Big Data and Information Analytics

https://www.aimspress.com/journal/BDIA

BDIA, 9: 1–28.
DOI:10.3934/bdia.2025001
Received: 09 December 2024
Revised: 01 February 2025
Accepted: 07 February 2025
Published: 21 March 2025

Research article

Deep neural networks with application in predicting the spread of avian influenza through disease-informed neural networks

Nickson Golooba[1], Woldegebriel Assefa Woldegerima[2,3,*] and Huaiping Zhu[1,2]

[1] Laboratory of Mathematical Parallel Systems (LAMPS), CDM, and OMNI-RÉUNIS, Department of Mathematics and Statistics, York University, Toronto, Canada

[2] Centre for Disease Modeling (CDM), Department of Mathematics and Statistics, York University, Toronto, Canada

[3] Laboratory for Industrial and Applied Mathematics (LIAM), Department of Mathematics and Statistics, York University, Toronto, Canada

* **Correspondence:** Email: wassefaw@yorku.ca.

Abstract: Deep learning has emerged in many fields in recent times where neural networks are used to learn and understand data. This study combined deep learning frameworks with epidemiological models and was aimed specifically at the creation and testing of disease-informed neural networks (DINNs) with a view of modeling the infection dynamics of epidemics. Our research thus trained the DINN on synthetic data derived from a susceptible infected-susceptible infected removed (SI-SIR) model designed for avian influenza and showed the model's accuracy in predicting extinction and persistence conditions. In the method, a twelve hidden layer model was constructed with sixty-four neurons per layer and the rectified linear unit activation function was used. The network was trained to predict the time evolution of five state variables for birds and humans over 50,000 epochs. The overall loss minimized to 0.000006, characterized by a combination of data and physics losses, enabling the DINN to follow the differential equations describing the disease progression.

Keywords: disease-informed neural network; physics-informed neural networks; deep neural networks; avian influenza; SI-SIR model; data and physics loss functions; neural network architecture

1. Introduction

One of the most advanced artificial intelligence (AI) technologies of the last decade is deep learning, which uses neural networks to process and understand complex data. In the last few decades, we have seen deep learning advance from just a theory to something hugely useful. Our research is centered on developing technology to apply deep learning research to fields like epidemiology by building on the so-called disease-informed neural networks (DINNs) designed by [1].

1.1. Background

The concept of artificial neural networks originated in the 1940s when neuroscientists Warren McCulloch and Walter Pitts noticed that human brain neurons work similarly to operators producing results based on thresholds [2]. In 1943, they envisioned introducing intelligence to the world [3]. Their approach involved mimicking the brain's functionality using neurons as units, inspired by the structure of human brain neurons transmitting signals to make decisions, thereby pioneering an artificial neural network [4]. According to [2], this neural network model they developed generated interest in the AI community despite limitations at that time. A breakthrough came in the 1970s with Seppo Linnainmaa's discovery of backpropagation, which greatly improved network performance. It was not until 2012, when students made advancements in the ImageNet competition, that neural networks gained recognition and popularity [2]. The achievement propelled networks, like convolutional neural networks (CNNs) and long short-term memory (LSTM) networks, to the forefront of AI research. The advancements in technology have allowed deep learning systems to tackle pattern recognition issues in fields marking a step forward in the field of AI [5].

1.2. Gaps and opportunities

Epidemiological models, like the susceptible infected recovered (SIR) model, have played a role in understanding how diseases spread [6, 7]. However, they are sometimes challenged in capturing the details of real-world data [8]. They face two key limitations in epidemic prediction: limited accuracy when fitting complex disease patterns, and difficulty in handling incomplete data [6, 9]. Specifically, their rigid parameter estimation approach often struggles to capture the nonlinear dynamics seen in avian influenza transmission between bird and human populations [10, 11]. Today, the challenge and urgency in understanding the dynamics of infectious diseases, as exemplified by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) pandemic, has led to the use of advanced AI techniques. The development of DINNs, which combines AI techniques with these models, represents an innovative integration of epidemiological data into the neural network framework and can improve prediction accuracy through advanced data analysis methods, improved parameter estimations, and modeling complex relationships between different variables [12, 13]. DINNs directly address the limitations mentioned earlier in two ways. First, they achieve higher prediction accuracy by simultaneously optimizing model parameters while enforcing epidemiological constraints through a unified loss function [1, 14]. This dual optimization will allow our DINN to better capture the complex transmission patterns of avian influenza. Second, DINNs handle modeling complexities more effectively through their neural network architecture [15, 16], which can automatically adapt to nonlinear incidence rates and varying transmission patterns without requiring mathematical reformulation of the underlying equations.

While deep learning models have shown promise, there are still obstacles to overcome, such as issues related to data quality and quantity, ensuring model transparency, and effectively incorporating real-time data [9, 13]. Tackling these challenges presents opportunities for advancing modeling and creating resilient and understandable models [12].

1.3. Integration of mathematics with AI

The solid mathematical foundation underlying deep learning is crucial, as it enhances the ability of neural networks to process visual information and enables them to solve complex mathematical puzzles, thus expanding what computers can do beyond previous limitations. This collaboration between deep learning and mathematics is essential, as highlighted by noted researchers [4, 17]. Despite its capabilities, deep learning as a field is still maturing, with ongoing research focused on uncovering its theoretical foundations to ensure these systems are both effective and equitable [17].

Building on these ideas, the DINNs are designed to utilize the detailed compartmentalization from classical epidemiology and the learning capabilities of neural networks to offer more precise predictions and effective management strategies for infectious diseases. The ongoing research on DINNs goes beyond modifying existing network structures; it involves establishing a reliable methodology that can aid real-world decision-making in public health [15].

This work will explore the mathematics behind neural networks, discuss their development over time, and explain how they are being used to address the challenges presented by infectious diseases using DINNs. By combining theoretical analysis with real-world applications in predicting diseases, this study intends to add to academic understanding and effective approaches in tackling worldwide health emergencies. It is important to note that for the remainder of this work, whenever we refer to “neuron”, we are actually referring to an “artificial neuron”.

Research questions

1) How can traditional epidemiological models be enhanced using deep neural networks?
2) What mathematical foundations are needed to integrate epidemiological parameters into neural network architectures?
3) How can the integration of classical epidemiological models with modern neural network architectures improve public health decision-making?

1.4. Brief review of the mathematics of DNNs

Here, we state some important definitions and concepts that are useful for this research and that have been adapted from [17–19].

Definition 1.1. *An artificial neuron with weights $w_1, \cdots, w_n \in \mathbb{R}$, bias $b \in \mathbb{R}$, and activation function $\rho : \mathbb{R} \to \mathbb{R}$ is defined as the function $f : \mathbb{R}^n \to \mathbb{R}$ given by*

$$\begin{aligned} f(x_1, \cdots, x_n) &= \rho\left(\sum_{i=1}^{n} x_i w_i + b\right) \\ &= \rho\left((x, w) + b\right), \end{aligned} \tag{1.1}$$

where $w = (w_1, \cdots, w_n)$ and $x = (x_1, \cdots, x_n)$.

Some common activation functions include sigmoid, Tanh, rectified linear unit (ReLU), etc. [20]. Every activation function comes with its benefits. For example, ReLU assists in addressing the issue of vanishing gradients often encountered in networks [18], and it is inspired by the thresholding behavior of neurons where only positive signals are transmitted. We can find more activation functions together with their benefits in [18] or other sources before selecting a particular activation function. For this research, we used ReLU and Tanh.

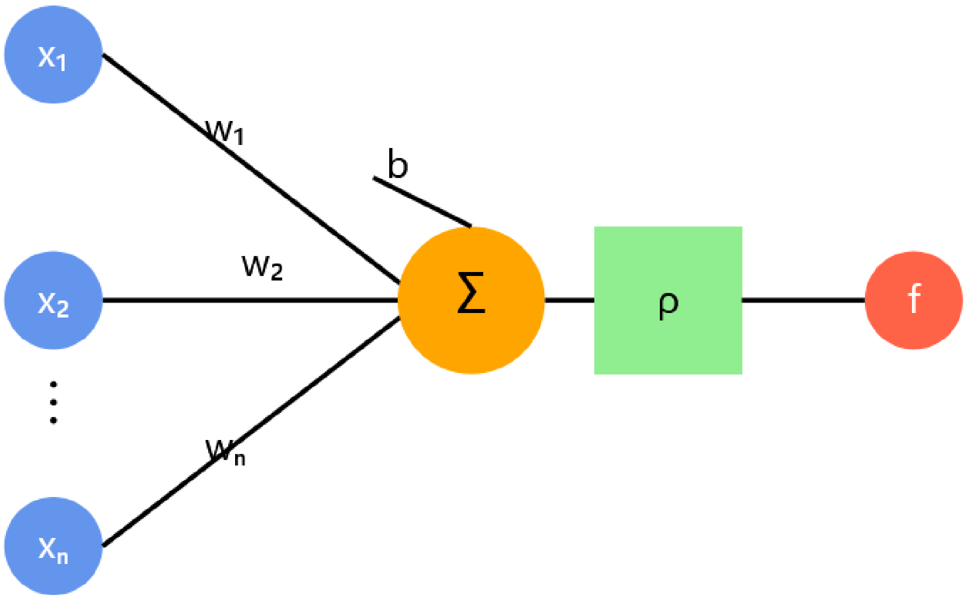

Figure 1. Graphical representation of an artificial neuron $f : \mathbb{R}^n \to \mathbb{R}$.

Layer: In a neural network, a layer is a collection of nodes or neurons that operate parallelly, receiving either the input data or the output from the preceding layer. Neural networks can consist of input, hidden, and output layers.

Definition 1.2. *(Artificial neural network) Let $d \in \mathbb{N}$ be the dimension of the input layer, let L be the number of layers, let $N_0 := d, N_\ell, \ell = 1, \dots, L$, be the dimensions of the hidden and last layer, let $\rho : \mathbb{R} \to \mathbb{R}$ be a nonlinear activation function, and, for $\ell = 1, \dots, L$, let T_ℓ be the affine-linear functions*

$$T_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}, \quad T_\ell x = W^{(\ell)} x + b^{(\ell)},$$

with $W^{(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ being the weight matrices and $b^{(\ell)} \in \mathbb{R}^{N_\ell}$ the bias vectors of the ℓth layer. Then $\Phi : \mathbb{R}^d \to \mathbb{R}^{N_L}$ given by

$$\Phi(x) = T_L \rho \left(T_{L-1} \rho \left(\dots \rho \left(T_1(x) \right) \right) \right), \quad x \in \mathbb{R}^d,$$

is called a (deep) neural network. This composition of affine-linear functions with a nonlinear activation function, is referred to as a neural network of depth L.

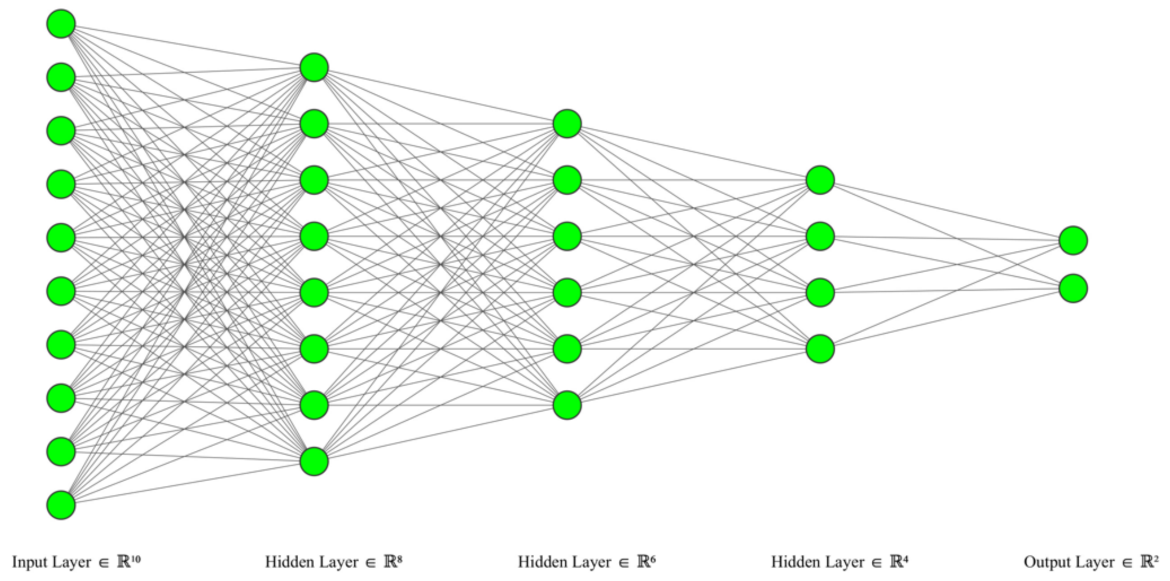
Input Layer ∈ $\mathbb{R}^{10}$
Hidden Layer ∈ $\mathbb{R}^{8}$
Hidden Layer ∈ $\mathbb{R}^{6}$
Hidden Layer ∈ $\mathbb{R}^{4}$
Output Layer ∈ $\mathbb{R}^{2}$

Figure 2. Deep Neural Network $\Phi : \mathbb{R}^{10} \to \mathbb{R}^2$ of depth 4. We plotted this using the software in [21].

Definition 1.3. *(Feed forward neural networks (FFNNs) process): Let* $\mathbf{a}^{(0)} = \mathbf{x} \in \mathbb{R}^d$ *be the input vector. For each value of* ℓ*, from 1, to L, the outputs of the layer denoted by* ℓ *are determined by the activations* $\mathbf{a}^{(\ell)}$.

$$\mathbf{a}^{(\ell)} = \rho(\mathbf{W}^{(\ell)}\mathbf{a}^{(\ell-1)} + \mathbf{b}^{(\ell)}),$$

where $\mathbf{W}^{(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ *is the weight matrix,* $\mathbf{a}^{(\ell-1)} \in \mathbb{R}^{N_{\ell-1}}$ *is the activation vector from the previous layer, and* $\mathbf{b}^{(\ell)} \in \mathbb{R}^{N_\ell}$ *is the bias vector.*

Optimization algorithms: Optimization algorithms are crucial for training neural networks. They iteratively adjust the network's weights to minimize the loss function, which measures the discrepancy between the predicted output and the actual target. Commonly used optimization algorithms include stochastic gradient descent (SGD), adaptive gradient gradient algorithm (AdaGrad), root mean square propagation (RMSprop), and Adam optimizer (adaptive moment estimation) [22].

During optimization, we aim to minimize the loss function. The common loss functions include the mean squared error (MSE) and cross-entropy loss, represented mathematically as:

$$L = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(y_i, \hat{y}_i),$$

where $\mathcal{L}$ is the loss function, y_i is the actual target, and $\hat{y}_i$ is the predicted output [18].

For regression tasks, the MSE is often used and it is defined as:

$$\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2.$$

For classification tasks, the cross-entropy loss is commonly used and it is defined as:

$$\mathcal{L}(y, \hat{y}) = -\sum_{c=1}^{C} y_c \log(\hat{y}_c),$$

where C is the number of classes, y_c is a binary indicator (0 or 1) if class label c is the correct classification for observation y, and $\hat{y}_c$ is the predicted probability of observation y being in class c.

During training, we use the backpropagation method to determine fine-tuning network parameters. Its primary objective is to minimize the loss function, which gauges the variance between the output and the desired target [19].

Mathematically, the gradient of the loss L with respect to the weights $w_{ji}^{(l)}$ is computed as:

$$\frac{\partial L}{\partial w_{ji}^{(l)}} = \frac{\partial L}{\partial a_j^{(l)}} \cdot \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial w_{ji}^{(l)}}, \quad i, j = 1, ..., n$$

For our work, we used the Adam optimizer as an optimization algorithm that combines the advantages of two other extensions of SGD, that is to say, AdaGrad and RMSprop. From AdaGrad, Adam adapts the learning rates for each parameter based on the magnitude of past squared gradients, ensuring infrequent parameters receive larger updates, which is especially useful for sparse data. In addition, from RMSprop, Adam improves on AdaGrad by using an exponentially decaying average of squared gradients to prevent the learning rate from shrinking too much, which ensures that the learning process remains efficient even in the later stages of training.

Adam maintains two moving averages of the gradients, which are used to adapt the learning rate per parameter.

The updated rules for Adam (used in our code) are as follows:

1) Compute the moving averages of the gradient (m_t) and the squared gradient (v_t) as

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla L \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)(\nabla L)^2. \end{aligned}$$

2) Correct the bias in the estimates as

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t}. \end{aligned}$$

3) Update the weights as

$$w_{ji}^{(l)} = w_{ji}^{(l-1)} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}.$$

Here, m_t and v_t are the first and second moment estimates, respectively, and β_1, β_2, and ϵ are hyperparameters. The default values recommended by the authors are $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ [23].

Expressivity of DNNs: Expressivity in neural networks aims to understand the extent to which aspects of a neural network's architecture affect its performance in deep learning. Methods from applied harmonic analysis and approximation theory are typically employed to study this aspect [17]. The universal approximation theorem states that a neural network with at least one hidden layer and nonlinear activation functions can approximate any continuous function to arbitrary precision, given sufficient neurons [24]. This theorem underscores the power and flexibility of neural networks in function approximation, asserting that a feedforward network with a single hidden layer containing a finite number of neurons can approximate any continuous function on compact subsets of $\mathbb{R}^n$, under mild assumptions on the activation function [25,26].

Theorem 1. *(Universal approximation theorem) [17] Let $d \in \mathbb{N}$, $K \subset \mathbb{R}^d$ be compact, $f : K \to \mathbb{R}$ be continuous, and $\rho : \mathbb{R} \to \mathbb{R}$ be continuous and not a polynomial. Then, for each $\epsilon > 0$, there exist $N \in \mathbb{N}$, $a_k, b_k \in \mathbb{R}$, and $w_k \in \mathbb{R}^d$, $1 \leq k \leq N$, such that*

$$\left\| f - \sum_{k=1}^{N} a_k \rho(\langle w_k, \cdot \rangle - b_k) \right\|_\infty \leq \epsilon.$$

In short, it states that each continuous function on a compact domain can be approximated to an arbitrary accuracy by a shallow neural network.

2. Predicting the spread of infectious diseases using a DINN

In this study, we examine an SI-SIR type of compartmental model focusing on avian influenza disease using a DINN. We begin by presenting the mathematical model formulation we will consider.

2.1. Avian influenza mathematical model

1) Avian influenza introduction and current context

Avian influenza, commonly known as bird flu, is a disease that mainly impacts birds but can also affect humans and other animals. This disease is caused by influenza A viruses from the Orthomyxoviridae family, known for their ability to change and create strains with levels of severity and ease of transmission [27].
Avian influenza viruses are categorized into two groups based on how harmful they are to poultry, that is to say, low pathogenic avian influenza (LPAI) and highly pathogenic avian influenza (HPAI). HPAI strains like H5N1 and H7N9 have led to outbreaks in birds and have occasionally crossed over to humans, causing respiratory problems and high mortality rates [28,29].
The transmission of avian influenza viruses to humans can happen through direct contact with infected birds, contaminated surroundings, or, less frequently, through human-to-human contact. The risk of these viruses causing pandemics is a concern in public health since they can undergo genetic changes that make them more easily transmissible among people [30]. Therefore, it is vital to understand how avian influenza spreads and develop strategies to control it in order to prevent and manage outbreaks. Recent occurrences have emphasized the significance of bird flu, as a public health issue. For example, on March 25, 2024, the Centers for Disease Control and Prevention (CDC) in the United States announced instances of H5N1 outbreaks with cases detected in animals like dairy cows. This marked the discovery of these avian flu strains in cattle [31].
There are worries about the virus's ability to change and spread easily among people, leading to increased efforts in monitoring and controlling outbreaks [32].
In Canada, authorities like the Canadian Food Inspection Agency (CFIA) and the Public Health Agency of Canada (PHAC) are taking steps to address the threat of avian influenza by strengthening

surveillance and biosecurity measures. Following the detection of H5N1 in U.S. dairy cattle, they have increased testing of imported dairy cattle, checking milk for traces, and conducting voluntary testing on asymptomatic cows to ensure safety [33]. These actions are aimed at preventing the virus from entering Canadian livestock and safeguarding the food supply. The rapid spread of the H5N1 clade 2.3.4.4b virus globally has caused concern with detections in wild bird species and domestic poultry, leading to mass culling efforts to contain it in North America and Canada where it has impacted commercial and non-commercially bird populations significantly [34].
Inspired by all these, we decided to choose this disease for our implementation.

2) Model equations for avian influenza
In our study, we use a version of the SI-SIR model to examine how avian influenza transmission occurs. This model is based on the work of Yu and Ma [11] which looks at nonlinear incidence and recovery rates in both deterministic and stochastic environments [11]. Unlike the model's varying recovery rate in their study, our simplified version assumes a constant recovery rate for humans. This simplification allows us to focus on key transmission dynamics without dealing with the complexities of changing recovery rates. The SI-SIR model's differential equations are as follows:

$$\frac{dS_a}{dt} = r_a S_a\left(1 - \frac{S_a}{K_a}\right) - \frac{\beta_a I_a S_a}{1 + bI_a} \tag{2.1}$$

$$\frac{dI_a}{dt} = \frac{\beta_a I_a S_a}{1 + bI_a} - (\mu_a + \delta_a)I_a \tag{2.2}$$

$$\frac{dS_h}{dt} = \Pi_h - \frac{\beta_h I_a S_h}{1 + cI_h^2} - \mu_h S_h \tag{2.3}$$

$$\frac{dI_h}{dt} = \frac{\beta_h I_a S_h}{1 + cI_h^2} - (\mu_h + \delta_h + \gamma)I_h \tag{2.4}$$

$$\frac{dR_h}{dt} = \gamma I_h - \mu_h R_h, \tag{2.5}$$

where the model variables and parameters are summarized in the tables below.

Table 1. Model variables and initial values.

Variable	Description	Initial Value	Reference
S_a	Susceptible avian population	9900	[10]
I_a	Infectious avian population	100	[10]
S_h	Susceptible human population	9999	[10]
I_h	Infectious human population	1	[10]
R_h	Recovered human population	0	[10]

Table 2. Model parameters.

Parameter	Description	Value	Reference
r_a	Intrinsic growth rate of avian population	5×10^{-3}	[10]
K_a	Carrying capacity for avian population	5×10^{4}	[10]
β_a	Transmission rate from avian to avian	Varied: 0.8×10^{-8} to 2.6×10^{-8}	[11]
β_h	Transmission rate from avian to human	8×10^{-7}	[10]
μ_a	Natural death rate for avian population	3.4246×10^{-4}	[10]
b	Nonlinear incidence coefficient (avian)	0.001	[11]
μ_h	Natural death rate for human population	3.91×10^{-5}	[10]
δ_a	Disease-induced death rate for avian population	4×10^{-4}	[10]
c	Nonlinear incidence coefficient (human)	0.01	[11]
γ	Constant recovery rate of human population	0.1	[10]
δ_h	Disease-induced death rate for human population	0.077	[10]
Π_h	Recruitment rate of human population	30	[10]

It is important to note that the transmission rate from avian to avian (β_a) was varied in the study conducted by [11] to examine different scenarios.
When $\beta_a = 0.8 \times 10^{-8}$, the basic reproduction number is $R_0 = 0.54 < 1$, leading to disease extinction.
When $\beta_a = 2.6 \times 10^{-8}$, $R_0 = 1.75 > 1$, resulting in disease persistence. According to [11], increasing the parameter β_a impacts whether the disease persists or is eliminated, resulting in number of infected individuals. It is also important to note that in [11], a comprehensive stability analysis of this model was performed, proving that the system exhibits a forward bifurcation behavior. Their analysis demonstrated that when $R_0 < 1$, the disease-free equilibrium is globally asymptotically stable, and when $R_0 > 1$, the endemic equilibrium is globally asymptotically stable. This mathematical property ensures the model has a clear threshold behavior, with no possibility of backward bifurcation or multiple endemic equilibria. These stability properties support our use of this model framework for examining disease dynamics through DINNs.
This model helps in understanding how avian influenza spreads and identifies strategies for controlling it. It explores how avian and human populations interact, taking into account transmission patterns and consistent recovery rates. While the original model by [11] offers insights by incorporating variable recovery rates, it also adds complexity. Through this approach, our goal is to offer an explanation of the fundamental mechanisms behind avian influenza spread and provide valuable insights, for shaping public health policies aimed at managing outbreaks.

2.2. Numerical simulation for the avian influenza model

To illustrate the dynamics of our avian influenza model under different scenarios, we vary the avian to avian transmission rate (β_a) to examine its impact on disease dynamics, as this parameter plays an important role in determining whether the disease persists or becomes extinct. We then conduct numerical simulations similar to one of those conducted by [11] using the same parameters as used in their study.

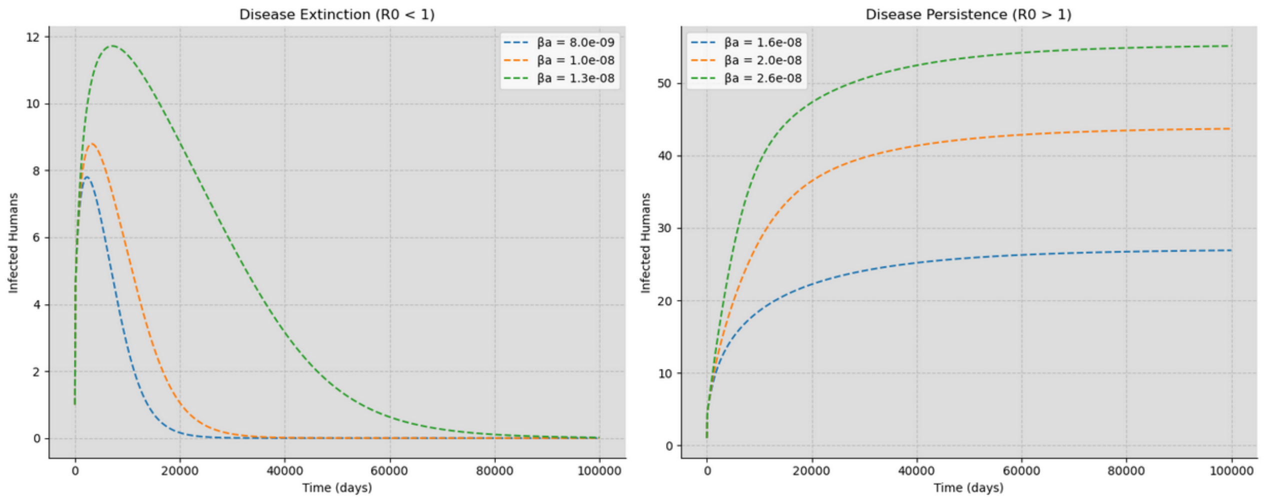

Figure 3. Avian influenza dynamics for different β_a values.

Figure 3 shows the results of these simulations for different values of β_a. The left subplot illustrates scenarios where the basic reproduction number is $R_0 < 1$, leading to disease extinction, while the right subplot shows cases where $R_0 > 1$, resulting in disease persistence.

In our simulations, we used the initial conditions as specified in Table 1 and the parameter values as detailed in Table 2. The time span for the simulation was set from 0 to $100,000$ days to better understand the long-term behavior of the pandemic. Additionally, it is worth noting the computational complexity underlying our implementation. The system is solved through numerical integration over $100,000$ days sampled at 1000 evenly spaced points, with each time point tracking five coupled state variables. This generates a dataset of 5000 data points in total, for all the 5 compartments, in capturing the transmission dynamics. The integration process handles multiple nonlinear terms simultaneously, including density-dependent growth and varying transmission rates. We examine six distinct transmission scenarios (β_a ranging from 0.8×10^{-8} to 2.6×10^{-8}), each requiring separate numerical integration and demonstrating the model's robustness in capturing both disease extinction and persistence regimes. This computational framework ensures accurate solutions across different parameter regimes while maintaining the biological constraints inherent in the epidemiological system.

As depicted in Figure 3, when $\beta_a \leq 1.3 \times 10^{-8}$, the number of infected humans initially increases but eventually declines to zero, indicating disease extinction. Conversely, for $\beta_a \geq 1.6 \times 10^{-8}$, the number of infected humans stabilizes at a nonzero value, signifying disease persistence.

These simulations confirm the findings of [11], that is to say, the avian-to-avian transmission rate β_a is a critical parameter in determining the long-term behavior of the epidemic.
Higher values of β_a lead to a larger number of infected humans at equilibrium, underscoring the importance of controlling avian-to-avian transmission in managing avian influenza outbreaks.

The data generated from these simulations will be instrumental in training and validating our DINN model, enabling us to assess its effectiveness in capturing both disease extinction and persistence scenarios in the next section.

3. DINNs

The combination of machine learning methods, with modeling has become a tool for comprehending and forecasting disease patterns [16]. DINNs, as introduced by [1], are a technique merging neural network adaptability with the expert knowledge inherent in epidemiological models. DINNs are a type of neural network that incorporates knowledge of disease mechanisms and biology into their architecture and learning process. By leveraging insights from disease pathology, DINNs aim to improve the diagnosis, prognosis, and treatment of diseases.

DINNs are a specialized type of physics-informed neural networks (PINNs) designed for epidemiological applications [14]. This approach uses effectively the universal approximation capabilities of neural networks while integrating the foundational structure of disease transmission models. This combination enables the DINNs to grasp the dynamics while upholding the core principles of epidemiology [35].

A key strength of the DINNs is their capacity to learn from both the data and the governing equations of disease models simultaneously. This dual learning approach improves the prediction accuracy in scenarios with noisy data, which is common during the early stages of disease outbreaks [35].

Although important work on DINNs has been done by [1] and other related works by [16] and [36], these studies have inspired us to further optimize the DINN model and apply it to a nonlinear avian influenza model for our study. Some of the choices made in our research such as the number of neurons in each layer of the network, learning rate, search range, and number of epochs, are based on the findings by [1].

3.1. Principles of DINNs

For our avian influenza model, we apply the DINN framework as illustrated in Figure 4. Our specific definition of the DINN, as illustrated in Figure 4, consists of a neural network defined in our code as `net_si_sir` with 12 hidden layers, each containing 64 neurons and using a ReLU activation function. The network takes time t as input and output the normalized compartment values $\hat{S}_a(t), \hat{I}_a(t), \hat{S}_h(t), \hat{I}_h(t), \hat{R}_h(t)$.

We incorporate 12 learnable parameters corresponding to the avian model parameters (r_a, K_a, μ_a, δ_a, Π_h, β_h, β_a, μ_h, δ_h, b, c, and γ). These parameters are initialized randomly and constrained to biologically feasible ranges using a *tanh* transformation. For example, the avian-to-avian transmission rate β_a is constrained to the range $[0.8 \times 10^{-8}, 2.6 \times 10^{-8}]$, which is within the values used in our previous simulations in Section 2.1.

All computational experiments were performed on a high-performance virtual machine provided by the Laboratory of Mathematical Parallel Systems (LAMPS) together with one health modeling network for infectious diseases (OMNI) at York University. The system specifications are as follows:

- **Device name:** omni.hpc1
- **Operating system:** Microsoft Windows Server 2022 Standard, Version 21H2 (OS Build 20348.2700)
- **Processor (CPU):** AMD EPYC 7513 32-Core Processor at 2.60 GHz (Max Clock Speed: 2.95 GHz)
- **Cores and logical processors:** 16 cores, 16 logical processors

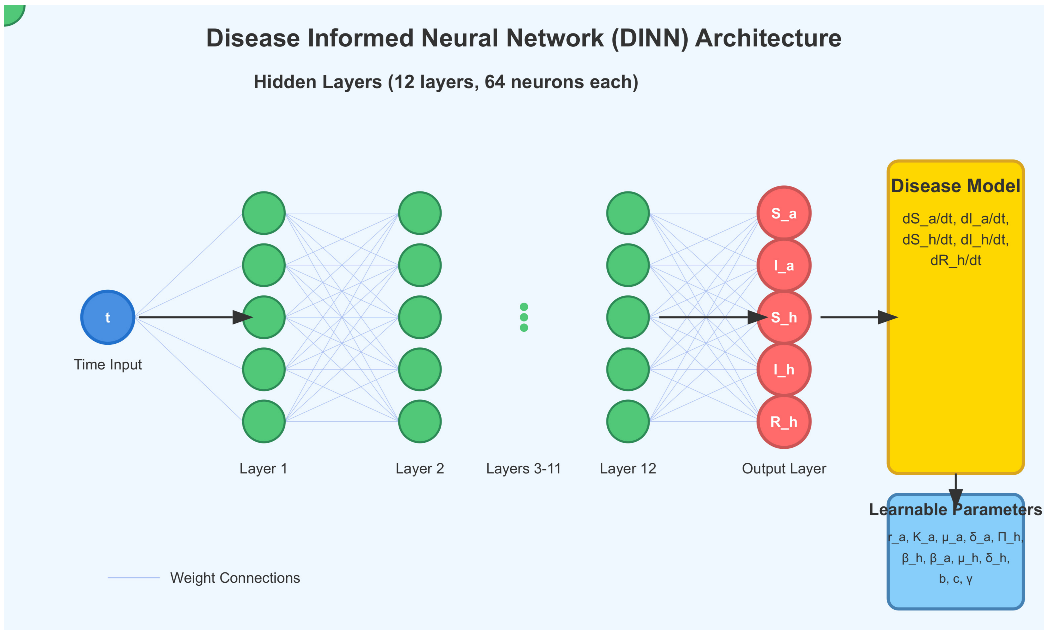
Disease Informed Neural Network (DINN) Architecture
Hidden Layers (12 layers, 64 neurons each)
t
Time Input
Layer 1
Layer 2
Layers 3-11
Layer 12
S_a
I_a
S_h
I_h
R_h
Output Layer
Disease Model
dS_a/dt, dI_a/dt, dS_h/dt, dI_h/dt, dR_h/dt
Learnable Parameters
r_a, K_a, μ_a, δ_a, Π_h, β_h, β_a, μ_h, δ_h, b, c, γ
Weight Connections

Figure 4. A DINN architecture for avian influenza model.

- **Memory (RAM):** 80.0 GB of installed physical memory; 75.2 GB available
- **Storage:** 699 GB total disk size with 643 GB available
- **Virtualization platform:** VMware 20.1 (System Manufacturer: VMware, Inc.)

This high-performance computing environment facilitated the efficient training and evaluation of our DINN model. The substantial processing power and memory were essential for handling the computational demands of our DINN training, especially when integrating physics-informed components.

In our implementation, we combine both data loss and physics loss in the loss function of our DINN to ensure that the model not only fits the data but also adheres to the underlying dynamics of the SI-SIR model for avian influenza.

The loss function is formulated as:

$$\text{Total Loss} = \text{Data Loss} + \text{Physics Loss}. \tag{3.1}$$

Data Loss: This measures the discrepancy between the network's predictions and the available synthetic data using the MSE:

$$\text{Data Loss} = \frac{1}{N}\sum_{i=1}^{N}\left((\hat{S}_{a,i} - S_{a,i})^2 + (\hat{I}_{a,i} - I_{a,i})^2 + (\hat{S}_{h,i} - S_{h,i})^2 + (\hat{I}_{h,i} - I_{h,i})^2 + (\hat{R}_{h,i} - R_{h,i})^2\right) \tag{3.2}$$

Here, $\hat{S}_{a,i}, \hat{I}_{a,i}, \hat{S}_{h,i}, \hat{I}_{h,i}, \hat{R}_{h,i}$ are the network's predictions at time t_i, and $S_{a,i}, I_{a,i}, S_{h,i}, I_{h,i}, R_{h,i}$ are the true values.

Physics loss: This quantifies how well the network's predictions satisfy the differential equations of our SI-SIR model. Using automatic differentiation, we compute the time derivatives of the network outputs and compare them with the righthand sides of the SI-SIR model equations:

$$\text{Physics Loss} = \frac{1}{N}\sum_{i=1}^{N}\left(f_1^2(t_i) + f_2^2(t_i) + f_3^2(t_i) + f_4^2(t_i) + f_5^2(t_i)\right), \tag{3.3}$$

where the residuals $f_j(t_i)$ are defined as:

$$f_1(t_i) = \frac{d\hat{S}_a}{dt} - \left(r_a\hat{S}_a\left(1 - \frac{\hat{S}_a}{K_a}\right) - \frac{\beta_a\hat{I}_a\hat{S}_a}{1 + b\hat{I}_a}\right) \tag{3.4}$$

$$f_2(t_i) = \frac{d\hat{I}_a}{dt} - \left(\frac{\beta_a\hat{I}_a\hat{S}_a}{1 + b\hat{I}_a} - (\mu_a + \delta_a)\hat{I}_a\right) \tag{3.5}$$

$$f_3(t_i) = \frac{d\hat{S}_h}{dt} - \left(\Pi_h - \frac{\beta_h\hat{I}_a\hat{S}_h}{1 + c\hat{I}_h^2} - \mu_h\hat{S}_h\right) \tag{3.6}$$

$$f_4(t_i) = \frac{d\hat{I}_h}{dt} - \left(\frac{\beta_h\hat{I}_a\hat{S}_h}{1 + c\hat{I}_h^2} - (\mu_h + \delta_h + \gamma)\hat{I}_h\right) \tag{3.7}$$

$$f_5(t_i) = \frac{d\hat{R}_h}{dt} - \left(\gamma\hat{I}_h - \mu_h\hat{R}_h\right). \tag{3.8}$$

By minimizing the *Total Loss* in Eq (3.1), the DINN learns to produce predictions that not only fit the data but also satisfy the SI-SIR model dynamics.

3.2. Implementation details of automatic differentiation using PyTorch

By carefully integrating the automatic differentiation capabilities of PyTorch and structuring the loss function to include both data fidelity and adherence to physical laws, our DINN effectively learns the dynamics of the avian influenza SI-SIR model. The combination of data loss and physics loss guides the neural network to produce accurate predictions that are consistent with both the observed data and the underlying epidemiological model. In our code, we utilize PyTorch's automatic differentiation capabilities to compute the time derivatives of the network outputs with respect to time t. The implementation details that describe the automatic differentiation capabilities of the PyTorch algorithm can be found in the Appendix.

We use PyTorch for the training process, employing the Adam optimizer and a learning rate scheduler that decreases the rate by 10% every 1000 epochs. The network undergoes training for $50,000$ epochs to ensure it reaches convergence.

Throughout the training phase, the model not only learns to fit the given data but also adheres to the underlying equations of the SI-SIR model. This dual learning approach enables DINN to capture both the dynamics and the essential epidemiological principles governing avian influenza spread.

Once trained, the DINN can forecast how avian influenza outbreaks evolve, potentially revealing insights that may not be clearly visible from the original differential equation model. Additionally, the trained network can offer parameter estimates that can be compared with values obtained through traditional estimation methods.

By applying this DINN architecture to our avian influenza model, we aim to enhance predictions of disease spread in situations where data may be scarce or uncertain. This strategy combines modeling and machine learning strengths, potentially leading to precise forecasts and informed public health decisions [37].

4. Results

In this section, we provide an account of how we implemented our DINN approach for modeling avian influenza. We describe the process of training, analyze the outcomes of parameter estimation, and assess how well the model predicts disease patterns in scenarios. Our focus is on how the model captures both disease disappearance and persistence while evaluating its performance across varying transmission rates.

4.1. DINN training and parameter estimation

In this section, we describe the training setup of our DINN model discussing the selected optimization method and the hyperparameters used. We then proceed to evaluate the accuracy of estimating parameters of the model by comparing the DINN's estimations with the target values and then provide a brief discussion regarding the conclusion and implication of the study.

In the table below, we compare the final estimated parameters by our DINN model with their target values from the original model (see Eqs (2.1)–(2.5)). It is important to note that these results were obtained with specific hyperparameters: a learning rate of 1×10^{-4}, 1000 time points, and a parameter search range of approximately 1000% for the same architecture as illustrated in Figure 4.

Table 3. Comparison of the DINN's estimated parameters with target values (learning rate: 1×10^{-4}, time points: 1000).

Parameter	DINN Estimate	Target Value	Relative Error	Search Range
r_a	0.004777	0.005000	4.46%	[0.0045, 0.0055]
K_a	50,005	50,000	0.01%	[0, 100,000]
μ_a	0.00029792	0.00034246	13.01%	[0.0002, 0.0004]
δ_a	0.0004085	0.0004000	2.13%	[0.0003, 0.0005]
Π_h	31	30	3.33%	[−70, 130]
β_h	0.0000009723	0.0000008000	21.54%	[0.0000006, 0.000001]
β_a	0.00000002402	0.000000026	7.62%	[8×10^{-9}, 2.6×10^{-8}]
μ_h	0.00003732	0.00003910	4.55%	[0.000029, 0.000049]
δ_h	0.070335	0.077000	8.66%	[0.05, 0.09]
b	0.000930	0.001000	7.00%	[−0.001, 0.003]
c	0.01111	0.01000	11.10%	[−0.01, 0.03]
γ	0.12409	0.10000	24.09%	[−0.15, 0.45]

In the Figure below, we show the evolution of the loss function over the course of training.

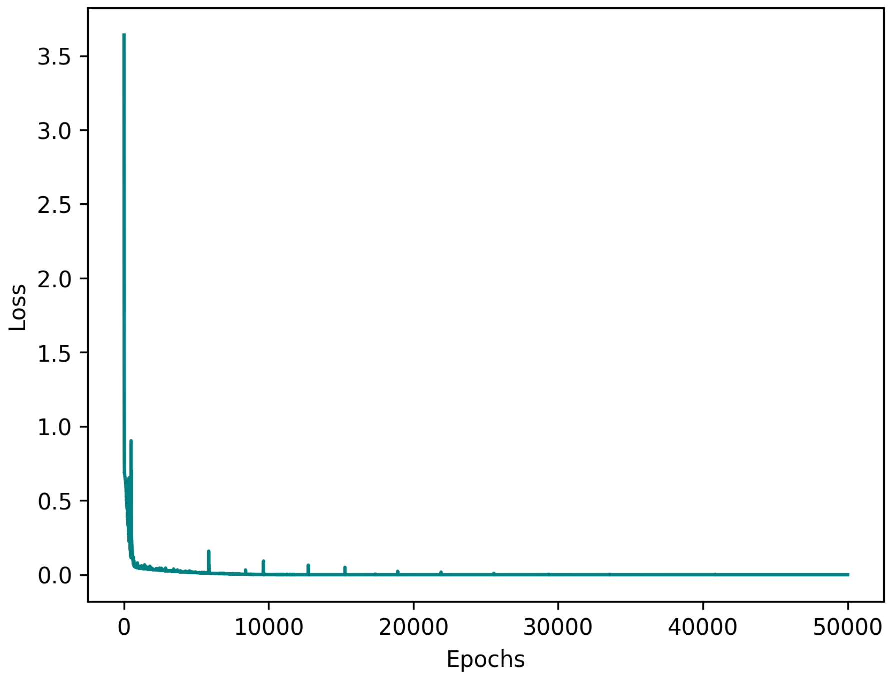

Figure 5. Loss vs. Epochs during DINN Training.

We conduct training for our DINN model for over 50,000 epochs using the Adam optimizer with a learning rate scheduler. The training process lasts around 28 minutes in total, and Figure 5 illustrates how the loss function evolved during training.

4.2. Analysis of DINN training and parameter estimation

In this section, we review the outcomes from Section 4.1 of our DINN training process and assess how accurately parameters are estimated. We discuss the impact of hyperparameters on model performance, analyze the variation in parameter estimations across runs, and interpret how the loss function behaves during training. Furthermore, we investigate how the model reacts to parameters and discuss what our discoveries mean for applications of DINN in epidemiological modeling.

It's important to stress that the values shown in Table 3 heavily rely on the chosen hyperparameters. The learning rate, number of time points, and parameter search range all have impacts on the DINN performance and the resulting parameter estimations. Thoughtful consideration of these hyperparameters is crucial for implementing the DINN model. For example, we noticed that when using a 0% search range (where the parameter was limited to its target value), DINN consistently replicated the target values regardless of epochs. This underscores how crucial the search range is to both the model's behavior and outcomes.

The search ranges for each parameter specified in our code are displayed in the rightmost column of Table 3. These ranges were established using the " tanh " function to keep the parameters within biologically reasonable limits while allowing for exploration around the desired values. For instance, the search range for r_a was defined as [0.0045, 0.0055], centered around its target value of 0.005. The broad search ranges (sometimes covering orders of magnitude) were selected to assess the DINNs capability to converge on accurate parameter values from an initial distribution. However, this decision also introduces variability in the outcomes as the model needs to navigate a large parameter space.

The DINN model exhibited a strong performance in estimating most parameters, showing particularly high accuracy for the K_a, r_a, δ_a, and Π_h parameters. The carrying capacity K_a was estimated with outstanding precision, displaying just a 0.01% relative error. Parameters like μ_a, β_h, and γ demonstrated errors which may indicate areas where the model is less sensitive or where there might be interactions between parameters that make precise estimation more challenging. It is important to note that for parameters with higher relative errors, the DINN estimates fall within the defined search ranges and biologically feasible limits. This suggests that the DINN approach can offer insights into parameter values even when exact estimation proves challenging.

During our project, we conducted several simulations with varying hyperparameters. Like the findings of [1], in their simulations with other infectious diseases, we noticed that using lower learning rates and more time points resulted in longer training durations. For instance, when we had $100,000$ time points and a learning rate of 1×10^{-6}, the training process took around 5 hours and 39 minutes. However, when we reduced the time points to 1000 and increased the learning rate to 1×10^{-4}, we were able to reduce the training time to 28 minutes for $50,000$ epochs. Conversely, keeping our time points at 1000 and lowering the learning rate to 1×10^{-6} led to a training duration of 29 minutes for $50,000$ epochs.

In our DINN model's normalization and loss calculation stages, we introduced an epsilon value (self.epsilon 1×10^{-8}). This tiny constant helped to prevent division by zero and enhanced numerical stability, especially when working with parameters that might have approached very small values.

During our simulations, we noticed variations in how parameters were estimated across runs even when using the same code. While some parameters consistently matched the target values closely, others showed differences. This variability stayed consistent across runs without any changes to the code, indicating that factors like the nature of optimization, the complexity of parameter space, and wide search ranges all play a role in this behavior.

These results underscore the importance of multiple runs and conducting analyses in implementing DINN. The variation in outcomes emphasizes the need for robust validation methods, possibly using ensemble techniques to enhance parameter estimation reliability. Additionally, it highlights the role of tuning hyperparameters in defining appropriate search ranges for each parameter to achieve accurate and consistent results with DINN models.

In Figure 5 we see that, initially, there is a decrease in loss, indicating that DINN quickly grasps the dynamics of the avian influenza model. Following this drop, the decline in loss becomes more gradual, suggesting adjustments are being made to fine-tune model parameters. The continuous decrease in loss signifies that our DINN is effectively learning to balance both data-driven and physics-informed aspects within its loss function. We think that the smoothness of the curve, with no fluctuations, indicates a stable training process. It's interesting to note that the loss does not level off completely after $50,000$ epochs. This could mean that additional training could lead to improvements or it might indicate that the model has struck a balance between fitting the data and satisfying the physical constraints set by the differential equations.

The final loss value reached is 0.000006, which reflects both the error in fitting the data and ensuring that the model's differential equations are satisfied. We believe that this low loss value indicates that our DINN has effectively learned to replicate the dynamics of the avian influenza model while adhering to physical principles.

4.3. Selection of neural network architecture:

To determine the optimal depth for our DINN, we experimented with neural networks of varying depths: 8, 10, and 12 layers. We trained each network using the same hyperparameters and compared their performance in terms of parameter estimation accuracy and computational efficiency.

Table 4. Comparison of DINN performance with different network depths.

Parameter	**Target Value**	**8 Layers**	**10 Layers**	**12 Layers**
r_a	0.005000	0.003044	0.003692	**0.004777**
Relative Error (%)		39.12%	26.16%	**4.46%**
K_a	50000.0	66220.4	50055.8	**50005.0**
Relative Error (%)		32.44%	0.11%	**0.01%**
μ_a	0.00034246	0.00028369	0.00030127	**0.00029792**
Relative Error (%)		17.15%	12.04%	**13.01%**
δ_a	0.0004000	0.0003941	0.0004117	**0.0004085**
Relative Error (%)		1.48%	2.93%	**2.13%**
Π_h	30.00	20.14	35.38	**31.00**
Relative Error (%)		32.87%	17.93%	**3.33%**
β_h	8×10^{-7}	9.564×10^{-7}	9.734×10^{-7}	9.723×10^{-7}
Relative Error (%)		19.55%	21.68%	**21.54%**
β_a	2.6×10^{-8}	1.951×10^{-8}	2.442×10^{-8}	2.402×10^{-8}
Relative Error (%)		24.96%	6.08%	**7.62%**
μ_h	0.00003910	0.00003797	0.00003753	**0.00003732**
Relative Error (%)		2.89%	4.01%	**4.55%**
δ_h	0.077000	0.071843	0.070740	**0.070335**
Relative Error (%)		6.70%	8.13%	**8.66%**
b	0.001000	0.001635	0.000947	**0.000930**
Relative Error (%)		63.50%	5.30%	**7.00%**
c	0.01000	0.01628	0.01075	**0.01111**
Relative Error (%)		62.80%	7.50%	**11.10%**
γ	0.10000	0.14560	0.13007	**0.12409**
Relative Error (%)		45.60%	30.07%	**24.09%**
Average Relative Error		26.84%	12.81%	**9.72%**
Training Time		19 min 26 s	23 min 40 s	**28 min 5 s**

From Table 4, the estimates of the critical parameters like r_a, K_a, and Π_h, for the 12-layer network are significantly closer to the target values. While the 12-layer network requires slightly more training time than the 8-layer and 10-layer networks, it remains computationally feasible, with a training time of approximately 28 minutes. In fact, the 12-layer network consistently outperforms the 8-layer and 10-layer networks in terms of lower relative errors across most parameters. This means that increasing our number of layers to above 12 will help us have lower percentage relative errors, at least lower than the higher ones as seen in γ, from 24.09% to a lower percentage value.

Calculation of Relative Errors: The relative error for each parameter is calculated using the formula:

$$\text{Relative Error (\%)} = \left(\frac{|\text{Estimated Value} - \text{Target Value}|}{\text{Target Value}} \right) \times 100\%$$

For example, for r_a in the 8-layer network:

$$\text{Relative Error} = \left(\frac{|0.003044 - 0.005000|}{0.005000} \right) \times 100\% = 39.12\%$$

Therefore, based on the comparative analysis in Table 4, a 12-layer DINN is the most suitable choice for our framework because it balances accuracy and computational efficiency and we are happy with the results obtained with this choice of layers. This decision is supported by the empirical evidence from our experiments, but we expect better performance as we increase the number of layers of our DINN despite the high computational power that will be required in such a case. This expectation can be supported by the work done by [1].

4.4. DINN disease model prediction

In this section, we evaluate the DINN model's ability to predict disease dynamics under various scenarios, focusing on its performance in capturing both disease extinction and persistence.

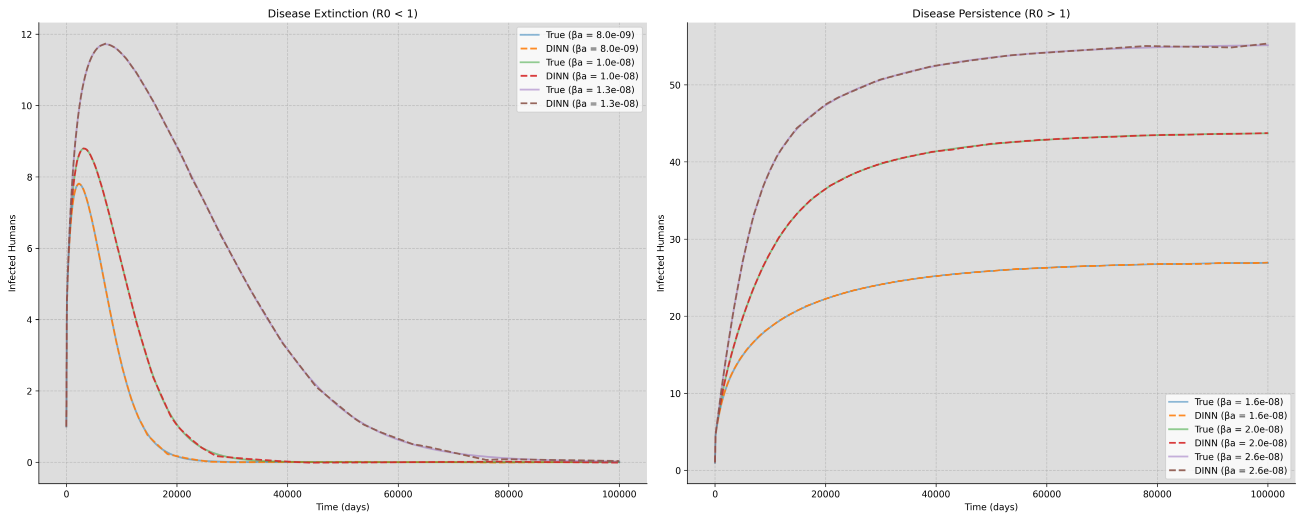

Figure 6. DINN predictions vs. true values for disease extinction ($R_0 < 1$) and persistence ($R_0 > 1$), showing the model performance across different β_a values.

To generate the prediction plots in Figure 6, we use the same DINN methodology as described in Section 4.1.

For each β_a value ($8.0 \times 10^{-9}, 1.0 \times 10^{-8}, 1.3 \times 10^{-8}, 1.6 \times 10^{-8}, 2.0 \times 10^{-8},$ and 2.6×10^{-8}), we follow these steps:

1) We load the corresponding synthetic data generated from the original SI-SIR model.
2) We train a separate DINN model for each β_a value, using the loaded data.

3) After training, we use each DINN model to predict the number of infected humans (I_h) over the same time period as the training data.
4) We plot both the true values from the original model (solid lines) and the DINN predictions (dashed lines) on the same graph.

The results are displayed in two sections: the left side illustrates scenarios where the disease becomes extinct ($R_0 < 1$), while the right side shows scenarios where the disease persists ($R_0 > 1$). Each line on the graph is color-coded and marked with its value of β_a for comparison.

This method enabled us to compare the predictions made by the DINN model with values across various transmission rates, showcasing the model's capacity to capture both extinction and persistence dynamics.

4.5. Analysis of DINN's disease model prediction

In this section, we present the forecasts generated by the DINN model for avian-to-avian transmission rates (β_a) and compare them with the true data obtained from the SI SIR model. Additionally, we evaluate the model's precision by exploring the compartments of the system and analyzing its performance under varying transmission rates.

4.5.1. Disease extinction ($R_0 < 1$)

In the left panel of Figure 6, we can see that the number of infected individuals changes over time for the three β_a values; 8.0×10^{-9}, 1.0×10^{-8}, and 1.3×10^{-8}. Our findings are:

- Initially, there is a spike in infections that later decreases to zero. This trend indicates that the disease is declining, as expected when $R_0 < 1$.
- Our DINN model (represented by dashed lines) closely mirrors the values (solid lines), capturing the initial peak and subsequent decline accurately.
- The model performs well across different β_a values, suggesting that it is robust to changes in the transmission rate.

4.5.2. Disease persistence ($R_0 > 1$)

The right panel of Figure 6 shows what happens with higher β_a values, that is to say, 1.6×10^{-8}, 2.0×10^{-8}, and 2.6×10^{-8}. The obtained results vary from when $R_0 < 1$, as elaborated.

- Instead of dying out, the number of infected individuals levels off at a steady state. This aligns with what we would expect when $R_0 > 1$ where the disease sticks around in the population.
- Again, our DINN model predictions (dashed lines) match up well with the true values (solid lines). It's good to see this consistency even as we crank up the β_a values.
- The model captures the different steady-state levels for each β_a value, showing it can adapt to various epidemic scenarios.

4.5.3. DINN performance across β_a values

We also examine how effectively our model performs for different compartments at certain β_a values. The comparison of RMSE losses is displayed in the figure below.

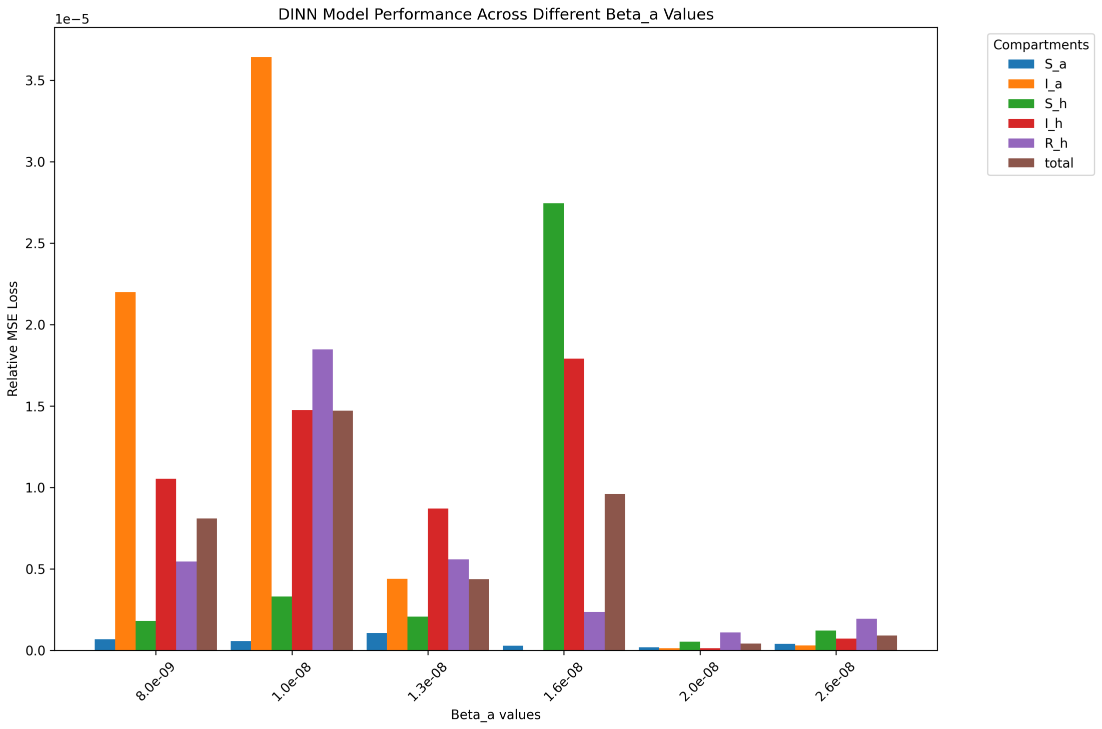

Figure 7. DINN model performance for different β_a values.

By analyzing the graph in Figure 7, we can conclude the following;

- The I_a compartment (population) poses the greatest challenge for our model as it exhibits the highest relative MSE loss, particularly at lower β_a values. This difficulty may stem from the complexity of predicting the infectious avian population when transmission rates are reduced. Nonetheless, there could be factors not addressed in this research.
- Despite some variation, the overall total loss stays pretty low. This suggests that our DINN is generally doing a good job across different β_a values.
- As the avian-to-avian transmission rate increases (β_a), there are changes in relative MSE losses at various parts of the DINN model. The shifts in error characteristics show how the framework behaves to the shifts in patterns of the disease. Thus, when the transmission rates are high, this substantially influences the speed and area where the diseases spread – starting with birds and then humans. Our introduced DINN model is able to capture such changes as depicted by the losses above. Therefore, the above adaptive response indicates that our model is flexible, in a way that it is able to attend to conditions and make predictions correspondingly depending on transmission rate.

5. Discussion

In recent studies, DINNs have been designed to utilize the detailed compartmentalization from classical epidemiology and the learning capabilities of neural networks to offer more precise predictions and effective management strategies for infectious diseases.

The DINN designed in our work managed to capture the dynamics of both disease extinction and persistence scenarios. Our DINN model is not perfect but we believe it is a considerable improvement in the combination of neural networks with epidemiological modeling. While we are still in the process of improvement and expansion around this very approach, it has the potential to become a tool of great potential for the understanding of and prediction of the complex dynamics of diseases.

Concerning the lower values of β_a, we noticed that the typical outbreak curve ratio grew to the maximum and then had periodic declines to zero. The DINN behaved almost like this curve, which is essential for forecasting the possible extinction of a disease. On the other hand, when β_a was increased, the disease prevalence remained in the population at a certain level in the long run. This flexibility is significant, as this suggests that our proposed DINN could, in principle, have to deal with a lot of real-world scenarios.

The rate at which avian influenza spreads within bird populations is heavily influenced by the transmission rate among birds (β_a) in the SI-SIR model for influenza. Changes in β_a can have an impact, on the progression of the epidemic. Ultimately we decide whether the disease will fade away or continue to exist in the population. Therefore, analyzing β_a provides essential insights into the effectiveness of intervention strategies aimed at reducing transmission among birds, which is a key factor in controlling the spread of avian influenza to human populations.

Moreover, we observed the sensitivity of the DINN especially in the I_a compartment as the number of new cases of transmission fell. We believe that the model seems to have struggled to approximate the population of infected birds, especially when rates were lower. This could possibly be worth further investigation.

It was very informative to look at how hyperparameters act. The modification of the learning rate with the amount of time points affected the performance of the DINN. It is like a situation! Since the decision is made by us, or by some of us at least, we have some levers that we can pull if we want to increase the model's efficiency.

On the parameter constraints, we were satisfied with the way the *tanh* function was performing. It tried to remain biologically realistic but also allowed the model to be somewhat loose. That said, though, it was refreshing to see how different the parameter estimates are in different runs, given that one is really optimizing a very complex function here.

Last but not least, the loss that was observed during the course of the training was encouraging, too. It confirms that our DINN is developing as expected and manages to strike a balance in the model between learning the data and respecting the differential equations.

6. Conclusions

In summation, we say our DINN implementation for avian influenza modeling shows great promise in model predictions to inform the public health sectors. We have created a tool that can successfully simulate different disease scenarios (including adapting to varying transmission rates) with a high degree of accuracy.

Our main reason for excitement is that the model was able to predict both extinction and persistence scenarios with high accuracy. Such flexibility in the model can prove to be highly beneficial in the public health sector, especially for planning, as it can anticipate a variety of possible outcomes.

Our investigation into using DINNs for modeling avian influenza has revealed some avenues for future research. One challenge we encountered with the I_a compartment at low transmission rates is a potential area we might look into for future refinements, as there is always room for improvement. One could investigate deeper into this by:

1) Experimenting with various neural network architectures or exploring the use of a network or a transformer-based model.
2) Adjusting the loss function to prioritize giving importance to the I_a component during training.

Moreover, although our synthetic data has served well in demonstrating the concept, there may be a desire to validate our model in real-world scenarios. In future work, we plan to compare our DINN approach against other learning frameworks such as sparse identification of nonlinear dynamics (SINDy) and neural ordinary differential equations (ODEs). This will help us better understand the relative advantages, computational costs, and robustness to noisy data of each method, which will, in the end, help us make more informed choices for epidemiological modeling and prediction. A study could be conducted to answer the questions below.

1) How does our DINN handle noisy or incomplete data, which is a challenge in the field of epidemiology?
2) How could one explore the methods for filling in missing data or reducing noise? This could be beneficial as part of the data preparation process.

We intend to incorporate advanced uncertainty quantification techniques, such as Bayesian methods or bootstrapping, to derive confidence intervals for the model parameters. This will offer a statistically rigorous measure of reliability for our DINN-based estimates, complementing the multiple-run approach and biologically constrained parameter ranges presented in this work. We also plan to use LSTMs, as a substitute for the FFNNs employed in this research project. LSTMs are adept at handling time series information, and they have the capability to capture sequential patterns by retaining knowledge across extended sequences. Enhancing the model's capacity to grasp patterns in epidemiological data by employing an LSTM-based DINN could result in more accurate forecasts of disease trends.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgment

W.A.W acknowledges financial support from the NSERC Discovery Grant (Appl No.: RGPIN-2023-05100) used to support the student. W.A.W also acknowledges support from the Canadian Institute for Health Research (CIHR) under the Mpox and Other Zoonotic Threats Team Grant (FRN. 187246). HZ acknowledges financial support from the Natural Sciences and Engineering Research Council of Canada and the Public Health Agency of Canada (NSERC-PHAC). Nickson Golooba would like to express his gratitude for the financial support received from Woldegebriel Assefa Woldegerima and Huaiping Zhu, and for their invaluable guidance and mentorship throughout this research. Appreciation is also extended to the Canadian "One Health Modelling Network for Emerging Infections" (OMNI) for their support and provision of computational resources.

Conflict of interest

The authors declare that there is no conflict of interest.

References

1. Shaier S, Raissi M, Seshaiyer P, Data-driven approaches for predicting spread of infectious diseases through dinns: Disease informed neural networks. preprint, arXiv:2110.05445. https://doi.org/10.48550/arXiv.2110.05445
2. Lichtner-Bajjaoui A, *A Mathematical Introduction to Neural Networks*. Master's thesis, Universitat de Barcelona, 2021.
3. McCulloch WS, Pitts W, (1943) A logical calculus of the ideas immanent in nervous activity. *Bull Math Biophys* 5: 115–133. https://doi.org/10.1007/BF02478259
4. Kutyniok G, The mathematics of artificial intelligence. preprint, arXiv:2203.08890. https://doi.org/10.48550/arXiv.2203.08890
5. Galván E, Mooney P, (2021) Neuroevolution in deep neural networks: Current trends and future challenges. *IEEE Trans Artif Intell* 2: 476–493. https://doi.org/10.1109/TAI.2021.3067574
6. Hethcote HW, (2000) The mathematics of infectious diseases. *SIAM Rev* 42: 599–653. https://doi.org/10.1137/S0036144500371907
7. Brauer F, Castillo-Chavez C, (2012) *Mathematical Models in Population Biology and Epidemiology*, 2 Eds., New York: Springer. https://doi.org/10.1007/978-1-4614-1686-9
8. Brauer F, Castillo-Chavez C, Feng Z, (2019) *Mathematical Models in Epidemiology*, New York: Springer. https://doi.org/10.1007/978-1-4939-9828-9
9. Holzinger A, Biemann C, Pattichis CS, Kell DB, What do we need to build explainable ai systems for the medical domain?. preprint, arXiv:1712.09923. https://doi.org/10.48550/arXiv.1712.09923
10. Liu S, Ruan S, Zhang X, (2017) Nonlinear dynamics of avian influenza epidemic models. *Math Biosci* 283: 118–135. https://doi.org/10.1016/j.mbs.2016.11.014
11. Yu X, Ma Y, (2022) An avian influenza model with nonlinear incidence and recovery rates in deterministic and stochastic environments. *Nonlinear Dyn* 108: 4611–4628. https://doi.org/10.1007/s11071-022-07422-6
12. Yang S, Santillana M, Kou SC, (2015) Accurate estimation of influenza epidemics using google search data via ARGO, In: *Proceedings of the National Academy of Sciences*, 112: 14473–14478. https://doi.org/10.1073/pnas.1515373112
13. Wiens J, Shenoy ES, (2018) Machine learning for healthcare: On the verge of a major shift in healthcare epidemiology. *Clin Infect Dis* 66: 149–153. https://doi.org/10.1093/cid/cix731
14. Raissi M, Perdikaris P, Karniadakis GE, (2019) Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J Comput Phys* 378: 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

15. Nguyen L, Raissi M, Seshaiyer P, (2022) Modeling, analysis and physics informed neural network approaches for studying the dynamics of COVID-19 involving human-human and human-pathogen interaction. *Comput Math Biophys* 10: 1–17. https://doi.org/10.1515/cmb-2022-0001

16. Ning X, Jia L, Wei Y, Li XA, Chen F, (2023) Epi-DNNs: Epidemiological priors informed deep neural networks for modeling COVID-19 dynamics. *Comput Biol Med* 158: 106693. https://doi.org/10.1016/j.compbiomed.2023.106693

17. Kutyniok G, (2023) An introduction to the mathematics of deep learning, In: *European Congress of Mathematics*, 73–91. https://doi.org/10.4171/8ECM/30

18. Goodfellow I, Bengio Y, Courville A, (2016) *Deep learning*, MIT Press. https://doi.org/10.4258/hir.2016.22.4.351

19. Rumelhart DE, Hinton GE, Williams RJ, (1986) Learning representations by back-propagating errors. *Nature* 323: 533–536. https://doi.org/10.1038/323533a0

20. Glorot X, Bordes A, Bengio Y, (2011) Deep sparse rectifier neural networks, In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 315–323.

21. Lenail A, (2019) NN-SVG: Publication-ready neural network architecture schematics. *J Open Source Softw* 4: 747. https://doi.org/10.21105/joss.00747 .

22. Tieleman T, (2012) Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA Neural Networks Mach Learn* 4: 26.

23. Kingma DP, Ba J, Adam: A method for stochastic optimization. preprint, arXiv:1412.6980. https://doi.org/10.48550/arXiv.1412.6980

24. Hornik K, (1991) Approximation capabilities of multilayer feedforward networks. *Neural Networks* 4: 251–257. https://doi.org/10.1016/0893-6080(91)90009-T

25. Hornik K, Stinchcombe M, White H, (1989) Multilayer feedforward networks are universal approximators. *Neural Networks* 2: 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

26. Funahashi KI, (1989) On the approximate realization of continuous mappings by neural networks. *Neural networks* 2: 183–192. https://doi.org/10.1016/0893-6080(89)90003-8

27. Shao W, Li X, Goraya MU, Wang S, Chen JL, (2017) Evolution of influenza a virus by mutation and re-assortment. *Int J Mol Sci* 18: 1650. https://doi.org/10.3390/ijms18081650

28. Peiris JM, De Jong MD, Guan Y, (2007) Avian influenza virus (H5N1): A threat to human health. *Clin Microbiol Rev* 20: 243–267. https://doi.org/10.1128/cmr.00037-06

29. Alexander DJ, (2007) An overview of the epidemiology of avian influenza. *Vaccine* 25: 5637–5644. https://doi.org/10.1016/j.vaccine.2006.10.051

30. Gao R, Cao B, Hu Y, Feng Z, Wang D, Hu W, et al. (2013) Human infection with a novel avian-origin influenza A (H7N9) virus. *N Engl J Med*, 368: 1888–1897. https://doi.org/10.1056/NEJMoa1304459

31. Centers for Disease Control and Prevention, Avian Influenza in Mammals, 2024. Available from: https://www.cdc.gov/bird-flu/situation-summary/mammals.html.

32. USDA, USDA, FDA and CDC Share Update on HPAI Detections in Dairy Cattle, 2024. Available from: https://www.aphis.usda.gov/news/agency-announcements/usda-fda-cdc-share-update-hpai-detections-dairy-cattle.

33. Government of Canada, The Government of Canada provides an update on Highly Pathogenic Avian Influenza, 2024. Available from: https://www.canada.ca/en/food-inspection-agency/news/2024/05/the-government-of-canada-provides-an-update-on-the-highly-pathogenic-avian-influenza.html.
34. Government of Canada, Animals susceptible to H5N1 highly pathogenic avian influenza (HPAI), 2024. Available from: https://inspection.canada.ca/en/animal-health/terrestrial-animals/diseases/reportable/avian-influenza/animals-susceptible-h5n1-hpai.
35. Peng L, Yang W, Zhang D, Zhuge C, Hong L, Epidemic analysis of COVID-19 in China by dynamical modeling. preprint, arXiv:2002.06563. https://doi.org/10.48550/arXiv.2002.06563
36. Ning X, Guan J, Li XA, Wei Y, Chen F, (2023) Physics-informed neural networks integrating compartmental model for analyzing COVID-19 transmission dynamics. *Viruses* 15: 1749. https://doi.org/10.3390/v15081749
37. Yang Z, Zeng Z, Wang K, Wong SS, Liang W, Zanin M, et al. (2020) Modified SEIR and AI prediction of the epidemics trend of COVID-19 in China under public health interventions. *J Thorac Dis*, 12: 165. https://doi.org/10.21037/jtd.2020.02.64

Appendix

PyTorch algorithm for automatic differentiation

In our code, we utilize PyTorch's automatic differentiation capabilities to compute the time derivatives of the network outputs with respect to time t. The key steps in our implementation are as follows:

Computing Time Derivatives:

To compute the time derivatives $\frac{d\hat{S}_a}{dt}, \frac{d\hat{I}_a}{dt}, \frac{d\hat{S}_h}{dt}, \frac{d\hat{I}_h}{dt}, \frac{d\hat{R}_h}{dt}$, we use the following procedure:

- Define masks `self.m1`, `self.m2`, `self.m3`, `self.m4`, `self.m5`, which are zero tensors except for a one in the position corresponding to the variable of interest.
- Perform a backward pass with respect to the input time t using these masks to compute the gradients.

Code Snippet: Here is the relevant portion of our code:

```
# Compute gradients for each compartment with respect to time:
# S_a_t
si_sir_hat.backward(self.m1, retain_graph=True)
S_a_hat_t = self.t.grad.clone()
self.t.grad.zero_()

# I_a_t
si_sir_hat.backward(self.m2, retain_graph=True)
I_a_hat_t = self.t.grad.clone()
self.t.grad.zero_()
```

```
# S_h_t
si_sir_hat.backward(self.m3, retain_graph=True)
S_h_hat_t = self.t.grad.clone()
self.t.grad.zero_()

# I_h_t
si_sir_hat.backward(self.m4, retain_graph=True)
I_h_hat_t = self.t.grad.clone()
self.t.grad.zero_()

# R_h_t
si_sir_hat.backward(self.m5, retain_graph=True)
R_h_hat_t = self.t.grad.clone()
self.t.grad.zero_()
```

In this code:

- `si_sir_hat` is the output of the neural network, which is a tensor containing $\hat{S}_a, \hat{I}_a, \hat{S}_h, \hat{I}_h, \hat{R}_h$ for all time points.
- The `backward` function computes the gradient of the network outputs with respect to the input t.
- The masks `self.m1`, $\ldots$, `self.m5` ensure that we compute the gradient for only one output at a time.
- We clone the gradients and reset them after each computation to avoid accumulation.

Calculating the Residuals:

Once we have the time derivatives, we calculate the residuals $f_j(t_i)$ by substituting the network's predictions and the computed derivatives into the SI-SIR model equations.

In our code, this is implemented as:

```
# Unnormalize the predictions
S_a = self.S_a_min + (self.S_a_max - self.S_a_min) * S_a_hat
I_a = self.I_a_min + (self.I_a_max - self.I_a_min) * I_a_hat
S_h = self.S_h_min + (self.S_h_max - self.S_h_min) * S_h_hat
I_h = self.I_h_min + (self.I_h_max - self.I_h_min) * I_h_hat
R_h = self.R_h_min + (self.R_h_max - self.R_h_min) * R_h_hat



# Compute the residuals (normalized)
f1_hat = S_a_hat_t -
(self.r_a * S_a * (1 - S_a / (self.K_a + self.epsilon))-
(self.beta_a * I_a * S_a) / (1 + self.b * I_a + self.epsilon)) /
(self.S_a_max - self.S_a_min + self.epsilon)

f2_hat = I_a_hat_t - ((self.beta_a * I_a * S_a) /
(1 + self.b * I_a + self.epsilon) - (self.mu_a + self.delta_a) * I_a) /
(self.I_a_max - self.I_a_min + self.epsilon)

f3_hat = S_h_hat_t - (self.Pi_h - (self.beta_h * I_a * S_h) /
(1 + self.c * I_h**2 + self.epsilon) - self.mu_h * S_h) /
(self.S_h_max - self.S_h_min + self.epsilon)

f4_hat = I_h_hat_t - ((self.beta_h * I_a * S_h) /
(1 + self.c * I_h**2 + self.epsilon) -
(self.mu_h + self.delta_h + self.gamma) * I_h) /
(self.I_h_max - self.I_h_min + self.epsilon)

f5_hat = R_h_hat_t - (self.gamma * I_h - self.mu_h * R_h) /
(self.R_h_max - self.R_h_min + self.epsilon)
```

Here:

- `S_a_hat_t`, `I_a_hat_t`, ... are the time derivatives of the normalized compartment values.
- We un-normalize the predictions before substituting them into the SI-SIR equations.
- The residuals f_j are computed in their normalized form to maintain consistency with the normalized predictions and derivatives.
- ϵ is a small constant added to prevent division by zero.

Total Loss Calculation:
Finally, we compute the total loss by summing the data loss and the physics loss:

```
loss = (torch.mean(torch.square(self.S_a_hat - S_a_pred)) +
        torch.mean(torch.square(self.I_a_hat - I_a_pred)) +
        torch.mean(torch.square(self.S_h_hat - S_h_pred)) +
        torch.mean(torch.square(self.I_h_hat - I_h_pred)) +
        torch.mean(torch.square(self.R_h_hat - R_h_pred)) +
        torch.mean(torch.square(f1_hat)) +
        torch.mean(torch.square(f2_hat)) +
        torch.mean(torch.square(f3_hat)) +
        torch.mean(torch.square(f4_hat)) +
        torch.mean(torch.square(f5_hat))
        )
```

In this expression:

- The first five terms correspond to the data loss (MSE between the normalized true values and predictions).
- The last five terms correspond to the physics loss (MSE of the residuals).

Optimization: During training, we minimize the total loss using an optimizer (for example, Adam optimizer) and update the network parameters accordingly.

```
self.optimizer.zero_grad()
loss.backward()
self.optimizer.step()
self.scheduler.step()
```

By incorporating both data and physics losses, the DINN is guided to produce solutions that are consistent with both the observed data and the underlying physical laws described by the differential equations.

Normalization and Stability Considerations: Normalization is crucial in our implementation to ensure numerical stability and effective training:

- We normalize the compartment values to the range [0, 1] using the maximum and minimum values from the data.
- During the computation of residuals, we un-normalize the predictions to substitute them into the SI-SIR equations.
- We use a small epsilon ($\epsilon = 1 \times 10^{-8}$) to prevent division by zero.

Parameter Constraints: We enforce biologically feasible ranges for the model parameters using the hyperbolic tangent (tanh) function:

$$\text{Parameter} = \tanh(\text{Parameter}_{\text{tilda}}) \times \text{Scale} + \text{Shift} \tag{6.1}$$

For example:

```
@property
def r_a(self):
  return torch.tanh(self.r_a_tilda) * 0.01 + 0.005
 # Range: [0.0045, 0.0055]
```

This approach ensures that during training, the parameters stay within specified bounds reflecting realistic biological values.