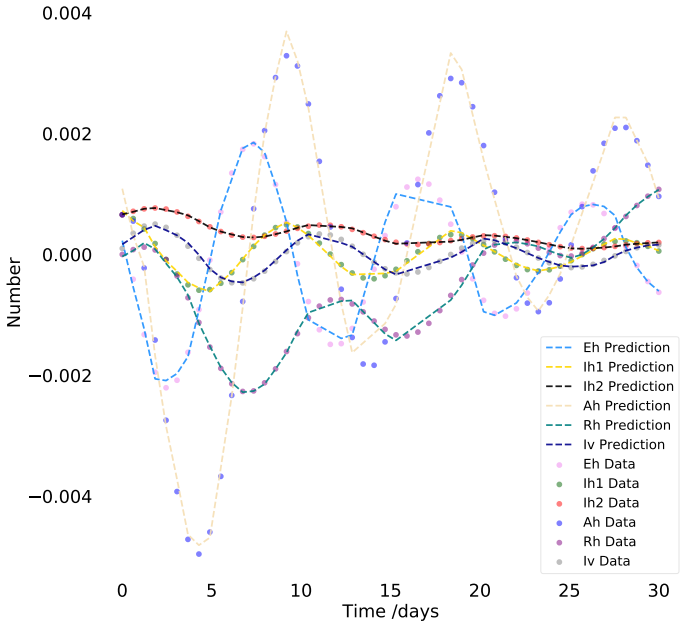


Figure 26: Zika: Neural Network Output.

Building on a simple SIRD model for COVID-19, we used it to model eleven infectious diseases and show the simplicity, efficacy, and generalization of DINNs in their respective applications. These diseases were modeled into various differential equations systems with various number of learnable parameters. We found that DINNs can quite easily learn systems with a low number of parameters and dimensions (e.g., COVID), and when the learnable parameters are known the training time can change from 50 hours to a few minutes. Moreover, it appears as if the number of dimensions does not affect the performance as much as the number of learnable parameters (e.g., see pneumonia vs ebola). From the anthrax model result we see that it is far more difficult for DINNs to learn systems which have numerous quick and sharp oscillations. That being said, looking at the polio and zika models results we can see that it is not impossible, but rather more time consuming (both in training and hyperparameter search). Also, based on the measles, tuberculosis, and smallpox models results we can see that a low number of sharp oscillations are relatively easy to learn.

It maybe noted that while the goal of this work was to introduce a powerful algorithm for predicting infectious diseases, the algorithms have the potential to be applied to complex models (for example, involving spatial dependencies, using facemasks, impact of lockdowns, etc.). Also, while DINNs presented here is shown to be robust and reliable, it can be slow to train on particular problems and there is no known theoretical guarantee of corresponding error bounds. These will be explored in forthcoming papers.

Acknowledgments

This work has been supported in part by the National Science Foundation DMS 2031027 and DMS 2031029.

References

Akman, D., O. Akman, and E. Schaefer (2018). Parameter estimation in ordinary differential equations modeling via particle swarm optimization. *Journal of Applied Mathematics 2018*. 72

Akman, O., M. R. Corby, and E. Schaefer (2016). Examination of models for cholera. *Letters in Biomathematics 3*(1), 93–118. 72

Alshakhoury, N. S. et al. (2017). *Mathematical modeling and control of MERS-CoV epidemics*. Ph. D. thesis. 72

Anastassopoulou, C., L. Russo, A. Tsakris, and C. Siettosid (2019). Data-based analysis, modelling and forecasting of the COVID-19 outbreak. 77, 86

Baydin, A. G., B. A. Pearlmutter, A. A. Radul, and J. M. Siskind (2018). Automatic differentiation in machine learning: a survey. *Journal of machine learning research 18*. 75

Bell, J. G. (1990). Mathematical biology (jd murray). *SIAM Review 32*(3), 487–489. 72

Blower, S. M. and H. Dowlatabadi (1994). Sensitivity and uncertainty analysis of complex models of disease transmission: an hiv model, as an example. *International Statistical Review/Revue Internationale de Statistique*, 229–243. 72

Brauer, F. (2017). Mathematical epidemiology: Past, present, and future. *Infectious Disease Modelling 2*(2), 113–127. 71

Brauer, F. and C. Castillo-Chávez (2001). Basic ideas of mathematical epidemiology. In *Mathematical Models in Population Biology and Epidemiology*, pp. 275–337. Springer. 77

Brauer, F., C. Castillo-Chavez, and C. Castillo-Chavez (2012). *Mathematical models in population biology and epidemiology*, Volume 2. Springer. 71, 73

Capaldi, A., S. Behrend, B. Berman, J. Smith, J. Wright, and A. L. Lloyd (2012). Parameter estimation and uncertainty quantification for an epidemic model. *Mathematical biosciences and engineering*, 553. 72

Castillo-Chavez, C. (2013). *Mathematical and statistical approaches to AIDS epidemiology*, Volume 83. Springer Science & Business Media. 72

Castillo-Chavez, C., S. Blower, P. Van den Driessche, D. Kirschner, and A.-A. Yakubu (2002). *Mathematical approaches for emerging and reemerging infectious diseases: models, methods, and theory*, Volume 126. Springer Science & Business Media. 72

Castillo-Chavez, C. and Z. Feng (1997). To treat or not to treat: the case of tuberculosis. *Journal of mathematical biology 35*(6), 629–656. 88

Chatterjee, S., A. Sarkar, S. Chatterjee, M. Karmakar, and R. Paul (2021). Studying the progress of covid-19 outbreak in india using sird model. *Indian Journal of Physics 95*(9), 1941–1957. 77

Chitnis, N., J. M. Hyman, and J. M. Cushing (2008). Determining important parameters in the spread of malaria through the sensitivity analysis of a mathematical model. *Bulletin of mathematical biology 70*(5), 1272. 72

Chowell, G., P. Diaz-Duenas, J. Miller, A. Alcazar-Velazco, J. Hyman, P. Fenimore, and C. Castillo-Chavez (2007). Estimation of the reproduction number of dengue fever from spatial epidemic data. *Mathematical biosciences 208*(2), 571–589. 72

Coelho, F. C., C. T. Codeço, and M. G. M. Gomes (2011). A bayesian framework for parameter estimation in dynamical models. *PloS one 6*(5), e19616. 72

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. 72

Dong, E., H. Du, and L. Gardner (2020, May). An interactive web-based dashboard to track COVID-19 in real time. *The Lancet Infectious Diseases 20*(5), 533–534. 86

Dye, C. and N. Gay (2003). Modeling the sars epidemic. *Science 300*(5627), 1884–1885. 72

E, W., J. Han, and A. Jentzen (2017, Nov). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics 5*(4), 349–380. 72

Eldan, R. and O. Shamir (2016). The power of depth for feedforward neural networks. 74

Fernández-Villaverde, J. and C. I. Jones (2020). Estimating and simulating a sird model of covid-19 for many countries, states, and cities. Technical report, National Bureau of Economic Research. 77

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep learning*. MIT press. 75

Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). Generative adversarial networks. 72

Hadeler, K. (2011). Parameter identification in epidemic models. *Mathematical biosciences 229*(2), 185–189. 72

Hagge, T., P. Stinis, E. Yeung, and A. M. Tartakovsky (2017). Solving differential equations with unknown constitutive relations as recurrent neural networks. 72

Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In *Neural networks for perception*, pp. 65–93. Elsevier. 75

Hethcote, H. W. (2009). The basic epidemiology models: models, expressions for r0, parameter estimation, and applications. In *Mathematical understanding of infectious disease dynamics*, pp. 1–61. World Scientific. 71

Hindmarsh, A. C. and L. R. Petzold (Sep 2005). Lsoda, ordinary differential equation solver for stiff or non-stiff system. 76

Hornik, K., M. Stinchcombe, and H. White (1989). Multilayer feedforward networks are universal approximators. *Neural Networks 2*(5), 359–366. 72

Huang, Y., D. Liu, and H. Wu (2006). Hierarchical bayesian methods for estimation of parameters in a longitudinal hiv dynamic system. *Biometrics 62*(2), 413–423. 72

Kermack, W. O. and A. G. McKendrick (1927). A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character 115*(772), 700–721. 71

Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 77

Knuth, D. E. (1984). Literate programming. *The computer journal 27*(2), 97–111. 76

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems*, Volume 25. Curran Associates, Inc. 72

LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *nature 521*(7553), 436–444. 74

Legrand, J., R. F. Grais, P.-Y. Boelle, A.-J. Valleron, and A. Flahault (2007). Understanding the dynamics of ebola epidemics. *Epidemiology & Infection 135*(4), 610–621. 88

Li, J. (2011). Malaria model with stage-structured mosquitoes. *Mathematical Biosciences & Engineering 8*(3), 753. 72

Ling, J., A. Kurzawski, and J. Templeton (2016, 10). Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics 807*. 72

Longini Jr, I. M., J. S. Koopman, M. Haber, and G. A. Cotsonis (1988). Statistical inference for infectious diseases: risk-specific household and community transmission parameters. *American Journal of Epidemiology 128*(4), 845–859. 72

Luo, J., W. Wang, H. Chen, and R. Fu (2016). Bifurcations of a mathematical model for hiv dynamics. *Journal of Mathematical Analysis and Applications 434*(1), 837–857. 72

Macey, R., G. Oster, and T. Zahnley (2000). Berkeley madonna user's guide. *University of California, Berkeley, CA*. 73

Maier, B. F. and D. Brockmann (2020). Effective containment explains subexponential growth in recent confirmed covid-19 cases in china. *Science 368*(6492), 742–746. 77

Martcheva, M. (2015). *An introduction to mathematical epidemiology*, Volume 61. Springer. 71, 72

Matrajt, L. and T. Leung (2020). Evaluating the effectiveness of social distancing interventions to delay or flatten the epidemic curve of coronavirus disease. *Emerging infectious diseases 26*(8), 1740. 77

McKay, M. D., R. J. Beckman, and W. J. Conover (2000). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 42*(1), 55–61. 72

Neilan, R. M. and S. Lenhart (2010). An introduction to optimal control with an application in disease modeling. In *Modeling Paradigms and Analysis of Disease Trasmission Models*, pp. 67–81. 72

Nguyen, L., M. Raissi, and P. Seshaiyer (2022). Modeling, analysis and physics informed neural network approaches for studying the dynamics of covid-19 involving human-human and human-pathogen interaction. *Computational and Mathematical Biophysics 10*(1), 1–17. 72

Nsoesie, E. O., R. J. Beckman, S. Shashaani, K. S. Nagaraj, and M. V. Marathe (2013). A simulation optimization approach to epidemic forecasting. *PloS one 8*(6), e67164. 73

Ohajunwa, C., K. Kumar, and P. Seshaiyer (2020). Mathematical modeling, analysis, and simulation of the covid-19 pandemic with explicit and implicit behavioral changes. *Computational and Mathematical Biophysics 8*(1), 216–232. 72, 77

Ohajunwa, C. and P. Seshaiyer (2021). Mathematical modeling, analysis, and simulation of the covid-19 pandemic with behavioral patterns and group mixing. *Spora: A Journal of Biomathematics 7*(1), 46–60. 72, 77

O'Dea, E. B., K. M. Pepin, B. A. Lopman, and C. O. Wilke (2014). Fitting outbreak models to data from many small norovirus outbreaks. *Epidemics 6*, 18–29. 72

Padmanabhan, P. and P. Seshaiyer (2017). Computational and mathematical methods to estimate the basic reproduction number and final size for single-stage and multistage progression disease models for zika with preventative measures. *Computational and mathematical methods in medicine 2017*. 72

Padmanabhan, P., P. Seshaiyer, and C. Castillo-Chavez (2017). Mathematical modeling, analysis and simulation of the spread of zika with influence of sexual transmission and preventive measures. *Letters in Biomathematics 4*(1), 148–166. 72

Pollicott, M., H. Wang, and H. Weiss (2012). Extracting the time-dependent transmission rate from infection data via solution of an inverse ode problem. *Journal of biological dynamics 6*(2), 509–523. 72

Raissi, M. and G. E. Karniadakis (2018, Mar). Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics 357*, 125–141. 72, 75

Raissi, M., P. Perdikaris, and G. Karniadakis (2019). 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. 72, 75

Raissi, M., N. Ramezani, and P. Seshaiyer (2019). On parameter estimation approaches for predicting disease transmission through optimization, deep learning and statistical inference methods. *Letters in Biomathematics 6*(2), 1–26. 72

Redmon, J., S. Divvala, R. Girshick, and A. Farhadi (2016). You only look once: Unified, real-time object detection. 72

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks 61*, 85–117. 75

Sen, D. and D. Sen (2021). Use of a modified sird model to analyze covid-19 data. *Industrial & Engineering Chemistry Research 60*(11), 4251–4260. 77

Smirnova, A., L. deCamp, and G. Chowell (2019). Forecasting epidemics through nonparametric estimation of time-dependent transmission rates using the seir model. *Bulletin of mathematical biology 81*(11), 4343–4365. 72

Tan, M., R. Pang, and Q. V. Le (2020). Efficientdet: Scalable and efficient object detection. 72

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017). Attention is all you need. 72

Yong, K. E., A. Mubayi, and C. M. Kribs (2015a). Agent-based mathematical modeling as a tool for estimating trypanosoma cruzi vector–host contact rates. *Acta tropica 151*, 21–31. 72

Yong, K. E., A. Mubayi, and C. M. Kribs (2015b). Agent-based mathematical modeling as a tool for estimating trypanosoma cruzi vector–host contact rates. *Acta tropica 151*, 21–31. 73