THE MATHEMATICS OF DEEP NEURAL NETWORKS WITH APPLICATION IN PREDICTING THE SPREAD OF INFECTIOUS DISEASES THROUGH DISEASE-INFORMED NEURAL NETWORKS (DINNs)

NICKSON GOLOOBA

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF ARTS

GRADUATE PROGRAM IN MATHEMATICS AND STATISTICS

YORK UNIVERSITY
TORONTO, ONTARIO

September, 2024

Abstract

Deep learning has emerged in many fields in recent times where neural networks are used to learn and understand data. This thesis combines deep learning frameworks with epidemiological models and is aimed specifically at the creation and testing of DINNs with a view to modeling the infection dynamics of epidemics. This research thus trains the DINN on synthetic data derived from an SI-SIR model designed for Avian influenza and shows the models accuracy in predicting extinction and persistence conditions. In the method, a twelve hidden layer model was constructed with sixty-four neurons per layer and ReLU activation function was used. The network is 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, was characterized of the loss of data and physics, which made the DINN follow the differential equations that fundamentally described the disease progression.

Acknowledgment

I want to express my gratitude to God for providing me with the strength and wisdom needed to finish this journey. I truly believe that, without His guidance and blessings, I wouldn't have been able to accomplish any of this. A big thank you to my parents; Justine Nabwami and Godfrey Kayondo, for their support, on this journey of mine! Mom I truly appreciate your love and encouragement that never wavered in the toughest times. Your strength and wisdom have always kept me going. My sincere thanks go out to my beloved sister Joan Nakimera for being there for me, and providing me with love and support that has truly meant everything to me.

This project is also dedicated, to my brother Allan Genza who left us on August 18th of 2023. Your memory has been a guiding light and a source of motivation, especially to decide and do research in disease modelling since you were taken away from us by a deadly disease. I am grateful, for my friends' support on this journey of mine right from Italy to Canada. I'm particularly thankful to Samuel Agenorwoth's companionship and assistance since the beginning of our shared adventure in Italy despite us going our ways now. My deepest gratitude to Professor Woldegebriel Assefa Woldegerima for his support through funding and in other ways, guidance and mentorship throughout this research project, because his expertise and encouragement have been invaluable to the completion of this work. I truly appreciate it! I'm grateful that he never gave up on me and made sure everything was done on schedule while also assisting me in choosing an significant topic, for my study. I'm extremely grateful to Professor Huaiping Zhu for serving so well on my supervisory committee and providing me support through the Ontario Modeling Network for Infectious Diseases (OMNI) program. The necessary computational resources and funding that OMNI has provided me for this research, is extremely appreciated.

My heartfelt thanks go out to the University of L'Aquila for their support, in both academic aspects throughout my journey to meet deadlines successfully. A big thanks also goes to York University, for their funding that made my studies possible. To all my friends and study buddies with whom I exchanged thoughts and embarked on this adventure together, in particular Roosvel Tatsinkou Tenekeu with whom I shared a lot with great experiences in Canada including studying together all the time and helping each other in every kind of situation. I genuinely appreciate your companionship and support. Lastly, Grammarly and my great friend, Derek Mak have been of great help too, in fixing any English grammar errors and poor English phrases that I made while writing my work and this made the work appear much clearer and understandable.

Contents

List of Tables

List of Figures

Chapter 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 thesis 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 [49]. Paving the way for such enhanced modeling frameworks became especially important in the context of the COVID-19 pandemic. The basis of this work therefore involves the integration of deep learnings strengths with the traditional compartmental models of epidemiology including the SIR model and its extensions. It could be the first time for some readers to encounter the combined use of epidemiological models mathematical formulas within a flexible and powerful neural network structure, but this constitutes the main accomplishment of this thesis.

Deep learning, a profound advancement in AI, employs complex neural networks to enable computers to interpret vast and intricate data, like images and audio. This technology didn't emerge abruptly but evolved through decades of innovative thinking and modern computational advancements [32]. Essential to this technological progress are neural networks and layered structures resembling a simplified brain, which gradually learn complex features from data, assisting in various sectors including health care and broader technological applications [33], [37].

The solid mathematical foundation underlying deep learning is crucial. It not only enhances the ability of neural networks to process visual information but

also 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 [33], [32]. 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 [32].

Neural networks started back in the 1940s, thanks to the amazing ideas of Walter Pitts and Warren McCulloch, with significant milestones like the development of backpropagation in the 1970s enhancing their practical applicability. A renewed focus on this field and progress made around 2012 have solidified the position of neural networks as a leading area of study, in AI research.[37].

Today, the challenge and urgency in understanding the dynamics of infectious diseases, as exemplified by the SARS-CoV-2 pandemic, has led to the use of advanced AI techniques. The development of DINNs represents an innovative integration of epidemiological data into neural network frameworks. This integration aims to increase the accuracy and speed of modeling and predicting the spread of infectious diseases, by drawing on historical insights from foundational models like those of Kermack and McKendrick along with the computing power of modern deep learning systems[28]; [47].

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 [41].

This thesis 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 analysis, with real world applications in predicting diseases, this study intends to add to academic understanding and effective approaches, in tackling worldwide health emergencies.

Chapter 2

Literature review

This chapter discusses the existing research literature concerning the use of deep learning in fields specifically emphasizing public health and epidemiology. It covers the development of neural network technologies and how they are utilized in health information systems for studying diseases. Additionally it investigates the incorporation of AI technologies into the traditional epidemiological models.

2.1 Evolution of Deep Learning

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 [37]. In 1943 they envisioned introducing intelligence to the world [39]. 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 and so that's how they pioneered a neural network [33]. According to [37], 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 [37]. The achievement propelled networks, like Convolutional Neural Networks (CNNs) and Long Short Term Memory (LSTM) networks to the forefront of AI research. Advancements, in technology have allowed deep learning systems to tackle pattern recognition issues in fields

marking a step forward, in the field of artificial intelligence [15].

In recent years, the introduction of Transformers by [56], has had an impact on the field of Natural Language Processing (NLP). This innovative architecture has revolutionized the way long range dependencies are modeled leading to advancements in areas such as image processing and time series analysis.

Following the success of Transformers, advanced Large Language Models (LLMs) like GPT-4 from OpenAI, Claude from Anthropic and Gemini from Google DeepMind have further pushed the boundaries of AI. These models utilize the capabilities of Transformers to create contextually relevant text showcasing performance in tasks like text generation, summarization and translation. GPT-4 and its counterparts have also been integrated into various applications, including chatbots like ChatGPT, for engaging conversational experiences.

2.2 Applications in Epidemiology

Deep learning has proved useful in predicting the spread of infectious diseases for instance, when the COVID-19 pandemic hit, deep learning models were used to predict how the disease would spread, helping to allocate vital healthcare resources [41, 42, 49]. However, they often rely on large datasets and extensive computer resources and can be difficult to interpret, making them challenging to use in public health, where explanations of predictions are often crucial [29].

2.3 Integration of Traditional Models with AI

Epidemiological models, like the SIR model have played a role in our understanding of how diseases spread [22, 5]. However, they sometimes struggle to capture the details of real-world data [6]. By combining AI techniques with these models we can improve prediction accuracy through advanced data analysis methods, improved parameter estimations, and modeling complex relationships between different variables [58, 57].

2.4 Gaps and Opportunities

While deep learning models have shown promise there are still obstacles to overcome such as issues related to data quality and quantity ensuring model trans-

parency and effectively incorporating real-time data [57, 25]. Tackling these challenges presents opportunities, for advancing modeling and creating resilient and understandable models [58].

Research Questions

a) How can traditional epidemiological models be enhanced using deep neural networks?

b) What mathematical foundations are needed to integrate epidemiological parameters into neural network architectures?

c) How can the integration of classical epidemiological models with modern neural network architectures improve public health decision-making?

Chapter 3

The Mathematics of Deep Neural Networks (DNNs)

In this chapter, we start with the foundations of DNNs and then dive into the theoretical insights and architectural variations concerning these networks. It is important to note that for the rest of the work discussed here, when we mention "Neuron", we are actually referring to an "Artificial Neuron (AN)".

Neurons: An artificial neuron functions as a unit that receives inputs, computes them with a weighted sum by adding them up with a bias value and then processes the outcome through an activation function.

Definition 3.0.1. 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

$$\begin{aligned} f(x_1, \cdots, x_n) &= \rho\left(\sum_{i=1}^{n} x_i w_i + b\right) \\ &= \rho\left((x, w) + b\right), \end{aligned} \tag{3.1}$$

where $w = (w_1, \cdots, w_n)$ and $x = (x_1, \cdots, x_n)$. See, Figure 3.1 for a graphical representation of an Artificial Neuron structure.

Weights (w): These are the parameters that determine the strength and sign of the input at each neuron. During network training adjustments to these parameters are usually made to reduce prediction errors.

Biases (b): The bias allows the activation function to be shifted left or right, which can be crucial for learning the relationships in data.

Activation Functions (ρ): Activation functions add non-linear elements to the network enabling it to to learn more complex functions. Biology-inspired activation functions are mathematical functions that mimic the behavior of neurons in the brain. Common activation functions include [17],

$$\textbf{Sigmoid } \sigma(x) = \frac{1}{1+e^{-x}} \tag{3.2}$$

$$\textbf{Tanh } \sigma(x) = \tanh(x), \tag{3.3}$$

$$\textbf{ReLU (Rectified Linear Unit) } \sigma(x) = \max(0,x)., \tag{3.4}$$

$$\textbf{Leaky ReLU Function } \sigma(x) = \max(\alpha x, x). \tag{3.5}$$

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. Note that the Leaky ReLU Function is a variation of ReLU, where a small fraction α of the input is allowed to pass through, even when negative. There are several other biology-inspired activation functions, such as the Softmax function which is used for multi-class classification, inspired by the normalized firing rates of neurons [7], Spiking Activation Functions inspired by the spiking behavior of biological neurons for example; integrate-and-fire [34] and FitzHugh-Nagumo [13];[40] and lastly the other activation functions to talk about are Bursting activation functions for example Hindmarsh-Rose [23].

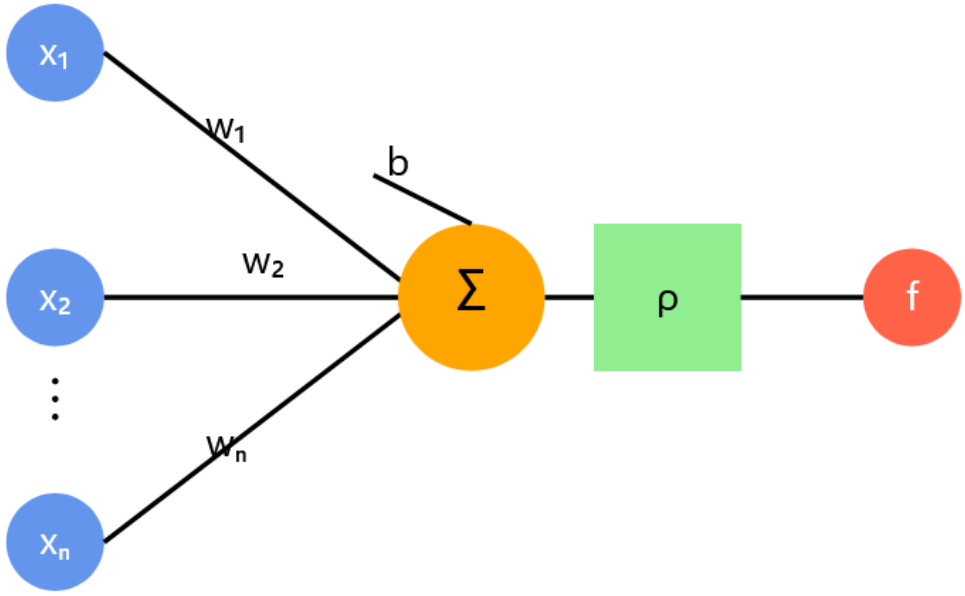

Figure 3.1: An Artificial Neuron $f : \mathbb{R}^n \to \mathbb{R}$

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 the following types of layers:

(i). **Input Layer:** This is comprised of nodes or units where the model takes its input data.

(ii). **Hidden Layers:** These layers are made of neurons to perform most of the computational lifting, learning the intrinsic patterns within data. If there is often more than one hidden layer in a neural network, then we have what is known as a "deep" neural network.

(iii). **Output Layer:** This is the final layer also comprised of neurons that provide the output of the model, structured to suit the specific type of prediction or classification problem being addressed.

At each step the data is converted into formats that showcase distinctive features, prepared for examination, at all levels. The arrangement and intricacy of these layers greatly impact the networks capacity to grasp patterns and tackle diverse

challenges ranging from recognizing images and speech, to mastering intricate strategy games.

3.1 Artificial Neural Network

After covering all the elements needed for constructing a neural network, lets now delve into what it actually is.

Definition 3.1.1. : Let $d \in \mathbb{N}$ be the dimension of the input layer, L the number of layers, $N_0 := d$, N_ℓ, $\ell = 1, \cdots, L$, the dimensions of the hidden and last layer, $\rho : \mathbb{R} \to \mathbb{R}$ a non-linear activation function, and, for $\ell = 1, \cdots, L$, 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. The composition of affine-linear functions with a non-linear activation function, given by $\Phi(x) = T_L \rho(T_{L1} \rho(\cdots \rho(T_1(x))))$ where $x \in$, is referred to as a neural network of depth L.

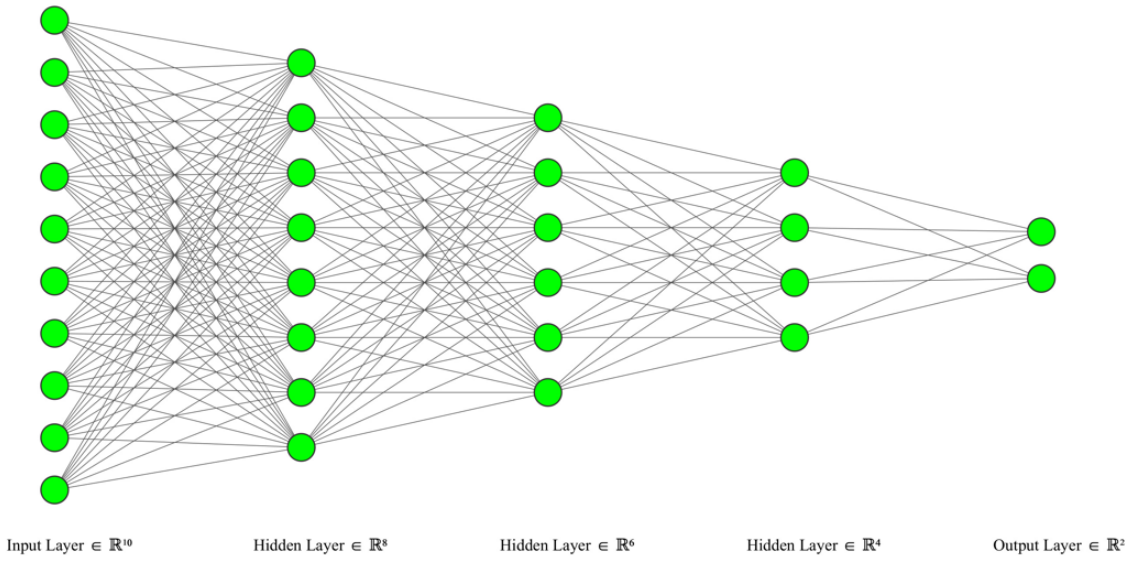

Figure 3.2: Deep Neural Network $\Phi : \mathbb{R}^{10} \to \mathbb{R}^2$ with depth 4. [36]

3.2 Backpropagation Algorithm

In the realm of neural networks the backpropagation algorithm plays a role, in fine-tuning network parameters. Its primary objective is to minimize the loss function, which gauges the variance, between the output and the desired target [48]. This procedure can be divided into two steps as described below:

a) Forward pass: In this step, we calculate the loss by running the input through the network and comparing the expected output, with the target.

b) Backward pass: In this step we determine the gradient of the loss concerning each weight using the chain rule and adjust the weights to reduce the loss

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

$$\frac{\partial L}{\partial w_{ji}^{(l)}} = \frac{\partial L}{\partial a_j^{(l)}} \cdot \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} \cdot \frac{\partial z_j^{(l)}}{\partial w_{ji}^{(l)}}.$$

Loss Function: The common loss functions includes both the Mean Squared Error (MSE) and Cross-Entropy Loss represented mathematically as:

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

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

For regression tasks, the Mean Squared Error (MSE) is often used and it is defined as:

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

For classification tasks, the Cross-Entropy Loss is commonly used and it is defined as:

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

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.

3.3 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.

Stochastic Gradient Descent (SGD): The most basic and widely used optimization algorithm is Stochastic Gradient Descent (SGD). In SGD, the gradient of the loss function is computed for each training example, and the weights are updated accordingly.
The update rule for SGD is given by

$$w_{ji}^{(l)} = w_{ji}^{(l-1)} - \eta \frac{\partial L}{\partial w_{ji}^{(l)}},$$

where η is the learning rate.

The learning rate determines the size of the steps taken towards the minimum of the loss function. A smaller learning rate may lead to slow convergence, while a larger learning rate may cause the algorithm to overshoot the minimum.

Mini-batch Gradient Descent: In practice, a variant called mini-batch gradient descent is often used. Instead of computing the gradient for a single training example (as in SGD) or the entire training set (as in batch gradient descent), mini-batch gradient descent computes the gradient over a small batch of training examples.
This approach provides a good trade-off between the computational efficiency of batch gradient descent and the convergence stability of SGD [3].

Adam Optimizer: Adam (Adaptive Moment Estimation) is an optimization algorithm that combines the advantages of two other extensions of SGD: AdaGrad and RMSprop. Adam maintains two moving averages of the gradients, which are used to adapt the learning rate for each parameter.

The update rules for Adam are as follows:

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

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

b) Correct the bias in the estimates as

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

c) Update the weights as

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

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 $\varepsilon = 10^{-8}$ [30].

RMSprop: RMSprop (Root Mean Square Propagation) is another adaptive learning rate method designed to maintain a per-parameter learning rate that improves the training speed. The RMSprop update rule is given by

$$\begin{aligned} v_t &= \beta v_{t-1} + (1-\beta)(\nabla L)^2 \\ w_{ji}^{(l)} &= w_{ji}^{(l-1)} - \eta \frac{\nabla L}{\sqrt{v_t} + \varepsilon}. \end{aligned}$$

where v_t is the moving average of the squared gradients and β is a decay hyperparameter, typically set to 0.9 [53].

Convergence of Gradient Descent: One of the key theoretical aspects of gradient descent and its variants is the convergence properties. Gradient descent algorithms are guaranteed to converge to a local minimum for convex loss functions, provided the learning rate is chosen appropriately. For non-convex loss

functions, such as those encountered in neural networks, gradient descent can converge to a local minimum, a saddle point, or a global minimum depending on the initialization and the landscape of the loss function [21].

Theorem 1 (Convergence Rate of Gradient Descent)**.** *For a convex and differentiable function f with L-Lipschitz continuous gradients, the gradient descent method with a fixed step size $\eta = \frac{1}{L}$ converges at a rate of*

$$f(x^{(k)}) - f(x^*) \leq \frac{L\|x^{(0)} - x^*\|^2}{2k},$$

where x^* is the optimal solution.

This result provides a theoretical guarantee on the convergence rate of gradient descent for convex optimization problems.

Sketch Proof

a) Gradient descent update rule

$$x^{(k+1)} = x^{(k)} - \eta \nabla f(x^{(k)}).$$

b) Using the Lipschitz continuity of the gradient

$$f(x^{(k+1)}) \leq f(x^{(k)}) + \langle \nabla f(x^{(k)}), x^{(k+1)} - x^{(k)} \rangle + \frac{L}{2}\|x^{(k+1)} - x^{(k)}\|^2.$$

c) Substitute the update rule

$$f(x^{(k+1)}) \leq f(x^{(k)}) - \frac{\eta}{2}\|\nabla f(x^{(k)})\|^2.$$

d) Choosing $\eta = \frac{1}{L}$

$$f(x^{(k+1)}) \leq f(x^{(k)}) - \frac{1}{2L}\|\nabla f(x^{(k)})\|^2.$$

e) Summing Over Iterations

$$f(x^{(k)}) - f(x^*) \leq \frac{L\|x^{(0)} - x^*\|^2}{2k}.$$

3.4 Convergence Properties

Gradient descent convergence refers to the conditions under which gradient descent algorithms converge to a local or global minimum. For convex loss functions, gradient descent is guaranteed to converge to a global minimum, provided the learning rate is appropriately chosen [4]. However, for non-convex loss functions, such as those encountered in neural networks, convergence can lead to a local minimum, saddle point, or global minimum depending on the initialization and the landscape of the loss function [3, 51].

One of the key factors affecting convergence is the learning rate. If the learning rate is too high, the algorithm may overshoot the minimum, leading to divergence. Conversely, if the learning rate is too low, the convergence process can be excessively slow. The choice of an adaptive learning rate, as used in optimization algorithms like Adam and RMSprop, can help mitigate these issues by adjusting the learning rate dynamically during training [30, 53].

Mathematically, gradient descent can be expressed as:

$$w_{t+1} = w_t - \eta \nabla L(w_t).$$

where η is the learning rate, and $\nabla L(w_t)$ is the gradient of the loss function L with respect to the weights w at iteration t.

3.5 Generalization Bounds

Generalization bounds provide insight into the capacity of neural networks to perform well on unseen data. Measures such as VC (Vapnik-Chervonenkis) dimension and Rademacher complexity are used to quantify the capacity of neural networks and their generalization properties [55, 2].

The VC dimension is a measure of the capacity of a statistical classification algorithm, defined as the cardinality of the largest set of points that the algorithm can shatter. A model with a high VC dimension can fit a wide variety of functions, indicating high capacity but also a potential for overfitting [55].

Rademacher complexity measures the richness of a class of functions by evaluating the model's ability to fit random noise. It provides a way to quantify how well a learning algorithm can generalize from the training data to unseen data. Lower Rademacher complexity indicates better generalization performance [2].

3.6 Expressivity of Deep Neural Networks (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 [32].

The Universal Approximation Theorem is a foundational result in the theory of neural networks. It states that a neural network with at least one hidden layer and non-linear activation functions can approximate any continuous function to arbitrary precision, given sufficient neurons [26]. 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 [27, 14].

According to [32], the universal approximation theorem states that each continuous function on a compact domain can be approximated to an arbitrary accuracy by a shallow neural network.

Theorem 2. *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 $\varepsilon > 0$, 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*

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

Historical Context and Significance: The theorem was independently proven by George Cybenko in 1989 for sigmoid activation functions [10], and by Kurt Hornik in 1991, who generalized the result to a broader class of activation functions, including the ReLU (Rectified Linear Unit) [26]. These results provided a theoretical foundation for the empirical success of neural networks in various applications, demonstrating their capacity to serve as universal function approximators.

Implications for Deep Learning: The Universal Approximation Theorem is significant because it ensures that neural networks have the ability to approximate any function accurately given resources, like neurons and computational power. This fundamental concept is what drives the success of learning models in fields such as image recognition, natural language processing and gaming [31].

It is worth noting that while the theorem guarantees the existence of a network for approximation it doesn't offer a clear method, for determining the network structure or training process needed to achieve this accuracy. In practice implementing networks involves dealing with challenges related to optimization, generalization and computational efficiency that go beyond what the theorem covers [18].

Limitations and Extensions: While the Universal Approximation Theorem is indeed a discovery, it does come with its set of limitations. This theorem specifically applies to smooth functions and it does not directly tackle the approximation of functions that are discontinuous or defined on domains that are not compact [46]. Moreover the theorem is focused on networks, with one hidden layer whereas in deep learning practices, networks with multiple layers (deep networks) are commonly utilized. Studies have indicated that deep networks can offer efficiency in terms of neuron count needed to achieve a level of approximation, as shallow networks [52]. In fact, we will employ Deep Neural Networks (DNNs) in this work, instead of traditional shallow neural networks in order to capture the complex, non-linear relationships inherent in epidemiological dynamics. The additional layers and neurons in a DNN enable the model to learn high level features and elaborate patterns in the data, which are essential for accurately modeling the spread of infectious diseases like Avian influenza.

Recent work has extended the universal approximation capabilities to other architectures and settings, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), demonstrating their ability to approximate complex functions and dynamic processes [61]

After having explored the aspects and theoretical insights of DNNs, let us now delve into their architectural variations.

3.7 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are created to process data that follows a grid structure, such, as images. CNNs use layers that apply a convolution operation to the input and pass the outcome to the next layer. This operation is represented mathematically as;

$$(f * g)(t) \quad = \quad \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau.$$

where f is the input function and g is the kernel or filter. In discrete terms, for an input x and a filter w, the convolution operation is

$$\begin{aligned} S(i,j) &= (x * w)(i,j) \\ &= \sum_m \sum_n x(i+m, j+n) w(m,n). \end{aligned}$$

Pooling layers, like max pooling or average pooling decrease the dimensions of the data. For instance the max pooling operation can be written as;

$$y_{i,j} \quad = \quad \max_{(m,n) \in \text{pool}} x_{i+m, j+n}.$$

Pooling offers translation invariance and reduces computational load by down-sampling the input representation, which is essential for efficiently handling large images [35, 31].

3.8 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are tailored for sequential data. They maintain a hidden state h_t that retains information from previous time steps.
The update of the state is defined by;

$$h_t \quad = \quad \sigma(W_h h_{t-1} + W_x x_t + b).$$

where W_h and W_x are weight matrices, x_t is the input at time t, b is the bias and σ is the activation function.

Standard RNNs face the challenge of the disappearing gradient issue, where gradients of the loss function with respect to earlier layers diminish significantly. To mitigate this problem Long Short Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) incorporate gating mechanisms . LSTM are comprised of a memory cell and gates which are activated by a sigmoid function that takes on output values 1 to mean "remember" and 0 to mean "forget" [24, 9].

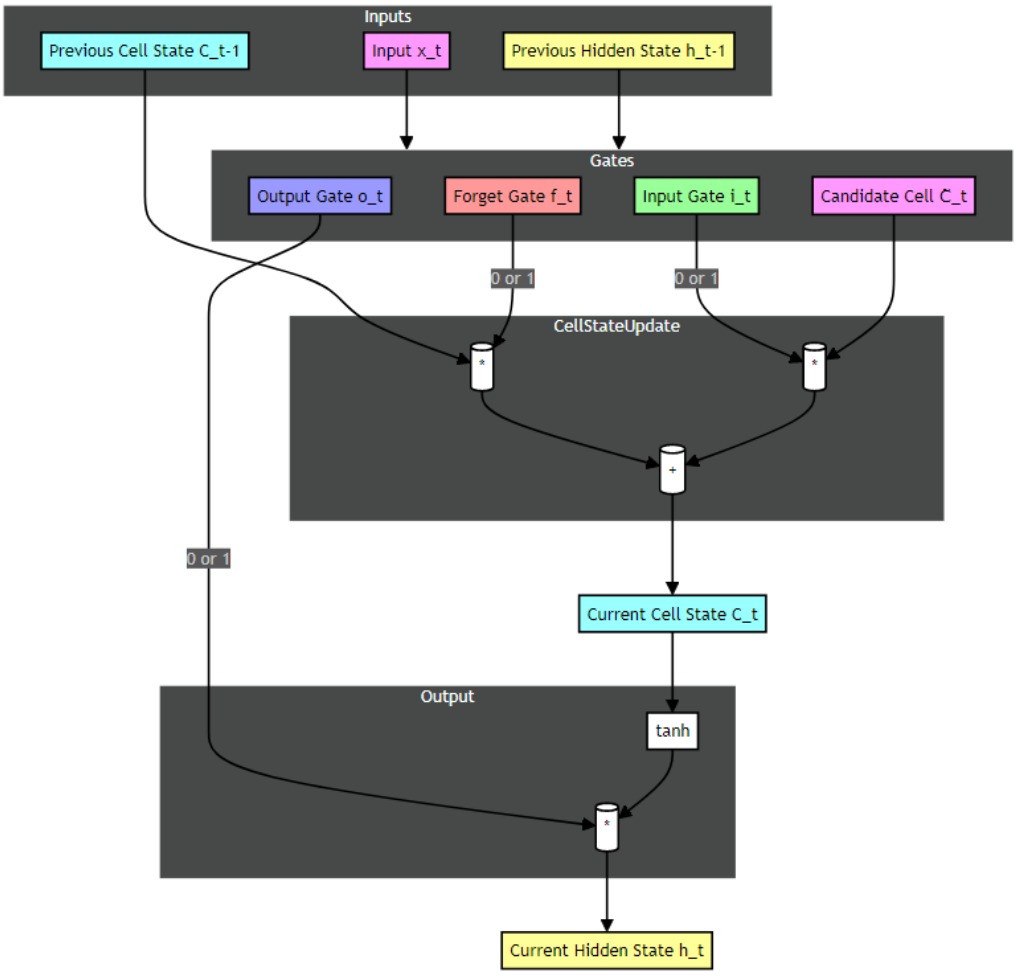

Figure 3.3: LSTM Cell Structure

The cell state plays a role, in an LSTM cell enabling it to store long term information across time steps and tackle the vanishing gradient issue in recurrent neural networks. In Figure 3.3 we can see how the LSTM cell operates and gets updated.

The **cell state**, C_t is updated by combining the retained information from the previous cell state C_{t-1} with new candidate values. This is done using the forget

gate f_t and the input gate i_t:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.6}$$

where $*$ denotes element-wise multiplication.

The **forget gate**, f_t determines what should be forgotten from the cell state.

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right) \tag{3.7}$$

Here, σ is the sigmoid function, W_f is the weight matrix, h_{t-1} is the previous hidden state, x_t is the current input, and b_f is the bias term.

The **input gate**, i_t decides which values to modify in the cell state;

$$i_t = \sigma\left(W_i[h_{t-1}, x_t] + b_i\right) \tag{3.8}$$

where W_i and b_i are the weight matrix and bias term for the input gate, respectively.

The **candidate cell state**, $\tilde{C}_t$ represents new information that could supplement the existing one;

$$\tilde{C}_t = \tanh\left(W_C[h_{t-1}, x_t] + b_C\right) \tag{3.9}$$

where tanh is the hyperbolic tangent function, W_C is the weight matrix, and b_C is the bias term for the candidate cell state.

The **output gate** o_t, controls which hidden state that will be carried to the next time step:

$$o_t = \sigma\left(W_o[h_{t-1}, x_t] + b_o\right), \tag{3.10}$$

where W_o and b_o are the weight matrix and bias term for the output gate, respectively.

Subsequently the **hidden state** h_t is calculated as a result of multiplying the output gate by the activated cell state and delivers the output for the next time step;

$$h_t \quad = \quad o_t * \tanh(C_t).$$

The updating mechanism of the cell state plays a role in LSTMs ability to maintain long term dependencies making it successful, in sequence based tasks.

3.9 Feed Forward Neural Networks (FFNNs)

The Feed Forward Neural Networks (FFNNs) represent the simplest form of neural network architecture in which information flows in one forward direction from input nodes, through any hidden nodes and to the output nodes without any loops or cycles present, in the structure.

During the pass, you calculate the activation of each layer one after the other, starting from the input layer, and moving toward the output layer. This can be explained mathematically as in the definition below.

Definition 3.9.1. 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 $\mathbf{a}^{(\ell)}$.

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

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.

3.10 Transformer Neural Networks

Transformers are a type of network architecture designed to handle sequential data, commonly applied in natural language processing (NLP). In contrast to recurrent networks (RNNs), transformers do not process data sequentially but rely on self attention to establish global relationships between input and output [56].

Mathematical Representation:

1. **Self-Attention Mechanism**: The self attention mechanism in transformers calculates a weighted sum of all representations, for each input token by utilizing Query (Q) Key (K) and Value (V) matrices.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

   where d_k is the dimension of the keys, and the softmax function ensures the weights sum to one.

2. **Multi-Head Attention**: Instead of using one attention function the transformer utilizes multiple attention heads each having its own Q, K and V matrices. The outputs are then concatenated and linearly transformed;

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\text{head}_2,\ldots,\text{head}_h)W^O,$$

   where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$.

3. **Positional Encoding**: Transformers lack a sense of sequence order in the data, so positional encodings are included in the input embeddings to represent the sequences order.

$$\begin{aligned}\text{PE}_{(pos,2i)} &= \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)\\ \text{PE}_{(pos,2i+1)} &= \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right).\end{aligned}$$

   Transformers overcome the limitations of RNNs by enabling processing during training leading to speed enhancements and improved handling of long range dependencies. The success of this architecture, in NLP tasks is credited to its capability to capture relationships among all words, in a sentence irrespective of their positions through self attention mechanisms [56].

Chapter 4

Predicting the Spread of Infectious Diseases

In this chapter we will now discuss classical epidemiological models that influence our decision, in selecting the disease model utilized in our implementation through Neural networks following a Physics Informed Neural Networks (PINNs) approach. These particular types of Neural Networks are referred to as Disease Informed Neural Networks (DINNs) as explored by Shaier, Raissi and Seshaiyer [49]. The detailed information about DINNs is discussed in Section 4.1.

4.1 Epidemiological Models

Classical epidemiological models are representations that explain the transmission of diseases among a population. Among the models are the Susceptible Infectious Recovered (SIR) model and its variants, such, as the Susceptible Exposed Infectious Recovered (SEIR) model.

4.1.1 SIR Model

The SIR model is comprised of the following population division compartments;

a) Susceptible (S): Individuals who can contract the disease.

b) Infectious (I): Individuals who have contracted the disease and can spread it.

c) Recovered (R): Individuals who have recovered from the disease and are assumed to be immune.

The model is described by the following differential equations [28]

$$\begin{aligned} \frac{dS}{dt} &= -\beta SI \\ \frac{dI}{dt} &= \beta SI - \gamma I \\ \frac{dR}{dt} &= \gamma I, \end{aligned}$$

where β is the transmission rate and γ is the recovery rate. These equations can be analyzed to understand the dynamics of disease spread. The basic reproduction number R_0 is a key parameter, defined as

$$R_0 = \frac{\beta}{\gamma}.$$

If $R_0 > 1$, the disease will spread in the population, while if $R_0 < 1$, the disease will eventually die out [11].

4.1.2 SEIR Model

The SEIR model extends the SIR model by including an exposed (E) compartment for individuals who have been exposed to the disease but are not yet infectious. The model equations are:

$$\begin{aligned} \frac{dS}{dt} &= -\beta SI \\ \frac{dE}{dt} &= \beta SI - \sigma E \\ \frac{dI}{dt} &= \sigma E - \gamma I \\ \frac{dR}{dt} &= \gamma I, \end{aligned}$$

where σ is the rate at which exposed individuals become infectious.

The inclusion of the exposed compartment, in the SEIR model allows for capturing the period during which the disease remains dormant (incubation period) an aspect

when dealing with diseases that have a delay between exposure and infectiousness. While sharing the reproduction number as the previously mentioned SIR model the SEIR models dynamics can become more intricate due to this additional compartment [22].

4.1.3 Extensions and Applications

The extensions of both the SIR and SEIR models give rise to variations like the SIRS model, which considers waning immunity over time and the SEIRS model incorporating both an exposed phase and diminishing immunity [22]. These models have been applied to a good number of infectious diseases such as influenza, measles and more notably COVID-19. They play a role in informing health strategies and predicting the outcomes under different intervention scenarios [12].

In this study we examine a special extension known as the SI-SIR model focusing on Avian influenza modeling. Before going into the details of this model it is essential to provide an overview of Avian influenza to establish our rationale before selecting this disease for detailed examination in our thesis work. We break it down into three sections that is to say; introduction and current context, explanation of the SI-SIR model, and numerical simulations, for the Avian influenza.

a) 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 [50].
Avian influenza viruses are categorized into two groups based on how harmful they're to poultry; Low Pathogenic Avian Influenza (LPAI). 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 [44], [1].

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 [16].

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 the 25th March, 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 cows according to a report by CDC, in 2024 [8].

There are worries, about the virus's ability to change and spread easily among people leading to increased efforts in monitoring and controlling outbreaks [54].
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 [20]. These actions are aimed at preventing the virus from entering the 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 including Canada where it has impacted commercial and non commercial bird populations significantly [19].

Inspired by all these, we decided to choose this disease for our implementation.

b) SI-SIR Model for Avian Influenza

In our study we use a version of the SI-SIR model to examine how Avian influenza transmission. This model is based on the work of Yu and Ma (2022) which looks at nonlinear incidence and recovery rates in both deterministic and stochastic environments [60]. Unlike the model's varying recovery rate in their study, our simplified version assumes a constant recovery rate for the humans. This simplification allows us to focus on key transmission dynamics without dealing with the complexities of changing recovery rates.

The SI-SIR models differential equations are as follows;

$$\frac{dS_a}{dt} = r_a S_a \left(1 - \frac{S_a}{K_a}\right) - \frac{\beta_a I_a S_a}{1 + bI_a} \tag{4.1}$$

$$\frac{dI_a}{dt} = \frac{\beta_a I_a S_a}{1 + bI_a} - (\mu_a + \delta_a) I_a \tag{4.2}$$

$$\frac{dS_h}{dt} = \Pi_h - \frac{\beta_h I_a S_h}{1 + cI_h^2} - \mu_h S_h \tag{4.3}$$

$$\frac{dI_h}{dt} = \frac{\beta_h I_a S_h}{1 + cI_h^2} - (\mu_h + \delta_h + \gamma) I_h \tag{4.4}$$

$$\frac{dR_h}{dt} = \gamma I_h - \mu_h R_h, \tag{4.5}$$

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

Variable	**Description**	**Initial Value**	**Reference**
S_a	Susceptible avian population	9900	[38]
I_a	Infectious avian population	100	[38]
S_h	Susceptible human population	9999	[38]
I_h	Infectious human population	1	[38]
R_h	Recovered human population	0	[38]

Table 4.1: Model variables and initial values.

Parameter	Description	Value	Reference
r_a	Intrinsic growth rate of avian population	5×10^{-3}	[38]
K_a	Carrying capacity for avian population	5×10^{4}	[38]
β_a	Transmission rate from avian to avian	Varied: 0.8×10^{-8} to 2.6×10^{-8}	[60]
β_h	Transmission rate from avian to human	8×10^{-7}	[38]
μ_a	Natural death rate for avian population	3.4246×10^{-4}	[38]
b	Nonlinear incidence coefficient (avian)	0.001	[60]
μ_h	Natural death rate for human population	3.91×10^{-5}	[38]
δ_a	Disease-induced death rate for avian population	4×10^{-4}	[38]
c	Nonlinear incidence coefficient (human)	0.01	[60]
γ	Constant recovery rate of human population	0.1	[38]
δ_h	Disease-induced death rate for human population	0.077	[38]
Π_h	Recruitment rate of human population	30	[38]

Table 4.2: Model parameters.

It is important to note that the transmission rate from avian to avian (β_a) was varied in the study conducted by [60] to examine different scenarios.
When $\beta_a = 0.8 \times 10^{-8}$, the basic reproduction number $R_0 = 0.54 < 1$, leading to disease extinction.
When $\beta_a = 2.6 \times 10^{-8}$, $R_0 = 1.75 > 1$, resulting in disease persistence. According to [60], increasing the parameter β_a impacts whether the disease persists or is eliminated resulting in numbers of infected individuals.

This model helps in understanding how Avian influenza spreads and identify 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 [60] 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.

c) Numerical simulation for Avian influenza

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 [60] using the same parameters as used in their study.

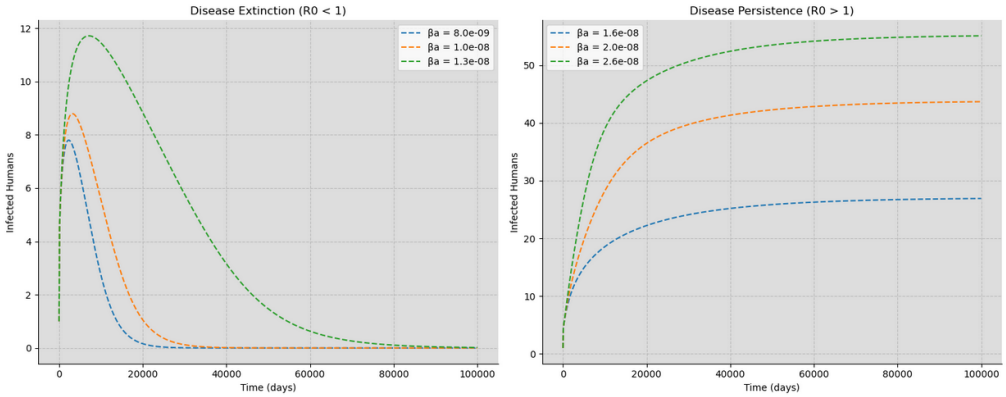

Figure 4.1: Avian influenza dynamics for different β_a values.

Figure 4.1 shows the results of these simulations for different values of β_a. The left subplot illustrates scenarios where the basic reproduction number $R_0 < 1$, 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 4.1 and the parameter values as detailed in Table 4.2. The time span for the simulation was set from 0 to 100,000 days to better understand the long-term

behavior of the pandemic.

As depicted in Figure 4.1, 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 non-zero value, signifying disease persistence.

These simulations confirm the findings of [60], 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 Disease Informed Neural Network (DINN) model enabling us to assess its effectiveness, in capturing both disease extinction and persistence scenarios in the next chapter.

4.2 Disease Informed Neural Networks (DINNs)

The combination of machine learning methods, with modeling has become a tool for comprehending and forecasting disease patterns [42]. Disease Informed Neural Networks (DINNs) as introduced by [49], are a technique merging neural networks 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, a specialized type of Physics Informed Neural Networks (PINNs) designed for epidemiological applications[47]. This approach uses effectively the universal approximation 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 [45].

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 [45].

Although an amazing work on DINNs has been done by [49] and other related work by [42];[43], these 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 for example on; the number of neurons in each layer of the network, learning rate, search range, and number of epochs are based on the findings by [49].

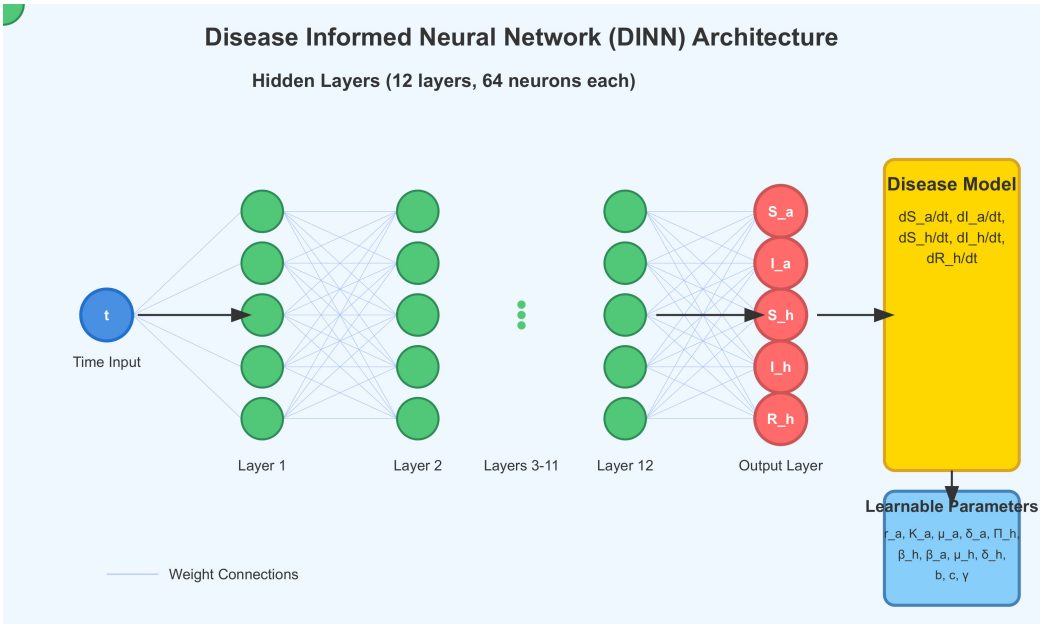

Figure 4.2: Disease Informed Neural Network (DINN) architecture for avian influenza model

For our Avian influenza model, we apply the DINN framework as illustrated in Figure 4.2. Our specific definition of the DINN, as illustrated in Figure 4.2, 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 Sub-section 4.1.3.

All computational experiments were performed on a high-performance virtual machine provided by the Ontario Modeling Network for Infectious Diseases (OMNI). 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
- **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:

$$\text{Total Loss} = \text{Data Loss} + \text{Physics Loss} \tag{4.6}$$

Data Loss: This measures the discrepancy between the network's predictions and the available synthetic data using the Mean Squared Error (MSE):

$$\text{Data Loss} = \frac{1}{N}\sum_{i=1}^{N}\left((\hat{S}_{a,i}-S_{a,i})^2+(\hat{I}_{a,i}-I_{a,i})^2+(\hat{S}_{h,i}-S_{h,i})^2+(\hat{I}_{h,i}-I_{h,i})^2+(\hat{R}_{h,i}-R_{h,i})^2\right) \tag{4.7}$$

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 right-hand sides of the SI-SIR model equations:

$$\text{Physics Loss} = \frac{1}{N}\sum_{i=1}^{N}\left(f_1^2(t_i)+f_2^2(t_i)+f_3^2(t_i)+f_4^2(t_i)+f_5^2(t_i)\right) \tag{4.8}$$

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

$$f_1(t_i) = \frac{d\hat{S}_a}{dt} - \left(r_a\hat{S}_a\left(1-\frac{\hat{S}_a}{K_a}\right) - \frac{\beta_a\hat{I}_a\hat{S}_a}{1+b\hat{I}_a}\right) \tag{4.9}$$

$$f_2(t_i) = \frac{d\hat{I}_a}{dt} - \left(\frac{\beta_a\hat{I}_a\hat{S}_a}{1+b\hat{I}_a} - (\mu_a+\delta_a)\hat{I}_a\right) \tag{4.10}$$

$$f_3(t_i) = \frac{d\hat{S}_h}{dt} - \left(\Pi_h - \frac{\beta_h\hat{I}_a\hat{S}_h}{1+c\hat{I}_h^2} - \mu_h\hat{S}_h\right) \tag{4.11}$$

$$f_4(t_i) = \frac{d\hat{I}_h}{dt} - \left(\frac{\beta_h\hat{I}_a\hat{S}_h}{1+c\hat{I}_h^2} - (\mu_h+\delta_h+\gamma)\hat{I}_h\right) \tag{4.12}$$

$$f_5(t_i) = \frac{d\hat{R}_h}{dt} - \left(\gamma\hat{I}_h - \mu_h\hat{R}_h\right) \tag{4.13}$$

By minimizing the *Total Loss* in Equation (4.6), the DINN learns to produce predictions that not only fit the data but also satisfy the SI-SIR model dynamics.

Implementation Details Using 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 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`, ..., `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 unnormalize 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 unnormalize the predictions to substitute them into the SI-SIR equations.
- We use a small epsilon ($\varepsilon = 1 \times 10^{-8}$) to prevent division by zero.

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

$$\text{Parameter} = \tanh(\text{Parameter}_{\text{tilda}}) \times \text{Scale} + \text{Shift} \tag{4.14}$$

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.

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.

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 [59].

Chapter 5

Results

In this chapter 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.

5.1 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 DINNs 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 equations (4.1)–(4.5)). It is important to note that these results were obtained with specific hyperparameters; a learning rate of $1e^{-4}$, 1000 time points, and a parameter search range of approximately 1000% for the same architecture as in Figure 4.2.

Parameter	DINN Estimate	Target Value	Relative Error	Search Range
r_a	0.004777	0.005000	4.46%	[0.0045, 0.0055]
K_a	50005.0	50000.0	0.01%	[0, 100000]
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	31.00	30.00	3.33%	[-70, 130]
β_h	0.0000009723	0.0000008000	21.54%	[0.0000006, 0.000001]
β_a	0.00000002402	0.000000026	7.62%	[0.8e-8, 2.6e-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]

Table 5.1: Comparison of DINN's estimated parameters with target values (learning rate: $1e^{-4}$, time points: 1000).

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

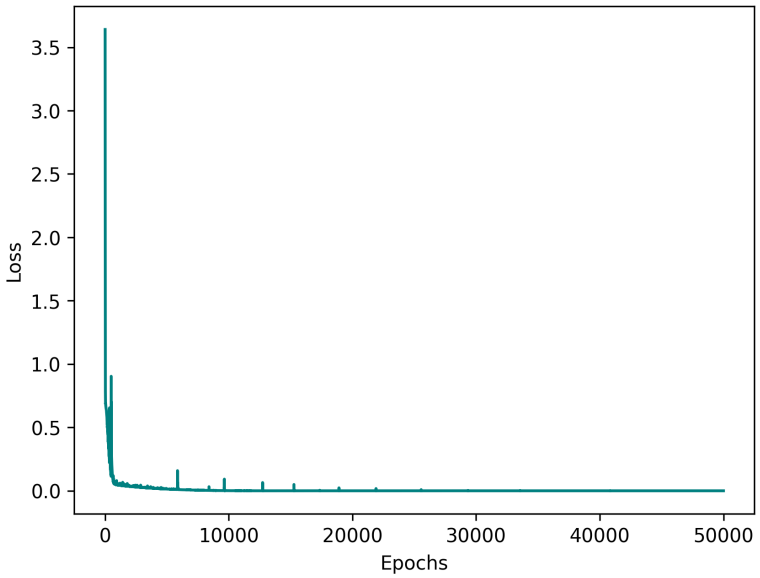

Figure 5.1: Loss vs. Epochs during DINN Training

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 5.1 illustrates how the loss function evolved during training.

5.2 Analysis of DINN training and parameter estimation

In this section, we review the outcomes from Section 5.1 of our DINN training process. 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 5.1 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, on both the models behavior and outcomes.

The search ranges for each parameter specified in our code are displayed in the rightmost column of Table 5.1. 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 an 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 [49], 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 $1e^{-6}$, 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 $1e^{-4}$, 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 $1e^{-6}$ 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 $= 1e^{-8}$). 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 5.1 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 models 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.

Selection of Neural Network Architecture: To determine the optimal depth for our Disease Informed Neural Network (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 5.2: 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	8.00×10^{-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 5.2, 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.

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 5.2, a 12-layer DINN is the most suitable choice for our framework because it balances accuracy and computational efficiency. This decision is supported by empirical evidence from our experiments but we expect a better performance as we increase the number of layers of our DINN. This expectation can be supported by the work done by [49].

5.3 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.

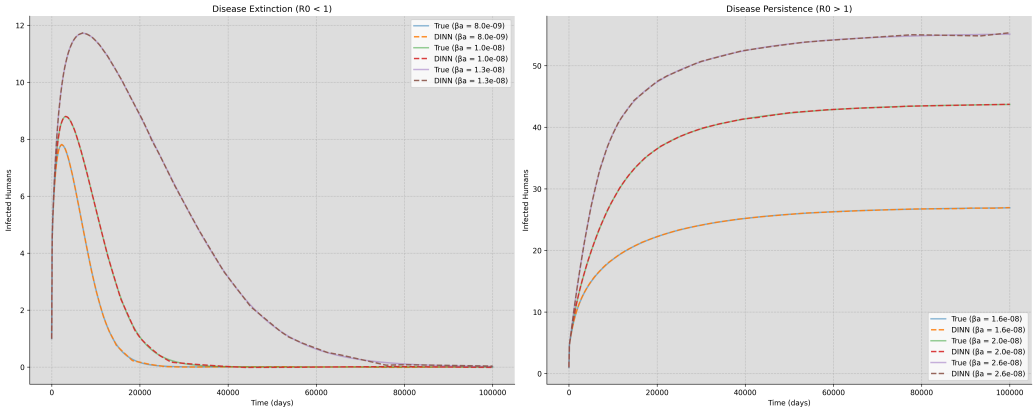

Figure 5.2: DINN predictions vs. true values for disease extinction $(R_0 < 1)$ and persistence $(R_0 > 1)$.

To generate the prediction plots in Figure 5.2, we use the same DINN methodology as described in Section 5.1.

For each β_a value $(8.0\times10^{-9}, 1.0\times10^{-8}, 1.3\times10^{-8}, 1.6\times10^{-8}, 2.0\times10^{-8},$ and $2.6\times10^{-8})$, 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 diseases become extinct ($R_0 < 1$) while the right side shows scenarios where diseases persist ($R_0 > 1$). 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 models capacity to capture both extinction and persistence dynamics.

5.4 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 compartments of the system and analyzing its performance, under varying transmission rates.

5.4.1 Disease Extinction $(R_0 < 1)$

In the left panel of Figure 5.2, 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 its robust to changes in the transmission rate.

5.4.2 Disease Persistence $(R_0 > 1)$

The right panel of Figure 5.2 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 $R_O < 1$ 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.

5.4.3 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 Relative Mean Squared Error (RMSE) losses is displayed in the figure below.

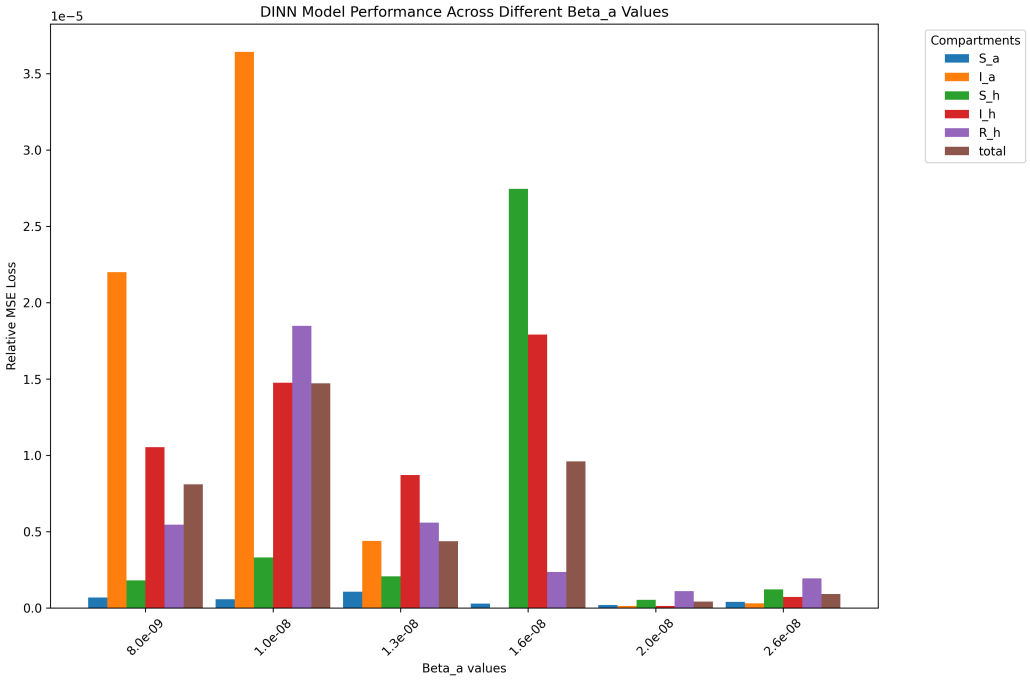

Figure 5.3: DINN Model Performance for Different β_a Values

By analyzing the graph in Figure 5.3 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 is able to attend to conditions and make predictions correspondingly depending on transmission rate.

Chapter 6

Discussion, Conclusion and Future work

Here in this chapter, it will be our endeavor to discuss more about our DINN in the context of Avian influenza modeling. Looking back to the disease data, we consider the extent to which the model is able to accommodate different disease conditions, contemplate its advantages and disadvantages, and think about our contributions to the improvement of the methods used in epidemic modeling. We then provide a brief discussion of what can be done next to improve and extend this work.

6.1 Discussion

In this section we discuss our thesis findings regarding the implementation and performance of our DINN, for avian influenza modeling. We explore the models strengths, weaknesses and implications drawing insights from our results within the context of modeling and machine learning applications in public health.

We are quite impressed by how the DINN managed both disease extinction and persistence scenarios. Capturing these dynamics was a challenging task but our model consistently handled it well.
Concerning the lower values of β_a we noticed 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 was 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 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 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 point affected the performance of the DINN. It is like 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 models 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.

6.2 Conclusion

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 very beneficial in the public health sector concerning planning since it can anticipate a variety of possible outcomes.

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. Also, if we can find ways to diminish the differences in parameter estimates between runs, it could add great robustness to our model as well. This leads us to the next section where we discuss this possible future work.

To sum up, 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 dance 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.

6.3 Future Work

Our investigation, into using DINNs for modeling Avian influenza has revealed some avenues for future research. In future studies, one might want to explore the following research directions;

1. Solving the puzzle of the I_a compartment
   It has been observed that our model sometimes struggles with predicting the infectious Avian population particularly when transmission rates are low. One could delve deeper into this by;

   (a) Experimenting with various neural network architectures or exploring the use of a network or a transformer-based model.

   (b) Adjusting the loss function to prioritize giving importance to the I_a component during training.

2. Strength in collaboration; Embracing teamwork
   Considering approaches to address fluctuations in parameter estimates one could explore teaming up by;

(a) Training multiple DINNs and combining their predictions to yield results.

(b) Delving into techniques, like bagging or boosting within this context.

3. Exploring new horizons. Other diseases
   It could also be interesting to explore the applicability of our DINN method to other various infectious diseases, addressing the following questions.

   (a) Can we adapt our model for conditions with transmission patterns like vector-borne diseases?

   (b) Is it feasible to assess our approach to a disease with a complex compartmental model?

4. Diving into reality. Dealing with data
   Although our synthetic data has served well for demonstrating the concept there may be a desire to validate our model in real-world scenarios. A study could be conducted to answer the questions below.

   (a) How does our DINN handle noisy or incomplete data which is a challenge in the field of epidemiology?

   (b) 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.

5. Understanding the image. Making it clear and understandable.
   It is important to make sure that our model is transparent and not seen as something useless.

   (a) Is there a way we can use techniques like SHAP (SHapley Additive exPlanations) values or integrated gradients to understand the factors affecting our predictions?

   (b) Can we explore the learned representations of our network (DNN) that might provide us with insights, into how it understands disease patterns?

The other last potential area we can think of is using LSTMs as a substitute, for the feedforward neural network structure employed in this research project. LSTMs are adept at handling time series information and they have a 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.

Here are some pathways to consider. As we improve and expand our DINN approach, we are optimistic about its potential to become a tool, in the arsenal of epidemiologists helping them understand and forecast disease patterns in an ever-changing environment.

Bibliography

[1] Dennis J Alexander. "An overview of the epidemiology of avian influenza". In: *Vaccine* 25.30 (2007), pp. 5637–5644.

[2] Peter L Bartlett and Shahar Mendelson. "Rademacher and Gaussian complexities: Risk bounds and structural results". In: *Journal of Machine Learning Research* 3.Nov (2002), pp. 463–482.

[3] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer. 2010, pp. 177–186.

[4] S Boyd. "Vandenberghe., L.(2004) Convex Optimization". In: *Cambridge university press* 29.30 (), p. 223.

[5] Fred Brauer, Carlos Castillo-Chavez, and Carlos Castillo-Chavez. *Mathematical models in population biology and epidemiology*. Vol. 2. 40. Springer, 2012.

[6] Fred Brauer, Carlos Castillo-Chavez, Zhilan Feng, et al. *Mathematical models in epidemiology*. Vol. 32. Springer, 2019.

[7] John S. Bridle. "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition". In: *Neurocomputing: Algorithms, Architectures and Applications*. Ed. by F. Fogelman-Soulié and J. Hérault. Berlin, Heidelberg: Springer, 1990, pp. 227–236. DOI: 10.1007/978-3-642-76153-9_28.

[8] Centers for Disease Control and Prevention. *Avian Influenza in Mammals*. 2024. URL: https://www.cdc.gov/bird-flu/situation-summary/mammals.html (visited on 07/22/2024).

[9] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[10] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[11] Odo Diekmann, Johan Andre Peter Heesterbeek, and Johan Anton Jacob Metz. “On the definition and the computation of the basic reproduction ratio R 0 in models for infectious diseases in heterogeneous populations”. In: *Journal of mathematical biology* 28 (1990), pp. 365–382.

[12] NM Ferguson. “LD, Nedjati-Gilani G, Imai N, Ainslie K, Baguelin M, Bhatia S, Boonyasiri A, Cucunubá Z, Cuomo-Dannenburg G, Dighe A”. In: *Impact of non-pharmaceutical interventions (NPIs) to reduce COVID-19 mortality and healthcare demand. Imperial College COVID-19 Response Team, London* (2020).

[13] Richard FitzHugh. “Impulses and Physiological States in Theoretical Models of Nerve Membrane”. In: *Biophysical Journal* 1.6 (1961), pp. 445–466. DOI: 10.1016/S0006-3495(61)86902-6.

[14] Ken-Ichi Funahashi. “On the approximate realization of continuous mappings by neural networks”. In: *Neural networks* 2.3 (1989), pp. 183–192.

[15] Edgar Galván and Peter Mooney. “Neuroevolution in deep neural networks: Current trends and future challenges”. In: *IEEE Transactions on Artificial Intelligence* 2.6 (2021), pp. 476–493.

[16] Rongbao Gao et al. “Human infection with a novel avian-origin influenza A (H7N9) virus”. In: *New England Journal of Medicine* 368.20 (2013), pp. 1888–1897.

[17] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep sparse rectifier neural networks”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[19] Government of Canada. *Animals susceptible to H5N1 highly pathogenic avian influenza*. 2024. URL: https://inspection.canada.ca/en/animal-health/terrestrial-animals/diseases/reportable/avian-influenza/animals-susceptible-h5n1-hpai (visited on 07/22/2024).

[20] Government of Canada. *The Government of Canada provides an update on the highly pathogenic avian influenza*. 2024. URL: 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 (visited on 07/22/2024).

[21] Moritz Hardt, Ben Recht, and Yoram Singer. "Train faster, generalize better: Stability of stochastic gradient descent". In: *International conference on machine learning*. PMLR. 2016, pp. 1225–1234.

[22] Herbert W Hethcote. "The mathematics of infectious diseases". In: *SIAM review* 42.4 (2000), pp. 599–653.

[23] J. L. Hindmarsh and R. M. Rose. "A Model of Neuronal Bursting Using Three Coupled First Order Differential Equations". In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 221.1222 (1984), pp. 87–102. DOI: 10.1098/rspb.1984.0024.

[24] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[25] Andreas Holzinger et al. "What do we need to build explainable AI systems for the medical domain?" In: *arXiv preprint arXiv:1712.09923* (2017).

[26] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2 (1991), pp. 251–257.

[27] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[28] William Ogilvy Kermack and Anderson G McKendrick. "A contribution to the mathematical theory of epidemics". In: *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character* 115.772 (1927), pp. 700–721.

[29] Daniel S Kermany et al. "Identifying medical diagnoses and treatable diseases by image-based deep learning". In: *cell* 172.5 (2018), pp. 1122–1131.

[30] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[32] Gitta Kutyniok. *An introduction to the mathematics of deep learning*. 2023.

[33] Gitta Kutyniok. "The mathematics of artificial intelligence". In: *Proceedings of the International Congress of Mathematicians*. 2022.

[34] Louis Lapicque. "Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation". French. In: *Journal de Physiologie et de Pathologie Générale* 9 (1907), pp. 620–635.

[35] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[36] Alex Lenail. “NN-SVG: Publication-Ready Neural Network Architecture Schematics”. In: *Journal of Open Source Software* 4.33 (2019), p. 747. DOI: 10.21105/joss.00747. URL: https://doi.org/10.21105/joss.00747.

[37] Aisha Lichtner-Bajjaoui. “A Mathematical Introduction to Neural Networks”. In: (2021).

[38] Sanhong Liu, Shigui Ruan, and Xinan Zhang. “Nonlinear dynamics of avian influenza epidemic models”. In: *Mathematical biosciences* 283 (2017), pp. 118–135.

[39] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5 (1943), pp. 115–133.

[40] Jin-ichi Nagumo, Suguru Arimoto, and Shuji Yoshizawa. “An Active Pulse Transmission Line Simulating Nerve Axon”. In: *Proceedings of the IRE* 50.10 (1962), pp. 2061–2070. DOI: 10.1109/JRPROC.1962.288235.

[41] Long Nguyen, Maziar Raissi, and Padmanabhan Seshaiyer. “Modeling, Analysis and Physics Informed Neural Network approaches for studying the dynamics of COVID-19 involving human-human and human-pathogen interaction”. In: *Computational and Mathematical Biophysics* 10.1 (2022), pp. 1–17.

[42] Xiao Ning et al. “Epi-DNNs: Epidemiological priors informed deep neural networks for modeling COVID-19 dynamics”. In: *Computers in biology and medicine* 158 (2023), p. 106693.

[43] Xiao Ning et al. “Physics-informed neural networks integrating compartmental model for analyzing COVID-19 transmission dynamics”. In: *Viruses* 15.8 (2023), p. 1749.

[44] JS Malik Peiris, Menno D De Jong, and Yi Guan. “Avian influenza virus (H5N1): a threat to human health”. In: *Clinical microbiology reviews* 20.2 (2007), pp. 243–267.

[45] Liangrong Peng et al. “Epidemic analysis of COVID-19 in China by dynamical modeling”. In: *arXiv preprint arXiv:2002.06563* (2020).

[46] Allan Pinkus. “Approximation theory of the MLP model in neural networks”. In: *Acta numerica* 8 (1999), pp. 143–195.

[47] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse

problems involving nonlinear partial differential equations". In: *Journal of Computational physics* 378 (2019), pp. 686–707.

[48] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[49] Sagi Shaier, Maziar Raissi, and Padmanabhan Seshaiyer. "Data-driven approaches for predicting spread of infectious diseases through DINNs: Disease Informed Neural Networks". In: *arXiv preprint arXiv:2110.05445* (2021).

[50] Wenhan Shao et al. "Evolution of influenza a virus by mutation and reassortment". In: *International journal of molecular sciences* 18.8 (2017), p. 1650.

[51] Ruoyu Sun. "Optimization for deep learning: theory and algorithms". In: *arXiv preprint arXiv:1912.08957* (2019).

[52] Matus Telgarsky. "Benefits of depth in neural networks". In: *Conference on learning theory*. PMLR. 2016, pp. 1517–1539.

[53] Tijmen Tieleman. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* 4.2 (2012), p. 26.

[54] USDA. *USDA, FDA, CDC Share Update on HPAI Detections in Dairy Cattle*. 2024. URL: https://www.aphis.usda.gov/news/agency-announcements/usda-fda-cdc-share-update-hpai-detections-dairy-cattle (visited on 07/22/2024).

[55] Vladimir Naumovich Vapnik, Vlamimir Vapnik, et al. "Statistical learning theory". In: (1998).

[56] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[57] Jenna Wiens and Erica S Shenoy. "Machine learning for healthcare: on the verge of a major shift in healthcare epidemiology". In: *Clinical infectious diseases* 66.1 (2018), pp. 149–153.

[58] Shihao Yang, Mauricio Santillana, and Samuel C Kou. "Accurate estimation of influenza epidemics using Google search data via ARGO". In: *Proceedings of the National Academy of Sciences* 112.47 (2015), pp. 14473–14478.

[59] Zifeng Yang et al. "Modified SEIR and AI prediction of the epidemics trend of COVID-19 in China under public health interventions". In: *Journal of thoracic disease* 12.3 (2020), p. 165.

[60] Xingwang Yu and Yuanlin Ma. “An avian influenza model with nonlinear incidence and recovery rates in deterministic and stochastic environments”. In: *Nonlinear Dynamics* 108.4 (2022), pp. 4611–4628.

[61] Ding-Xuan Zhou. “Universality of deep convolutional neural networks”. In: *Applied and computational harmonic analysis* 48.2 (2020), pp. 787–794.