DATA-DRIVEN DEEP NEURAL NETWORKS FOR EPIDEMIOLOGICAL AND BIOCHEMICAL MODELS

By

Kayode Daniel Olumoyin

A Dissertation

Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Computational Science

Middle Tennessee State University

May, 2022

Dissertation Committee:

Dr. Abdul Q.M. Khaliq, (Mathematical Sciences), Chair.

Dr. Wandi Ding, (Mathematical Sciences).

Dr. William Robertson, (Physics).

Dr. Chris Stephens, (Mathematical Sciences)

ABSTRACT

In recent years, the efficiency of deep neural networks to forward and inverse problems that have found applications in biochemical and epidemiological models has been demonstrated. In these systems, it has been observed that the system parameters fit unknown nonlinear functions. Classical models mostly assume these system parameters to be constants. Some researchers explicitly chose a form for these nonlinear parameters using intuition and good reasoning. In this work, we will study mathematical models with nonlinear dynamics that occur in biochemical and epidemiological models and we will develop data-driven deep learning approaches to learn the nonlinear parameters in these models and thereby detect hidden patterns in these complex systems. In our study, we will employ a modification of the physics-informed neural network. The modified network, which we call an epidemiology informed neural network, allows us to predict nonlinear system parameters. Here, multilayer perceptrons are connected to a larger multilayer perceptron that learns the solution to a system of partial differential equations or a system of ordinary differential equations. The Neural network approaches we present are suitable for partial differential equations and ordinary differential equations because they are meshless and can scale to high spatial dimensions. They can also solve forward and inverse problems with sparse data. We enforce the physics of our model in the objective function and device efficient methods that allows us to train with a small dataset. The adaptive neuro-fuzzy inference system, a widely used Neural network in time series forecast, is combined with an epidemiology informed neural network. We demonstrate that this hybrid network is an improvement over the adaptive neuro-fuzzy inference system. We also demonstrate that the epidemiology informed neural network combined with a recurrent neural network such as the long short-term memory network provides a more accurate short-term forecast than a plain recurrent neural network. Next, we develop an attention-based neural network that is capable of learning nonlinear dynamics from noisy data.

Dedicated to
my wife Bukola, my daughter Semilore,
my Dad Gabriel Adeniyi Olumoyin, my Mom Patricia Fehintola Olumoyin,
and to the loving memory of my brother Bolaji Olumoyin.

ACKNOWLEDGMENTS

"May the LORD now show you kindness and faithfulness, and I too will show you the same favor because you have done this."

2 Samuel 2:6

I am grateful to my advisor Professor Abdul Khaliq for his patience, leadership, and encouragement throughout my doctoral program at Middle Tennessee State University (MTSU). I am thankful to Professor John Wallin, the program director of the Computational Science (COMS) program, for funding my studies at MTSU and for all his support to students in the COMS program. I am indebted to Professor Khaled Furati for the countless meetings and his many questions that help shape this dissertation and my research work. In the Spring of 2019, Dr. Guofei Pang visited MTSU at the invitation of Professor Abdul Khaliq. I am grateful for those meetings our group had with Dr. Pang. I am also thankful to the CRUNCH group at Brown University, led by Professor George Em Karniadakis, for the opportunity to be a part of the weekly CRUNCH seminar. I congratulate Dr. Maziar Raissi, Dr. Paris Perdikaris, and Professor George Em Karniadakis for their landmark paper on the Physics-informed Neural Network and their continued research on Deep Learning.

I am thankful to Professor Wandi Ding, late Professor Tibor Koritsanszky, Professor William Robertson, and Professor Chris Stephens for agreeing to be on my committee. I took COMS 7840 with Professor Robertson in Fall 2019 and it is one of my favorite experiences at MTSU. I am thankful to Dr. Joshua Phillips for granting me access to the BIOSIM cluster. I also want to thank him for his approach to the deep learning course he taught in Fall 2021 where I first learned about Attention mechanism models. The models described in the dissertation were trained using the BIOSIM cluster.

The chair of the University Studies department at MTSU, Dr. Marva Lucas, was instrumental to my coming to MTSU in 2016, when she gave me a job as a lecturer in the University Studies department. I am thankful for her generosity to me and my family. I am also thankful to the following people of the University studies department for their kindness over the years- Dr. Marva Lucas, Dr. Linda Clark, Dr. M.A. Higgs, Dr. Vivian Alley, Dr. Tim Nelson, Ms. Gina Johnson, and Mr. Thomas Torku.

This pursuit has been a journey of several sacrifices for Bukola. It will pay off, I promise.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

1

INTRODUCTION

"With the right features, almost any machine learning algorithm will find what you are looking for. Without good features; none will."

Unknown

In the late 1500s, Danish astronomer, Tycho Brahe collected accurate data about the orbit of Mars. Many years later, his assistant Johannes Kepler mapped Brahe's data to an ellipse showing that it was mathematically possible to predict planetary motions [1]. This was one of the earliest recorded use of a data-driven approach to formulate and discover the governing laws of our physical and celestial world. In contrast, about 300 years after Brahe, Joseph Fourier formulated the heat equation [2]-an example of what is called a parabolic partial differential equation. Fourier's work on the heat equation was a triumph of reasoning. And for many decades, such reasoning influenced computational modeling and the formulation of governing equations for differential equation models.

Machine Learning has been described as the field of study that gives computers the ability to learn without being explicitly programmed [3]. In [4], Tom Mitchell describes Machine Learning as the study of computer algorithms that improve automatically through experience. Some of the common machine learning task includes: Classification, Regression, Machine translation, Anomaly detection [5]. For instance, in a regression task, we may describe the accuracy of a machine learning algorithm by how much its prediction matches a target function at domain values where the target function is known.

In recent years, we are seeing a resurgence of data-driven discovery and identification of differential equation models [6, 7, 8, 9]. Advancements in neural networks and data science have made it possible to learn complex patterns from data [10, 11]. This chapter

describes deep neural networks. In Section 1.1, we provide a mathematical introduction of deep neural networks. In Section 1.2, we discuss the deep neural network architectures that motivate the neural network models and solvers in this dissertation. In Section 1.3 we provide a description of the deep neural network solvers.

1.1 Deep Neural Networks

We define a deep neural network (DNN) to be a composition of functions. Here, we consider a DNN to be a neural network architecture called the Feedforward Neural Network (FNN). The FNN is acyclic, that is, the neural network flows in one direction and it is usually a fully connected network.

$$\mathscr{N}(U;\theta) = \sigma(W_L\sigma(\dots\sigma(W_2\sigma(W_1U+b_1)+b_2)\dots)+b_L), \tag{1.1}$$

where $U = [u_1, u_2, \dots, u_n]^T$ is the input vector and $\mathscr{N}(\mathscr{U}) = [\mathscr{N}_1, \mathscr{N}_2, \dots, \mathscr{N}_m]^T$ is the output vector. W_k and b_k, $k = 1, \dots, L$, are the neural network weights and biases. The neural network described in (1.1) are compositions of linear functions together with $\sigma(\cdot)$. σ is usually a nonlinear function, it is called an activation function. It enforces some rules on the neural network. For instance, if we want our neural network to match a non-negative target function, we can choose a non-negative σ. There are different types of activation functions. It is important to choose an appropriate activation function, see figure (1.2) when constructing a neural network architecture. There are $L-1$ hidden layers in (1.1). Figure (1.1) is a simple schematic of a deep neural network with an input layer, 2 hidden layers, and an output layer. The backpropagation algorithm [11], one of the reasons deep neural networks are successful, is used to adjust the neural network weights and biases. This is called Training the deep neural network. The trainable parameters $\theta := (W_1, \dots, W_L, b_1, \dots, b_L)$ are updated after each epoch. This is an optimization problem (1.2) where we find the optimal parameters θ^* that yield the optimal output $\mathscr{N}$ of the deep neural network (1.1).

$$\theta^* = \arg\min_{\theta} \mathscr{N}(U;\theta). \tag{1.2}$$

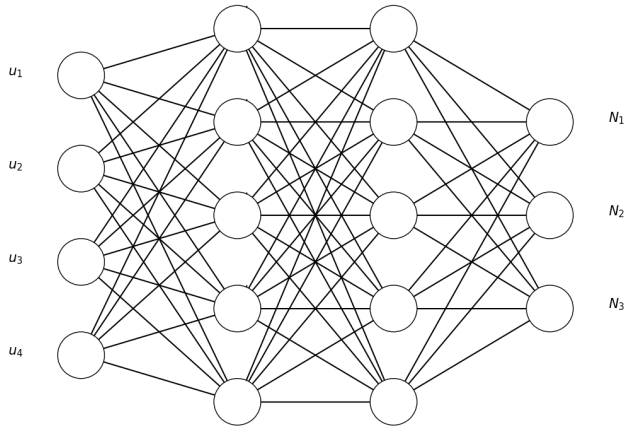
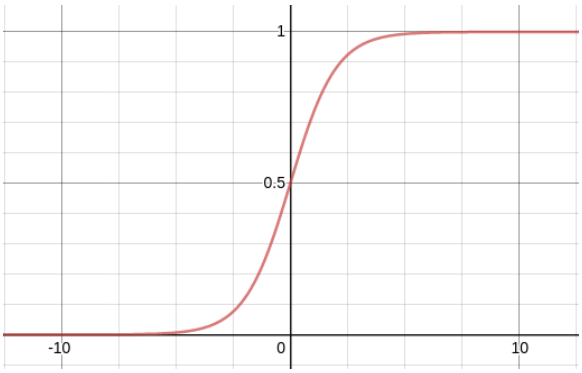


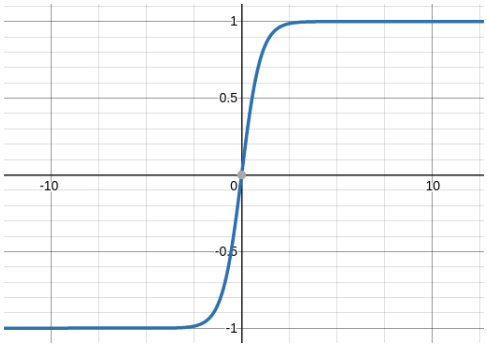
Figure 1.1: Schematic of a Deep Neural Network with input $U = [u_1, u_2, u_3, u_4]$ and output $\mathscr{N}(U) = [\mathscr{N}_1, \mathscr{N}_2, \mathscr{N}_3]$

A major task in training a network is hyperparameters tuning, that is, determining the suitable number of layers, the number of neurons per layer needed, the learning rate, the choice of activation function, and an appropriate optimizer for the loss function [5]. FNN has been used to learn approximate solutions to differential equations. In [12], FNN was combined with the traditional Cox model for survival analysis to predict the clinical outcome of COVID-19 patients. In [9], FNN was used to develop differential equation solvers and parameter estimators by constraining the residual. This FNN is called the Physics Informed Neural Network (PINN).

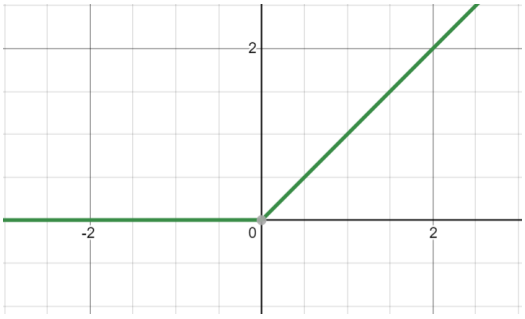
In the early 1990s, it was known that neural networks are universal approximators of continuous functions [13]. The earliest universal approximation theorems for deep neural networks that guarantee the approximation of any continuous function albeit under some conditions was presented in [14, 15]. In recent years, we know of more robust approximation theorems for deep neural networks [16, 17].



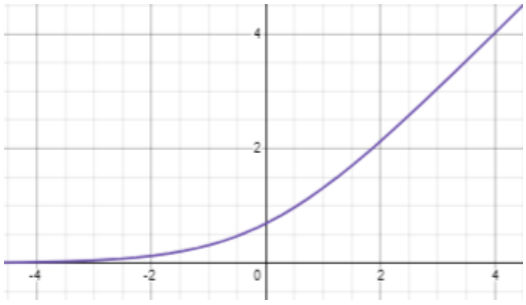
(a) Sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$



(b) Hyperbolic tangent function, $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$



(c) Rectified linear unit (ReLU) function, $\max\{0, x\}$



(d) Softplus function, $\ln(1 + e^x)$

Figure 1.2: Some activation functions

1.2 Deep Neural Network Architectures

If you want to Solve any task by a deep neural network, you will start by considering which of the many neural network architectures best suit your task. Deep learning [10] and deep neural networks have found applications in function approximation tasks, since neural networks are known to be universal approximators of continuous functions [13, 14]. In this dissertation, we have used deep neural networks in the construction of differential equation solvers or in the discovery of differential equations from data. In this section, we discuss some of these deep neural network architectures.

1.2.1 Residual Neural Network (ResNet)

In [18], it is demonstrated that increasing network depth is beneficial [19]. However, merely adding more layers to a deep network leads to higher training error [20]. A major challenge is the vanishing and exploding gradients which hampers convergence. The problem of vanishing and exploding gradients has been largely addressed by normalized initialization [21] and intermediate normalization layers. These techniques work because of the backpropagation algorithm [11].

ResNet is probably the most popular neural network in the computational mathematics community due to its resemblance to a numerical scheme, in particular, the forward Euler scheme (1.3) [18]. This observation was first made in [22] and later by [23, 24, 25]

$$Y_{j+1} = Y_j + hf(Y_j, \theta_j), \tag{1.3}$$

Where Y_{j+1} represents the output features of the $j-th$ layer, θ_j are the parameters of the neural network and f is a nonlinear function.

In [26], the authors transformed (1.3) into an ODE (1.4). They called this neural network architecture an ODENet.

$$\frac{dY(t)}{dt} = f(Y(t), \theta(t)). \tag{1.4}$$

So that starting from the input layer $Y(0)$, we can define the output layer $Y(T)$ to be the solution to (1.4). One challenge with this new neural network is how to backpropagate. The way the authors fixed this is through the adjoint sensitivity method [27]. In their paper [26] they demonstrated that ODENet performs like a continuous solution of an ODE while ResNet performs like a discrete approximation to some ODE.

1.2.2 Recurrent Neural Network (RNN)

They are widely used in natural language processing (NLP). When unfolded into a time-layered network, they resemble a feedforward neural network. They suffer from vanishing and exploding gradients during backpropagation. The key limitation in recurrent neural networks as its sequential nature, which makes the recurrent neural network very difficult to parallelize, especially when the sequence becomes very long. This sequential nature of the recurrent neural networks also has memory constraints during training. Even though there have been significantly improved recurrent models [28, 29], the sequential constraints still persist in recurrent models. RNNs have many variants like the echo state network, long short-term memory (LSTM), and gated recurrent unit (GRU). In [30], an algorithm that implements LSTM is presented to solve an epidemiological model and identify weekly and daily time-varying parameters.

1.2.3 Attention-based Network

The sequential nature of the recurrent neural networks has memory constraints during training especially when the sequence becomes very long. In [31], the authors proposed a model called the 'Transformer' as a viable alternative to recurrent architectures. In that, the transformer model relies solely on an attention mechanism to draw global dependencies between input and output. An encoder-decoder is used to build neural sequence transduction [32]. Encoder-decoder models are auto-regressive because they use previously generated states as additional input for generating future states.

The scaled Dot-Product Attention (1.5) consists of the dot product of a matrix Q and

a matrix K. The shape of Q is $[n_q, d_k]$, where n_q denotes the number of queries and d_k denotes the number of dimensions of each query and each key, while the shape of K is $[n_k, d_k]$, where n_k is the number of keys and values. QK^T is divided by some scaling factor $\sqrt{d_k}$ and then a softmax activation is applied followed by the dot product with a matrix V with the shape $[n_k, d_v]$, where d_v is the number of dimensions of each value [33, 31].

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V. \tag{1.5}$$

Since there are no recurrence or convolution in the 'Transformer', additional information is injected into the model by adding 'positional encodings' to the input embeddings at the bottoms of the encoder and decoder. These 'positional encodings' may allow the model to extrapolate to sequence lengths longer than the ones used in training the model [31].

1.3 Deep Neural Networks solvers

In the past five years, there have been several deep learning frameworks geared at solving Partial Differential Equations (PDEs) and Ordinary Differential Equations (ODEs). In [34], a deep learning algorithm is presented for simulating a noisy dynamical system using an LSTM based network. In [35], the authors introduced a deep learning algorithm, where the goal is to solve high-dimensional PDEs such as the Hamilton-Jacobi-Bellman equation and the Black-Scholes equations. In their approach, as the number of parameters of the neural network increases, the neural network minimizer approaches the ground truth.

One of the earliest implementations of a deep neural network for solving differential equations was presented in [36]. Their approach can be summarized as follows. Suppose we seek a solution to a partial differential equation (PDE) of the form below (1.6),

$$G(x, u(x), \nabla u(x), \nabla^2 u(x)) = 0 \quad for\ all \quad x \in D \subset \mathbb{R}^d, \quad d \in \mathbb{N}. \tag{1.6}$$

And for a subset of the domain D, $(x_i)_{i \in I} \subset D$, where I is an index set, we define u_{NN} to be a neural network satisfying a system of equation (1.7)

$$G(x_i, u_{NN}(x_i;\theta), \nabla u_{NN}(x_i;\theta), \nabla^2 u_{NN}(x_i;\theta)) = 0 \quad for\ all \quad i \in I. \tag{1.7}$$

Using some collocation points, the PDE can be satisfied on a subset of the domain D. θ corresponds to the weights and biases of this deep neural network.

1.3.1 Sparse Regression for Nonlinear dynamics

Data-driven approaches including neural networks and nonlinear regression have been used to learn nonlinear dynamics from data [6, 7, 37, 38, 39]. In [6], the authors developed a sparse regression algorithm for the identification of nonlinear dynamics in a dynamical system. In [7], the authors developed machine learning algorithms for the discovery of governing equations from data. The approach in [6, 7] is a blend of linear algebra and sparse regression. The sparse regression approach in [7] is called PDE-FIND and it can be described as follows, suppose we want to discover the governing equation for a discretized dataset which have been assumed to be the solution of a PDE of the form (1.8)

$$u_t = N(u, u_x, u_{xx}, \ldots, x, t, \lambda) \tag{1.8}$$

Here, λ denotes the parameters in the system. PDE-FIND is a dictionary-based sparse regression. It has the following form (1.9)

$$\Theta(\mathbf{U}, \mathbf{Q}) = [1 \quad \mathbf{U} \quad \mathbf{U}^2 \quad \ldots \quad \mathbf{Q} \quad \ldots \quad \mathbf{U}_x \quad \mathbf{U}\mathbf{U}_x \quad \ldots \quad \mathbf{Q}^2\mathbf{U}^3\mathbf{U}_{xxx}] \tag{1.9}$$

Where $\mathbf{U}$ is the matrix containing u and $\mathbf{Q}$ denotes the matrix containing additional information on u such as a time-varying function interacting with u. So for a problem of the form (1.8), we have the linear system (1.10), where Θ contains all values of candidate functions.

$$\mathbf{U}_t = \Theta(\mathbf{U}, \mathbf{Q})\xi \tag{1.10}$$

If it is assumed that Θ is a sufficiently rich library such that it can contain the dynamics

in (1.8), so that (1.9) becomes a representation of (1.8). The fit variable ξ in (1.9) is a sparse vector that is used to pick the candidate functions in Θ that may be used to write the PDE in (1.8). Using an algorithm called Sequential Threshold Ridge regression (STRidge) [7], a sparse approximation to ξ is obtained (1.11)

$$\hat{\xi} = \arg\min_{\xi} ||\Theta(\mathbf{U}, \mathbf{Q})\xi - \mathbf{U}_t||_2^2 + \tau||\xi||_0 \tag{1.11}$$

The performance of STRidge depends on how robust Θ is. That is, we need to have some prior knowledge about the PDE which can be incorporated into the construction of Θ. This sparse regression and linear dictionary framework of PDE-FIND [7] and its dynamical system counterpart introduced in [6] called sparse identification of nonlinear dynamics (SINDy) can be challenging in practice especially when less prior knowledge is known about the differential system represented by the dataset.

1.3.2 Physics-informed Neural Network (PINN)

One of the most successful data-driven deep neural network in the last few years is the physics-informed neural network (PINN) introduced in [9], where the form of the differential system is assumed to be known and the task is either to learn the parameters of the differential system from data (inverse problem) or PINN is used as a differential system solver (forward problem). To see how PINN works, let us consider a nonlinear partial differential equation of the general form (1.12)

$$\begin{aligned} \frac{\partial u(x,t)}{\partial t} + \mathscr{N}[u(x,t)] &= f(x,t) \qquad x \in \Omega, t > 0 \\ u(x,0) &= h(x), \qquad x \in \Omega \\ u(x,t) &= g(x,t), \qquad x \in \partial\Omega, t > 0 \end{aligned} \tag{1.12}$$

where $\mathscr{N}[\cdot]$ is a differential operator. Now suppose f and g are known at some sample points, that is we do not know f and g entirely on the domain Ω and the boundary $\partial\Omega$ respectively. PINN seeks to find a function u_{NN} satisfying (1.13), here we represent the

weights and biases of PINN by λ.

$$\begin{aligned} \frac{\partial u_{NN}(x,t)}{\partial t} + \mathcal{N}[u_{NN}(x,t;\lambda)] = f(x,t) \qquad & x \in \Omega, t > 0 \\ u_{NN}(x,0) = h(x), \qquad & x \in \Omega \\ u_{NN}(x,t) = g(x,t), \qquad & x \in \partial\Omega, t > 0 \end{aligned} \tag{1.13}$$

PINN solves (1.13). We demonstrate this in Figure (1.3). In [40], the authors consider when (1.12) is a second-order linear parabolic equations and under certain assumptions, they show that PINN also solves (1.12).

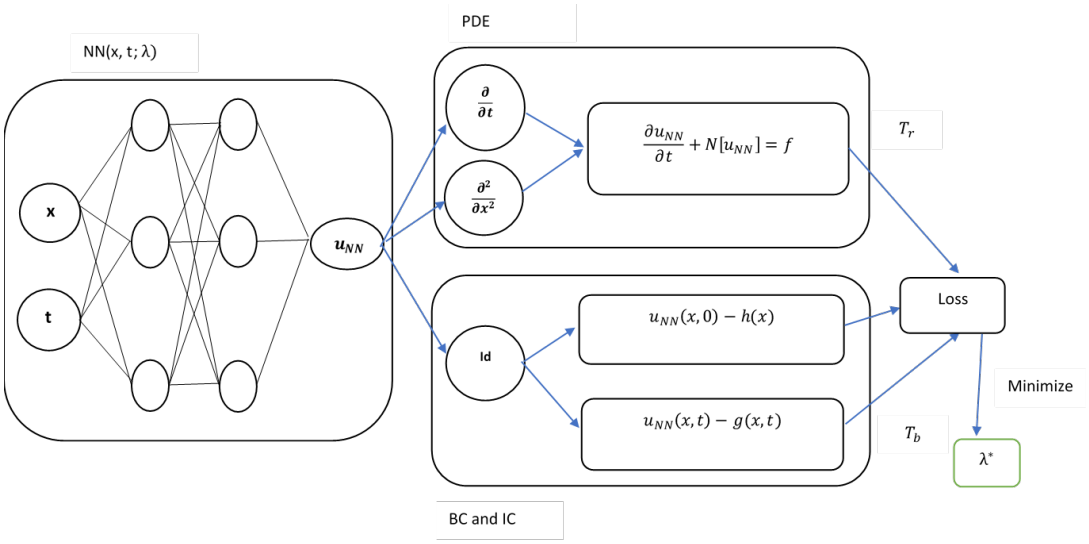


Figure 1.3: Schematic of PINN for solving the general form PDE (1.12)

The schematic of PINN (1.3) can be described in the following steps:

1. Construct the neural network $u_{NN}(x,t;\lambda)$, where $\lambda = \{w^l, b^l\}_{1 \le l \le L}$ is the set of all weights matrices and bias vectors in the neural network. The input to this neural network is x and t and the output of the neural network is the approximate solution of (1.13).

2. In order to enforce the physics of the problem and make u_{NN} satisfy the boundary and initial conditions of the PDE, we restrict u_{NN} to some scattered points in the

domain and on the boundary. These are the training points $\mathscr{T} = \mathscr{T}_r \cup \mathscr{T}_b$ and $\mathscr{T}_r \in \Omega$, $\mathscr{T}_b \in \partial\Omega$.

3. To measure the accuracy of our training, we compute the loss function defined as a weighted summation of the $\mathscr{L}^2$ norm of residuals for the PDE and the boundary and initial conditions:

$$\mathscr{L}(u_{NN}(x,t;\lambda);w,\mathscr{T}) = w_r\mathscr{L}_r(u_{NN}(x,t;\lambda);\mathscr{T}_r) + w_b\mathscr{L}_b(u_{NN}(x,t;\lambda);\mathscr{T}_b) \quad (1.14)$$

where $w = \{w_r, w_b\}$, and $\mathscr{L}_r(u_{NN}(x,t;\lambda);\mathscr{T}_r)$ and $\mathscr{L}_b(u_{NN}(x,t;\lambda);\mathscr{T}_b)$ are defined in (1.15)

$$\begin{aligned}
\mathscr{L}_r(u_{NN}(x,t;\lambda);\mathscr{T}_r) &= \frac{1}{|\mathscr{T}_r|}\sum_{(x,t)\in\mathscr{T}_r}\left\|\frac{\partial u_{NN}(x,t)}{\partial t} + \mathscr{N}[u_{NN}(x,t)] - f(x,t)\right\|_2^2 \\
\mathscr{L}_b(u_{NN}(x,t;\lambda);\mathscr{T}_b) &= \frac{1}{|\mathscr{T}_b|}\sum_{(x,t)\in\mathscr{T}_b}\left\{\left\|u_{NN}(x,0) - h(x)\right\|_2^2 + \left\|u_{NN}(x,t) - g(x,t)\right\|_2^2\right\}
\end{aligned} \quad (1.15)$$

Regularization term can be added to (1.14) to obtain (1.16)

$$\mathscr{L}(u_{NN};w,w^R,\mathscr{T}) = w_r\mathscr{L}_r(u_{NN};\mathscr{T}_r) + w_b\mathscr{L}_b(u_{NN};\mathscr{T}_b) + w_r^R R_r(u_{NN}) + w_b^R R_b(u_{NN}) \quad (1.16)$$

where the regularization weights are $w^R = (w_r^R, w_b^R)$ and $R_r(\cdot)$, $R_b(\cdot)$ are regularization functionals.

4. The last step is a search for 'good' λ^* by minimizing the loss function $\mathscr{L}(u_{NN}(x,t;\lambda);\mathscr{T})$. We minimize the loss function by gradient-based optimizers, such as gradient descent, Adam optimizer [41], and L-BFGS [42].

In the cases when we only know f and g at a few points in the domain Ω and the boundary $\partial\Omega$ respectively in (1.12), then PINN is one order more accurate than a finite difference

method (FDM). This is because an FDM will require interpolation to fill the missing information about f but since PINN is mesh-free, we only need some scattered points in the domain and boundary of the PDE. We can get a good approximation with PINN even with small information about f and g. In the case of an inverse problem, that is, solving for some parameters such as the diffusivity term (constant or a nonlinear function) in (1.12), FDM we will have to solve the forward problem many times but in PINN, we only add the parameter to the loss function. We can also solve nonlinear PDEs without linearization of the nonlinear term or use time-stepping schemes. This is because PINN uses automatic differentiation in its loss function. In this dissertation, we present a modification of PINN that can learn time-dependent parameters in systems of differential equations from data. PINN has been used to simulate pandemic spread, see [43], where the epidemiology parameters were assumed to be constants [9, 44]. PINN has found application in solving nonlinear partial differential equations from data [39]. PINN has also been used to solve systems of ordinary differential equations [45] as well as systems of fractional differential equations [46].

2

LEARNING AN EPIDEMIOLOGICAL MODEL FROM DATA

A good method, like a good spouse, is reliable, stable, efficient.

A. Q. M. Khaliq

It has been said that "all models are wrong but some models are useful."... Nevertheless, enormous progress has been made by entertaining such fictions and using them as approximations.

George Box

The earliest infectious disease model is the SIR model [47], it is composed of interactions between compartments: Susceptible, Infectious and Removed. There is a transmission rate that determines flow from the Susceptible compartment to the Infectious compartment. Most mathematical models of infectious diseases take this transmission rate to be constant. More recently, there are mathematical models of infectious diseases that take the transmission rate to be an explicitly defined function [48, 49, 50].

In December 2019, an infectious disease began to spread throughout Wuhan, China. The virus is the SARS-CoV-2 and the disease COVID-19. However, given that the data we have on COVID-19 from different countries and different regions reflects transmission patterns that are unique to each region due to the various levels of public health interventions such as lockdown, social distancing, early detection of infectives, contact tracing, and vaccination, and the public response that follows such measures [49, 51, 52, 50]. We saw governments of different Countries initiate lock-down and lift lock-down restrictions at dif-

ferent stages of the pandemic and some countries did not institute any lock-down [53]. One of the widely debated issues in the early days of the COVID-19 pandemic is the impact of the widespread adoption of facial masks on transmission.

The SIR model has inspired several epidemiological studies of diseases like Malaria and Dengue fever [54] and recently COVID-19. A widely used threshold parameter for the spread or extinction of an infectious disease in an epidemiology model is the basic reproduction number [55]. It is defined as the average number of persons an infected person can infect. When the basic reproduction number is less than one, the infectious disease vanishes. In the SIR model [47], the basic reproduction number is computed as the ratio of the transmission rate to the recovery rate.

2.1 Asymptomatic-SIR Model

The asymptomatic-SIR model introduced in [56] assumes that some of the infectives are asymptomatic infectives. This group is infectious despite not showing symptoms of COVID-19, probably are not tested, and are usually unreported in the various publicly available data.

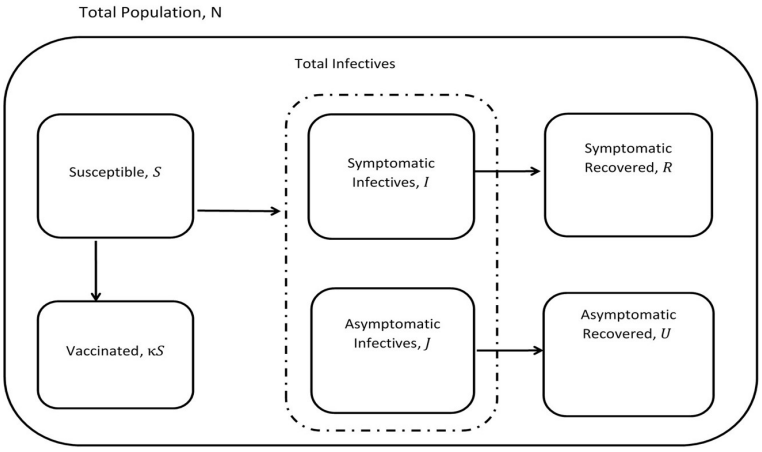


Figure 2.1: Compartments in Asymptomatic-SIR model with vaccination

The asymptomatic-SIR model considers the following population compartments: the Susceptible (S), the symptomatic Infectives (I) which correspond to the reported infectives

in the publicly available data, and the asymptomatic Infectives (J) which correspond to the unreported infectives. The total infectives are $I+J$. The rest of the compartments are the symptomatic Recovered (R) and the asymptomatic Recovered (U). The symptomatic Infectives (I) recover at the rate γ, and the asymptomatic Infectives (J) recover at the rate μ. I recover through isolation in the hospital or at home. On the other hand, J recovers spontaneously. The vaccinated population, $(V=\kappa S)$, is a loss from the susceptible compartment: they are added to the recovered compartments. $\beta(t)$ is the time-varying transmission rate, it usually depends on the infection vector. In the COVID-19 pandemic, $\beta(t)$ depends also on contacts between individuals. κ is the average percentage of individuals that are vaccinated daily. ξ represents the probability that an infective individual is reported, while $(1-\xi)$ is the probability that an infective is an asymptomatic infective. The portion of the total infectives that are symptomatic and reported corresponds to $\xi(I+J)$. On the other hand, $(1-\xi)(I+J)$ represents the asymptomatic infectives. N represents the total population (2.2). It is assumed that N does not change throughout the pandemic and that infective individuals are immediately infectious. The dynamics of the interactions between the compartments in Figure 2.1 can be represented by the following system of ordinary differential equations with a time-varying transmission rate $\beta(t)$.

$$\begin{aligned}
\frac{dS(t)}{dt} &= -\frac{1}{N}\beta(t)\Big(I(t)+J(t)\Big)S(t)-\kappa S(t)\\
\frac{dI(t)}{dt} &= \frac{1}{N}\beta(t)\xi\Big(I(t)+J(t)\Big)S(t)-\gamma I(t)\\
\frac{dJ(t)}{dt} &= \frac{1}{N}\beta(t)\Big(1-\xi\Big)\Big(I(t)+J(t)\Big)S(t)-\mu J(t)\\
\frac{dR(t)}{dt} &= \gamma I(t)+\kappa\xi S(t)\\
\frac{dU(t)}{dt} &= \mu J(t)+\kappa(1-\xi)S(t).
\end{aligned} \tag{2.1}$$

The continuity equation is given by

$$N(t)=S(t)+I(t)+J(t)+R(t)+U(t), \quad t\geq t_0. \tag{2.2}$$

The initial conditions are denoted by $S(t_0) = S_0$, $I(t_0) = I_0$, $J(t_0) = J_0$, $R(t_0) = R_0$, and $U(t_0) = U_0$, where $t \geq t_0$ represent time in days and t_0 is the start date of the pandemic in the model. The model parameters are summarized in Table 2.1.

Parameter	**Notation**	**Range**	**Remark**	**Reference**
Baseline transmission rate	β_0	[0,1)	fitted using early data	[57, 58]
Probability that an Infected person is reported	ξ	$[0,1)$	constant	[56]
Proportions of daily vaccinated individuals	κ	$[0,1)$	constant	[57, 52]
recovery rate of symptomatic infectives	γ	[0,1)	constant	[56]
recovery rate of asymptomatic infectives	μ	[0,1)	constant	[56]

Table 2.1: Summary table of parameters in (2.1)

There is an asymptomatic period for every infective individual in the range of 7 to 14 days [57]. There are also asymptomatic infectives that never show symptoms but are infectious [56]. Early studies of the spread of COVID-19 show that some of the infectives are asymptomatic infectives [59, 60] and they are mostly unreported in the publicly available data [56]. In [61], it was reported that the asymptomatic infectives can spread the virus efficiently, and they are the silent spreaders of COVID-19, which has caused difficulties in the control of the pandemic. Early in the pandemic, the Centers for Disease Control and Prevention (CDC) estimates the proportion of the asymptomatic infectives to be 40% of the total infectives in the USA [60]. A high proportion of asymptomatic infectives was estimated in [59] for China and Singapore. In [61], the proportion of Asymptomatic infectives in Wanzhou district before 10 April 2020 was 20%. In [56], the author reported that 10% of the total infectives were asymptomatic in northern Italy. In a study conducted in England from June through September 2020 and in Spain from 27 April to 11 May 2020, the proportions of asymptomatic infectives in England and Spain were reported to be 32.4% and 33.0% respectively [62].

To overcome the limitations of statistical approaches, we present an Epidemiology-

Informed Neural Network (EINN) inspired by applying a PINN to epidemiology models. Given that it may not be possible to know the most accurate form of a time-varying transmission rate, the EINN algorithm is a viable option to learn the time-varying transmission rate and detect the impact of mitigation measures from data. The EINN loss function is extended to include some known epidemiology facts about infectious diseases. To detect hidden details in the training data, cubic spline interpolation is used to generate sufficient training data. EINN algorithm can capture the dynamics of the spread of the disease and the influence of various mitigation measures. Since asymptomatic infectives population is unreported in the publicly available data [63]. EINN algorithm learns asymptomatic infectives population by training on symptomatic infectives data that are available in the reported public data.

2.2 Epidemiology-informed Neural Network (EINN) Algorithm

We present the EINN Algorithm 1 for the asymptomatic-SIR model with constant parameters. That is, in Equation (2.1), we set $\beta(t) = \beta$. The learned cumulative infectives and the recovered solution is matched against the cumulative infectives and recovered data. In this algorithm, the parameters represent average rates. We implement Algorithm 1 using publicly available COVID-19 data [63].

Algorithm 1 EINN algorithm for Asymptomatic-SIR model with constant parameters

1: Construct EINN

specify the input: t_j, $j = 1,\ldots,M$

Initialize EINN parameter: θ

Initialize the epidemiology and vaccination parameters: $\lambda = [\beta, \gamma, \mu, \xi, \kappa]$

Output layer: $S(t_j;\theta;\lambda), I(t_j;\theta;\lambda), J(t_j;\theta;\lambda), R(t_j;\theta;\lambda), U(t_j;\theta;\lambda)$, $j = 1,\ldots,M$.

2: Specify the training set

Training data: using cubic spline, generate $\tilde{I}(t_j)$, $\tilde{R}(t_j)$, $j = 1,\ldots,M$ and $\tilde{V}(t_j)$, $j = 1,\ldots,M_\kappa$. from given dataset.

Initialize the Asymptomatic population: $\tilde{J}(0) = (1-\xi)\tilde{I}(0)/\xi$ and $\tilde{U}(0) = (1-$

$\xi)\tilde{R}(0)/\xi$.

3: Train the neural network

Specify an MSE loss function:

$$\begin{aligned} MSE = & \frac{1}{M}\sum_{j=1}^{M}||I(t_j;\theta;\lambda)-\tilde{I}(t_j)||_2^2+\frac{1}{M}\sum_{j=1}^{M}||R(t_j;\theta;\lambda)-\tilde{R}(t_j)||_2^2 \\ & +\frac{1}{M_\kappa}\sum_{j=1}^{M_\kappa}||\kappa S(t_j;\theta;\lambda)-\tilde{V}(t_j)||_2^2 \\ & +||J(0;\theta;\lambda)-\tilde{J}(0)||_2^2+||U(0;\theta;\lambda)-\tilde{U}(0)||_2^2 \\ & +\frac{1}{M}\sum_{i=1}^{6}\sum_{j=1}^{M}||L_i(t_j;\theta;\lambda)||_2^2. \end{aligned} \tag{2.3}$$

Minimize the MSE loss function: compute $\arg\min_{\{\theta;\lambda\}}(MSE)$ using an optimizer such as the adam optimizer.

4: return EINN solution

$S(t_j;\theta;\lambda), I(t_j;\theta;\lambda), J(t_j;\theta;\lambda), R(t_j;\theta;\lambda), U(t_j;\theta;\lambda),\ j=1,\ldots,M.$

parameters: $\beta,\gamma,\mu,\xi,\kappa$.

It was assumed in [56] that When the transmission rate in (2.1) is constant, that is, $(\beta(t)=\beta)$, the basic reproduction number can be given by the ratio of the transmission rate to a weighted sum of the symptomatic and asymptomatic recovery rates. However, we observed that this under-estimate the basic reproduction number ($\mathscr{R}_0$) for the asymptomatic-SIR model Equation (2.1). Assuming a disease-free equilibrium of (2.1), given by

$$(S^*,I^*,J^*,R^*,U^*)=(S_0,0,0,0,0)$$

Applying the next generation matrix approach [64], the basic reproduction number ($\mathscr{R}_0$) is obtained as the spectral radius of the next generation matrix FV^{-1}, where

$$F=\begin{pmatrix}\beta\xi & \beta\xi\\ \beta(1-\xi) & \beta(1-\xi)\end{pmatrix},\quad V=\begin{pmatrix}\gamma & 0\\ 0 & \mu\end{pmatrix}.$$

so that

$$\mathscr{R}_0 = \frac{\beta\left(\xi\mu + (1-\xi)\gamma\right)}{\mu\gamma} \quad \xi \in (0,1). \tag{2.4}$$

If $\xi = 0$, $\mathscr{R}_0 = \beta/\mu$, when all the infective population are asymptomatic.

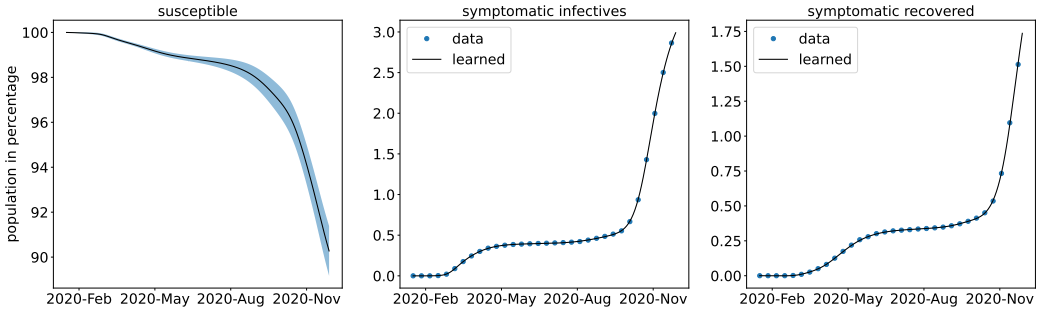
If $\xi = 1$, $\mathscr{R}_0 = \beta/\gamma$, when all the infective population are symptomatic.

Using data from Italy, South Korea, and the United States starting from the date of the first reported cases in the respective countries to the day before vaccination data were reported. The cumulative infective and recovered population data are observed to be non-exponential whenever a mitigation measure such as a comprehensive lockdown is detected in the data. We take the total population N to be 60.36×10^6, 51.64×10^6, and 328.2×10^6 in Italy, South Korea and the USA, respectively. In Figures 2.2a–2.4a, M_κ is zero and so $\kappa = 0$ for all the period from the first reported cases to the day before vaccination data are reported. In addition to learning the parameters, EINN learns ξ, the probability that an infective is reported. A high value of ξ indicates a large number of reported infectives.

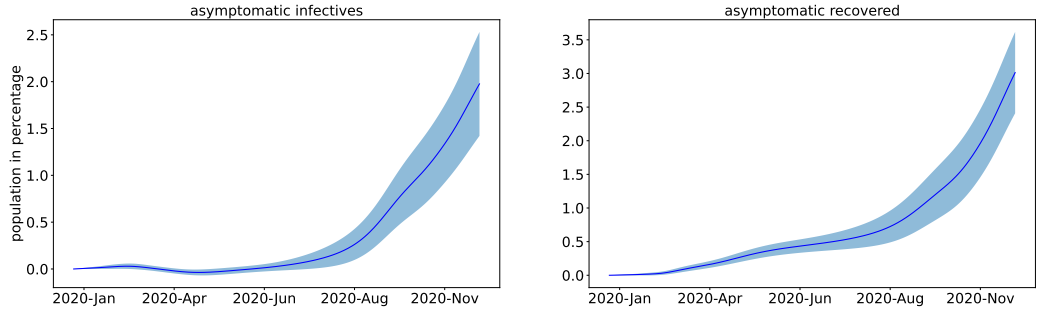
Parameters	Mean	Std
β	0.03773	0.00276
ξ	0.55699	0.07896
γ	0.01327	0.00027
μ	0.02906	0.017478
$\mathscr{R}_0$	2.32770	0.06014

Table 2.2: In Figure (2.2), EINN Algorithm 1 learns the constant model parameters β γ, μ, ξ, and $\mathscr{R}_0$ from 31 January 2020 to 11 December 2020

As shown in Figures (2.2)a–(2.4)a, early in the pandemic, the cumulative infective and recovered data closely resemble an exponential function. Cubic Spline interpolation is used to generate 3000 training points from the cumulative symptomatic infective and recovered data. In Tables (2.2)–(2.4) the mean and standard deviation of the the parameters β γ, μ, ξ, and $\mathscr{R}_0$ are presented after 10 runs of EINN Algorithm (1)



(a)

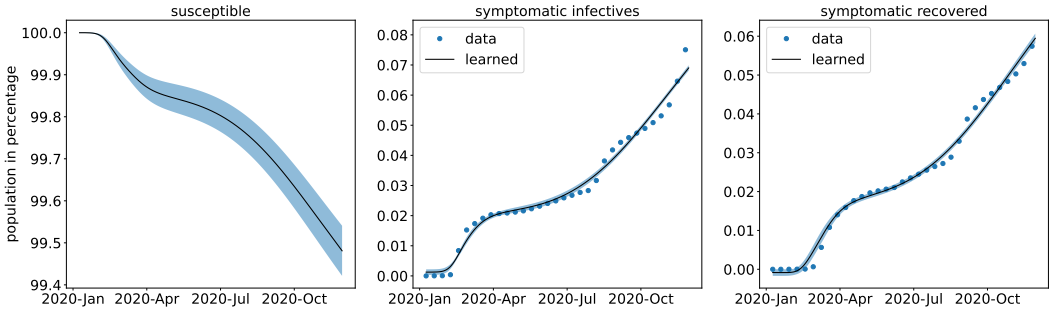


(b)

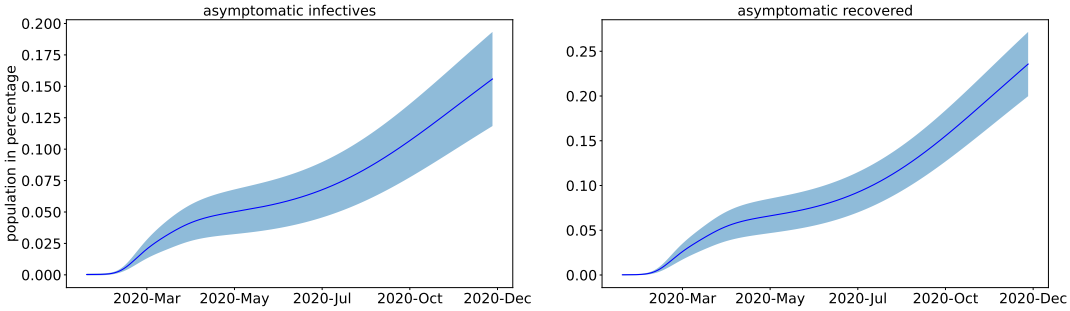
Figure 2.2: Simulation of Italy COVID-19 data ; (**a**) The learned symptomatic infectives and recovered population by the EINN Algorithm 1; (**b**) EINN Algorithm 1 learns the cumulative population of Italy that are asymptomatic infectives and asymptomatic recovered from 31 January 2020 to 11 December 2020

Parameters	Mean	Std
β	0.01537	0.00350
ξ	0.24862	0.04333
γ	0.00537	0.00013
μ	0.01174	0.00587
$\mathscr{R}_0$	1.84796	0.16187

Table 2.3: In Figure (2.3), EINN Algorithm 1 learns the constant model parameters β γ, μ, ξ, and $\mathscr{R}_0$ from 22 January 2020 to 11 December 2020



(a)

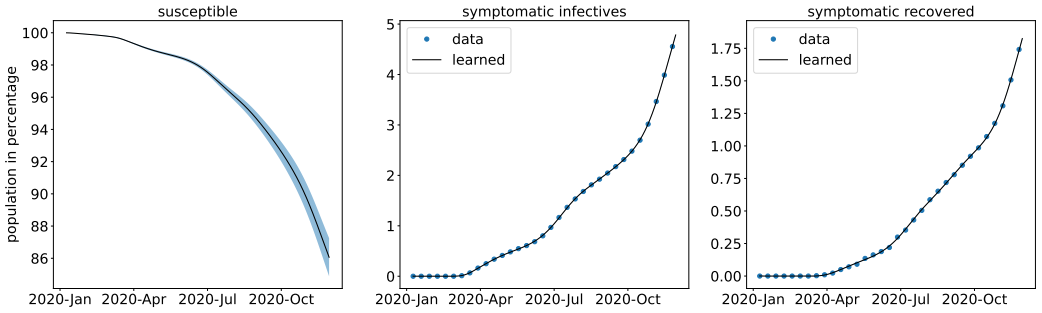


(b)

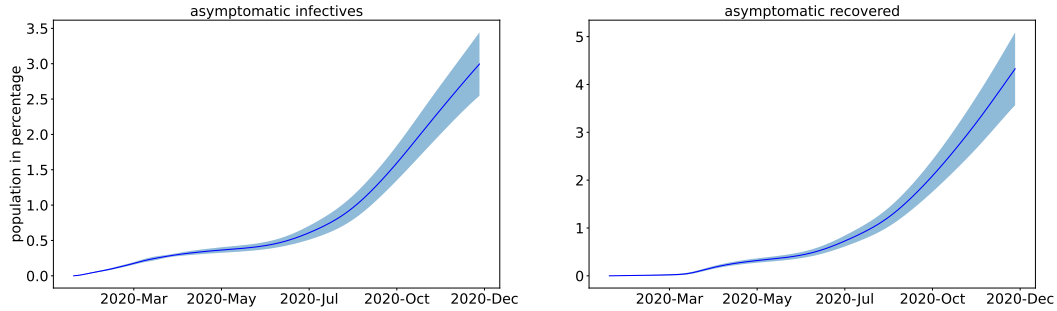
Figure 2.3: Simulation of South Korea COVID-19 data; (**a**) The learned symptomatic infectives and recovered population were obtained by the EINN Algorithm 1; (**b**) EINN Algorithm 1 learns the cumulative population of South Korea that are asymptomatic infectives and asymptomatic recovered from 22 January 2020 to 11 December 2020

Parameters	Mean	Std
β	0.02130	0.00144
ξ	0.49176	0.06541
γ	0.00437	0.000046
μ	0.01499	0.00199
$\mathscr{R}_0$	3.10406	0.09609

Table 2.4: In Figure (2.4), EINN Algorithm 1 learns the constant model parameters β γ, μ, ξ, and $\mathscr{R}_0$ from 22 January 2020 to 11 December 2020



(a)



(b)

Figure 2.4: Simulation of USA COVID-19 data; (**a**) The learned symptomatic infectives and recovered population were obtained by the EINN Algorithm 1; (**b**) EINN Algorithm 1 learns the cumulative population of USA that are asymptomatic infectives and asymptomatic recovered from 22 January 2020 to 11 December 2020.

2.2.1 Data-Driven Simulation for Non-Pharmaceutical Mitigation Measures

The model parameters in an epidemiology model are influenced by mitigation measures. For instance, social distancing corresponds to reducing the transmission rate by reducing human contact. In this section, we simulate different levels of various non-pharmaceutical mitigation measures, and we demonstrate their impact on $\mathscr{R}_0$ and the spread of COVID-19. The epidemiological meaning of each of the model parameters in Equation (2.1) including ξ are presented in Sections 2.2.1.1–2.2.1.3.

2.2.1.1 Early Detection of Infectives

Early detection of infectives population leads to higher reported infectives. This results in an early isolation of individuals who have had contact with infective individuals. There are no reported data for the asymptomatic infectives populations. Simulating with higher ξ increases the symptomatic infectives population. This corresponds to higher reported cases. Simulations are presented for Italy, South Korea, and the USA see Tables 2.5–2.7.

		β	γ	μ	$\beta\xi$	$\beta(1-\xi)$	$\mathscr{R}_0$
$\xi = \mathbf{0.1}$	**Mean**	0.03161	0.00119	0.03125	0.00316	0.02845	4.32459
	Std	0.00376	0.00047	0.02510	0.00038	0.00338	0.96941
$\xi = \mathbf{0.25}$	**Mean**	0.03827	0.00810	0.02418	0.00957	0.02870	2.42050
	Std	0.00307	0.00122	0.00456	0.00077	0.00230	0.08582
$\xi = \mathbf{0.50}$	**Mean**	0.03698	0.01208	0.03253	0.01849	0.01849	2.33102
	Std	0.00304	0.00152	0.02876	0.00152	0.00152	0.11068
$\xi = \mathbf{0.75}$	**Mean**	0.03700	0.01435	0.03027	0.02775	0.00925	2.35074
	Std	0.00262	0.00122	0.01801	0.00196	0.00065	0.09155

Table 2.5: The learned parameters using EINN Algorithm 1 with fixed values of ξ based on Italy data from 31 January 2020 to 5 September 2020.

Higher ξ values in Tables 2.5–2.7, increase the symptomatic infectives population and reduce the asymptomatic population in general. This is reflected by the increase in the $\beta\xi$ column and the corresponding decrease in the $\beta(1-\xi)$ column. This means that more

		β	γ	μ	$\beta\xi$	$\beta(1-\xi)$	$\mathscr{R}_0$
$\xi = \mathbf{0.1}$	**Mean**	0.01230	0.00179	0.00958	0.00123	0.01107	1.95802
	Std	0.00172	0.00041	0.00304	0.00017	0.00155	0.15387
$\xi = \mathbf{0.25}$	**Mean**	0.01326	0.00615	0.00792	0.00332	0.00995	1.83806
	Std	0.00101	0.00118	0.00148	0.00025	0.00076	0.11778
$\xi = \mathbf{0.50}$	**Mean**	0.01499	0.01222	0.00754	0.00749	0.00749	1.73132
	Std	0.00217	0.00156	0.00269	0.00109	0.00109	0.22998
$\xi = \mathbf{0.75}$	**Mean**	0.01195	0.01537	0.00318	0.00896	0.00299	1.64407
	Std	0.00186	0.00226	0.00224	0.00139	0.00047	0.28103

Table 2.6: The learned parameters using EINN Algorithm 1 with fixed values of ξ based on South Korea data from 22 January 2020 to 5 September 2020.

		β	γ	μ	$\beta\xi$	$\beta(1-\xi)$	$\mathscr{R}_0$
$\xi = \mathbf{0.25}$	**Mean**	0.02270	0.00224	0.01471	0.00568	0.01703	3.83612
	Std	0.00143	0.00056	0.00119	0.00036	0.00108	0.53227
$\xi = \mathbf{0.50}$	**Mean**	0.02126	0.00419	0.01639	0.01063	0.01063	3.20680
	Std	0.00071	0.00032	0.00239	0.00036	0.00036	0.13379
$\xi = \mathbf{0.75}$	**Mean**	0.02009	0.00514	0.02039	0.01507	0.00502	3.18964
	Std	0.00083	0.00026	0.00465	0.00062	0.00021	0.09912

Table 2.7: The learned parameters using EINN Algorithm 1 with fixed values of ξ based on USA data from 22 January 2020 to 5 September 2020.

people will be in hospitalization/isolation. This translates to more recovery in the symptomatic compartment. We see that the detection of early infectives alone is not enough to mitigate an infectious disease such as COVID-19 as demonstrated in the $\mathscr{R}_0$ column in Tables 2.5–2.7. It should be combined with other measures such as contact tracing of infectives.

2.2.1.2 Social Distancing

It is widely understood that measures such as a lockdown, social distancing, and widespread adoption of facial coverings result in the mitigation of COVID-19. Social distancing is often the most sought-after measure at reducing the $\mathscr{R}_0$. The goal of social distancing is to reduce the average number of human contacts. This is demonstrated by reducing β, the transmission rate [56]. The impact of social distancing on the $\mathscr{R}_0$ is presented in the following Tables 2.8–2.10.

		γ	ξ	μ	$\beta\xi$	$\beta(1-\xi)$	$\mathscr{R}_0$
$\beta = \mathbf{0.1}$	**Mean**	0.01371	0.69461	0.36808	0.06946	0.03054	5.15392
	Std	0.00020	0.04506	0.07684	0.00451	0.00451	0.24303
$\beta = \mathbf{0.05}$	**Mean**	0.01361	0.55675	0.30860	0.02784	0.02216	2.11654
	Std	0.00000	0.00000	0.14114	0.00000	0.00000	0.00000
$\beta = \mathbf{0.025}$	**Mean**	0.01163	0.54429	0.021192	0.01361	0.01139	1.81348
	Std	0.00032	0.09003	0.01164	0.00225	0.00225	0.45399

Table 2.8: The learned parameters using EINN Algorithm 1 with fixed values of β based on Italy data from 31 January 2020 to 5 September 2020

Reducing β in Tables 2.8–2.10 correspond to a reduced symptomatic infectives population I. There is an increase in the asymptomatic infectives population J. Social distancing is effective when the asymptomatic infective population J diminishes. $\beta\xi$ and $\beta(1-\xi)$ both decreases. Social distancing should be combined with contact tracing and early detection of infectives population.

		γ	ξ	μ	$\beta\xi$	$\beta(1-\xi)$	$\mathscr{R}_0$
$\beta = \mathbf{0.05}$	**Mean**	0.00571	0.64985	0.16848	0.03249	0.01751	7.69632
	Std	0.00229	0.18684	0.03398	0.00934	0.00934	4.61209
$\beta = \mathbf{0.025}$	**Mean**	0.00539	0.30717	0.03705	0.00768	0.01732	1.90597
	Std	0.00000	0.00000	0.01178	0.00000	0.00000	0.00000
$\beta = \mathbf{0.01}$	**Mean**	0.00285	0.09468	0.00819	0.00095	0.00905	1.37149
	Std	0.00103	0.07145	0.002224	0.00071	0.00071	0.48262

Table 2.9: The learned parameters using EINN Algorithm 1 with fixed values of β based on South Korea data from 22 January 2020 to 5 September 2020

		γ	ξ	μ	$\beta\xi$	$\beta(1-\xi)$	$\mathscr{R}_0$
$\beta = \mathbf{0.05}$	**Mean**	0.00398	0.32026	0.64670	0.01601	0.03399	4.53989
	Std	0.00115	0.05013	0.17882	0.00251	0.00251	1.61048
$\beta = \mathbf{0.025}$	**Mean**	0.00458	0.85867	0.03475	0.02147	0.00353	3.34950
	Std	0.00048	1.22557	0.01843	0.03064	0.03064	1.52165
$\beta = \mathbf{0.01}$	**Mean**	0.00314	0.59924	0.00532	0.00599	0.00401	3.52616
	Std	0.00009	0.21138	0.00613	0.00211	0.00211	1.52799

Table 2.10: The learned parameters using EINN Algorithm 1 with fixed values of β based on USA data from 22 January 2020 to 5 September 2020

2.2.1.3 Contact Tracing of Infectives

Contact tracing is equivalent to increasing the symptomatic recovery and asymptomatic recovery rates [56]. However, since we do not have reported data for the asymptomatic population, in this paper, we pursue contact tracing as an increase in the symptomatic recovery rate. This is equivalent to reducing the number of days an infective individual stays infective. In Tables 2.11–2.13, the impact of contact tracing is demonstrated by increasing the symptomatic recovery rate.

		β	ξ	μ	$\beta\xi$	$\beta(1-\xi)$	$\mathscr{R}_0$
$\gamma = \mathbf{0.001}$	**Mean**	0.03235	0.37063	0.02157	0.01208	0.02027	13.05386
	Std	0.00251	0.05151	0.00443	0.00245	0.00127	2.29273
$\gamma = \mathbf{0.005}$	**Mean**	0.03386	0.43284	0.02479	0.01484	0.01902	3.86667
	Std	0.00372	0.08534	0.01229	0.00406	0.00258	0.50726
$\gamma = \mathbf{0.01}$	**Mean**	0.03564	0.49312	0.02306	0.01771	0.01793	2.62805
	Std	0.00332	0.07613	0.00791	0.00373	0.00216	0.12344
$\gamma = \mathbf{0.05}$	**Mean**	0.04573	0.85962	0.01113	0.03924	0.00649	1.36939
	Std	0.00223	0.06479	0.00436	0.00270	0.00319	0.09824

Table 2.11: The learned parameters using EINN Algorithm 1 with fixed values of γ based on Italy data from 31 January 2020 to 5 September 2020

		β	ξ	μ	$\beta\xi$	$\beta(1-\xi)$	$\mathscr{R}_0$
$\gamma = \mathbf{0.001}$	**Mean**	0.01274	0.17877	0.00811	0.00225	0.01049	3.59469
	Std	0.00161	0.02628	0.00199	0.00027	0.00153	0.19821
$\gamma = \mathbf{0.005}$	**Mean**	0.01386	0.24278	0.00890	0.00335	0.01051	1.90717
	Std	0.00157	0.02331	0.00261	0.00039	0.00134	0.17414
$\gamma = \mathbf{0.01}$	**Mean**	0.01410	0.27970	0.00841	0.00399	0.01012	1.74857
	Std	0.00239	0.02634	0.00350	0.00097	0.00149	0.30925
$\gamma = \mathbf{0.05}$	**Mean**	0.01804	0.69863	0.00552	0.01269	0.00534	1.31972
	Std	0.00309	0.08219	0.00234	0.00311	0.00137	0.27368

Table 2.12: The learned parameters using EINN Algorithm 1 with fixed values of γ based on South Korea data from 22 January 2020 to 5 September 2020

		β	ξ	μ	$\beta\xi$	$\beta(1-\xi)$	$\mathscr{R}_0$
$\gamma = \mathbf{0.001}$	**Mean**	0.02089	0.38780	0.01533	0.00808	0.01281	8.92979
	Std	0.00091	0.04234	0.00219	0.00073	0.00127	0.60324
$\gamma = \mathbf{0.005}$	**Mean**	0.02121	0.50579	0.01446	0.01071	0.01049	2.88515
	Std	0.00105	0.04284	0.00240	0.00084	0.00120	0.07506
$\gamma = \mathbf{0.01}$	**Mean**	0.02334	0.56126	0.01437	0.01308	0.01026	2.02548
	Std	0.00087	0.03675	0.00156	0.00069	0.00117	0.02210

Table 2.13: The learned parameters using EINN Algorithm 1 with fixed values of γ based on USA data from 22 January 2020 to 5 September 2020

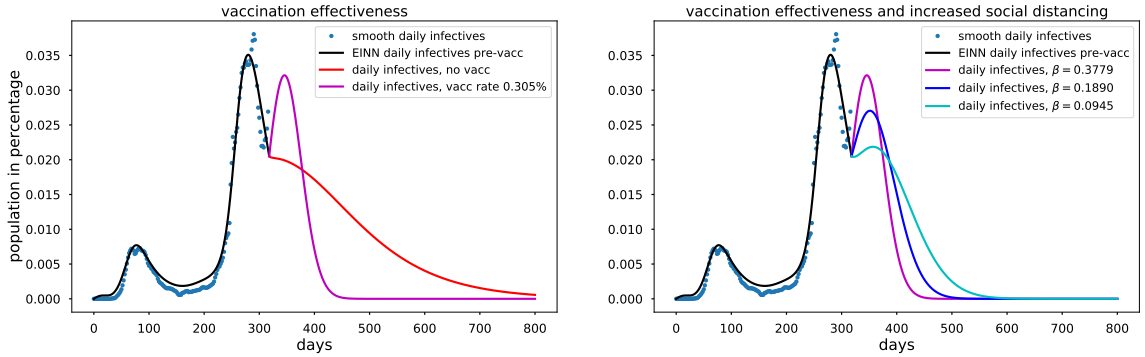
The raising of γ in Tables 2.11–2.13, increases the symptomatic infectives population I which is demonstrated in increased ξ and increased β. $\beta(1-\xi)$ decreases while $\beta\xi$ increases. This also results in a reduced $\mathscr{R}_0$. Contact tracing is an efficient mitigation measure in lowering the spread of COVID-19.

The COVID-19 infectious population surge witnessed in March and April 2020 around the world forced many countries to institute strict lockdown measures. This was largely successful in reducing the $\mathscr{R}_0$ in many countries, unfortunately, it also resulted in economic hardship, such that we seek other measures that also reduce the $\mathscr{R}_0$ to a number less than 1. In recent months, the measures that are promoted in most countries include contact tracing, social distancing, and facial covering.

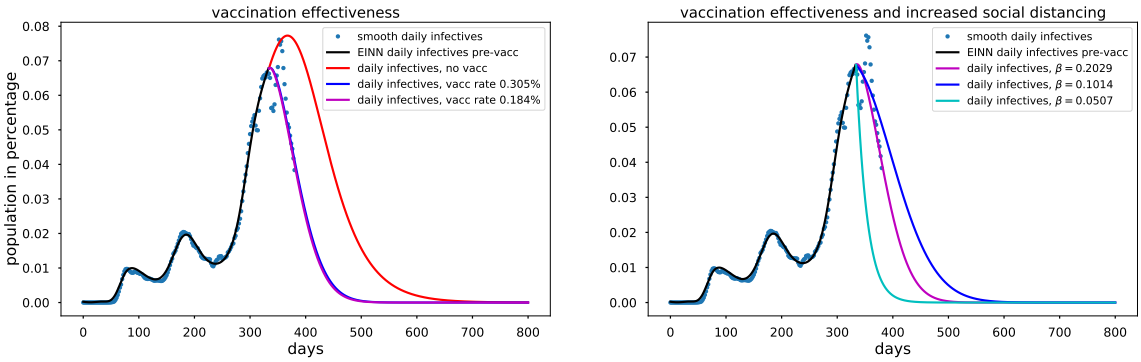
2.2.2 Data-Driven Simulation for Vaccination Efficacy

The mitigation measures described in Section 2.2.1 are non-pharmaceutical measures. In this Section, we discuss vaccination. In the fight against COVID-19, countries such as USA and United Kingdom began to vaccinate in December 2020. A major goal of vaccination is to reduce the susceptible population, i.e., people recover without becoming infected. This constitutes a pharmaceutical mitigation measure. We considered the vaccination data for the USA and the United Kingdom, and simulate the effectiveness of vaccination on the daily reported infectives. A hybrid neural network is used to simulate an

efficient vaccination strategy in [65]. We show that the implementation of Algorithm 1 for the asymptomatic-SIR model (2.1), we can demonstrate the efficacy of vaccination in combination with some mitigation measures. In Figure 2.5 we present a simulation of the effectiveness of vaccination in combination with an increase in social distancing in the USA and in the United Kingdom.



(a) UK



(b) USA

Figure 2.5: Vaccination efficacy

We used the USA projection of $1,000,000$ daily vaccination. In the case of the magenta curve, we learned κ using the daily vaccination data. The first reported case was $01/22/2020$, Vaccination data were first reported on 19 December 2020. In 2.5(**a**) the model is extrapolated for 2 cases. The red curve is the case of no vaccination, here $\kappa = 0$. In the magenta curve, we learned κ using the daily vaccination data. The first reported case was on 31 January 2020, Vaccination data were first reported on 13 December 2020. The effectiveness of vaccination is demonstrated by learning the pre-vaccination and post-

vaccination epidemiology parameters using smooth daily reported infectious data from the USA. In 2.5(**b**) the effectiveness of vaccination is demonstrated by learning the pre-vaccination and post-vaccination epidemiology parameters using smooth daily reported infectives data from the United Kingdom.

In Figure 2.5(**b**), using USA data, the mitigation effect of vaccination on the daily infectives is demonstrated. Implementing Algorithm 1, we obtained $\kappa = 0.00184$, which is slightly different from the projection of κ = 0.00305, corresponding to 1 million people vaccinated per day. In Figure 2.5(**a**), using United Kingdom data, we simulate the impact of vaccination on the daily reported infectives, using a smoothed daily vaccination data from 13 December 2020 to 5 February 2020 and smoothed daily reported infectives data. We implement Algorithm 1 and we obtained $\kappa = 0.00305$. We demonstrate the impact of increased social distancing together with the vaccination effort. Social distancing corresponds to decreasing the transmission rate β. Increased social distancing reduces the daily reported infectives but it extends the number of days daily infectives data is significant.

2.3 Error Metrics for Data-Driven Simulation

The performance of the neural network training is demonstrated in Table 2.14, where the random and shuffle splits [66] has been used to generate the training and testing dataset. The random split performed better than the shuffle split. In Figure 2.6, we present the training and testing MSE at different epochs, depths and widths. We observe that it is more beneficial to increase the width before increasing the depth [67].

Data Split	R_2 **score**	**MSE**	**MAE**	**Max Error**
Random split	9.9994×10^{-1}	3.9365×10^{-4}	1.2440×10^{-2}	6.6720×10^{-2}
Shuffle split	9.2104×10^{-1}	4.4006×10^{-1}	4.9789×10^{-1}	1.3683×10^{0}

Table 2.14: Error metrics for the infected cases (I) using the random and shuffle splits for Italy COVID data, where we use 40% of the dataset for testing.

We have presented a data-driven deep-learning algorithm that discovers transmission

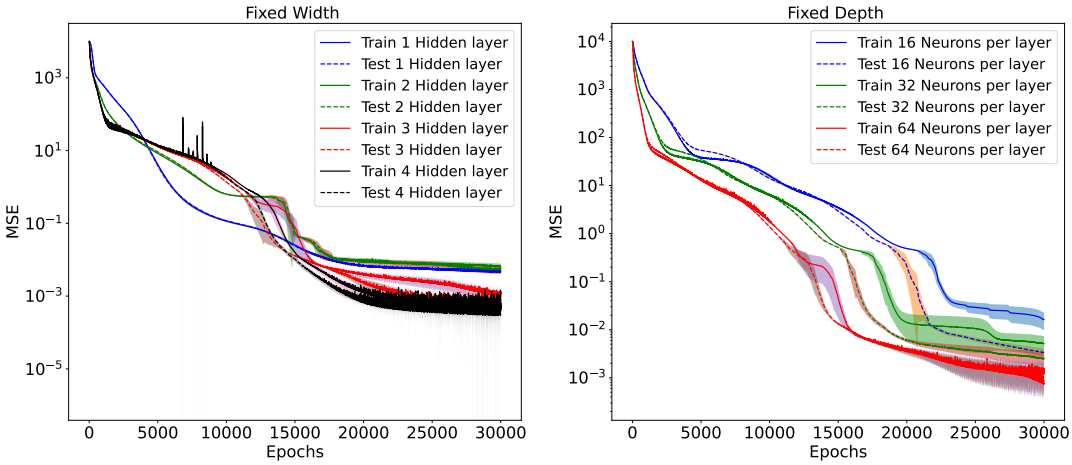


Figure 2.6: Training and testing Errors in EINN for Italy data

rate patterns in an epidemiology model using cumulative and daily reported symptomatic infective and recovered data. The algorithm predicts asymptomatic infectives and asymptomatic recovered populations. The asymptomatic population is usually unreported in the publicly available data. The asymptomatic population is learned from the symptomatic population data. It is demonstrated that a time-varying function models the nonlinear transmission rate. The EINN algorithms presented, learns the nonlinear time-varying transmission rate without a pre-assumed pattern. This approach is useful when the dynamics of an epidemiological model are impacted by various mitigation measures. The algorithm can be adapted to most epidemiology models.

In the proposed model, we have demonstrated the impact of public health actions on the transmission of COVID-19. The effect of pharmaceutical mitigation measures such as vaccination is presented. Non-pharmaceutical mitigation measures such as early detection of symptomatic infectious population, contact tracing, and social distancing are promoted by showing their impact on the spread of COVID-19. This study is useful in the event of a pandemic such as COVID-19, where governmental interventions and public response and perceptions interfere in the interaction of the compartments in an epidemiology model.

3

LEARNING TIME-VARYING TRANSMISSION RATES of EPIDEMIOLOGICAL MODELS

"Nature forms patterns. Some are orderly in space but disorderly in time, others orderly in time but disorderly in space. . . . The dynamics seem so basic—shapes changing in space and time—yet only now are the tools available to understand them."

James Gleick

Time-varying transmission rates have been suggested to efficiently model the spread of COVID-19. For example, fast methods for estimating time-varying transmission rate were introduced in [68]; however, they reported that their method suffers from extreme sensitivity to noise. In [58], a first-principle machine learning approach was presented to predict time-dependent parameters, but these parameters require good initial guesses.

3.1 Time-Varying Transmission Rate

Time-varying transmission rate $\beta(t)$ in (2.1) incorporates the impact of public health actions and the public response to the actions [69, 49]. The formulation of $\beta(t)$ introduced in [69] includes temperature. However, temperature is not considered as a parameter in the formulation of $\beta(t)$ presented in [49], since there is no evidence that temperature plays a role in the transmission of COVID-19. Early in the transmission of COVID-19, the major public health action was lockdown, which was followed by other measures such as social distancing, contact tracing, masking, early detection of infectives, and so on. We chose a formulation of $\beta(t)$ that strongly reflects the pre and post-lockdown periods. In [70] a sigmoid function is used to model a time-dependent decrease in the transmission of COVID-

19. In [56], a piecewise constant function is used to model $\beta(t)$. Our formulation of $\beta(t)$ follows the approach presented in [57]. The following exponentially decreasing function (3.1) is used to represent the transmission rate $\beta(t)$ in (2.1) to model the impact of lockdown.

$$\beta(t) = \begin{cases} \beta_0, & 0 \leq t \leq K, \\ \beta_0 \exp\left(-\eta(t-K)\right), & K < t \end{cases} \tag{3.1}$$

where K denotes the number of days between the date of the first reported case of COVID-19 and lockdown. So K signifies the onset of government intervention including isolation, quarantine, and lockdown. η is the rate at which human contact decreases.

When the transmission rate is time-varying, we use a modified reproduction, which we call the time-varying reproduction $\mathscr{R}_t$. This time-varying reproduction number, $\mathscr{R}_t$, demonstrates the spread pattern of COVID-19 throughout the duration of the pandemic [56].

$$\mathscr{R}_t = \frac{\beta(t)\left(\xi\mu + (1-\xi)\gamma\right)}{\mu\gamma} \quad \xi \in (0,1). \tag{3.2}$$

3.2 EINN for Time-Varying Transmission rates

EINN is a Feedforward Neural Network that includes the known epidemiology dynamics in its loss function. In this chapter, EINN is adapted for the asymptomatic-SIR model (2.1), where the Mean Square Error (MSE) of this neural network's loss function includes the known epidemiology dynamics such as a lockdown, while other mitigation measures such as social distancing, and contact tracing are detected by the time-varying transmission rate. The output of EINN are the learned solutions to the asymptomatic-SIR model (2.1) denoted by $S(t_j;\theta;\lambda)$, $I(t_j;\theta;\lambda)$, $J(t_j;\theta;\lambda)$, $R(t_j;\theta;\lambda)$, $U(t_j;\theta;\lambda)$, $j = 1,\ldots,M$. Where θ represents the neural network weights and biases and λ represents the epidemiology parameters. M is the number of the training set. The network representing the time-varying transmission rate is denoted by $\beta(t_j;\phi;\eta)$, $j = 1,\ldots,M$, The parameter ϕ represents the weights and biases of this network and η is the exponential decay parameter. The training

data are generated using cubic spline and denoted by $\tilde{I}(t_j)$, $\tilde{R}(t_j)$, $j = 1,\ldots,M$ and $\tilde{V}(t_j)$, $j = 1,\ldots,M_\kappa$ from the given dataset. Here M_κ is the number of vaccination days. We observe that training data are not available for all the compartments in the asymptomatic-SIR model; however, EINN can capture the epidemiology interactions between the compartments because the epidemiology model residual is included in the MSE loss function. The MSE loss function for EINN with the time-varying transmission rate is given by

$$\begin{aligned} MSE = & \frac{1}{M}\sum_{j=1}^{M} ||I(t_j;\theta;\lambda) - \tilde{I}(t_j)||_2^2 + \frac{1}{M}\sum_{j=1}^{M} ||R(t_j;\theta;\lambda) - \tilde{R}(t_j)||_2^2 \\ & + \frac{1}{M_\beta}\sum_{j=1}^{M_\beta} ||\beta(t_j;\phi;\eta) - \tilde{\beta}(t_j)||_2^2 \\ & + \frac{1}{M_\kappa}\sum_{j=1}^{M_\kappa} ||\kappa S(t_j;\theta;\lambda) - \tilde{V}(t_j)||_2^2 \\ & + ||J(0;\theta;\lambda) - \tilde{J}(0)||_2^2 + ||U(0;\theta;\lambda) - \tilde{U}(0)||_2^2 \\ & + \frac{1}{M}\sum_{i=1}^{6}\sum_{j=1}^{M} ||L_i(t_j;\theta;\phi;\lambda;\eta)||_2^2, \end{aligned} \tag{3.3}$$

where the residual L_i, $i = 1,\ldots 6$ is as follows

$$
\begin{aligned}
L_1(t_j;\theta;\phi;\lambda;\eta) &= \frac{dS(t_j;\theta;\lambda)}{dt_j} + \frac{1}{N}\beta(t_j;\phi;\eta)\Big(I(t_j;\theta;\lambda)+J(t_j;\theta;\lambda)\Big)S(t_j;\theta;\lambda) \\
&\quad + \kappa S(t_j;\theta;\lambda) \\
L_2(t_j;\theta;\phi;\lambda;\eta) &= \frac{dI(t_j;\theta;\lambda)}{dt_j} - \frac{1}{N}\beta(t_j;\phi;\eta)\xi\Big(I(t_j;\theta;\lambda)+J(t_j;\theta;\lambda)\Big)S(t_j;\theta;\lambda) \\
&\quad + \gamma I(t_j;\theta;\lambda) \\
L_3(t_j;\theta;\phi;\lambda;\eta) &= \frac{dJ(t_j;\theta;\lambda)}{dt_j} - \frac{1}{N}\beta(t_j;\phi;\eta)\Big(1-\xi\Big)\Big(I(t_j;\theta;\lambda)+J(t_j;\theta;\lambda)\Big)S(t_j;\theta;\lambda) \\
&\quad + \mu J(t_j;\theta;\lambda) \\
L_4(t_j;\theta;\phi;\lambda;\eta) &= \frac{dR(t_j;\theta;\lambda)}{dt_j} - \gamma I(t_j;\theta;\lambda) - \kappa\xi S(t_j;\theta;\lambda) \\
L_5(t_j;\theta;\phi;\lambda;\eta) &= \frac{dU(t_j;\theta;\lambda)}{dt_j} - \mu J(t_j;\theta;\lambda) - \kappa(1-\xi)S(t_j;\theta;\lambda) \\
L_6(t_j;\theta;\phi;\lambda;\eta) &= N - (S(t_j;\theta;\lambda)+I(t_j;\theta;\lambda)+J(t_j;\theta;\lambda)+R(t_j;\theta;\lambda)+U(t_j;\theta;\lambda)).
\end{aligned}
\tag{3.4}
$$

In Figure 3.1, EINN includes the time-varying infection as an output of the neural network. *ICs* represents the loss in the neural network output for the asymptomatic infectives $J(0;\theta)$ and the asymptomatic recovered $U(0;\theta)$ at $t=0$. Asymptomatic infectives at $t=0$ is determined according to the formula $\tilde{J}(0) = (1-\xi)\tilde{I}(0)/\xi$. Similarly, Asymptomatic recovered at $t=0$ is determined according to the formula $\tilde{U}(0) = (1-\xi)\tilde{R}(0)/\xi$. *KPs* represent the known dynamics in the transmission rate pattern. M is the number of training points. M does not necessarily correspond to the number of available data. M is generated by fitting the data with cubic splines. For instance, $\tilde{I}(t_j)$, $j=1,\ldots,M$ is the training data for the infectives after fitting with an interpolation function. M_β is the number of training points used to enforce the known dynamics of the transmission rates pattern. Since κ is the average percentage of individuals that are vaccinated daily, M_κ is the number of vaccination days in the model, we can also describe M_κ as the number of days κ is not zero. $\tilde{V}(t_j) = \kappa\tilde{S}(t_j)$, $j=1,\ldots,M_\kappa$, is the daily vaccination data. The input to EINN is t_j, $j=1,\ldots,M$. The output of EINN solves the asymptomatic-SIR model (2.1), because they

are enforced in the residual (3.4).

To achieve good accuracy in the neural network, we tune the hyperparameters; such as the number of layers, number of training points, and the learning rate. In all the simulations presented in this paper, we used 4 hidden layers, 64 neurons per layer, and the training loss was minimized in 40,000 iterations. Cubic splines are used to generate 3000 training points from the original dataset. The loss function is minimized by a gradient-based optimizer such as the adam optimizer [71].

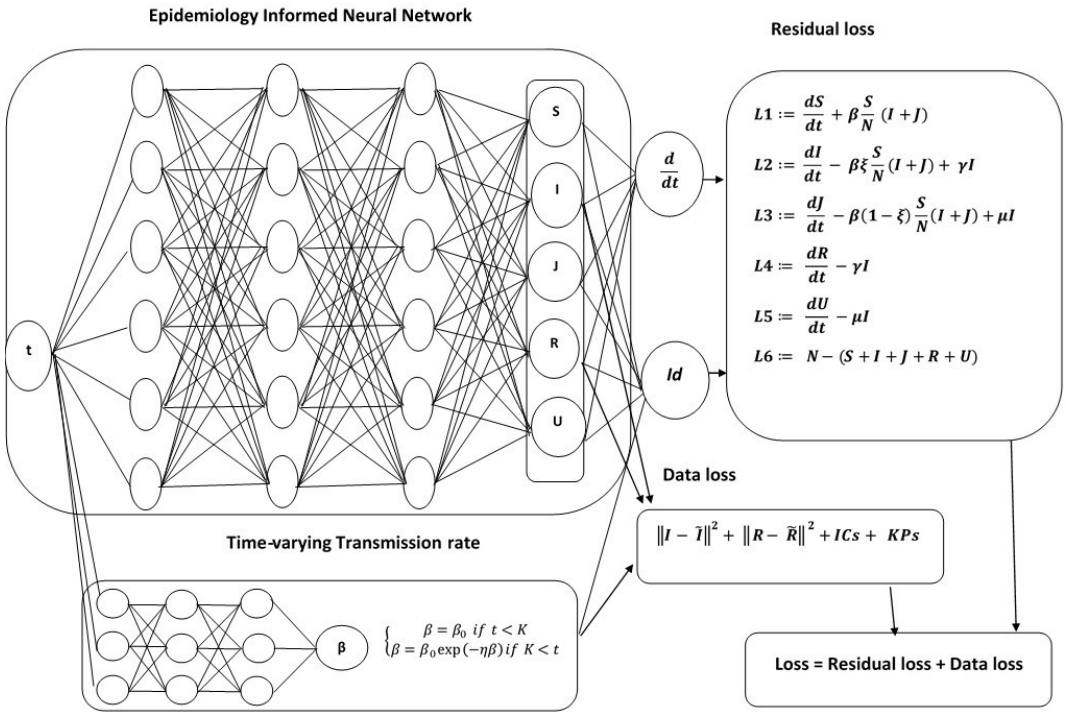


Figure 3.1: Schematic diagram of the Epidemiology-Informed Neural Network with non-linear time-varying transmission rate. The term *KPs* represent the known dynamics in the transmission rates pattern and *ICs* represent the initial condition for the asymptomatic population.

3.2.1 Delayed-mitigation exponential Time-Varying Transmission Rate

The time-varying transmission rate is non-constant in the presence of mitigation measures in the cumulative infective data. In [57, 52], it was shown that during the early phase of the COVID-19 pandemic when the cumulative infection population grew exponentially,

the transmission rate was constant. This coincides with the period before any mitigation measure. Incorporating measures such as social distancing, lockdown, and widespread adoption of facial covering in an epidemiology model is complex.

We learn an exponentially decreasing time-varying transmission rate, we observe that it takes the form of Equation (3.1). However, because we learn this time-varying transmission rate by using a neural network, our approach also detects various other post-lockdown mitigation measures. We use EINN Algorithm 2 to obtain $\beta(t)$.

Algorithm 2 EINN algorithm for Asymptomatic-SIR model with delayed-mitigation exponential time-varying transmission rate

1: Construct EINN

specify the input: t_j, $j = 1,\ldots,M$

Initialize EINN parameter: θ

Initialize the epidemiology and vaccination parameters: $\lambda = [\gamma, \mu, \kappa]$

Output layer: $S(t_j;\theta;\lambda), I(t_j;\theta;\lambda), J(t_j;\theta;\lambda), R(t_j;\theta;\lambda), U(t_j;\theta;\lambda)$, $j = 1,\ldots,M$

2: construct neural network: β

specify the input: t_j, $j = 1,\ldots,M$

Initialize the neural network parameter: ϕ

Specify β_0 obtained by nonlinear regression of early cumulative infective population data

Initialize the exponential decay parameter: η

Output layer: $\beta(t_j;\phi;\eta)$

$$\beta(t_j;\phi;\eta) = \begin{cases} \beta_0 & 0 \le t_j \le M_\beta \\ \beta_0 \exp\left(-\eta\beta(t_j;\phi;\eta)\right) & M_\beta < t_j, \end{cases} \tag{3.5}$$

3: Specify EINN training set

Training data: using cubic spline, generate $\tilde{I}(t_j)$ and $\tilde{R}(t_j)$, $j = 1,\ldots,M$.

Set ξ to the value obtained for ξ from EINN Algorithm 1

Initialize the Asymptomatic population: $\tilde{J}(0) = (1-\xi)\tilde{I}(0)/\xi$ and $\tilde{U}(0) = (1-\xi)\tilde{R}(0)/\xi$.

4: Train the neural networks

Specify an MSE loss function:

$$\begin{aligned} MSE = & \frac{1}{M}\sum_{j=1}^{M} ||I(t_j;\theta;\lambda) - \tilde{I}(t_j)||_2^2 + \frac{1}{M}\sum_{j=1}^{M} ||R(t_j;\theta;\lambda) - \tilde{R}(t_j)||_2^2 \\ & + \frac{1}{M_\beta}\sum_{j=1}^{M_\beta} ||\beta(t_j;\phi;\eta) - \beta_0||_2^2 \\ & + \frac{1}{M_\kappa}\sum_{j=1}^{M_\kappa} ||\kappa S(t_j;\theta) - \tilde{V}(t_j)||_2^2 \\ & + ||J(0;\theta;\lambda) - \tilde{J}(0)||_2^2 + ||U(0;\theta;\lambda) - \tilde{U}(0)||_2^2 \\ & + \frac{1}{M}\sum_{i=1}^{6}\sum_{j=1}^{M} ||L_i(t_j;\theta;\phi;\lambda;\eta)||_2^2. \end{aligned} \tag{3.6}$$

Minimize the MSE loss function: compute $\underset{\{\theta;\phi;\lambda;\eta\}}{\arg\min}\,(MSE)$ using an optimizer such as the adam optimizer.

5: return EINN solution

$S(t_j;\theta;\lambda)$, $I(t_j;\theta;\lambda)$, $J(t_j;\theta;\lambda)$, $R(t_j;\theta;\lambda)$, $U(t_j;\theta;\lambda)$, $j = 1,\ldots,M$.

epidemiology parameters: γ and μ

vaccination parameter: κ

6: return time-varying epidemiology parameter:

$\beta(t_j;\phi;\eta)$, $j = 1,\ldots,M$.

Rate of human contact decrease: η.

3.2.2 Piecewise time-varying transmission rate

A piecewise time-varying transmission rate (3.7) is used to learn a time-dependent transmission rate β in eq. (2.1). In [72, 73], the piecewise $\beta(t)$ is defined as follows,

$$\beta(t) = \begin{cases} \beta_0 q_1 & t \leq M_1 \\ \beta_0 q_2 & M_1 < t \leq M_2 \\ \beta_0 q_3 & M_2 < t \leq M_3 \\ \beta_0 q_4 & M_3 < t \leq M_4 \\ \quad \vdots & \\ \beta_0 q_n & M_n < t. \end{cases} \tag{3.7}$$

The goal of the parameters $q_1, \ldots, q_n$ in (3.7) is to capture the exponential decrease observed in the transmission rate $\beta(t)$. We choose $M_1, \ldots, M_n$ in order to partition the pandemic timeline, according to the onset of various mitigation measures, and we use EINN algorithm 3 to learn $q_1, \ldots, q_n$.

Algorithm 3 EINN algorithm for Asymptomatic-SIR model with piecewise time-varying transmission rate

1: Construct EINN

specify the input: t_j, $j = 1, \ldots, M$

Initialize EINN parameter: θ

Initialize the epidemiology and vaccination parameters: $\lambda = [\gamma, \mu, \kappa]$

Output layer: $S(t_j;\theta;\lambda), I(t_j;\theta;\lambda), J(t_j;\theta;\lambda), R(t_j;\theta;\lambda), U(t_j;\theta;\lambda)$, $j = 1, \ldots, M$

2: construct neural network: β

specify the input: t_j, $j = 1, \ldots, M$

Initialize the neural network parameter: ϕ

Specify β_0 obtained by nonlinear regression of early cumulative infective population data

Initialize the decay parameters: $q_1, q_2, q_3, q_4, \ldots, q_n$

Output layer: $\beta(t_j;\phi;q_1,q_2,q_3,q_4,\ldots,q_n)$

$$\beta(t_j;\phi;q_1,q_2,q_3,q_4,\ldots,q_n)=\begin{cases}\beta_0 q_1\beta(t_j;\phi;q_1) & 0\le t_j\le M_1\\ \beta_0 q_2\beta(t_j;\phi;q_2) & M_1<t_j\le M_2\\ \beta_0 q_3\beta(t_j;\phi;q_3) & M_2<t_j\le M_3\\ \beta_0 q_4\beta(t_j;\phi;q_4) & M_3<t_j\le M_4\\ & \vdots\\ \beta_0 q_n\beta(t_j;\phi;q_n) & M_n<t_j,\end{cases} \tag{3.8}$$

3: Specify EINN training set

Training data: using cubic spline, generate $\tilde{I}(t_j)$ and $\tilde{R}(t_j)$, $j=1,\ldots,M$.

Set ξ to the value obtained for ξ from EINN Algorithm 1

Initialize the Asymptomatic population: $\tilde{J}(0)=(1-\xi)\tilde{I}(0)/\xi$ and $\tilde{U}(0)=(1-\xi)\tilde{R}(0)/\xi$.

4: Train the neural networks

Specify an *MSE* loss function:

$$\begin{aligned}MSE=&\frac{1}{M}\sum_{j=1}^{M}||I(t_j;\theta;\lambda)-\tilde{I}(t_j)||_2^2+\frac{1}{M}\sum_{j=1}^{M}||R(t_j;\theta;\lambda)-\tilde{R}(t_j)||_2^2\\&+\sum_{i=1}^{n}\frac{1}{M_i}\sum_{j=1}^{M_i}||\beta(t_j;\phi;\eta)-\beta_0||_2^2\\&+\frac{1}{M_\kappa}\sum_{j=1}^{M_\kappa}||\kappa S(t_j;\theta)-\tilde{V}(t_j)||_2^2\\&+||J(0;\theta;\lambda)-\tilde{J}(0)||_2^2+||U(0;\theta;\lambda)-\tilde{U}(0)||_2^2\\&+\frac{1}{M}\sum_{i=1}^{6}\sum_{j=1}^{M}||L_i(t_j;\theta;\phi;\lambda;\eta)||_2^2.\end{aligned} \tag{3.9}$$

Minimize the *MSE* loss function: compute $\underset{\{\theta;\phi;\lambda;\eta\}}{\arg\min}\,(MSE)$ using an optimizer such as the adam optimizer.

5: return EINN solution

$S(t_j;\theta;\lambda), I(t_j;\theta;\lambda), J(t_j;\theta;\lambda), R(t_j;\theta;\lambda), U(t_j;\theta;\lambda),\ j = 1,\ldots,M.$

epidemiology parameters: γ and μ

vaccination parameter: κ

6: return time-varying epidemiology parameter:

$\beta(t_j;\phi;q_1,q_2,q_3,q_4,\ldots,q_n),\ j = 1,\ldots,M.$

Rate of human contact decrease: $q_1,q_2,q_3,q_4,\ldots,q_n$.

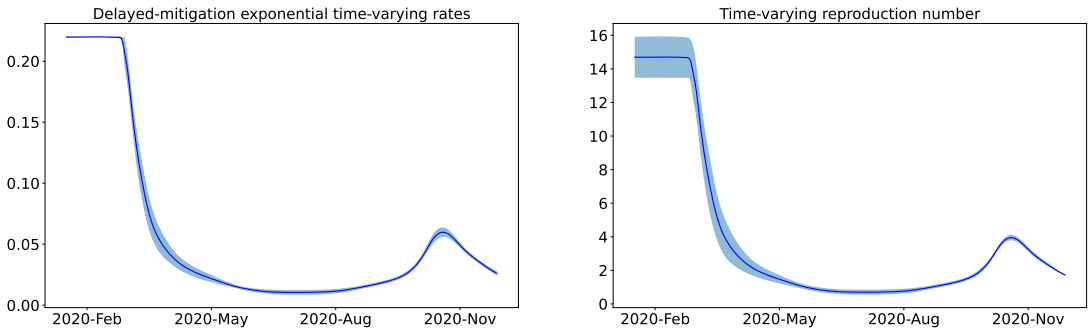
3.3 Data-Driven Simulation for Time-Varying Transmission Rate

In this section, We present a discussion of the time-varying transmission rates obtained using the delayed-mitigation exponential transmission rate and the piecewise transmission rate.

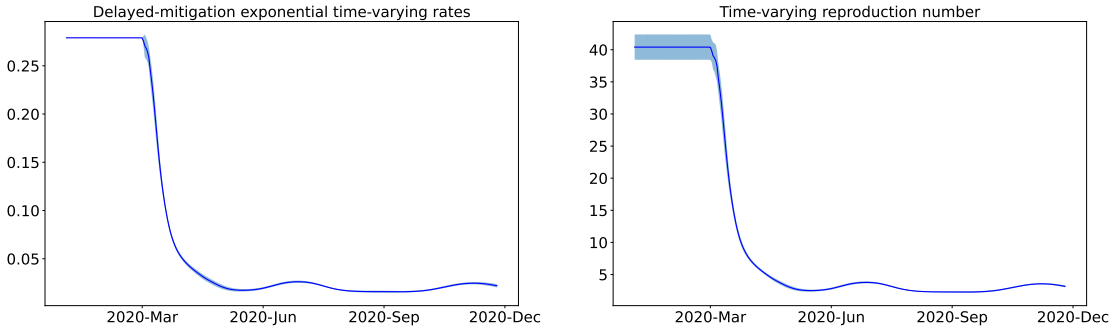
3.3.1 Data-Driven Simulation for Delayed-mitigation exponential Transmission Rate

In the EINN Algorithm 2, M_β corresponds to the number of days mitigation is delayed in the data, which is equal to K in Equation (3.1). M_κ is the number of vaccination days. In Figures 3.2(**a**) and 3.2(**b**), time-varying transmission rates learned by the EINN Algorithm 2.

In Figure 3.2(**a**) A learned delayed-mitigation exponential time-varying transmission rate β is plotted for cumulative Italy COVID-19 data from January 31, 2020 to December 11, 2020. $N = 60.36 \times 10^6$. The plotted time-varying basic reproduction rate $\mathscr{R}_t$ shows the impact of lockdown and the mitigation measures post-lockdown. The relaxation that followed is due to the COVID-19 surge and is detected in the learned β and $\mathscr{R}_t$. The EINN Algorithm 2 also learns $\gamma = 0.0121$ and $\mu = 0.0106$. The MSE in (I) is 7.5×10^{-5}. In Figure 3.2(**b**) A learned delayed-mitigation exponential time-varying transmission rate β is plotted for cumulative U.S.A COVID-19 data from 22 January 2020 to December 11, 2020. $N = 328.2 \times 10^6$. The time-varying basic reproduction rate R_t is underestimated pre-lockdown. The EINN Algorithm 2 also learns $\gamma = 0.001$ and $\mu = 0.0224$. The MSE



(a) Italy



(b) USA

Figure 3.2: Delayed-mitigation exponential time-varying rates.

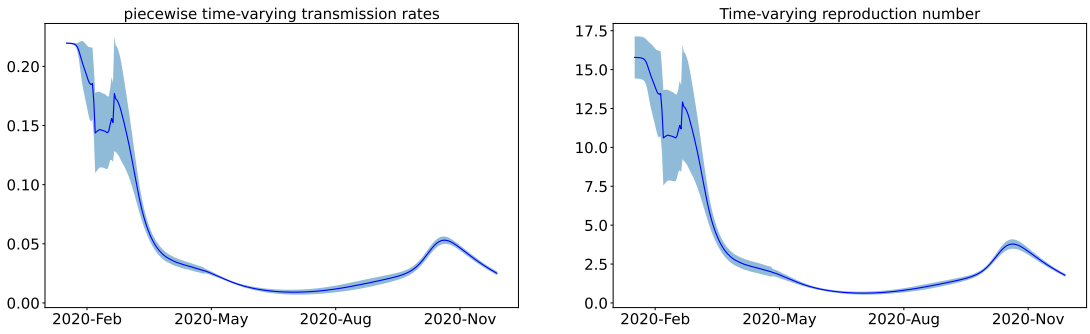
in (I) is 3.88×10^{-4}. In 3.2(**a**) The delayed-mitigation exponential transmission rate is learned using Equation (3.1) in Equation (2.1). We set $K = 40$ and we and fix $\xi = 0.37$ in EINN Algorithm 2. We take $\beta_0 = 0.22$, obtained using early data and nonlinear regression. EINN Algorithm 2 learns $\eta = 0.87$, the rate at which human contact decreases. In 3.2(**b**) The delayed-mitigation exponential transmission rate is learned using Equation (3.1) in Equation (2.1). We set $K = 57$ and we fix $\xi = 0.46$ in EINN Algorithm 2. We take $\beta_0 = 0.279$, obtained using early data and nonlinear regression. EINN Algorithm 2 learns $\eta = 0.60$, the rate at which human contact decreases.

In Section 3.3.1, the delayed-mitigation exponential time-varying transmission rate detects the impact of 2020 COVID-19 lockdown, as well as the other mitigation measures post-lockdown using the parameter η. It is however difficult to know if η captures all the pattern in the time-varying transmission rate as demonstrated in Figure 3.2a,b, i.e., whether or not Equation (3.1) helps us to learn the most accurate form of β. For instance, the time-varying basic reproduction rate $\mathscr{R}_t$ is underestimated pre-lockdown in the USA data and overestimated pre-lockdown in Italy data.

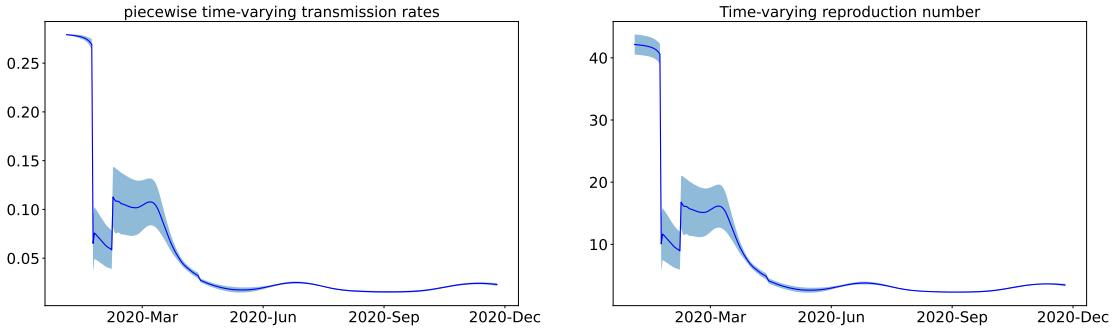
3.3.2 Data-Driven Simulation for Piecewise Transmission Rate

In the EINN Algorithm 3, $M_i, 1 \leq i \leq n$ are chosen to corresponds to a partitioning in the data. Time-varying transmission rates learned by the EINN Algorithm 3 are presented in Figures 3.3(**a**) and 3.2(**b**). For Italy and USA data, we used the following formulation for $\beta(t)$ in Algorithm 3

$$\beta(t) = \begin{cases} \beta_0 q_1 & 0 \leq t_j \leq 20 \\ \beta_0 q_2 & 20 < t_j \leq 35 \\ \beta_0 q_3 & 35 < t_j \leq 100 \\ \beta_0 q_4 & 100 < t. \end{cases} \tag{3.10}$$



(a) Italy



(b) USA

Figure 3.3: Piecewise-constant time-varying rates.

Parameters	Mean	Std
γ	0.00459	0.00013
μ	0.01202	0.00238
q_2	0.29297	0.17053
q_3	0.53369	0.16459
q_4	0.49833	0.07585

Table 3.1: Setting $q_1 = 1$, EINN Algorithm 3 learns q_2, q_3, and q_4 for Italy data from 31 January 2020 to 11 December 2020

Parameters	Mean	Std
γ	0.01338	0.00062
μ	0.01700	0.00704
q_2	0.73672	0.19446
q_3	0.84209	0.16659
q_4	0.84166	0.09445

Table 3.2: Setting $q_1 = 1$, EINN Algorithm 3 learns q_2, q_3, and q_4 for USA data from 31 January 2020 to 11 December 2020

4

A MULTI-VARIANT MATHEMATICAL MODEL WITH HETEROGENEOUS TRANSMISSION RATES

"There are three kinds of epidemiologist: those who can count and those who can't."

Unknown

COVID-19 was first reported in China in 2019 [74], it has since become a global pandemic. In recent months, there have been reports of mutating variants of the virus [75]. In 2021, the dominant mutant variant of COVID-19 was the B.1.617.2 delta variant [76]. Effort to combat the spread of COVID-19 have included combinations of pharmaceutical (vaccination and hospitalization) and non-pharmaceutical (social distancing, contact tracing, and facial mask) measures.

Prior to the onset of COVID-19 mutating variants in the US, the progress seen in the data from several states prompted the ease of the various non-pharmaceutical measures. Amid the news that several states had vaccinated over 70% of its population and a few states had vaccinated between $60\% - 70\%$ of its population, vaccination effort began to slow down in many US states. As a result, the existence of mutating variants resulted in a resurgence in cases of infections. The Center for Disease Control and Prevention (CDC) reported that the dominant variant in the US in 2021 was the B.1.617.2 delta variant. According to the World Health Organization (WHO), many variants were first reported in the United Kingdom and South Africa and in recent months, the USA, Europe, China, Brazil, and Japan have all reported mutating variant infected cases.

In this chapter, a Susceptible-Exposed-Infected-Recovered, mathematical model (SEIR) is developed. We present a data-driven deep learning algorithm for a model consisting of time-varying transmission rates for each active variant. Using infected daily cases data, we

learn the form of the time-varying transmission rates, to reveal a timeline of the impact of mitigation measures on the transmission of COVID-19 [77, 72].

4.1 Multi-variant SEIR model

We assume that the total population $N(t) = N$ at any given time is distributed among the following compartments: susceptible (S), exposed (E), Infectious (I_i), $i = 1,\ldots,M$, and recovered (R), where M is the number of different variants. The interaction between the compartments is shown in Figure 4.1.

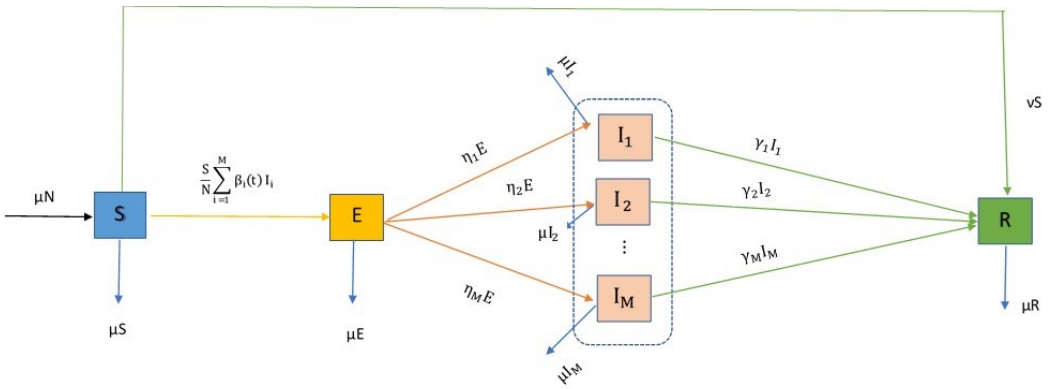


Figure 4.1: Transfer diagram between the compartments

As shown in Figure (4.1), the susceptible individuals enter the exposed compartment at the rate $\frac{1}{N}\sum_{i=1}^{M}\beta_i(t)I_i$, where $\beta_i(t)$ is the transmission rate of variant i. The exposed individuals progress to the ith infected compartment at the rate η_i. The ith infected compartment recover at the rate of $\gamma_i I_i$.

We assume a natural death rate μ, given by $\mu = \frac{1}{LFE\times 365}$, where LFE denotes the life expectancy. For simplicity, it is assumed that the birth rate of the population is equal to the death rate. The parameter η is the transmission rate from the exposed to the various infectious sub-compartments, γ_i^{-1} is the mean symptomatic infectious period for the ith variant. The parameter $\nu(t)$ represent the time-dependent removal rate of the vaccinated individuals from the susceptible compartment. We assume that any variant does not super-infect another variant, so there are no interactions between the infectious sub-compartments.

Based on the transfer diagram depicted in Figure 4.1, the mathematical model for a multi-variant COVID-19 pandemic with heterogeneous transmission rates is given by:

$$\begin{aligned}
\frac{dS}{dt} &= \mu N - \frac{S}{N}\sum_{i=1}^{M}\beta_i(t)I_i - \Big(\nu(t)+\mu\Big)S \\
\frac{dE}{dt} &= \frac{S}{N}\sum_{i=1}^{M}\beta_i(t)I_i - (\eta+\mu)E \\
\frac{dI_i}{dt} &= \eta_i E - (\gamma_i+\mu)I_i, \quad i = 1,\ldots,M \\
\frac{dR}{dt} &= \sum_{i=1}^{M}\gamma_i I_i + \nu(t)S - \mu R
\end{aligned} \tag{4.1}$$

subject to non-negative initial conditions

$$S(0) = S_0, \quad E(0) = E_0, \quad I_i(0) = I_{i,0}, i = i,\ldots,M, \quad R(0) = R_0.$$

The parameter η is defined as: $\eta = \sum_{i=1}^{M}\eta_i$, and the total population is

$$N(t) = S(t) + E(t) + \sum_{i=1}^{M} I_i(t) + R(t).$$

The differential equation satisfied by the total population size is obtained by adding all the equations in (4.1), that is, $\frac{dN}{dt} = 0$ and thus N is constant. The model parameters are summarized in Table 4.1.

Time-varying transmission rates have been shown to efficiently model the spread of COVID-19 [77, 68]. Next, will discuss the form of the time-varying transmission rates for each variant.

4.1.1 Variant-based time-varying transmission rates

Time-varying transmission rates in (4.1) incorporates the impact of governmental actions, and the public response [69]. We consider the transmission rates of the form

$$\beta_i(t) = \beta_i^0 \exp(-\kappa_i t), \quad 1 \le i \le M. \tag{4.2}$$

where κ_i in (4.2) is the infectiousness factor for each i th variant. However, we define $(1+\tau_i)$ to be the factor by which a particular variant is more infectious than the original variant SARS-CoV-2. And so, the following relationship exist between each mutating variant and the SARS-CoV-2 variant, see equation (4.3).

$$\beta_i(t) = \beta_1(t)(1+\tau_i), \quad 2 \leq i \leq M. \tag{4.3}$$

In (4.3), $\beta_1(t)$ is the transmission rate for the original variant SARS-CoV-2. The transmission rates of the subsequent mutating variants are given by $\beta_i(t)$, $i = 2,\ldots,M$, where M represents the number of mutating COVID-19 variants. Although the publicly available data reports the daily infected cases, there were reports that suggest hat the dominant variant, B.1.617.2 delta variant, to be twice as infectious as the original variant SARS-CoV-2. According to CDC reports [76, 78], the delta variant accounted for 1.3% of total infected cases in May, 2021, 9.5% in June, and in August it accounted for 93% of the total infected cases.

Parameter	**Notation**	**Range**	**Remark**	**Reference**
Baseline transmission rate for each ith variant	β_i^0	[0,1)	fitted from data	[77]
Emigration rate	μ	$\frac{1}{LFE\times 365}$	constant	[79]
Mean latent period	η^{-1}	$2-14(days)$	constant	[79]
recovery rate for each ith variant	γ_i	$[0,1)$	constant	
infectiousness factor for each ith variant	κ_i	$[0,1)$	constant	

Table 4.1: Summary table of parameters in model (4.1)

4.1.2 Well-posedness of the model

Definition 1 ([80], Locally Lipschitz continuity)**.** *Let $d_1, d_2 \in \mathbf{N}$ and $\mathbf{S}$ be a subset of $\mathbf{R}^{d_1}$. A function $\mathbf{F} : \mathbf{S} \to \mathbf{R}^{d_2}$ is Lipschitz continuous on $\mathbf{S}$ if there exists a nonnegative constant $L \geq 0$ such that*

$$|\mathbf{F}(x) - \mathbf{F}(y)| \leq L|x-y|, \quad x, y \in \mathbf{S}. \tag{4.4}$$

Let $\mathscr{U}$ be an open subset of $\mathbf{R}^{d_1}$, and let $\mathbf{F}:\mathscr{U}\to\mathbf{R}^{d_2}$. We shall call $\mathbf{F}$ locally Lipschitz continuous if for every point $x_0\in\mathscr{U}$ there exists a neighborhood V of x_0 such that the restriction of $\mathbf{F}$ to V is Lipschitz continuous on V.

We consider a more general framework of model (4.1)

$$z'(t)=G(z(t)),\quad z(0)=z_0, \tag{4.5}$$

where $z(t)=(x_1(t),x_2(t),\ldots,x_n(t))^T$ and $G(z(t))=(g_1(z(t)),g_2(z(t)),\ldots,g_n(z(t)))^T$, the initial condition $z_0\in\mathbf{R}^n$. We state the following theorem.

Theorem 4.1.1 ([80]). *If $\mathbf{G}:\mathbf{R}^n\to\mathbf{R}^n$ is locally Lipschitz continuous and if there exist nonnegative constants B, K such that*

$$|\mathbf{G}(z(t))|\le K\,|z(t)|+B,\quad z(t)\in\mathbf{R}^n, \tag{4.6}$$

then the solution of the initial value problem (4.5) *exists for $t>0$, and*

$$|(z(t))|\le|z_0|\cdot\exp(K\,|t|)+\frac{B}{K}\cdot(\exp(K\cdot|t|)-1),\quad t>0, \tag{4.7}$$

Lemma 4.1.2. *For each ith variant, $i\in\{1,\ldots,M\}$, the time varying transmission rates $\nu,\beta_i:[0,\infty)\to[0,\infty)$ are Lipschitz continuous and contnuously differentiable. There exists $\beta_{min},\beta_{max}>0$ and $\nu_{min}\nu_{max}>0$ such that $\beta_{min}\le\beta_i(t)\le\beta_{max}$, $\nu_{min}\le\nu(t)\le\nu_{max}$ for all t.*

Theorem 4.1.3. *The nonlinear first order system of differential equations* (4.1) *has at least one solution which exists for $t\in[0,\infty)$.*

Proof. Let $z(t)=(S(t),E(t),I_1(t),\ldots,I_M(t),R(t))^T$, we can set

$$G:\mathbf{R}^{M+3}\to\mathbf{R}^{M+3},\qquad z(t)\to\begin{pmatrix}\mu N-\frac{S}{N}\sum_{i=1}^{M}\beta_i(t)I_i-\Big(\nu(t)+\mu\Big)S\\ \frac{S}{N}\sum_{i=1}^{M}\beta_i(t)I_i-(\eta+\mu)E\\ \eta_iE-(\gamma_i+\mu)I_i,\quad i=1,\ldots,M\\ \sum_{i=1}^{M}\gamma_iI_i+\nu(t)S-\mu R\end{pmatrix} \tag{4.8}$$

G is locally Lipschitz continuous, using supremum norm $||f(t)|| := \sup_{t\in[a,b]} |f(t)|$, we have

$$
\begin{aligned}
||G(z(t))|| = \sup_{t\in[0,\infty)} \Bigg\{ & \left| \mu N - \frac{S(t)}{N}\sum_{i=1}^{M}\beta_i(t)I_i(t) - \Big(\nu(t)+\mu\Big)S(t) \right|, \left| \frac{S(t)}{N}\sum_{i=1}^{M}\beta_i(t)I_i - (\eta+\mu)E(t) \right|, \\
& \Big|\eta_i E(t) - (\gamma_i+\mu)I_i(t), \quad i=1,\ldots,M\Big|, \Big|\sum_{i=1}^{M}\gamma_i I_i(t) + \nu(t)S(t) - \mu R(t)\Big| \Bigg\} \\
\leq \sup_{t\in[0,\infty)} \Bigg\{ & \mu N + \beta_{max}\left|\frac{S(t)}{N}\sum_{i=1}^{M} I_i(t)\right| + (\nu_{max}+\mu)|S(t)|, \beta_{max}\left|\frac{S(t)}{N}\sum_{i=1}^{M} I_i(t)\right| + (\eta+\mu)|E(t)|, \\
& \eta_i|E(t)| + (\gamma_i+\mu)|I_i(t)|, \gamma_i\sum_{i=1}^{M}|I_i(t)| + \nu_{max}|S(t)| + \mu|R(t)| \Bigg\} \\
\leq \sup_{t\in[0,\infty)} \Big\{ & \mu N + \beta_{max}|S(t)| + (\nu_{max}+\mu)|S(t)|, \beta_{max}|S(t)| + (\eta+\mu)|E(t)|, \\
& \eta_i|E(t)| + (\gamma_i+\mu)|I_i(t)|, \gamma_i\sum_{i=1}^{M}|I_i(t)| + \nu_{max}|S(t)| + \mu|R(t)| \Big\} \\
\leq & K||z(t)||.
\end{aligned}
$$

So by Theorem 4.1.1, and the boundedness of the time-varying nonlinear functions from Lemma 4.1.2, the nonlinear initial value problem 4.1 has a solution for all time. □

4.1.3 Basic reproduction number and equilibria stability

The basic reproduction number $\mathscr{R}_0$ is the expected number of secondary infections that a single infectious individual will generate on average within a susceptible population.

Definition 2. *The disease-free equilibrium of* (4.1) *is given by*

$$(S^*, E^*, I_1^*, \ldots, I_M^*, R^*) = (S_0, 0, 0, \ldots, 0, 0).$$

The basic reproduction number $\mathscr{R}_0$ is calculated for the case when $\beta_i(t) = \beta_i^0$, $i \in \{1,\ldots,M\}$. Applying the next-generation operator approach [64], the reproduction number

$\mathscr{R}_0$ is obtained as the spectral radius of the next generation matrix FV^{-1}, where

$$F = \begin{pmatrix} 0 & \beta_1^0 & \dots & \beta_M^0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}, \quad V = \begin{pmatrix} \eta+\mu & 0 & 0 & \dots & 0 \\ -\eta_1 & \gamma_1+\mu & 0 & \dots & 0 \\ -\eta_2 & 0 & \gamma_2+\mu & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\eta_M & 0 & 0 & \dots & \gamma_M+\mu \end{pmatrix}.$$

The basic reproduction number $\mathscr{R}_0$ is computed as follows in (4.9)

$$\mathscr{R}_0 = \sum_{i=1}^{M} \frac{\beta_i^0 \eta_i}{(\gamma_i+\mu)(\eta+\mu)}. \tag{4.9}$$

Next, we analyze the local asymptotic stability of the disease-free equilibrium in Definition 2.

Theorem 4.1.4. *The disease-free equilibrium* $(S^*, E^*, I_1^*, \dots, I_M^*, R^*)$ *of* (4.1) *is locally asymptotically stable if* $\mathscr{R}_0 < 1$.

Proof. The Jacobian of the right hand side of (4.1) at the equilibrium point is given by

$$J = \begin{pmatrix} -(\nu+\mu) & 0 & -\beta_1^0 & -\beta_2^0 & \dots & -\beta_M^0 & 0 \\ 0 & -(\eta+\mu) & \beta_1^0 & \beta_2^0 & \dots & \beta_M^0 & 0 \\ 0 & \eta_1 & -(\gamma_1+\mu) & 0 & \dots & 0 & 0 \\ 0 & \eta_2 & 0 & -(\gamma_2+\mu) & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \eta_M & 0 & 0 & \dots & -(\gamma_M+\mu) & 0 \\ \nu & 0 & \gamma_1 & \gamma_2 & \dots & \gamma_M & -\mu \end{pmatrix}$$

If $M = 1$, the eigenvalues of the Jacobian matrix are given as follows:

$$\lambda_1 = -\mu, \lambda_2 = -(\nu+\mu), \lambda_3 = \frac{1}{2}\Big(-A-B\Big), \lambda_4 = \frac{1}{2}\Big(A-B\Big).$$

where

$$A = \sqrt{4\beta_1^0\eta_1 + (\eta_1 - \gamma_1)^2},$$
$$B = \eta_1 + 2\mu + \gamma_1.$$

Clearly, $A, B > 0$ and $A < B$, so that $\lambda_1, \lambda_2, \lambda_3, \lambda_4 < 0$. Similarly, we can show negative eigenvalues for $M \geq 2$. So the disease-free equilibrium is locally asymptotically stable. □

4.2 EINN for SEIR model with time-varying transmission rate

We observe that training data is not available for all the compartments in the SEIR model, however, EINN is able to capture the epidemiology interactions between the compartments because the epidemiology model residual is included in the MSE loss function.

The MSE loss function for EINN with the time-varying transmission rate is given by,

$$\begin{aligned}
MSE = {} & \frac{1}{T_\delta}\sum_{j=1}^{T_\delta}\left|I_1(t_j;\psi;\rho) - \tilde{I}(t_j)\right|^2 + \frac{1}{|T-T_\delta|}\sum_{j=T_\delta}^{T}\left|\sum_{i=1}^{M} I_i(t_j;\psi;\rho) - \tilde{I}(t_j)\right|^2 \\
& + \frac{1}{T_v}\sum_{j=1}^{T_v}\left|\nu(t_j;\psi;\rho) - 0\right|^2 + \frac{1}{|T-T_v|}\sum_{j=T_v}^{T}\left|\nu(t_j;\psi;\rho)S(t_j;\psi;\rho) - \tilde{V}(t_j)\right|^2 \\
& + \sum_{i=2}^{M}\left|I_i(t_{\delta_1};\psi;\rho) - p_{\delta_1}\tilde{I}(t_{\delta_1})\right|^2 + \sum_{i=2}^{M}\left|I_i(t_{\delta_2};\psi;\rho) - p_{\delta_2}\tilde{I}(t_{\delta_2})\right|^2 \\
& + \frac{1}{T_\delta}\sum_{i=2}^{M}\sum_{j=1}^{T_\delta}\left|\beta_i(t_j;\pi_i;\kappa_i) - 0\right|^2 \\
& + \frac{1}{|T-T_\delta|}\sum_{i=2}^{M}\sum_{j=T_\delta}^{T}\left|\beta_1(t_j;\pi_1;\kappa_1)(1+\tau_i) - \beta_i(t_j;\pi_i;\kappa_i)\right| \\
& + \sum_{l=1}^{5} L_l,
\end{aligned} \tag{4.10}$$

where the residual L_l, $l = 1, \ldots 5$, is as follows

$$
\begin{aligned}
L_1 &= \frac{1}{T}\sum_{j=1}^{T}\left|\frac{dS(t_j;\psi;\rho)}{dt_j} + \frac{S(t_j;\psi;\rho)}{N}\Big(\sum_{i=1}^{M}\beta_i(t_j;\pi_i;\kappa_i)I_i(t_j;\psi;\rho)\Big) + \Big(\nu(t_j;\psi;\rho)+\mu\Big)S(t_j;\psi;\rho) - \mu N\right|^2 \\
L_2 &= \frac{1}{T}\sum_{j=1}^{T}\left|\frac{dE(t_j;\psi;\rho)}{dt_j} - \frac{S(t_j;\psi;\rho)}{N}\Big(\sum_{i=1}^{M}\beta_i(t_j;\pi_i;\kappa_i)I_i(t_j;\psi;\rho)\Big) + (\eta+\mu)E(t_j;\psi;\rho)\right|^2 \\
L_3 &= \frac{1}{T}\sum_{i=1}^{M}\sum_{j=1}^{T}\left|\frac{dI_i(t_j;\psi;\rho)}{dt_j} - \eta_i E(t_j;\psi;\rho) + (\gamma_i+\mu)I_i(t_j;\psi;\rho)\right|^2 \\
L_4 &= \frac{1}{T}\sum_{j=1}^{T}\left|\frac{dR(t_j;\psi;\rho)}{dt_j} - \sum_{i=1}^{M}\gamma_i I_i(t_j;\psi;\rho) - \nu(t_j;\psi;\rho)S(t_j;\psi;\rho) + \mu R(t_j;\psi;\rho)\right|^2 \\
L_5 &= \frac{1}{T}\sum_{j=1}^{T}\left|N - (S(t_j;\psi;\rho) + E(t_j;\psi;\rho) + \sum_{i=1}^{M} I_i(t_j;\psi;\rho) + R(t_j;\psi;\rho))\right|^2.
\end{aligned}
\tag{4.11}
$$

where $\eta = \sum_{i=1}^{M}\eta_i$, $i = 1, \ldots M$.

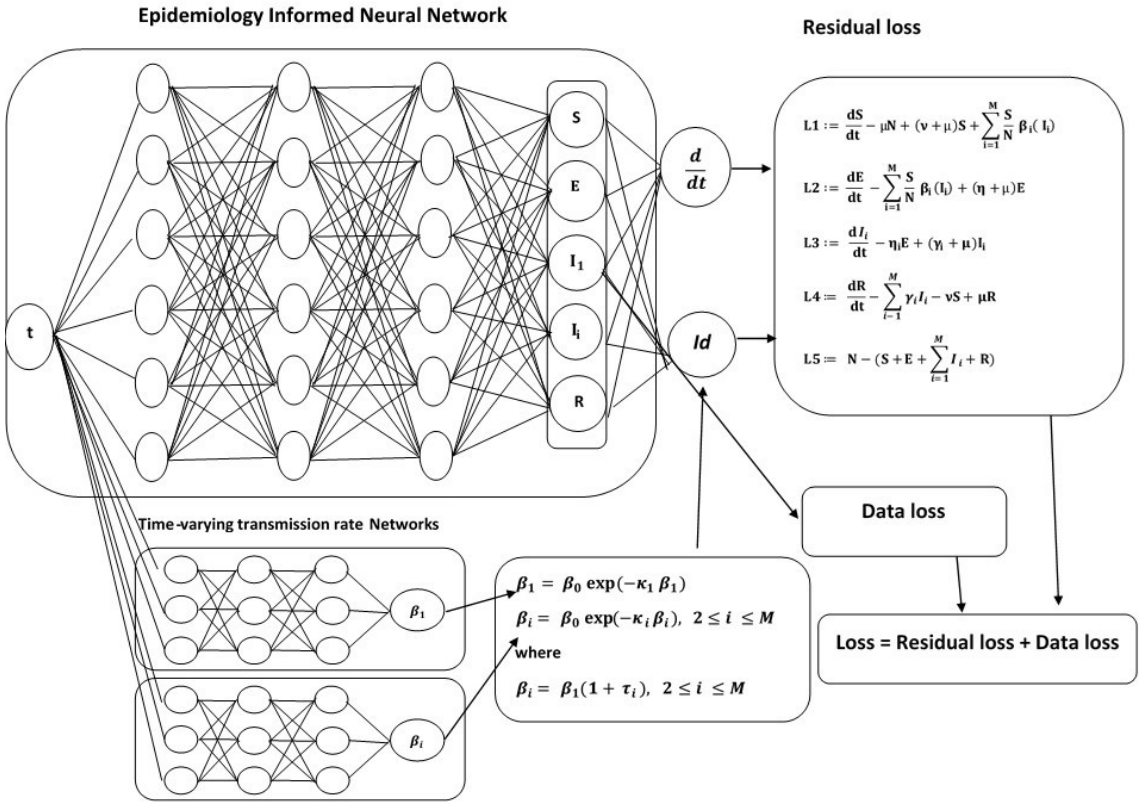


Figure 4.2: Schematic diagram of the Epidemiology Informed Neural Network with non-linear time-varying transmission rate.

The output of EINN are the learned solutions to the SEIR model (4.1) denoted by $S(t_j;\psi;\rho)$, $E(t_j;\psi;\rho)$, $I(t_j;\psi;\rho)$, $R(t_j;\psi;\rho)$, $j = 1,\ldots,T$. Here, ψ represent the neural network weights and biases while ρ represent the epidemiology parameters and T is the number of days in our dataset. Next, we set-up time-varying transmission rate networks whose outputs are $\beta_i(t_j;\pi_i;\kappa_i)$, $j = 1,\ldots,T$, for $i = 1,\ldots,M$. Each π_i represent the weights and biases of each ith network and κ_i is the infectiousness factor for each ith variant. The training data is generated using cubicspline and denoted by $\tilde{I}(t_j)$ and $\tilde{V}(t_j)$, $j = T_v,\ldots,T$. Here T_v is an integer that correspond to the vaccination start date in the dataset. The B.1.617.2 delta variant was first reported in the USA in May, T_δ is an integer that correspond to May $4th$, 2021. We observe that training data is not available for all the compartments in the SEIR model, however, EINN is able to capture the epidemiology interactions between the compartments because the residual of equation (4.1) is included in the MSE loss function.

The daily cases, the vaccinated cases, the known COVID-19 variants facts and the transmission rates are enforced in the mean square error (MSE) (4.10), see Figure (4.2). For instance, p_{δ_1} and p_{δ_2} correspond to the proportion of daily cases that was due to the mutating variants as reported by the CDC [78].

Algorithm 4 EINN algorithm for SEIR model with time-varying transmission rate

1: Construct EINN

specify the input: t_j, $j = 1,\ldots,T$

Initialize EINN parameter: ψ

Initialize the mathematical model parameters: $\rho = [\gamma_i]$, $i = 1,\ldots,M$.

Output layer: $S(t_j;\psi;\rho)$, $E(t_j;\psi;\rho)$, $I(t_j;\psi;\rho)$, $R(t_j;\psi;\rho)$, $j = 1,\ldots,T$

2: construct neural networks: β_i, $j = 1,\ldots,M$

specify the input: t_j, $j = 1,\ldots,T$

Initialize the neural network parameter: ϕ

Specify β_i^0 obtained by fitting daily cases

Output layers : $\beta_i(t_j;\pi_i;\kappa_i)$

$$\beta_i(t_j;\pi_i;\kappa_i) = (1+\tau_i)\beta_1(t_j;\pi_1;\kappa_1), \quad i \geq 2 \tag{4.12}$$

3: Specify EINN training set

Training data: using cubicspline, generate $\tilde{I}(t_j)$ and $\tilde{R}(t_j)$, $j = 1,\ldots,T$.

4: Train the neural networks

Specify an MSE loss function:

$$\begin{aligned}
MSE = &\frac{1}{T_\delta}\sum_{j=1}^{T_\delta}\left|I_1(t_j;\psi;\rho)-\tilde{I}(t_j)\right|^2 + \frac{1}{|T-T_\delta|}\sum_{j=T_\delta}^{T}\left|\sum_{i=1}^{M} I_i(t_j;\psi;\rho)-\tilde{I}(t_j)\right|^2 \\
&+\frac{1}{T_\nu}\sum_{j=1}^{T_\nu}\left|\nu(t_j;\psi;\rho)-0\right|^2 + \frac{1}{|T-T_\nu|}\sum_{j=T_\nu}^{T}\left|\nu(t_j;\psi;\rho)S(t_j;\psi;\rho)-\tilde{V}(t_j)\right|^2 \\
&+\sum_{i=2}^{M}\left|I_i(t_{\delta_1};\psi;\rho)-p_{\delta_1}\tilde{I}(t_{\delta_1})\right|^2 + \sum_{i=2}^{M}\left|I_i(t_{\delta_2};\psi;\rho)-p_{\delta_2}\tilde{I}(t_{\delta_2})\right|^2 \\
&+\frac{1}{T_\delta}\sum_{i=2}^{M}\sum_{j=1}^{T_\delta}\left|\beta_i(t_j;\pi_i;\kappa_i)-0\right|^2 \\
&+\frac{1}{|T-T_\delta|}\sum_{i=2}^{M}\sum_{j=T_\delta}^{T}\left|\beta_1(t_j;\pi_1;\kappa_1)(1+\tau_i)-\beta_i(t_j;\pi_i;\kappa_i)\right| \\
&+\sum_{l=1}^{5} L_l,
\end{aligned} \tag{4.13}$$

Minimize the MSE loss function: compute $\arg\min_{\{\psi;\pi_i\}}(MSE)$ using an optimizer such as the $L-BFGS-B$.

5: return EINN solution

$S(t_j;\psi;\rho)$, $E(t_j;\psi;\rho)$, $I_i(t_j;\psi;\rho)$, $R(t_j;\psi;\rho)$, $j = 1,\ldots,T$, $i = 1,\ldots,M$.

epidemiology parameters: γ_i, $i = 1,\ldots,M$.

vaccination parameter: ν

6: return time-varying epidemiology parameter:

$\beta_i(t_j;\pi_i;\kappa_i)$, $j = 1,\ldots,T$, $i = 1,\ldots,M$.

Infectiousness factor: κ_i, $i = 1,\ldots,M$.

parameter: τ_i, $i = 2, \ldots, M$.

4.3 Data-driven simulation of COVID-19 variants

We present results of the implementation of the EINN algorithm(4) for COVID-19 data from Alabama, Missouri, Tennessee, and Florida. We consider data from March 2020 to September 2021, during which there were two dominant variants;the original variant SARS-CoV-2 and the delta variant (B.1.617.2). CDC report indicate that 1.3% of the total infected cases were due to the delta variant in May 4th 2021 [78]. The EINN algorithm learns the infected cases, and the time-varying transmission rates due to each variant. In Table (4.2)–(4.5), pre-γ_1, post-γ_1, post-γ_2 denote the recovery rate of people infected due to the original variant SARS-CoV-2 before the onset of the delta variant, recovery rate of people infected due to the original variant SARS-CoV-2 after the onset of the delta variant, and recovery rate of people infected due to the delta variant after the onset of the delta variant respectively.

The CDC reports that by July 31st, 2021, the proportions of infected cases that are due to the B.1.617.2 delta variant in Alabama was 82.6%, Tennessee was 67.4%, Missouri 53.9%, and in Florida, it was 86.4% [76]. The CDC also reported that in the USA, the delta variant accounted for about 1.3% of the infected cases.

We seek to learn τ_i for an ith mutating variant. For the simulations in this section, we observed that the delta variant is a dominant mutating variant therefore we included only two variants, the SARS-CoV-2 and the delta variant.

Parameters	Mean	Std
pre- γ_1	0.02423	0.01266
post- γ_1	0.00395	0.00717
post- γ_2	0.00463	0.00768
η_1	0.12437	0.04933
η_2	0.20893	0.04933
κ_1	1.07385	0.06271
κ_2	1.13052	0.02809
$(1+\tau)$	1.22391	0.10176

Table 4.2: Using Alabama daily cases from March 2020 to September 2021, the EINN Algorithm (4) learns the model parameters

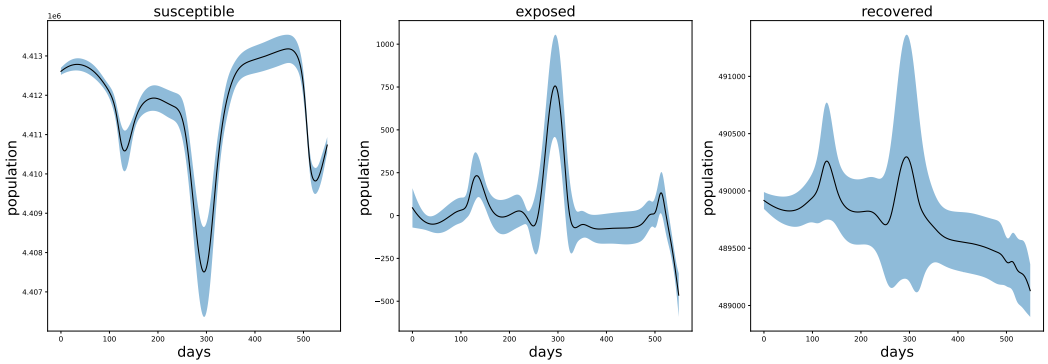


Figure 4.3: learned Alabama Susceptible, Exposed, and Recovered daily population

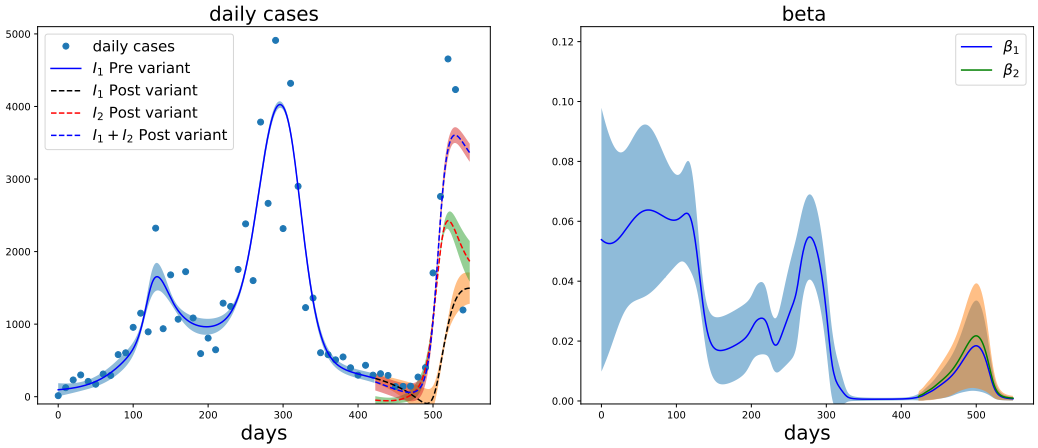


Figure 4.4: Alabama daily cases and time-varying transmission rates

Parameters	Mean	Std
pre- γ_1	0.02344	0.00613
post- γ_1	0.00911	0.00426
post- γ_2	0.02095	0.01794
η_1	0.15912	0.03381
η_2	0.17420	0.03383
κ_1	1.01114	0.03561
κ_2	1.10474	0.01358
$(1+\tau)$	1.15537	0.08817

Table 4.3: Using Missouri daily cases from March 2020 to September 2021, the EINN Algorithm (4) learns the model parameters

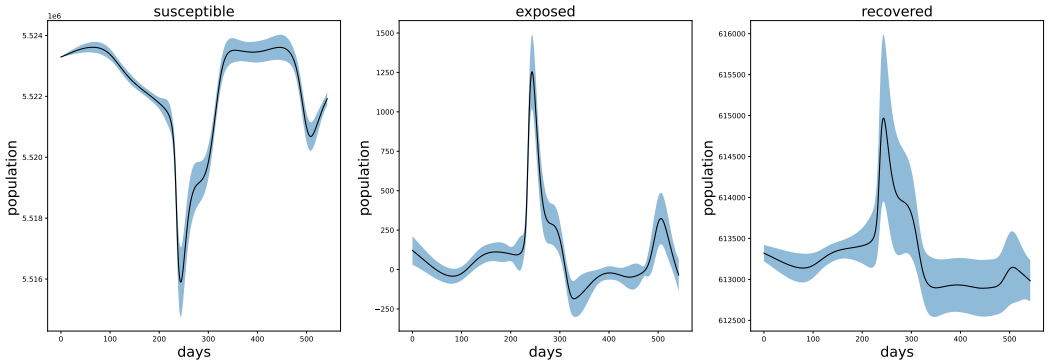


Figure 4.5: learned Missouri Susceptible, Exposed, and Recovered daily population

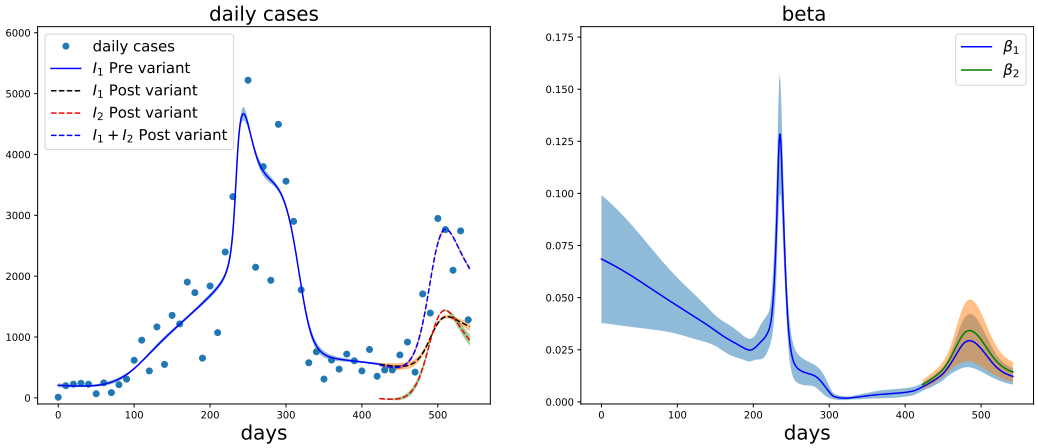


Figure 4.6: Missouri daily cases and time-varying transmission rates

Parameters	Mean	Std
pre- γ_1	0.01259	0.01111
post- γ_1	0.00587	0.00821
post- γ_2	0.00849	0.01081
η_1	0.13721	0.03429
η_2	0.19611	0.03427
κ_1	1.04761	0.03035
κ_2	1.13552	0.02867
$(1+\tau)$	1.09879	0.09738

Table 4.4: Using Tennessee daily cases from March 2020 to September 2021, EINN Algorithm (4) learns the model parameters

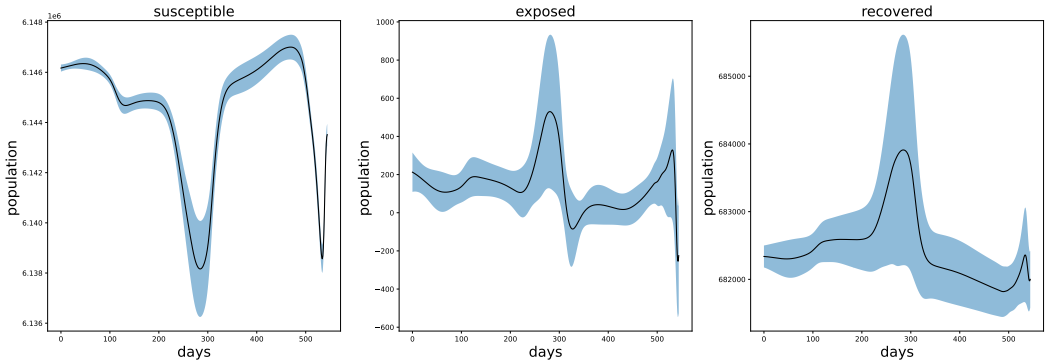


Figure 4.7: learned Tennessee Susceptible, Exposed, and Recovered daily population

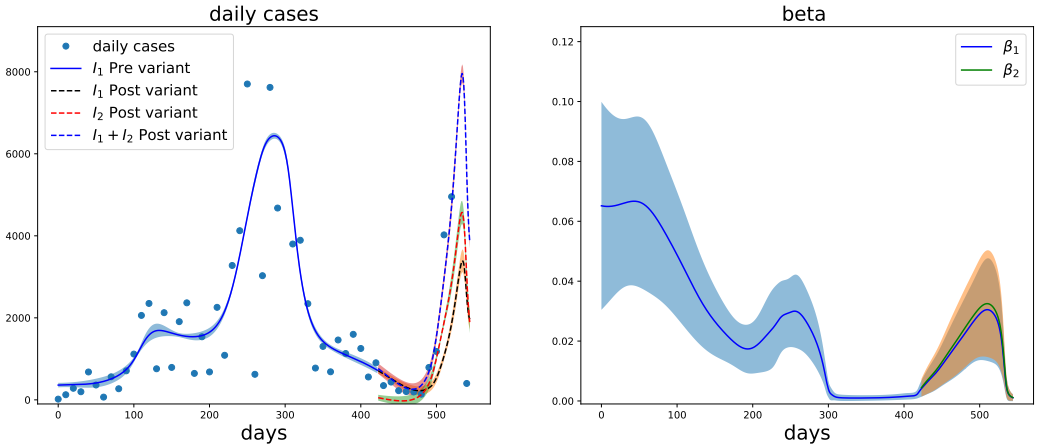


Figure 4.8: Tennessee daily cases and time-varying transmission rates

Parameters	Mean	Std
pre- γ_1	0.02968	0.01594
post- γ_1	0.00943	0.00985
post- γ_2	0.00576	0.00516
η_1	0.09304	0.06144
η_2	0.24027	0.06143
κ_1	1.03508	0.02477
κ_2	1.13773	0.00892
$(1+\tau)$	1.12553	0.11431

Table 4.5: Using Florida daily cases from March 2020 to September 2021, EINN Algorithm (4) learns the model parameters

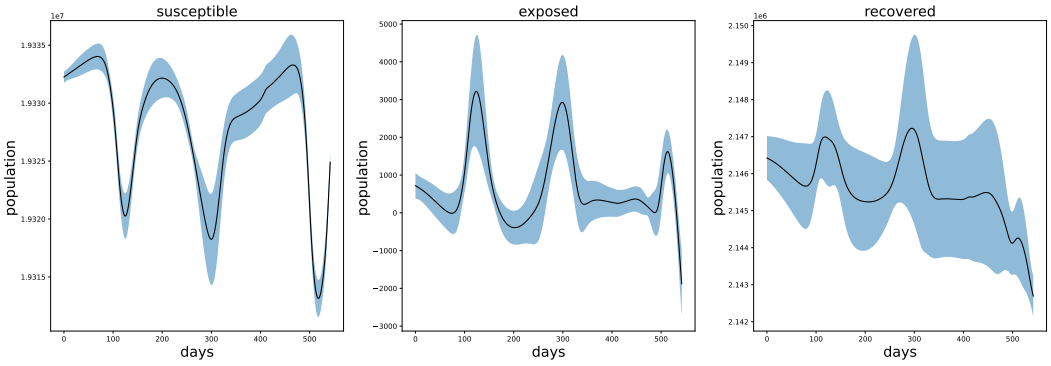


Figure 4.9: learned Florida Susceptible, Exposed, and Recovered daily population

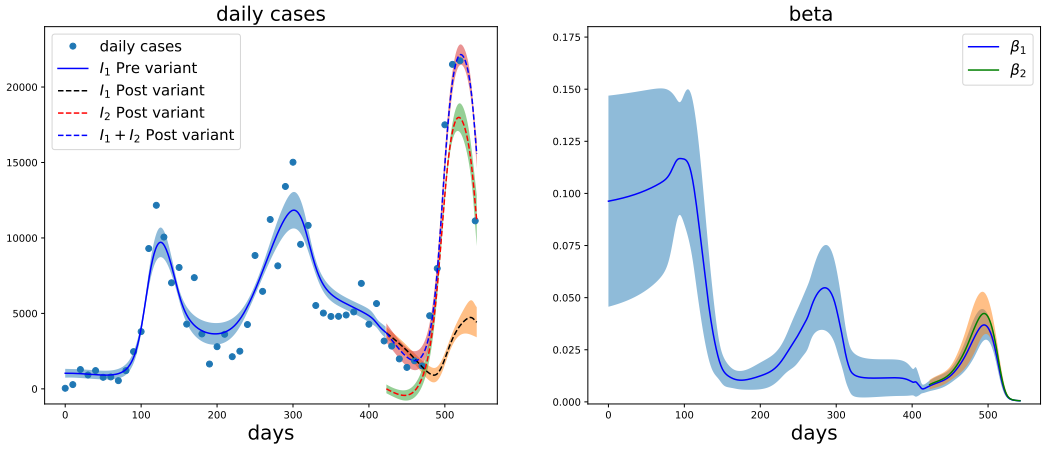


Figure 4.10: Florida daily cases and time-varying transmission rates

5

FORECASTING WITH RECURRENT NEURAL NETWORK AND AN ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM

The most reliable way to forecast the future is to try to understand the present.

John Naisbitt

Forecasting the spread of infectious diseases in many studies are based on multiple linear regression (MLR), ordinary least squares regression (OLSR), principal component regression (PCR) and partial least squares regression (PLSR) and statistical methods such as the Auto Regressive Moving Average (ARIMA) and its many variants [81, 82, 83]. These statistical methods are not optimal for nonlinear predictive task. This has motivated a shift towards techniques that rely on neural networks and neuro-fuzzy models [84]. In this Section, we present an hybrid neural network that combines the simplicity and nonlinear learning capabilities of the Epidemiology-informed neural network (EINN) as well as the fuzzy inference system (ANFIS).

Adaptive neuro-fuzzy inference system (ANFIS), an hybrid neural network itself, is a combination of fuzzy logic and a feedforward neural network. It incorporates the advantages of both methods including learning capabilities, interpretability, quick convergence, adaptability and high accuracy. ANFIS displays excellent performance in approximation and prediction of nonlinear relationships in various fields [85].

5.1 Adaptive Neuro-Fuzzy Inference System (ANFIS)

The Adaptive Neuro-Fuzzy Inference System (ANFIS) was introduced in [86]. It combines a neural network with a fuzzy inference system (FIS) based on "IF-THEN" rules. One major advantage of FIS is that it does not require knowledge of the main physical process as

a pre-condition. ANFIS combines FIS with a backpropagation algorithm. These techniques provide a method for the fuzzy modeling procedure to learn from the available dataset, in order to compute the membership function parameters that best allow the fuzzy inference system to track the given input/output data.

To forecast the transmission of a multi-variant COVID-19, we present an efficient deep learning forecast model which combines two neural networks, we solve the ODE system using an Epidemiology Informed Neural Network (EINN) and we forecast using an adaptive neuro-fuzzy system (ANFIS), which we called the EINN-ANFIS model.

5.2 Performance analysis of error metrics

The following error metrics are used in our data driven simulation:

- Root Mean Square Error (RMSE):

$$RMSE = \sqrt{\frac{1}{N_s}\sum_{i=1}^{N_s}(Y_i - \tilde{Y}_i)^2}$$

,

where Y and $\tilde{Y}$ are the predicted and original values, respectively.

- Mean Absolute Error (MAE):

$$MAE = \frac{1}{N_s}\sum_{i=1}^{N_s}|Y_i - \tilde{Y}_i|$$

.

- Mean Absolute Percentage Error (MAPE):

$$MAPE = \frac{1}{N_s}\sum_{i=1}^{N_s}|\frac{Y_i - \tilde{Y}_i}{Y_i}|$$

.

- Root Mean Squared Relative Error (RMSRE):

$$RMSRE = \sqrt{\frac{1}{N_s}\sum_{i=1}^{N_s}(\frac{Y_i - \tilde{Y}_i}{Y_i})^2}$$

,

N_s represents the sample size of the data.

In Table 5.1 We provide a comparison of error metrics for EINN using random splits for the training and test data.

State	*RMSE*	*MAE*	*MAPE*	*RMSRE*
Florida	0.0076864	0.0868862	0.6259307	1.4576211
Tennessee	0.0100392	0.0795658	1.4157410	3.1137207
Alabama	0.0077280	0.0829354	0.7699850	0.2481321
Missouri	0.0083841	0.0998531	1.6570308	0.4456926

Table 5.1: Error metrics when random split is used to split the training and test data

5.3 Results

We present results of the implementation of ANFIS, EINN-ANFIS, LSTM, EINN-LSTM for COVID-19 data from Alabama, Missouri, Tennessee, and Florida from March 2020 to September 2021. In the ANFIS approach, We used 4 regressors, 12 membership rules, and learning rate of 0.002. Training was done using 300 epochs, where we used the adams optimizer and for the loss function, we used the mean square error. The EINN-ANFIS is a hybrid neural network, where EINN is first used to train the daily cases dataset and a second round of training is done using ANFIS. In the LSTM approach, we used 4 input layers which corresponds to the daily cases at times t, $t+1$, $t+2$, and $t+3$. The adams optimizer is also used in training the LSTM with 20 epochs and the loss function also

uses the mean square error. In the EINN-LSTM approach, a first batch of training is done using the EINN algorithm and then a second batch of training is done using LSTM. In Tables (5.2)–(5.5), we present the validation loss of each method.

Method	Mean	Std
ANFIS	0.00048	0.00098
EINN-ANFIS	0.00032	0.00050
LSTM	0.00141	0.00004
EINN-LSTM	0.00110	0.00006

Table 5.2: Validation loss in the ANFIS, EINN-ANFIS, LSTM, and EINN-LSTM forecasting technique for Alabama daily cases from March 2020 to September 2021.

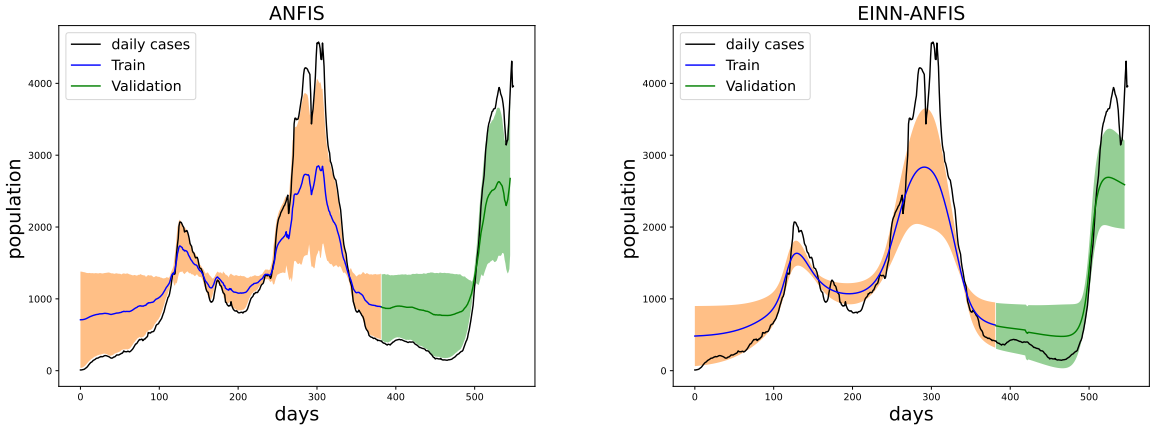
Method	Mean	Std
ANFIS	0.00011	0.00019
EINN-ANFIS	0.00004	0.00006
LSTM	0.00333	0.00003
EINN-LSTM	0.00118	0.00012

Table 5.3: Validation loss in the ANFIS, EINN-ANFIS, LSTM, and EINN-LSTM forecasting technique for Missouri daily cases from March 2020 to September 2021.

Method	Mean	Std
ANFIS	0.00061	0.00125
EINN-ANFIS	0.00033	0.00056
LSTM	0.00267	0.00011
EINN-LSTM	0.00183	0.00009

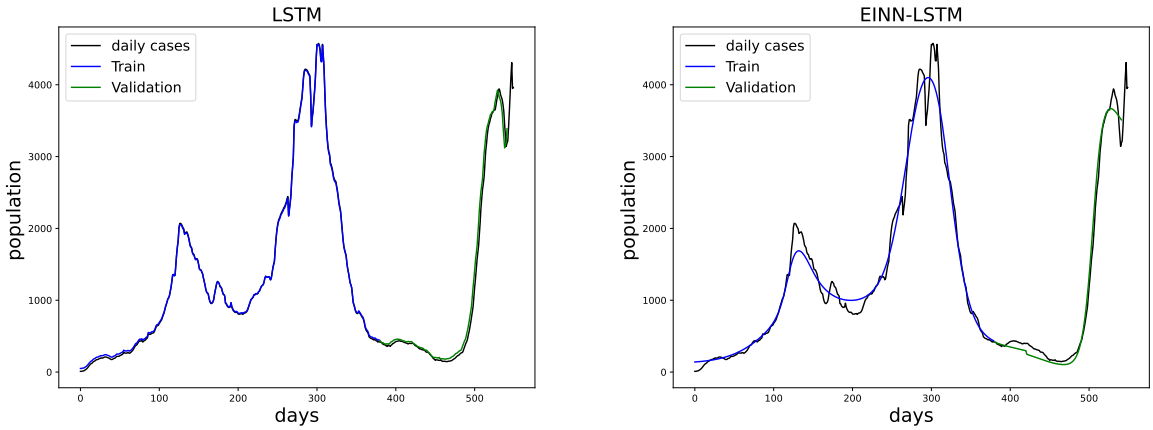
Table 5.4: Validation loss in the ANFIS, EINN-ANFIS, LSTM, and EINN-LSTM forecasting technique for Tennessee daily cases from March 2020 to September 2021.

As can be observed from these Tables (5.2)–(5.5) EINN-ANFIS is an improvement over ANFIS and similarly, EINN-LSTM is an improvement over LSTM.



(a) ANFIS

(b) EINN-ANFIS



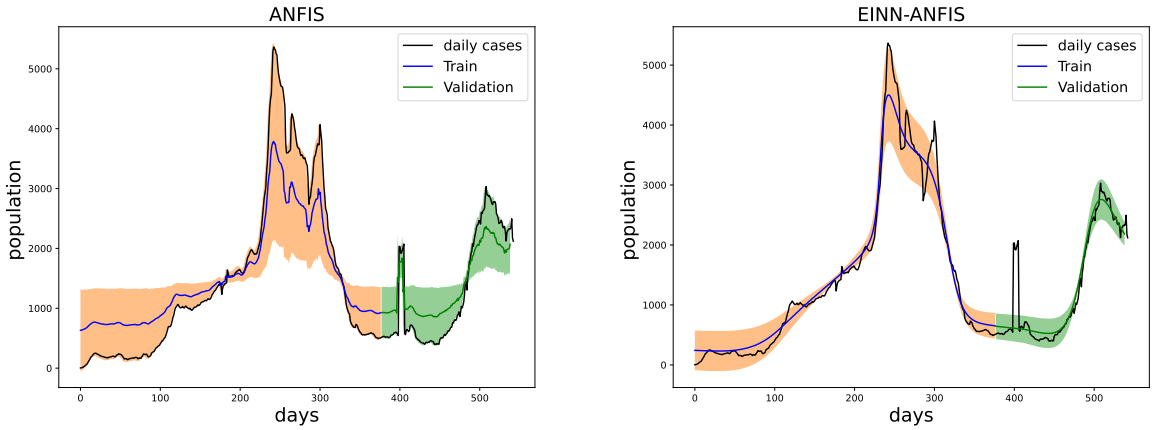
(c) LSTM

(d) EINN-LSTM

Figure 5.1: Alabama daily cases forecasting using ANFIS, EINN-ANFIS, LSTM, EINN-LSTM

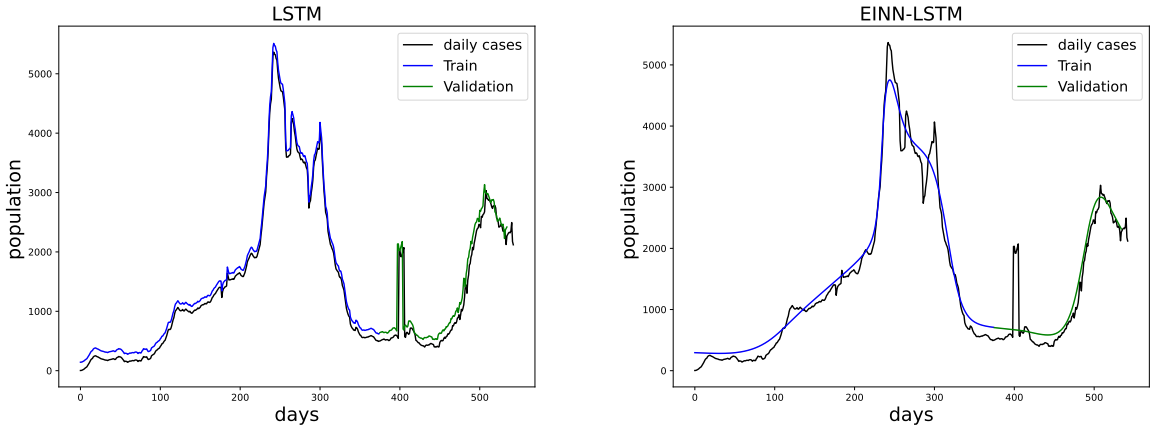
Method	Mean	Std
ANFIS	0.00199	0.00284
EINN-ANFIS	0.00249	0.00347
LSTM	0.00169	0.00009
EINN-LSTM	0.00149	0.00014

Table 5.5: Validation loss in the ANFIS, EINN-ANFIS, LSTM, and EINN-LSTM forecasting technique for Florida daily cases from March 2020 to September 2021.



(a) ANFIS

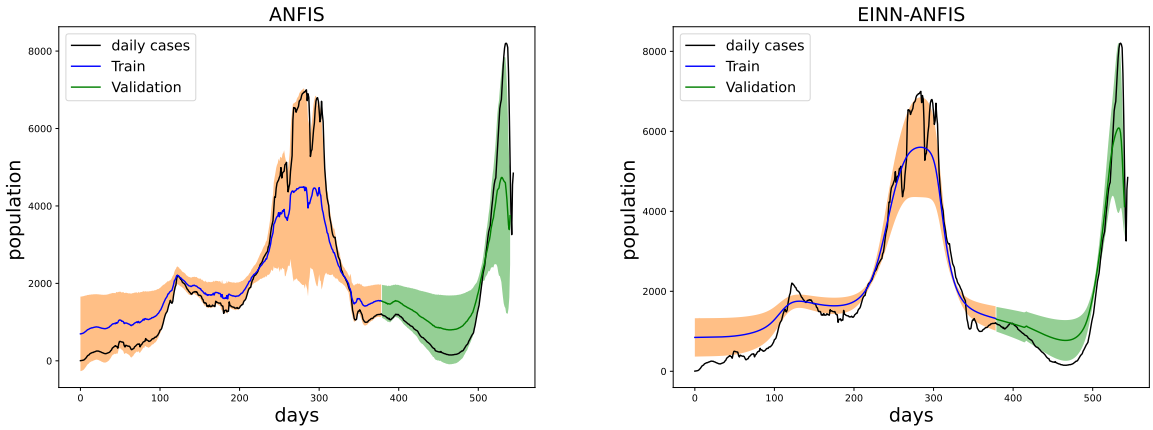
(b) EINN-ANFIS



(c) LSTM

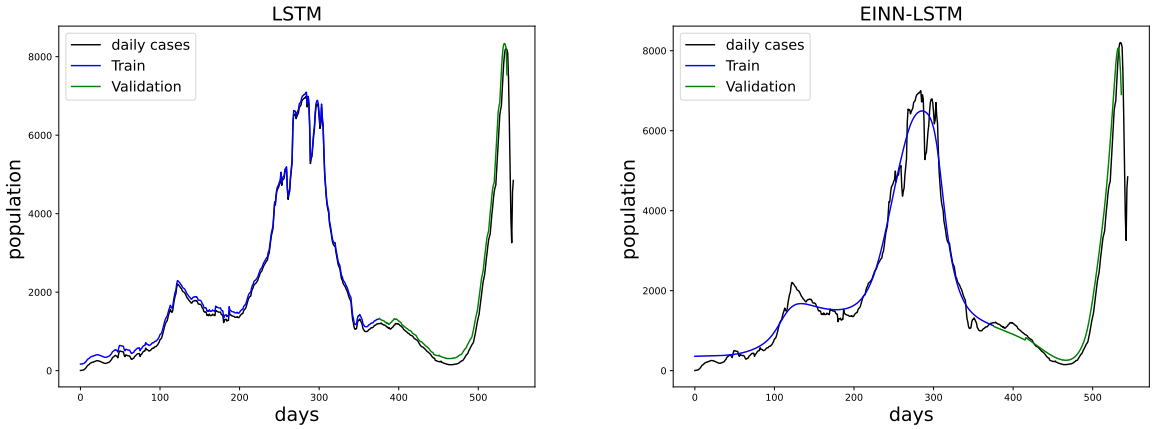
(d) EINN-LSTM

Figure 5.2: Missouri daily cases forecasting using ANFIS, EINN-ANFIS, LSTM, EINN-LSTM



(a) ANFIS

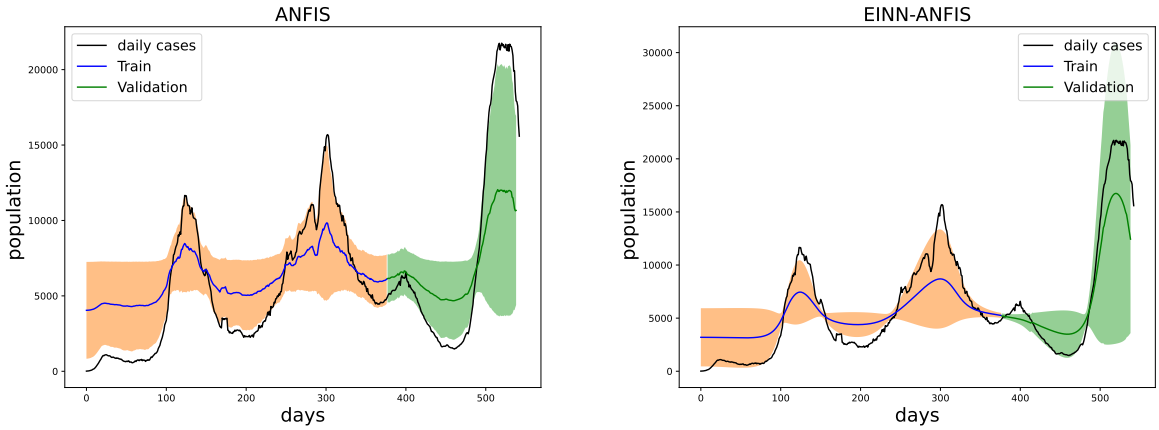
(b) EINN-ANFIS



(c) LSTM

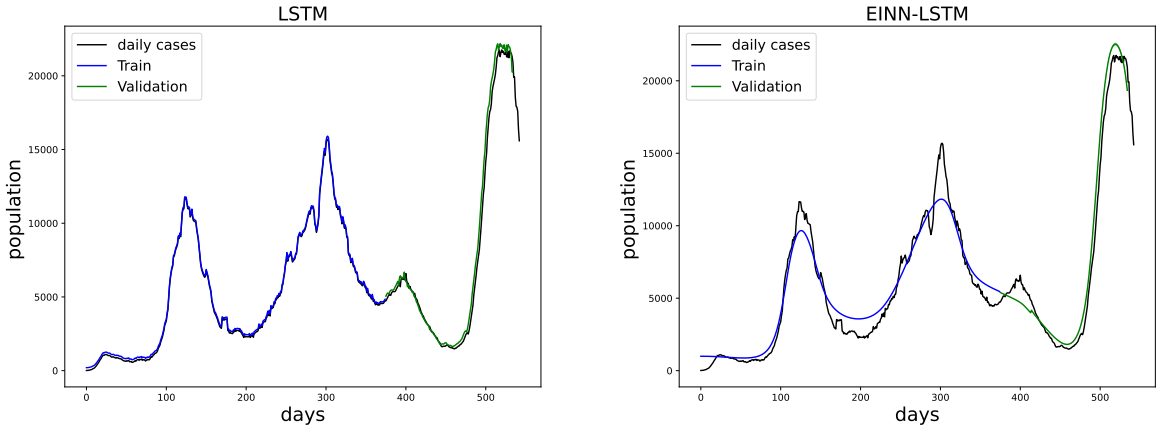
(d) EINN-LSTM

Figure 5.3: Tennessee daily cases forecasting using ANFIS, EINN-ANFIS, LSTM, EINN-LSTM



(a) ANFIS

(b) EINN-ANFIS



(c) LSTM

(d) EINN-LSTM

Figure 5.4: Florida daily cases forecasting using ANFIS, EINN-ANFIS, LSTM, EINN-LSTM

6

LEARNING BIOCHEMICAL MODELS FROM DATA

"All the world's a differential equation, and the men and women are merely variables."

Ben Orlin

6.1 FitzHugh Nagumo (FHN) model

FHN is a simplification of the Hodgkin-Huxley model [87]. It describes interaction between u, membrane potential, which is a voltage-like variable having cubic nonlinearity and v a recovery variable having a linear dynamics that provides a slower negative feedback.

The partial differential equation version of FHN is given below:

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{1}{c}(u - \frac{1}{3}u^3 - v) + \triangle u \\ \frac{\partial v}{\partial t} &= c(u - av + b) \end{aligned} \tag{6.1}$$

The parameters satisfies the following inequalities, $0 < a < 1$, $b > 0$, $c > 0$. An appropriate initial condition is chosen by finding an homogeneous solution of (6.1), which satisfies the ordinary differential equations below:

$$\begin{aligned} \frac{du}{dt} &= \frac{1}{c}(u - \frac{1}{3}u^3 - v) \\ \frac{dv}{dt} &= c(u - av + b) \end{aligned} \tag{6.2}$$

Setting $a = 0.5$, $b = 0.75$, $c = 0.3$, the equilibrium solution of (6.2) is (u_e, v_e) as shown in Figure (6.1).

We consider the following initial conditions $u(x,0)$ and $v(x,0)$ for system (6.1) given as follows:

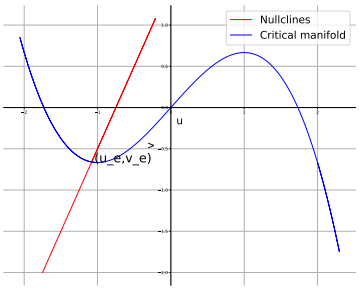


Figure 6.1: Diagram of the nullcline and critical manifold for homgeneous FitzHugh-Nagumo model.

$$u(x,0) = u_0(x), v(x,0) = v_0(x)$$

where,

$$u_0(x) = \begin{cases} u_e & \text{if } x \geq \frac{L}{2} \\ -u_e & \text{if } x < \frac{L}{2} \end{cases}$$

$$v_0(x) = -v_e$$

and $x \in [0, L]$.

6.2 Parameter estimation of FitzHugh Nagumo model

We start by providing a comparison between an optimization technique and the physics-informed neural network (PINN) in a parameter estimation task.

6.2.1 The Nelder-Mead algorithm

This approach is a classical example of how we find parameters (usually constants) in a dynamical system model. Here we used synthetic data, which we obtained by randomization of the numerical solution obtained from an ODE-solver on the FHN model (6.2) and where we took the initial conditions for the membrane potential and the recovery variable to be

$u(0) = 2$, $v(0) = -2$ respectively. The exact model parameters were $a = 0.5$, $b = 0.75$, $c = 0.3$.

This optimization approach is taken from pp 442 in [88], where the Error function is given by

$$E(c,b) = \sum_{i=1}^{n} \{|u(t_i) - u_i|^2 + |v(t_i) - v_i|^2\} \tag{6.3}$$

and where, we take (t_i, u_i) and (t_i, v_i) to be the training data, in our case, the synthetic data, and where $u(t)$ and $v(t)$ are the predicted solutions from our implementation of algorithm 5. It is assumed that we know one of the parameters $(a = 0.5)$, and we used a fitting method to recover the values of the other two parameters c and b. After 56 iterations, we obtained the parameters to be $c = 0.309626$ and $b = 0.7255445$.

Algorithm 5 Parameter Estimation using Nelder-Mead for FHN model (6.2)

1: fix $a = 0.5$ ▷ one of the system parameters
2: using the exact parameters $c = 0.3$ and $b = 0.75$
3: choose a coarse time step $k = 0.25$ and $m =$ max iteration
4: set $z(0) = [2, -2]$ ▷ Initial conditions
5: $[\hat{u}, \hat{v}] = rk4(t, z(0), k, m, c, b)$
6: set $t_{span} = [t_0, t_{final}]$, $dt = t_{i+1} - t_i$,
7: Construct an interpolation function using $\hat{u}, \hat{v}$ to generate u_i, v_i
8: $[u_{pred}, v_{pred}] = rk4(t_{span}, z(0), dt, m, c_{guess}, b_{guess})$
9: $E(c_{guess}, b_{guess}) = \sum_{i=1}^{n} \{|u_{pred}(t_i) - u_i|^2 + |v_{pred}(t_i) - v_i|^2\}$
10: obtain c_{guess}, b_{guess} using the nelder-mead algorithm ▷ go to line 8 until $E(c_{guess}, b_{guess})$ is minimized
11: **return** c_{guess}, b_{guess}

6.2.2 PINN algorithm for FHN model

Parameter identification with the PINN algorithm is an inverse problem. We solve the FHN model (6.2) for the system parameter from data. The goal here is to find the best network parameters λ (lambda represent the network weights and biases) that minimizes the loss function

$$\mathscr{L}(\lambda; m) = \mathscr{L}_r(\lambda; m_r) + \mathscr{L}_b(\lambda; m_b) \tag{6.4}$$

$L_r(\lambda; m_r)$ is called the residual loss and $L_b(\lambda; m_b)$ is the training loss. $m = m_r \cup m_b$ in the total number of training dataset. We define the following:

$$\mathscr{L}_r(\lambda; m_r) = \frac{1}{m_r} \sum_{j=1}^{2} \sum_{i=1}^{m_r} |r_j(t_i)|^2 \tag{6.5}$$

where

$$\begin{aligned} r_1(t) &= \frac{du_{NN}(t)}{dt} - \frac{1}{c}(u_{NN}(t) - \frac{1}{3} v_{NN}(t)^3 - v_{NN}(t)) \\ r_2(t) &= \frac{dv_{NN}(t)}{dt} - c(u_{NN}(t) - a\, v_{NN}(t) + b) \end{aligned} \tag{6.6}$$

and

$$\mathscr{L}_b(\lambda; m_b) = \frac{1}{m_b} \Big\{ \sum_{i=1}^{m_b} |u_{NN}(t_i) - u(t_i)|^2 + \sum_{i=1}^{m_b} |v_{NN}(t_i) - v(t_i)|^2 \Big\} \tag{6.7}$$

Physics-informed neural network only admits t as input, and the outputs are u_{NN} and v_{NN}. The network is trained to minimize (6.4), that is an optimizer such as the Adams has found the most optimal weights and biases that minimizes (6.4). Then the neural network also finds optimal values for the model parameters. In the loss function, $u(t_i)$ and $v(t_i)$ are the training data. We have embedded the physics into the residual in form of the ODE system and the initial conditions is also embedded into the training loss.

Using 3 hidden layers and 32 neurons per layer, after 2000 iterations and using noisy training data(here obtained from an ODE solver), PINN recover the system parameters $a = 0.48684078$, $b = 0.74607641$ and $c = 0.30332789$.

Algorithm 6 PINN algorithm for FHN model (6.2)

1: *construct* a NN ▷ input t, output u_{NN} and v_{NN}

2: *initialize* parameters λ of the NN

3: *specify* the training dataset $m = m_r \cup m_b$ of the NN

4: *specify* a loss function $\mathscr{L}(\lambda, m_r, m_b) = \mathscr{L}(\lambda, m_r) + \mathscr{L}(\lambda, m_b)$

5: where $\mathscr{L}(\lambda, m_r) = \frac{1}{m_r} \sum_{j=1}^{2} \sum_{i=1}^{m_r} |r_j(t_i)|^2$

6: and $\mathscr{L}(\lambda, m_b) = \frac{1}{m_b} \sum_{i=1}^{m_b} |u_{NN}(t_i) - u(t_i)|^2 + \sum_{i=1}^{m_b} |v_{NN}(t_i) - v(t_i)|^2\}$

7: $r_1(t) = \frac{du_{NN}(t)}{dt} - \frac{1}{c}(u_{NN}(t) - \frac{1}{3}u_{NN}(t)^3 - v_{NN}(t))$

8: $r_2(t) = \frac{dv_{NN}(t)}{dt} + c(u_{NN}(t) - a\, v_{NN}(t) + b)$

9: until $\min_m\{\mathscr{L}(\lambda; m)\}$ ▷ obtain optimal λ that minimizes $\mathscr{L}(\lambda; m)$

10: $[u_{NN}, v_{NN}] \to [u, v]$ for m large ▷ u and v are the target

11: The NN also recovers the model parameters a, b, c

6.3 A discrete physics loss function based Neural Network

Physics-informed neural network is generally limited to low-dimensional parameter identification. It has not performed well in modeling PDEs with sharp gradients, and it has become necessary to use collocation points in the PDE residuals. This results in a huge computational cost in training. However, in a true sense of learning PDEs and ODEs from data, we require approaches that do not assume that the form of the PDE is known. The following works have succeeded in modeling PDEs from data, where the form of the PDE is not assumed to be known [89] and [90]. In Section (6.3), we propose a forward Euler method using an attention mechanism and we will demonstrate that this neural network is an improvement over a feedforward neural network

We build the neural network as follows:

1. Formulate a neural network, This network is used to learn the dynamics of the PDE/ODE from a data.

2. Imposition of initial and boundary conditions so that we can have a well-posed opti-

mization problem during the training.

3. A residual PDE connection inspired by the forward-Euler scheme will be used to construct the loss function. If we seek to improve the network, we will have to use higher-order temporal schemes to design the loss function. The use of higher-order temporal schemes will require changes in the network architecture. In Section (6.3), we want to demonstrate that a loss function built using a forward Euler temporal scheme in the network formulation is capable to learn the form of a PDE/ODE from data to high accuracy.

We consider the general form of a multi-dimensional, nonlinear, coupled PDE system of the form

$$\mathbf{u}_t(x,t) = \mathscr{F}[\mathbf{u}, \mathbf{u}^2, \ldots, \mathbf{u}_x, \mathbf{u}_{xx}, \ldots; \lambda] \tag{6.8}$$

where $\mathbf{u}(x,t)$ is the physical quantity we want to study. The temporal domain is $t \in [0,T]$ and the spatial domain is $x \in \Omega \subset \mathbf{R}$. $\mathbf{u}_t(x,t)$ is the first-order time derivative term and $\mathbf{u}_x$ denotes the gradient operator with respect to the spatial variable x. $\mathscr{F}[\cdot]$ is the nonlinear functional parameterized by λ. Eq (6.8) satisfies appropriate initial and boundary conditions.

6.3.1 ForwardEulerNet

We will learn the FitzHugh-Nagumo system (FHN) [91, 92] from data. We consider (6.2) in 1 spatial dimensions, and $t > 0$.Here, we assume only a temporal domain $t \in [0,T]$, and $\mathbf{u}$ denotes the variables (u,v). In this approach, we do not assume the form of the right hand side of (6.2). Suppose $\hat{\mathbf{u}}$ is the target variable for which there is a known measurements, we can formulate a neural network that take as input; $\hat{\mathbf{u}}$, and the temporal domain $t \in [0,T]$. The output of this neural network is the learned right hand side of (6.2) denoted as $\mathscr{F}$. This neural network is a feedforward neural network(6.2).

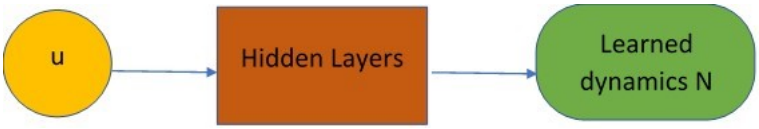


Figure 6.2: Schematic of ForwardEulerNet

The loss function can then be formulated as follows

$$\mathscr{R}(t;\theta) = \hat{\mathbf{u}}_t - \mathscr{F}[t,\hat{\mathbf{u}},\ldots;\lambda,\theta] \tag{6.9}$$

A discrete loss is used in training the neural network, where the discrete $\mathscr{F}$ is given by $\mathscr{N}$. The discrete loss function is inspired by the forward Euler method to connect the temporal state $\hat{\mathbf{u}}_i$ and temporal state $\hat{\mathbf{u}}_{i+1}$.

$$\hat{\mathbf{u}}_{i+1} = \hat{\mathbf{u}}_i + \partial t \cdot \mathscr{N}[\hat{\mathbf{u}};\theta], \quad \partial t = \frac{T}{K}, K \in \mathbf{R}^+ \tag{6.10}$$

The trainable parameters θ are optimized during training of the forward-Euler based neural network that minimizes the mean squared error loss function (MSE) given as follows

$$MSE = \sum_{i=1}^{K} |-\hat{\mathbf{u}}_{i+1} + \hat{\mathbf{u}}_i + \partial t \cdot \mathscr{N}[\hat{\mathbf{u}};\theta]|^2, \quad \partial t = \frac{T}{K}, K \in \mathbf{R}^+ \tag{6.11}$$

When training is done, we obtain $\mathscr{N}$, which is the learned dynamics of an ODE system such as eq (6.2). Now to obtain learned $\mathbf{u}$ denoted as $\mathbf{u}^{pred}$, we use an ODE solver such. Using a solver such as the scipy odeint. We observe that the learned model parameters $\lambda = [a,b,c]$ in eq (6.2) are embedded into the dynamics $\mathscr{N}$.

$$\mathbf{u}^{pred} = odeint(\mathscr{N}, (u(0), v(0)), tspan)$$

In Figures (6.3)–(6.8), we obtained $\mathbf{u}^{pred}$ and compared with the target $\hat{\mathbf{u}}$. In Figure (6.8), we observe that when we add 5% noise to the training data, we are no longer able to learn an accurate dynamics of (6.2) from data.

When tuning the hyperparameters of a neural network, we observe in Table (6.1) that

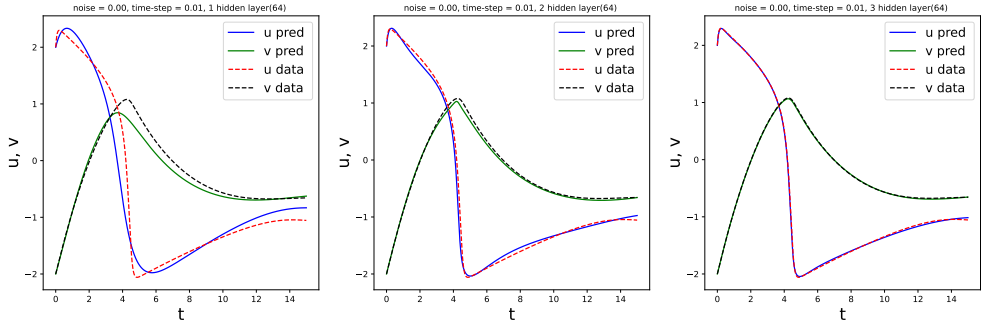


Figure 6.3: No noise, $\triangle t = 0.01$, 64 neurons

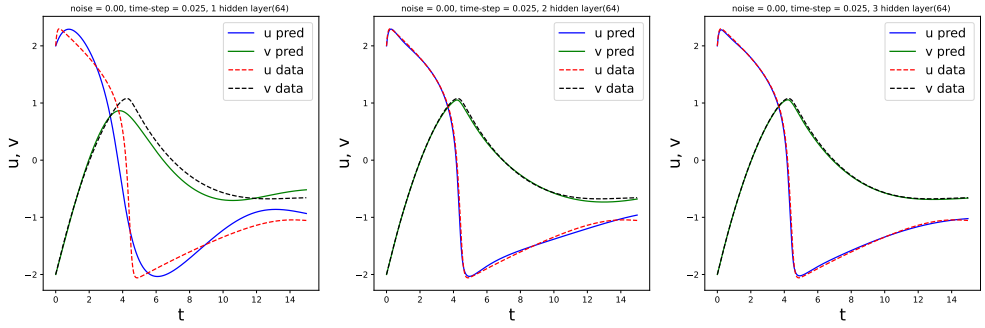


Figure 6.4: No noise, $\triangle t = 0.025$, 64 neurons

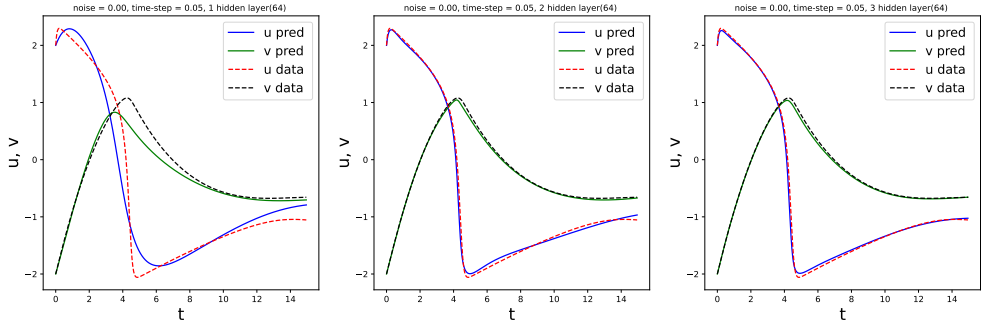


Figure 6.5: No noise, $\triangle t = 0.05$, 64 neurons

Hidden layers	32 Neurons	64 Neurons	128 Neurons	256 Neurons
1	7.6e-03	4.3e-03	1.3e-03	2.3e-03
2	4.8e-03	3.0e-03	2.3e-03	3.53-03
3	4.9e-03	4.4e-03	2.8e-03	3.3e-03

Table 6.1: Table showing MSE error of the variable u, where we assumed a noise free training data and set the time step to 0.01.

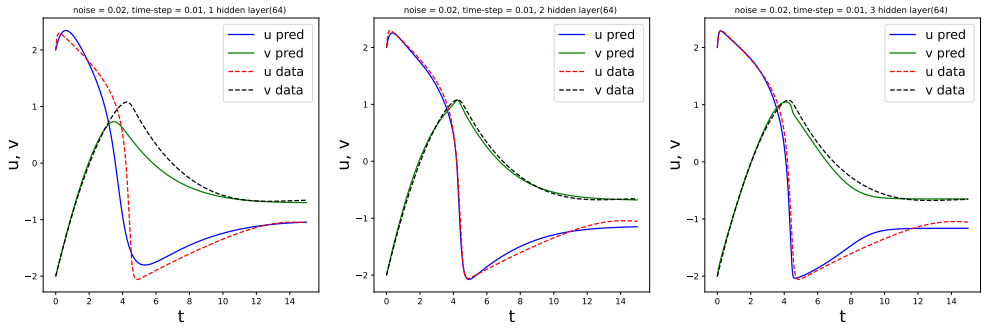


Figure 6.6: 2% noise, $\triangle t = 0.01$, 64 neurons

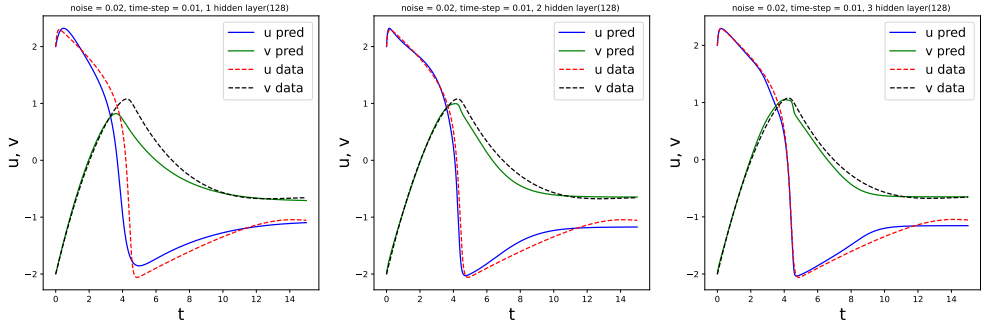


Figure 6.7: 2% noise, $\triangle t = 0.01$, 128 neurons

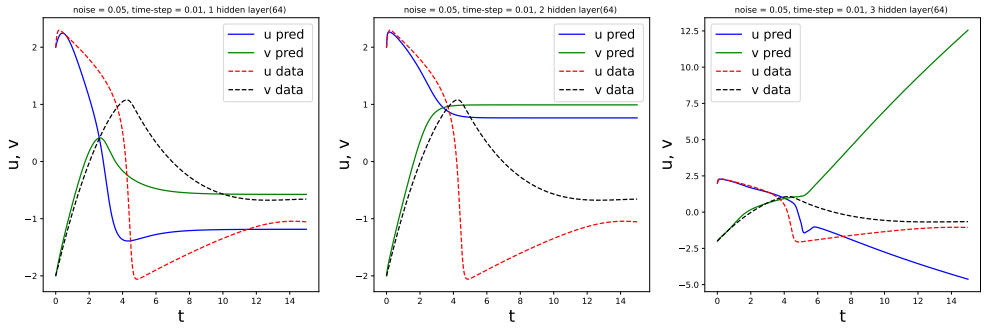


Figure 6.8: 5% noise, $\triangle t = 0.01$, 64 neurons

it is more beneficial to increase the width (neurons per layer) before increasing the depth (number of hidden layers).

6.3.2 PhyAttNet

Due to ForwardEulerNet's failure to learn the dynamics in (6.2) when there is a significant amount of noise in the training data, we propose a modification to ForwardEulerNet, where we replace the feedforward neural network in (6.2) with a scaled Dot-Product attention mechanism [33], we call this new neural network PhyAttNet. We observe that in the case of no noise, ForwardEulerNet performs better than PhyAttNet, however, when we add 5% noise to the training data, PhyAttNet is able to learn the dynamics in (6.2) much better than ForwardEulerNet, see Figures (6.9),(6.10), when we increase the neural network depth as demonstrated in the mean square errors presented in Tables (6.2)–(6.3).

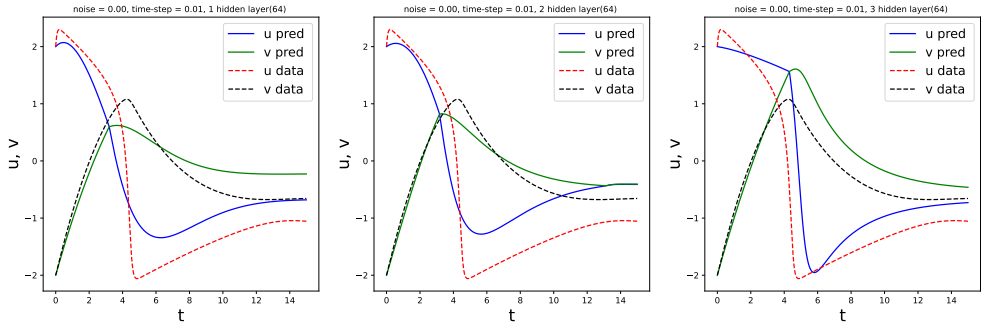


Figure 6.9: No noise, $\triangle t = 0.01$, 64 neurons using an attention network

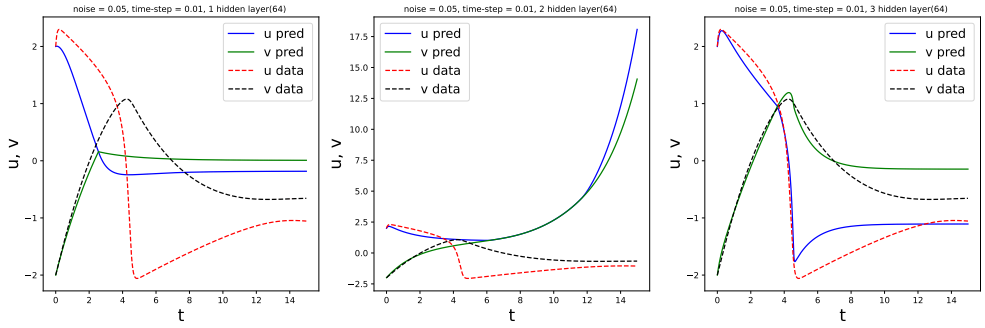


Figure 6.10: 5% noise, $\triangle t = 0.01$, 64 neurons using an attention network

Neural Network	u_{1h}	u_{2h}	u_{3h}
ForwardEulerNet	0.08593	0.00837	0.00036
PhyAttNet	0.25513	0.38623	0.39366

Table 6.2: MSE of ForwardEulerNet and PhyAttNet, for 0% noise at different neural network depth

Neural Network	u_{1h}	u_{2h}	u_{3h}
ForwardEulerNet	0.52710	3.53157	2.75154
PhyAttNet	1.44995	35.26668	0.09301

Table 6.3: MSE of ForwardEulerNet and PhyAttNet, for 5% noise at different neural network depth

6.4 Learning nonlinear biochemical parameters using a modified PINN

It has been widely observed that the physics-informed neural network is capable of learning approximate solutions to PDEs and ODE from data. We demonstrate in this section that in the case of a noisy data, the model parameters in (6.1) are nonlinear functions. We implement a modified PINN algorithm see Figure (6.11) with $x \in \Omega = [0, L]$, where $L > 0$.

We minimizes the following loss function

$$\mathscr{L}(\lambda; m) = \mathscr{L}_r(\lambda; m_r) + \mathscr{L}_b(\lambda; m_b) \tag{6.12}$$

where,

$$\mathscr{L}_r(\lambda; m_r) = \frac{1}{m_r} \sum_{k=1}^{2} \sum_{i=1}^{m_r} \sum_{j=1}^{m_r} |r_k(x_i, t_j)|^2 \tag{6.13}$$

and

$$\begin{aligned} r_1(x,t) &= \frac{\partial u_{NN}(x,t)}{\partial t} - \frac{1}{c}(u_{NN}(x,t) - \frac{1}{3} v_{NN}(x,t)^3 - v_{NN}(x,t)) \\ r_2(x,t) &= \frac{\partial v_{NN}(x,t)}{\partial t} - c(u_{NN}(x,t) - a\, v_{NN}(x,t) + b). \end{aligned} \tag{6.14}$$

The data loss is specified as follows:

$$\mathscr{L}_b(\lambda; m_b) = \frac{1}{m_b}\Big\{ \sum_{i=1}^{m_b}\sum_{j=1}^{m_b} |u_{NN}(x_i,t_j) - u(x_i,t_j)|^2 + \sum_{i=1}^{m_b}\sum_{j=1}^{m_b} |v_{NN}(x_i,t_j) - v(x_i,t_j)|^2 \Big\} \tag{6.15}$$

The modified Physics-informed neural network in Figure (6.11) admits x, and t as input, and the outputs are u_{NN} and v_{NN}. However, since we want to learn nonlinear biochemical functions $a(x,t)$, $b(x,t)$, and $c(x,t)$ from data, we construct 3 more feedforward neural networks that takes as input u_{NN} and v_{NN} and outputs a, b, and c respectively. This modified network is trained to minimize equation (6.12).

In figure (6.12), we present a comparison between solution obtained by PINN and the modified PINN. In Figure (6.13), we present the solution obtained by the modified PINN and the nonlinear parameters learned from noisy data. Here the training data is obtained by solving (6.1) by a finite difference method and where we have used Neumann boundary conditions.

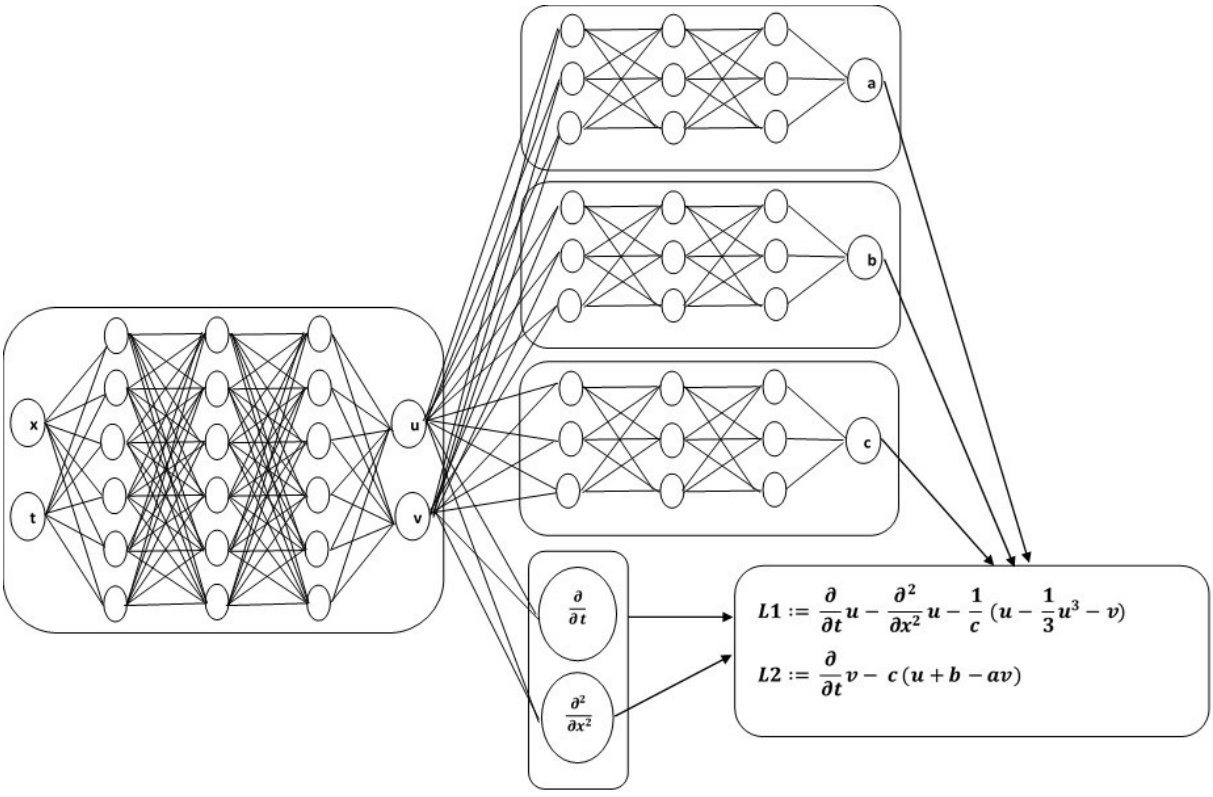


Figure 6.11: A modified PINN

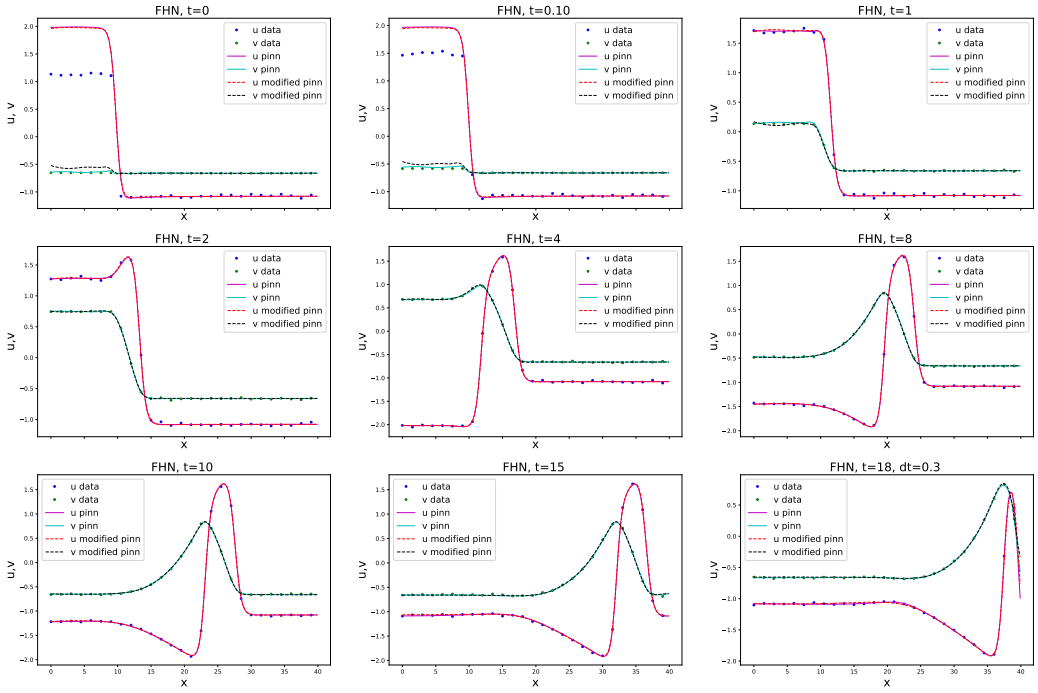


Figure 6.12: FHN at different time step

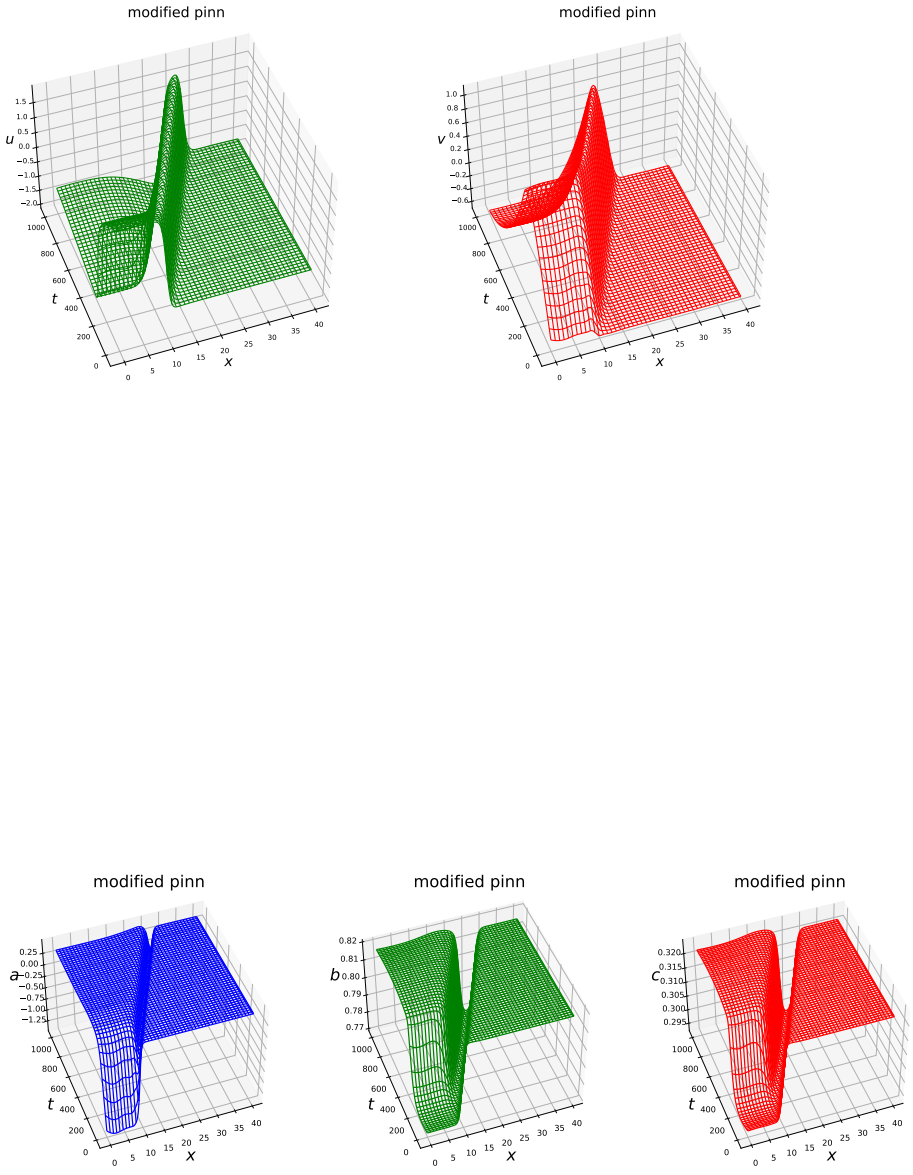


Figure 6.13: 2% noisy data, nonlinear parameters

7

CONCLUSION

"Oh falcon, do not be scared of this gusts of wind hitting at you, they are only meant to take you higher"

Muhammad Iqbal

In this dissertation, we set out to discover hidden patterns in spatio-temporal data that have been observed to be reaction-diffusion systems. The data-driven deep learning algorithms we present do not make prior assumptions about the system parameters, instead, we learn these parameters from data. Our approach discovers more detailed governing equations from data for biochemical and epidemiological models. To the best of our knowledge, this is the first work where data-driven deep learning algorithms have been used to discover time-dependent system parameters in reaction-diffusion systems and in Differential equation models. We have also presented deep learning algorithms that have the capability to learn heterogeneous time-dependent transmission rates due to the presence of multiple variants/strain of an infectious disease outbreak. In order to ameliorate the challenges that arise when performing forecasting task with non-smooth data, we present hybrid deep learning approaches that improves the forecast capabilities of recurrent neural networks and an adaptive neuro-fuzzy inference system.

Bibliography

[1] S. Hawkings. *The Illustrated On the Shoulders of Giants: The Great Works of Physics and Astronomy*. Running Press, 2004.

[2] J. Fourier. *Theorie Analytique de la Chaleur*. Firmin Didot (reissued by Cambridge University Press), 1822.

[3] A. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3:210–229, 1959.

[4] T.M. Mitchell. *Machine learning*. McGraw-Hill, New York, 1997.

[5] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, 2016.

[6] S.L. Brunton, J.L. Proctor, and J.N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113:3932 – 3937, 2016.

[7] S.H. Rudy, S.L. Brunton, J.L. Proctor, and J.N. Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3:e1602614, 2017.

[8] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Inferring solutions of differential equations using noisy multi-fidelity data. *Journal of Computational Physics*, 335:736–746, 2017.

[9] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[10] Y. LeCun, Y. Bengio, and G.E. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[11] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. *Parallel Distributed Processing*. MIT Press, 1987.

[12] W. Liang, J. Yao, A. Chen, and et al. Early triage of critically ill COVID-19 patients using deep learning. *Nature Communications*, 11:3543, 2020.

[13] G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathematics of control, signals and systems*, 1989.

[14] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[15] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861 – 867, 1993.

[16] B. Hanin. Universal function approximation by deep neural nets with bounded width and relu activations. *arXiv:1708.02691*, 2017.

[17] P. Kidger and T. Lyons. Universal approximation with deep narrow networks. *Conference on Learning Theory*, 2020.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv:1512.03385v1 [cs.CV]*, 2015.

[19] I. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015.

[20] R.K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv:1505.00387*, 2015.

[21] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural network. *AISTATS*, 2010.

[22] E. Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematical Science*, 5:1 – 11, 2017.

[23] E. Haber and L. Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 2017.

[24] L. Ruthotto and E. Haber. Deep neural networks motivated by partial differential equations. *Preprint arXiv:1804.04272v2*, 2018.

[25] E. Haber, K. Lensink, E. Treister, and L. Ruthotto. Imexnet- a forward stable deep neural. *Preprint arXiv:1903.02639v2*, 2019.

[26] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

[27] L.S. Pontryagin, V.G. Boltyanskii, r.V. Gamkrelidze, and E.F. Mischenko. The mathematical theory of optimal processes. *Wiley Interscience*, 1962.

[28] N. Shazeer, A. Mirhoseini, A. Maziarz, K. Davis, Q. Le, and G Hinton. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv:1701.06538*, 2017.

[29] O. Kuchaiev and B. Ginsburg. Factorization tricks for LSTM networks. *arXiv:1703.10722*, 2017.

[30] J. Long, A.Q.M Khaliq, and K.M. Furati. Identification and prediction of time-varying parameters of COVID-19 model: a data-driven deep learning approach. *International Journal of Computer Mathematics*, 98:1617–1632, 2021.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv:1706.03762v5*, 2017.

[32] K. Cho, B. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *CoRR*, page abs/1406.1078, 2014.

[33] A. Géron. *Hands-on machine learning with scikit-learn, keras & TensorFlow*. O'Reilly, 2019.

[34] K. Yeo and I. Melnyk. Deep learning algorithm for data-driven simulation of noisy dynamical system. *Journal of Computational Physics*, 376:1212 – 1231, 2019.

[35] J. Sirignano and K. Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339 – 1364, 2018.

[36] I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *arXiv:physics/9705023v1*, 1997.

[37] M. Raissi, P. Perdikaris, , and G. E. Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *Preprint arXiv:1801.0126v1*, 2018.

[38] W. Jin, E. T. Shah, C. J. Pennington, S. W. McCue, L. K. Chopin, and M. J. Simpson. Reproducibility of sratch assays is affected by the initial degree of confluence: Experiments, modelling and model selection. *Journal of Theoretical Biology*, 390:136–145, 2016.

[39] J. H. Lagergren, J. T. Nardini, R. E. Baker, M. J. Simpson, and K. B Flores. Biologically-informed neural networks guide mechanistic modeling from sparse experimental data. *PLoS Computational Biology*, 16(12):e1008462, 2020.

[40] Y. Shin, J. Darbon, and G. E. Karniadakis. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes. *Preprint arXiv:2004.01806v2*, 2020.

[41] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representation*, 2015.

[42] R.H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16:1190 – 1208, 1995.

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

[44] M. Raissi, A. Yazdani, and G. E. Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367:1026–1030, 2020.

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

[46] E. Kharazmi, M. Cai, X. Zheng, G. Lin, and G. E. Karniadakis. Identifiability and predictability of integer- and fractional-order epidemiological models using physics-informed neural networks. *medRxiv*, 2021.

[47] W.O. Kermack and A.G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Soceity A: Mathematical, Physical and Engineering Sciences*, 115:700 – 721, 1921.

[48] Z. Liu, P. Magal, O. Seydi, and G. Webb. Predicting the cumulative number of cases for the covid-19 epidemic in china from early data. *medRxiv preprint*, 2020.

[49] Q. Lin, S. Zhao, D. Gao, Y. Lou, S. Yang, S.S. Musa, M.H. Wang, Y. Cai, W. Wang, L. Yang, and D. He. A conceptual model for the coronavirus disease 2019 (COVID-19) outbreak in Wuhan, China with individual reaction and governmental action. *International journal of Infectious diseases*, 93:211–216, 2020.

[50] S. E. Eikenberry, M. Mancuso, E. Iboi, T. Phan, K. Eikenberry, Y. Kuang, E. Kostelich, and A. B. Gummel. To mask or not to mask: Modeling the potential for the face mask use by the general public to curtail the COVID-19 pandemic. *Infectious Disease Modelling*, 5:293–308, 2020.

[51] K-M Tam, N Walker, and J Moreno. Effect of mitigation measures on the spreading of COVID-19 in hard-hit states in the U.S. *PLoS Computational Biology*, 15(11):e0240877, 2020.

[52] Z. Liu, P. Magal, O. Seydi, and G. Webb. Understanding unreported cases in the COVID-19 epidemic outbreak in Wuhan, China, and the importance of major public health interventions. *biology*, 9(3), 2020.

[53] J. Kaplan, L. Frias, and M. McFall-Johnsen. Countries around the world are reopening - here's our list of how they're doing it and who remains under lockdown. *Business Insider*, 2020.

[54] L.M. Stolerman, D. Coombs, and S. Boatto. SIR-Network model and its application to dengue fever. *SIAM Journal on Applied Mathematics*, 75(6):2581–2609, 2015.

[55] P. Magal, G. Webb, and Y. Wu. On the basic reproduction number of reaction-diffusion epidemic models. *SIAM Journal on Applied Mathematics*, 79(1):284–304, 2019.

[56] G. Gaeta. A simple SIR model with a large set of asymptomatic infectives. *Mathematics in Engineering*, 3(2):1–39, 2021.

[57] Z. Liu, P. Magal, O. Seydi, and G. Webb. Predicting the cumulative number of cases for the COVID-19 epidemic in China from early data. *Mathematical Biosciences and Engineering*, 17(4):3040–3051, 2020.

[58] L. Magri and N. A. K. Doan. First-principles machine learning modelling of COVID-19. *arXiv preprint*, 2020.

[59] X. He, E. H. Y. Lau, P. Wu, X. Deng, J. Wang, X. Hao, Y. C. Lau, J. Y. Wong, Y. Guan, X. Tan, X. Mo, Y. Chen, B. Liao, W. Chen, F. Hu, Q. Zhang, M. Zhong, Y. Wu, L. Zhao, F. Zhang, B. J. Cowling, F. Li, and G. M. Leung. Temporal dynamics in viral shedding and transmissibility of COVID-19. *Nature Medicine*, 26:672–675, 2020.

[60] Centers for Disease Control and Prevention (CDC). COVID-19 pandemic planning scenarios. https://www.cdc.gov/coronavirus/2019-ncov/hcp/planning-scenarios.html, Accessed: 2020-12-11.

[61] Q.-X. Long, X.-J. Tang, Q.-L. Shi, Q. Li, H.-J. Deng, J. Yuan, J.L. Hu, W. Xu, Y. Zhang, and F.J. Lv. Clinical and immunological assessment of asymtomatic sars-cov-2 infections. *Nat. Med.*, 26:1200–1204, 2020.

[62] D.P. Oran and E.J. Topol. The proportion of sars-cov-2 infections that are asymptomatic. *Ann. Intern. Med.*, 174:655–662, 2021.

[63] E. Dong, H. Du, and L. Gardner. An interactive web-based dashboard to track COVID-19 in real time. *Lancet Infect. Dis.*, 20:533–534, 2020.

[64] P. Driessche and J. Watmough. Reproduction numbers and sub-threshold endemic equilibria for compartmental models of disease transmission. *Mathematical Biosciences*, 180:29–48, 2002.

[65] T. K. Torku, A. Q. M. Khaliq, and K. M. Furati. Deep-data-driven neural networks for covid-19 vaccine efficacy. *Epidemiologia*, 2(4):564–586, 2021.

[66] F. Chollet. *Deep Learning with Python*. Manning, 2017.

[67] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as gaussian processes. *arxiv:1711.00165*, 2017.

[68] M. Jagan, M. S. deJonge, O. Krylova, and D. J.D. Earn. Fast estimation of time-varying infectious disease transmission rates. *PLoS Computational Biology*, 16(9):e1008124, 2020.

[69] D. He, J. Dushoff, T. Day, J. Ma, and D.J.D. Earn. Inferring the causes of the three waves of the 1918 influenza pandemic in england and wales. *Proc. R. Soc.*, 280:20131345, 2013.

[70] B. Tepekule, A. Hauser, V.N. Kachalov, S. Andressen, T. Scheier, P.W. Schreiber, and et al. Assessing the potential impact of transmission during prolonged viral shedding on the effect of lockdown relaxation on COVID-19. *PLoS Computational Biology*, 17(1):e1008609, 2021.

[71] D.P. Kingma and J.L. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980[cs.LG]*, 2017.

[72] K. D. Olumoyin, A. Q. M. Khaliq, and K. M. Furati. Data-driven deep learning algorithms for time-varying infection rates of covid-19 and mitigation measures. *arXiv*, 2021.

[73] G Gaeta. Social distancing versus early detection and contacts tracing in epidemic management. *Chaos, Solitons, & Fractals*, 140:110074, 2020.

[74] World Health Organization (WHO). Archived: WHO Timeline-COVID-19. https://www.who.int/news/item/27-04-2020-who-timeline--covid-19, Accessed: 2021-08-12.

[75] E. Callaway. Making sense of coronavirus mutaions. *Nature*, 585:174–177, 2020.

[76] Centers for Disease Control and Prevention (CDC). Delta variant: What we know about the science. https://www.cdc.gov/coronavirus/2019-ncov/variants/delta-variant.html, Accessed: 2021-08-12.

[77] K.D. Olumoyin, A.Q.M. Khaliq, and K.M. Furati. Data-driven deep-learning algorithm for asymptomatic covid-19 model with varying mitigation measures and transmission rate. *Epidemiologia*, 2:471–489, 2021.

[78] Centers for Disease Control and Prevention (CDC). Variant proportions. https://covid.cdc.gov/covid-data-tracker/#variant-proportions, Accessed: 2021-08-20.

[79] K.M. Furati, I.O. Sarumi, and A.Q.M. Khaliq. Fractional model for the spread of COVID-19 subject to governmental intervention and public perception. *Applied Mathematical Modelling*, 95:89–105, 2021.

[80] D.G. Schaeffer and J.W. Cain. *Ordinary Differential Equations: Basics and beyond*. Springer, New York, 2016.

[81] M. Eftekhari, A. Yadollahi, A. Shojaeiyan, and M. Ayyari. Development of an artificial neural network as a tool for predicting the targeted phenolic profile of grapevine Vitis vinifera foliar wastes. *Frontiers in plant science*, page 9:837, 2018.

[82] C. Du, J. Wei, S. Wang, and Z. Jia. Genomic selection using principal component regression. *Heredity*, 121(1):12–23, 2018.

[83] V.K.R. Chimula and L. Zhang. Time series forecasting of COVID-19 transmission in canada using lstm networks. *Chaos Solitons Fractals*, 135:109864, 2020.

[84] M. Hossain, S. Mekhilef, F. Afifi, L.M. Halabi, L. Olatomiwa, M. Seyedmahmoudian, B. Horan, and A. Stojcevski. Application of the hybrid ANFIS models for long term wind power density prediction with extrapolation capability. *PLoSONE*, 13(4):e0193772, 2018.

[85] A.P. Vacilopoulos and R. Bedi. Adaptive neuro-fuzzy inference system in modelling fatigue life of multidirectional composite laminates. *Computational Materials Science*, 43(4):1086–1093, 2008.

[86] J-S R. Jang. Anfis: adaptive-network-based fuzzy inference system. *IEEE*, 23:665–685, 1993.

[87] A.L. Hodgekin and A.F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol*, 117:500 – 544, 1952.

[88] M. H. Holmes. *Introduction to scientific computing and data analysis*. Springer International Publishing, Switzerland, 2016.

[89] N. Geneva and N. Zabaras. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056, 2020.

[90] P. Ren, C. Rao, Y. Liu, J-X. Wang, and H. Sun. Phycrnet: Physics-informed convolutional-recurrent network for solving spatiotemporal pdes. *arXiv:2106.14103v1 [cs.LG]*, 2021.

[91] R FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical*, 1:445 – 466, 1961.

[92] J Nagumo. An active pulse transmission line simulating nerve axon. *Proc. IRE*, 50:2061 – 2070, 1962.

[93] N.L. Carothers. *Real Analysis*. Cambridge University Press, Cambridge, 2000.

[94] S. Safaei, V. Safaei, S. Safaei, Z. Woods, H. R. Arabnia, and J. B. Gutierrez. The swag algorithm; a mathematical approach that outperforms traditional deep learning. theory and implementation. *Preprint arXiv:1811.11813v1*, 2018.

[95] P. Kidger and T. Lyons. Universal approximation with deep narrow networks. *Conference on Learning Theory*, 2020.

Appendices

A.1 Weierstrass Approximation Theorem

This is a theorem from Real Analysis for a space of continuous functions-a collection of functions that have some 'nice properties.' For instance, if we consider a compact set X, we can define $C(X)$ to be a space of continuous functions. With these 'nice properties' on $C(X)$, the Weierstrass approximation theorem gives us a framework for a polynomial function that approximates any function in $C(X)$. In our case, we choose X to the the abstract set $[a,b]$.

Theorem A.1.1. *[93] Given $f \in C[a,b]$ and $\varepsilon > 0$, there is a polynomial p such that $||f-p||_\infty < \varepsilon$. Hence, there is a sequence of polynomials (p_n) such that p_n converges uniformly to f on $[a,b]$.*

In practice, an example of a sequence of polynomials used to approximate $f \in C(X)$ is the **Bernstein polynomials**. To see how this works, first we state the following lemma without proof

Lemma A.1.2. *There is a linear isometry from $C[0,1]$ onto $C[a,b]$ that maps polynomials to polynomials.*

let $f \in C[0,1]$, we can define the sequence $(B_n(f))_{n=1}^\infty$ of **Bernstein polynomials** for f by

$$(B_n(f))(x) = \sum_{k=0}^{n} f(\frac{k}{n}) \cdot \binom{n}{k} x^k (1-x)^{n-k}, \qquad 0 \leq x \leq 1 \tag{1}$$

The notion of approximating any continuous function by polynomial functions looks promising for the development of a neural network. However, it was proved in [15] that polynomials are not universal approximators.

Theorem A.1.3. *[15] Let M be the set of functions which are $L^\infty_{loc}(\mathscr{R})$ with property that the closure of the set points of discontinuity of any function in M has zero lebesgue measure. Let $\sigma \in M$. Then for a fixed $x \in \mathscr{R}^n$,*

$$span\{\sigma\{w \cdot x + b\} : w \in \mathscr{R}^n, b \in \mathscr{R}\} \tag{2}$$

is dense in $C(\mathscr{R}^n)$ if and only if σ is not an algebraic polynomial $(a.e.)$

Although, in [94], we see a framework where polynomials can be used as an activation function in the construction of a universal approximator. However, one of the reasons neural networks are widely used today is due in part to the use of non-polynomial nonlinear functions, namely the sigmoid function, hyperbolic tangent function, and Rectified Linear Unit.

B.2 Universal Approximation Theorems

In [13], we have one of the earliest universal approximation theorem for neural network. It describes a two layered neural network composed with a sigmoid function.

Theorem B.2.1. *[13] Let σ be any continuous discriminatory function. Then finite sums of the form*

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(w_j^T x + b_j) \tag{3}$$

are dense in $C(\mathbb{R}^n)$. In other words, given any $f \in C(\mathbb{R}^n)$ and $\varepsilon > 0$ there is a sum $G(x)$ of the above form, for which

$$|G(x) - f(x)| < \varepsilon \qquad for\ \ all\ \ x \in \mathscr{R}^n \tag{4}$$

The neural networks built from [13] and similar theorems did not work well in general. It was not until the development of deep learning that we begin to see how powerful neural networks can be especially in the study of differential equation solvers. In recent years, we have seen many efficient deep learning algorithms and architectures. Next, we consider a recent universal approximation theorem [95].

Theorem B.2.2 (Universal approximation theorem (nonaffine activation, arbitrary depth))**.** *[95] Let $\varphi : \mathbb{R} \to \mathbb{R}$ be any nonaffine continuous function which is continuously differentiable at at least one point, with nonzero derivative at that point. Let $K \subseteq \mathbb{R}^n$ be compact. The space of real vector-valued continuous functions on K is denoted by $C(K;\mathbb{R}^m)$. Let $\mathscr{N}$ denote the space of feedforward neural networks with n input neurons, m output neurons, and an arbitrary number of hidden layers each with $n+m+2$ neurons, such that every hidden neuron has activation function φ and every output neuron has the identity as its activation function. Then given any $\varepsilon > 0$ and any $f \in C(K;\mathbb{R}^m)$, there exists $F \in \mathscr{N}$ such that*

$$|F(x) - f(x)| < \varepsilon \qquad for\ \ all\ \ x \in K. \tag{5}$$

In other words, $\mathscr{N}$ is dense in $C(K;\mathbb{R}^m)$ with respect to the uniform norm.