Assessing Covid-19 Situation in Vietnam Using a Data-Driven Epidemiological Compartmental Model

Vo Le Tung
Department of Computer Science
Vietnamese - German University

Assessor: Assoc. Prof. Huynh Trung Hieu
Co-Assessor: Assoc. Prof. Nguyen Tuan Duc

December 11, 2021

Declaration

Acknowledgement

Abstract

Contents

List of Figures

List of Tables

1. Introduction

Since its emergence in 2019, the Covid-19 pandemic has evolved rapidly and is still affecting millions of lives around the globe. According to data from the World Health Organization (WHO) [1], Covid-19 has caused over 230 million infections and over 4.7 million death worldwide . Despite the high infection rate across the world, Vietnam had been fortunate during most of the pandemic's period, and the country was recognized for its effective government's interventions in reducing the number of cases. Until the first quarter of 2021, decisive actions and policies from the Vietnamese government have proven successful in containing the spread of Covid-19. But the situation has quickly changed since the end of April 2021, when a new wave of infection hit the country. The Delta variant of the SARS-CoV-2 virus, which is responsible for this infection wave, has a shorter incubation period and can spread quicker than the previous year [2]. As a result, policies makers in Vietnam were caught off-guard, and the number of infections has been growing exponentially for the last few months (May 2021 - September 2021). In response to the escalating number of daily reported cases, the Vietnamese government has enacted similar strategies as in 2020 and more stringent policies in regions with a high infection rate. Unlike before, these aggressive responses to the virus spread where all citizens had to stay at home were not as effective in containing the spread of the virus. At the time of this writing (October 2021), the number of daily infections is on the decline but remains high, while prolonged restrictions on movement have negatively impacted the livelihood of lots of people and the entire economy [3].

Like many other countries, one major factor that leads to the failure in swiftly containing the virus is the lack of knowledge in the dynamics of the pandemic under the effects of new variants and government interventions. Researchers have proposed diverse methods for modeling the disease to understand these dynamics. These methods use different techniques to investigate the strategies that can reduce the number of daily new infections. Rahimi, Chen, and Gandomi [4] gave a systematic review of these methods. Governments in different countries have employed these methods to aid in policy-making [5]. One popular method is an ensemble of multiple models submitted by researchers [6], used by the United States (US) Center for Disease Control and Prevention.

Although countries have been using these modeling and forecasting techniques to analyze the Covid-19 situation, many are not applied to Vietnam. One of the reasons is because there was not a high number of recorded cases. Thus the specific dynamics of Covid-19 in Vietnam have not been well studied. Admit policies from other countries can be adopted; the effectiveness may vary due to socioeconomic, demographic, and cultural factors. As the number of reported cases is dropping, the Vietnamese government is slowly removing its restrictions in many places to help improve the current economic recession. Hence having a model for the distinct circumstances and data availability in Vietnam can be beneficial for assessing the pandemic's situation, especially when restrictions are lifted. The model could be of help in the following aspects: (1) informing policies makers about the effects of their decision, (2) informing health care facilities about a possible surge in the number of cases to ensure the supply of personnel and equipment, (3) informing business owners about possible policies to plan their supply and demand needs. Lastly, the recent surge in infections has generated data on the number of Covid-19 cases in Vietnam in a much higher quantity and

quality that are publicly available and can be used for data-driven modeling.

Given that reasoning, this thesis focuses on implementing an interpretable disease model for Vietnam that can capture the current trends in the development of the pandemic at an early stage. The implemented model based on classical compartmental models where the mentioned issues are alleviated by using covariates to control the parameter(s). Instead of using a predefined multivariate function to adjust the parameters in the system of ODEs, an Artificial Neural Network (ANN) was used to encode the covariates. The method takes advantage of the recent development in ANN architectures for solving forward-inverse problems with ODEs [7]–[9]. Utilizing the capability of ANNs to approximate any arbitrary function [10]–[12], data-driven approaches can discover the underlying mechanics of Covid-19 without needing to define the governing multivariate function, which requires expert epidemiological knowledge.

The rest of the thesis is structured as follow:

- Chapter 2 gives an overview of the research backgrounds of the model. A list of related methods and techniques from other researchers is also presented.
- Chapter 3 discusses how the model was formulated, the used dataset, and the evaluation methods.
- Chapter 4 presents the findings and evaluations for the performance of the implemented model.
- Chapter 5 compares the results of the implemented model and its performance with related works produced by other researchers.This chapter includes a list of limitations of the model and further improvements that can be made to boost the effectiveness and performance of the model.
- Chapter 6 shows the overall achievement of this thesis.

2. Literature review

2.1 Related works

Ever since Covid-19 emerged, researchers have tried to model the dynamics of the disease with varying successes. In an ongoing epidemic, it is generally challenging to model the disease dynamics under partial observations. It is especially true with an unprecedented pandemic like Covid-19, where prior knowledge learned from other diseases can not be applied. There are two distinct approaches that previous researchers have used to model Covid-19: the mathematical approach and the data-driven approach. Each approach has its advantages and disadvantages, so the model depends on the questions that need to be answered and how the model will be used.

With the mathematical approach, the characteristics and behaviors of the disease are compressed down into parameters of a governing equation or system of equations. This type of model emphasizes prior knowledge about different diseases and the interpretability of the model. When using a mathematical model, the metrics, that help inform about the prevalence of the disease, can be derived from the model. One metric that epidemiologists typically want to know is the *basic reproduction number*, also known as $\mathcal{R}_0$. This number tells us whether a disease will be an epidemic, and this number must be identified as early as possible so that interventions can be in place.

With the data-driven approach, none of the knowledge about the disease that is being modeled is a priori. This type of modeling learns the dynamics of the disease from data which can help with understanding the data or help with predicting future changes. Data-driven models remove the biases and the simplistic assumptions made by mathematical models. Thus, unknown information about the disease can be implicitly captured by a data-driven model. However, epidemiologically significant metrics, that can inform scientists about the disease while using data-driven approaches, can not be derived. Furthermore, data-driven approaches can only be applied when data is available in a high quantity at a high quality. Therefore, this type of modeling is not commonly utilized at the early stage of the disease when data is not adequately collected.

In the subsequent sections, a non-exhaustive list of existing works in Covid-19 modeling and forecasting are presented. The list should give the reader a general view of some of the prior researches.

2.1.1 Forecasting Covid-19 with mathematical models

A mathematical model expresses real-world phenomenons and their relations using mathematical tools, and it is widely used in science and engineering. This kind of model relies on prior knowledge to arrive at certain constraints and assumptions about a process. Within those constraints and assumptions, a set of equations that determine how a system behaves are defined. Prior knowledge can be based on previous researches or the intuition of the researchers who have experience in the field. In the case of an infectious disease like Covid-19, the information that describes the system includes, but is not limited to, the transmission rate, the incubation period, the recovery period, and the mortality rate. The relations be-

tween these factors are intricate, and it is challenging to formulate an equation that can represent all the details that in play.

Compartmental models

Models such as the Susceptible-Infective-Removed (SIR) model and the Susceptible-Exposed-Infective-Removed (SEIR) model [13]–[16] are extensively used by researchers and public authorities. This technique divides the population into compartments and models the transition of the number of people between these compartments using a system of differential equations. In the effort to understand the development of Covid-19, compartmental modeling is among the most popular approaches due to its simplicity and interpretability. While Covid-19 exhibits similar characteristics to other transmissible diseases, its dynamics are quite different due to different government interventions, underreporting, or asymptomatic infections. Thus, many scientists have experimented with adding more compartments to the classic SIR/SEIR model to better characterize Covid-19. Commonly added compartments across literature are for asymptomatic individuals and quarantined individuals. Some researchers also model the number of patients in Intensive Care Units (ICUs) or the number of patients on ventilators, which helps forecast the number of occupied ICUs and ventilators so that healthcare facilities can prepare for a possible outbreak. Here a quick summary of some of the compartmental models created during the early phase of the pandemic are given while the problem they were solving are noted.

Zhao and Chen [17] proposed a new compartmental model tailored specifically for Covid-19 by considering three different classifications of the infective individuals: unquarantined infective, quarantined infective, and confirmed infective. This classification of infective cases follows the reality of data collecting and government intervention in China, where infective individuals were placed in quarantine immediately after they were tested positive for the SARS-NCoV-2 virus. It was demonstrated to be effective when applied to data from thirty other countries. The authors noted that this could be a helpful tool for quantifying parameters and variables concerning the effects of quarantine or confirmation method.

He, Peng, and Sun [18] modified the classic SEIR model with additional compartments to express government interventions through hospitalization and quarantine. The model was shown to have acceptable accuracy when applied with data from Hubei province. The authors argued that the inclusion of compartments for hospitalized individuals and quarantined individuals is more suitable in the case of Covid-19, and they suggested that the model can be used to inform policies on quarantine and treatment for patients. According to the authors, the results can improve by considering the model's parameters as time-dependent variables.

Ndaïrou, Area, Nieto, *et al.* [19] added compartments for hospitalized individuals, asymptomatic infective individuals, and super-spreaders to the classic SEIR model. This model was shown to adapt well to the real-world data of the early outbreak in Wuhan. The authors noted that the model could perform better when additional knowledge about the disease is added, and they suggested that later research should consider more appropriate parameters that could better indicate the characteristics of the disease.

Bastos and Cajueiro [20] modified the classic SIR model by adding compartments for asymptomatic infective individuals and introducing a parameter that took the effects of government interventions into account. This model was used to study Covid-19 in Brazil, and it showed that short-term government policies could only shift the peak of the disease further into the future. The authors also showed that the effectiveness of government interventions is in reducing the number of deaths and not in reducing the number of infective individuals. In addition, the authors found that the proportion of asymptomatic infective individuals affects the peaked number of symptomatic infective individuals, which suggested the importance of regularly testing the population.

Sarkar, Khajanchi, and Nieto [21] proposed a new compartmental model based on the classic SIR model. It takes the effects of quarantine and asymptomatic infective individuals into consideration. The model was then simulated and evaluated against data from seventeen different provinces in India, and it indicated that early lockdown is crucial for effective control of the spread. The authors showed that reducing contact between infective individuals and uninfected individuals through placing the susceptible individuals in quarantine can effectively reduce the basic reproduction number. Furthermore, they demonstrated that Covid-19 could be eradicated through a combination of social distancing and contact tracing.

Generally, it was common that early compartmental models were tailored to the specific characteristics of Covid-19 by considering two major factors: government interventions and asymptomatic infections. Different researchers had different methods for incorporating these factors into the models, but the conclusion based on such models are largely the same. They all indicated early on that quarantine, contact tracing, and testing for asymptomatic infective individuals are all effective methods for controlling the spread of the virus.

Agent-based models

This type of model performs simulations on the level of individuals to derive the epidemic trends for the entire population. Each *agents* represents a single person in the population, and the interactions between these agents can simulate the spread of the virus. In a sense, both agent-based models and compartmental models simulate the same phenomenons. However, when using agent-based models, fine-grained demographic data about a population can be utilized, and the situations of the disease can be assessed based on different scenarios of how people interact with each other. Therefore, agent-based models can easily be adapted to changing conditions and are more suitable for modeling the disease based on individualistic behaviors.

Kerr, Stuart, Mistry, *et al.* [22] developed an agent-based simulation tool that considers the population size, age structure, transmission networks for different locations such as households, schools, workplaces, and long-term care facilities. Different intervention scenarios are supported by the tool, and simulations can be run under many different assumptions. The model was calibrated for data from King County in the US in the US, and it exhibited many features of the disease in the calibration period. Once calibrated, the model is then used to evaluate different control measures and analyze how sensitive the dynamics are to these control measures. This tool is implemented in Python and is now available publicly for anyone to use.

Silva, Batista, Lima, *et al.* [23] developed an agent-based simulation tool to assess seven different intervention scenarios. A novelty of the model is that it considers both epidemiological and economic effects of Covid-19 using a wide variety of input parameters. Simulations from the model were shown to be in line with other researches. The authors also demonstrated with the simulations that policies in many different countries were ineffective.

Hoertel, Blachier, Blanco, *et al.* [24] developed an agent-based simulation tool and used it to assess Covid-19 scenarios in France. They showed that while lockdown is effective in slowing the spread, a rebound is likely to happen once lockdown is lifted. Simulations with this model also showed that both physical distancing and mask-wearing are effective in reducing the spread of the disease and lower the mortality rate. However, these preventative methods are not as effective in preventing hospitals from becoming overwhelmed.

Agent-based models are powerful tools for simulating multiple different scenarios, and they are easy to construct. However, the effectiveness of the model is depended on the quality of data on individuals that can be collected. In addition, this type of model is subject to the usual limitations of mathematical models since the parameters that are used to determine individual behaviors are subject to large uncertainties.

2.1.2 Forecasting Covid-19 with data-driven models

Because Covid-19 has prolonged, different datasets about the disease have been collected and are available publicly in high quantity. One of the frequently used datasets is from the John Hopkins University [25], which aggregates the daily number of cases, recoveries, and deaths in many countries. In addition, there is a worldwide effort in generating more datasets to help with analyzing Covid-19 spread, such as mobility indices from Apple [1], Google [2], and Facebook [3].

Statistical models

One technique that was widely used to forecast the number of Covid-19 infections based on data was the Autoregressive Integrated Moving Average (ARIMA) model [26]. The ARIMA model is one of the most used time-series models because the model takes changing trends, periodic changes, and random noises in the time-series into account. Moreover, the model is suitable for many types of data and cap capture the temporal dependency structure of a time-series.

Ceylan [27] was among the first to use this ARIMA model for forecasting Covid-19 in Italy, Spain, and France. The authors used the model to produce a 10-day forecast for Covid-19 cases, and the results on out-of-sample data show that the model can achieve high accuracy in the short-term forecast.

Ribeiro, da Silva, Mariani, *et al.* [28] proposed a framework for Covid-19 time-series forecast using an ensemble of the ARIMA model in combination and other machine learning approaches. The model was then used to forecast the number of new infections in Brazil and it achieved promising results when performing short-term forecasts.

Singh, Rani, Bhagavathula, *et al.* [29] used an ARIMA model to forecast the number of infections in 15 countries. In addition to applying the model for forecasting, the authors demonstrated that there many differences in the disease dynamics across the countries.

Although the ARIMA model is widely used and has shown that it can make reasonably good forecasts on the number of new infections, this type of model does not give much more information about the disease itself. Therefore, it is unsuitable for modeling tasks more than just the number of future cases are needed.

Deep-learning models

With an abundance of data, it has been shown that data-driven time-series forecasting techniques using deep learning can have high predictive performance. Hence, researchers have been experimenting with different architectures of ANNs for predicting the number of cases.

Chimmula and Zhang [30] used a Long Short Term Memory (LSTM) network for predicting future transmission in Canada. Predictions made by the model are based entirely on data of past transmissions. The authors demonstrated that their LSTM network could capture the complex dynamics of the disease without using a forecast complex encoding of multiple factors. However, they noted that although the model could capture the trends in transmission rate, the predictions could be highly incorrect due to external factors, and further studies are needed to precisely forecast and understand the disease dynamics.

Ramchandani, Fan, and Mostafavi [31] proposed a new ANN architecture that incorporates multivariate spatial time-series data to forecast the range of increase in COVID-19 infected cases in US counties. The model can utilize a wide range of heterogeneous features

[1] https://www.apple.com/covid19/mobility
[2] https://www.google.com/covid19/mobility
[3] https://dataforgood.facebook.com

and learn complex interactions between those features, and it demonstrated that an ANN could be used to encode the wide variety of external factors and improve the forecasting ability of ANN. Experiments from the research showed that the method obtained high predictive performance while remaining interpretable. Furthermore, the authors argued that the model could inform the effects of different mitigation and response strategies. Thus, the model can later be used by other researchers to evaluate the significance of a feature in their forecasting models.

Shahid, Zameer, and Muneeb [32] performed a comprehensive comparison between the ARIMA statistical model and multiple different Recurrent Neural Networks (RNNs), including LSTM, Bidirectional Long Short Term Memory (Bi-LSTM), and Gated Recurrent Unit (GRU). Inputs to these models are historical data on the number of cases, recoveries, and deaths. The research showed that Bi-LSTM achieved the best overall predictive performance and suggested that such a model can be used to aid in strategic planning for Covid-19.

While achieving high accuracy in predicting future cases, many of these models are black-box algorithms that are not interpretable, rendering it hard to quantify the causal effect of external factors on the progression of the pandemic. One exception is the model from [31], where how each of the external factors affects the dynamics can be deducted, but well-studied metrics in epidemiology still can not be derived from the model. Having an explainable model is extremely important for healthcare and public authorities to derive meaningful analyses that aid in the planning process. Moreover, these black-box algorithms might not capture the underlying dynamics of the disease due to under-reporting, asymptomatic infections, or a general lack of data, especially in the early stages of the outbreak.

2.1.3 Forecasting Covid-19 with data-driven compartmental models

Because of its assumptions, compartmental models typically have many drawbacks: (1) low representational capability due to the low number of parameters, (2) the represented dynamics are stationary due to the constant parameters used in the model, (3) the population is assumed to be well-mixed, i.e., every individual is statistically indifferent, and (4) non-identifiability since a different set of parameters may result in the same dynamics [33]. Many studies have attempted to overcome these limitations by varying the model's parameters, typically the transmission rate, based on spatial and temporal factors. Across the literature, many different methods and techniques have been employed to incorporate this knowledge into compartmental models.

Informing compartmental models with covariates

Since Covid-19 is an infectious disease, it is reasonable to assume that mobility plays an important role in dictating the disease dynamics. Furthermore, governments from different countries have tried different lockdown policies and quarantine policies, so it is important to justify the effectiveness of these policies. To quantify the effects of mobility on Covid-19, several researchers have proposed methods for adding this knowledge into compartmental models.

Li, Pei, Chen, *et al.* [34] modeled the spreads of Covid-19 in China using city-to-city movement data with a SIR model. Their results suggested that undocumented infective individuals are predominantly responsible for the spread of the disease before the implementation of travel restrictions in China, and travel restrictions can reduce the spread of the disease. But the length of time required for the travel restrictions was unknown, and they suggested that travel restrictions might need to be on a global scale to effectively eradicated the disease.

Chang, Pierson, Koh, *et al.* [35] used a fine-grained mobility network of the population's hourly movements, integrated it with a simple SEIR model by weighting the parameters with

the hourly movements data, and used the model to simulate hourly infections in the US. Their results showed that mobility had substantial effects on the dynamics of the disease, and individuals from groups of minorities are more likely to go to crowded locations to get their necessities. In addition, these results indicated that widespread lockdown was ineffective and harmful to the population with low income.

It is known that the population census and the quality of the healthcare system contribute to Covid-19 dynamics. Recently, researchers have been encoding different metrics into compartmental models, and some examples are Gross Domestic Product (GDP), the population age structure, the quality of the healthcare system These data points present an essential part in representing the true dynamics of Covid-19.

Schneider, Ngwa, Schwehm, *et al.* [36] used a modified SEIR model where transitions between compartments are modeled as a stepwise process to simulate Covid-19 in Austria under different scenarios, eliminating the assumption that time-delay in those transitions is exponentially distributed. A wide range of features was used to inform the transfer rate between the compartments and simulate different scenarios of interventions. The authors used the model to show that reducing contact is efficient in delaying the peak of the epidemic, and it might also be effective in decreasing the number of peak infections depending on the seasonal fluctuations in the transmissibility of the disease.

IHME COVID-19 Forecasting Team [37] used covariates to informed how the transmission rate changes over time to simulate different scenarios with Non-Pharmaceutical Interventions (NPIs) in the US. These covariates directly influence the contact rate in the modified SEIR model that they used, and how these covariates affect the contact rate is determined by a weights matrix that is learned from data using a regression technique. Their results showed that the model has high predictive performance when compared to other models.

Arık, Li, Yoon, *et al.* [38] used an SEIR-based model that takes the undocumented infective individuals, the number of hospitalized individuals, and reinfections into account. In this model, the rates of transfers between compartments are all dependent on different covariates, which are encoded by trainable weights matrices similar to an ANN. These weights matrices can be trained end-to-end with optimization techniques used in training ANN. This method has been one of the most accurate comparing with the methods used for state-level forecasts submitted for the US's ensemble model [6].

With these models, how the disease evolved under different circumstances can be simulated while still being able to derive epidemiologically significant metrics. The models in this form serve not only for understanding how different factors affect Covid-19 but also as a tool for forecasting a variety of future scenarios. Moreover, they demonstrated that compartmental models could be dramatically improved through the simple incorporation of additional information.

Informing compartmental models with artificial neural networks

Recent advances in machine learning and deep learning have enabled the ability to incorporate mathematical models with ANNs [7]–[9]. Because of the ability to approximate any possible function [10]–[12], ANNs can be used to discover unknown interactions within classical mathematical models. On the other hand, placing constraints on ANNs through mathematical models can help the ANNs learn quicker on limited data. This fusion helps to create a model that is both interpretable and flexible under rapidly changing circumstances.

Jung, Jo, Son, *et al.* [39] used multiple ANNs to learn from data how the parameters in the classical SEIR model change over time. This hybrid model was then trained using Physics-Informed Neural Network (PINN) [7] with data from South Korea and several different cities within the countries. The results from [39] showed that the ANNs could learn how these parameters change over time, and these time-depended parameters can improve the classical SEIR model and help avoid statistical uncertainties.

Dandekar, Rackauckas, and Barbastathis [40] modified the classic SIR model and introduced a new compartment to represent the number of infective individuals currently in quarantine. In addition, they introduced a time-varying term that represented the strength of quarantine and governed the number of individuals entering the quarantine compartment at each time step. The introduction of this term for the quarantine strength has the same effects as having a time-dependent contact rate. An ANN is then used to learn the function that computes the quarantine strength from data, creating a Universal Differential Equation (UDE) [9]. This model has been trained with data from 70 countries, and it can learn the correlation between increasing quarantine strength and a decrease in the spread of the disease. The authors identified in the research that this model lacks forecasting abilities and suggested the inclusion of real-time metrics on social distancing to enable robust forecasting.

These hybrid models utilized the best of both mathematical models, and ANNs showed promising results in both the ability to capture the dynamics of the disease and the predictive performance. However, not many studies have tried to extend this type of model, and existing literature only uses the prevalence of the disease as input to the embedded ANN. Since having covariates to inform the model can have positive effects, as shown in the previous section, it is not unreasonable to believe that the same covariates can be used to improve these hybrid models.

2.2 Compartmental modeling

One of the most transparent and most recognizable methods for modeling transmittable disease is through a compartmental model. This kind of modeling has been around for almost a century. The most basic was described by Kermack, McKendrick, and Walker [13]–[15] in 1927, 1932, and 1933. In the subsection, describe the basic concepts of compartmental models are described using two different models. First, the details of the basic SIR model are discussed, its assumptions, and the meaning of each term in the ODE system. Then, the SEIR model, an extension of the SIR model, is illustrated. While there have been countless extensions made to the basic SIR model, for brevity, not all of the extended versions are considered. Having a basic understanding of how the most basic models work is sufficient for grasping the concepts of more complicated models. Compartmental models are flexible such that one easily inserts a new compartment into an existing model to accommodate for the disease that is being studied.

2.2.1 SIR model

A compartmental model divides the population that is being modeled into compartments, i.e., a subpopulation. Individuals can be moved from one compartment to another under certain assumptions about the nature and time rate of transfer. As its name, the SIR model partitions the population into three subpopulations: susceptible, infective, and removed.

Let $S(t)$ be the number of people susceptible to the disease at time t. These are the people that have not been infected with the disease at time t. $I(t)$ denotes the individuals who were infected with the disease and can spread it by coming into contact with the susceptible population. $R(t)$ denotes the individuals who were infected with the disease and can not be infected with the disease again. Individuals can enter the removed compartment either through isolation, immunization against infection, recovery with immunity against reinfection, or death caused by the disease [16].

The movement between one compartment to another is captured by a system of ODEs, having time t and the transfer rates between the compartments are independent variables. The derivatives in the ODEs system express the changes in the size of each compartment with respect to time. When modeling with a system of ODEs, the number of individuals in

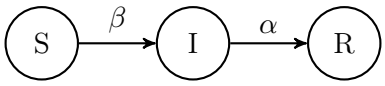

Figure 2.1: Graph of transitions between each compartment in the SIR model

each compartment is assumed to be differentiable, and the course of the epidemic is assumed to be deterministic [16].

The following ODEs system defines the SIR model [13], [16]

$$\begin{aligned} S' &= -\beta SI \\ I' &= \beta SI - \alpha I \\ R' &= \alpha I \end{aligned}$$

where interactions between the compartments can be visualized in Figure 2.1. The system is defined under the following assumptions [13], [16]

1. On average, a person makes contact sufficient to infect βN other people per unit time, where N is the total population (*mass action incidence*). Because the chance of an infective individual coming into contact with a susceptible individual is S/N, the number of new infections caused by a single infective individual per unit time is $(\beta N)(S/N)$. Thus, the total number of newly added infective individuals per unit time is $(\beta N)(S/N)I = \beta SI$.

2. Population in the infective compartment transfers to the removed compartment at a rate of αI per unit time. This assumption means that the length of the infective period is exponentially distributed with a mean of $1/\alpha$.

3. There is no new population entering into or leaving from the initial population, except deaths caused by the disease. The modeled period is short so demographic effects are negligible.

The system of ODEs can not be solved analytically, but its behavior can be acquired through quantitative approaches. It is known for a fact that the value $S(t)$ and $I(t)$ can not be negative. Thus the solving process can be terminated once $S(t) = 0$ or $I(t) = 0$. It can be observed that $S' < 0$ for all t and $I' > 0$ if and only if $S > \alpha/\beta$. Consequently, if $S(0) < \alpha/\beta$, I decreases to zero (no epidemic), whereas if $S(0) > \alpha/\beta$, I increases to a maximum achieved when $S = \alpha/\beta$ and then decreases to zero (epidemic) [16]. From the behavior, a threshold value of $\beta S(0)/\alpha$, commonly called the *basic reproduction number*, can be obtained

$$\mathcal{R}_0 = \frac{\beta S(0)}{\alpha}.$$

If $\mathcal{R}_0 < 1$, the number of infections dies out, whereas when $\mathcal{R}_0 > 1$, there will be an epidemic. By definition, the value $\mathcal{R}_0$ is the number of secondary infections caused by a single infective individual placed within a fully susceptible population of size $K \approx S(0)$ over the infective duration of this person. In this situation, the basic reproduction number becomes $\beta K/\alpha$. From here, the final size of the epidemic can be deducted by considering the *final size relation* [16]

$$ln\frac{S(0)}{S_\infty} = \mathcal{R}_0 \left[1 - \frac{S_\infty}{K}\right]. \tag{2.1}$$

Equation 2.1 also shows that $S_\infty > 0$ because its right-hand size is finite.

Estimating the contact rate β is a challenging task because it depends not only on the disease itself but also on social and behavioral factors. In retrospection, one can obtain the

values $S(0)$ and S_∞ and calculate $\mathcal{R}_0$ once an epidemic has ended. During the early phase of the epidemic, the number of infective individuals grows exponentially, where the equation for I can be approximated with [16]

$$I' = (\beta K - \alpha)I,$$

and the initial growth rate is given by

$$r = \beta K - \alpha = \alpha(\mathcal{R}_0 - 1).$$

Because both the values K and α can be measure, the contact rate can be calculated as [16]

$$\beta = \frac{r + \alpha}{K}.$$

Note that early on in the outbreak, this estimation could experience high inaccuracy due to incomplete data or underreporting of the number of cases. The estimation accuracy might also suffer significantly with an outbreak of new unknown diseases, where early cases are more likely to be diagnosed.

Lastly, the maximum number of infective individuals can be obtained, which is attained when $I' = 0$ [16]

$$I_{max} = S(0) + I(0) - \frac{\alpha}{\beta} ln S(0) - \frac{\alpha}{\beta} + \frac{\alpha}{\beta} ln \frac{\alpha}{\beta}$$

Generalized SIR model

Assumption (1) of the basic SIR model is unrealistic. It is better to assume that the contact rate is a non-increasing function of the population size. A more generalized SIR model can be formulated by replacing assumption (1) with the assumption that each person in the population makes $C(N)$ contacts in unit time on average with $C'(N) \geq 0$. The contact rate is then calculated as

$$\beta(N) = \frac{C(N)}{N},$$

and make a reasonable assumption that $\beta'(N) \leq 0$ express the saturation in the number of contacts [16]. With that assumption, a new model can be formulated

$$\begin{aligned} S' &= -\beta(N)SI \\ I' &= \beta(N)SI - \alpha I \\ R' &= f\alpha I \\ N' &= -(1-f)\alpha I \end{aligned}$$

In this model, the rate of change in the total population N is used because now the contact rate depends on it. Therefore, the distinction between the individuals who recover from the disease and the individuals who die of the disease has be be considered. Hence, assuming that a fraction f of the αI members leaving the infective compartment at time t recover. The remaining fraction $(1 - f)$ dies of the disease.

In this model, the standard reproduction number is [16]

$$\mathcal{R}_0 = \frac{K\beta(K)}{\alpha}. \tag{2.2}$$

A time-dependent reproduction number $\mathcal{R}^*$ can also be calculated. This value represents the number of secondary infections caused by an individual at time t. The equation for getting this value is [16]

$$\mathcal{R}^* = \frac{S\beta(N)}{\alpha}.$$

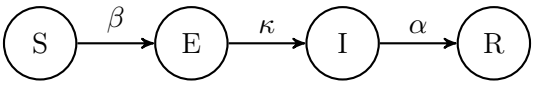

Figure 2.2: Graph of transitions between each compartment in the SEIR model

The final size relation can also be quantified under the assumption that $\lim_{N\to 0}\beta(N)$ is finite, giving the inequalities [16]

$$\mathcal{R}_0\left[1-\frac{S_\infty}{K}\right] \leq ln\frac{S(0)}{S_\infty} \leq \frac{\beta(0)(K-S_\infty)}{\alpha K}$$

2.2.2 SEIR model

With many infectious diseases, there is a period where an individual has been infected with the disease but can not transmit it. This could be modeled by adding an exposed compartment, labeled as E, where the exposed period is exponentially distributed with a mean of $1/\kappa$. The generalized version of the model with four compartments S, E, I, R, and $N = S + E + I + R$ is given as [16]

$$\begin{aligned} S' &= -\beta(N)SI \\ E' &= \beta(N)SI - \kappa E \\ I' &= \kappa E - \alpha I \\ R' &= f\alpha I \\ N' &= -(1-f)\alpha I \end{aligned}$$

and the interactions between these compartments are illustrated in Figure 2.2. The calculation of the basic reproduction for this model is similar to the calculation for the generalized SIR model, which is given by Equation 2.2.

Following the same analysis in the previous subsections, the final size relation is expressed by the inequality [16]

$$ln\frac{S(0)}{S_\infty} \geq \mathcal{R}_0\left[1-\frac{S_\infty}{K}\right]$$

For diseases that have an asymptomatic phase rather than an exposed stage, a factor ϵ can be introduced. This factor represents a reduction in the infectiousness of those who are infected but not showing symptoms. A model that represents this situation is given by [16]

$$\begin{aligned} S' &= -\beta(N)S(I+\epsilon E) \\ E' &= \beta(N)S(I+\epsilon E) - \kappa E \\ I' &= \kappa E - \alpha I \\ R' &= f\alpha I \\ N' &= -(1-f)\alpha I \end{aligned}$$

where the basic reproduction number is calculated with

$$\mathcal{R}_0 = \frac{K\beta(K)}{\alpha} + \epsilon\frac{K\beta(K)}{\kappa},$$

and the final size relation is the following inequality

$$ln\frac{S(0)}{S_\infty} \geq \mathcal{R}_0\left[1-\frac{S_\infty}{K}\right] - \frac{\epsilon\beta(K)}{\kappa}I_0.$$

2.3 Artificial Neural Networks

With the explosion of the amount of collected data and the rapid development of computer hardware in the twenty-first century, deep learning techniques have been used extensively for solving different classes of problems. Deep learning techniques are revolutionary in that they can automatically capture the relationship between the inputs and the outputs, allowing machines to perform complex tasks without having humans explicitly told them what to do. These capabilities are empowered by the underlying ANN architecture, allowing computing systems to mimic the functionality of biological neural networks, which comprise animal brains. An ANN simulates biological neural networks by having the ability to learn from experiences and examples. This process works by using a set of inputs with known outputs. The ANN is then used to guess the outputs from the inputs. The differences between the guess output values and the known output values indicate what changes need to be made to the ANN. These updates to the ANN are done multiple times until the guess error is minimized or some criteria are met. This process is now called *supervised learning*.

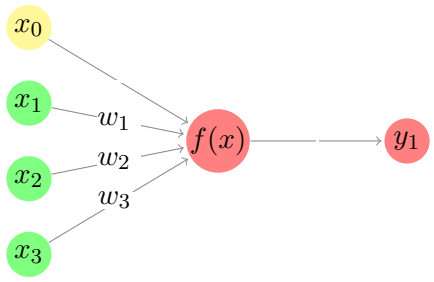

Figure 2.3: Graph representation of a perceptron described in Equation 2.3. x_0 is the bias b, and x_{1-3} are the input signals.

The idea for a computational model based on the brain was first proposed by McCulloch and Pitts [41] in 1943. They observed that nervous activity, neural events, and their relations could be demonstrated using propositional logic. Later on, researchers had been building upon that idea to find more methods for artificially representing the brain. In 1958, Rosenblatt [42] introduced the concept of perceptrons which is considered the first ANN. Mathematically, a perceptron is expressed as

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

where w and x are vectors of real value, $w \cdot x = \sum_{i=1}^{m} w_i x_i$ is the dot product between the two vectors, and b is the bias for shifting the decision boundary. The perceptron is illustrated in Figure 2.3. This initial version of ANN only works with classification problems in which the classes are linearly separable.

Nowadays, neural networks are highly complex because of the considerably higher computing power and the developments of special-purpose hardware, such as the Graphics Processing Unit (GPU). Multiple different architectures of neural networks exist, a non-exhaustive list includes: Convolutional Neural Networks (CNNs) are typically used for processing images or other two-dimensional data [43]; LSTMs solve the vanishing gradient problem and have the ability to handle data with a mix of low and high frequencies [44]; Generative Adversarial Networks (GANs) are designed to have competing ANNs on tasks such as playing a game [45].

Multi-Layer Perceptron (MLP) is a network that composes multiple nodes, called artificial neurons. These nodes in the network form a directed weighted graph, where each node can take input signals from nodes in the previous layer, performs some calculation and sends

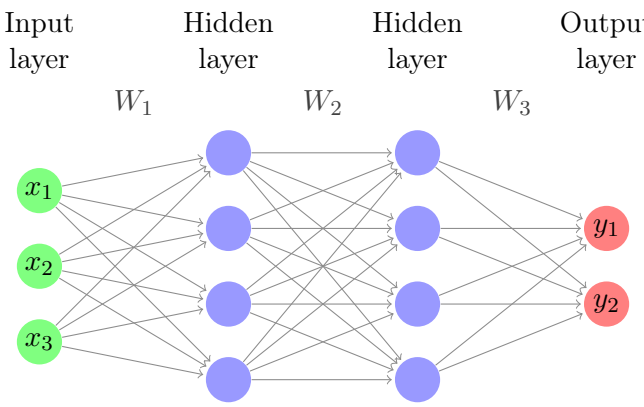

Figure 2.4: Graph representation of a multi-layer perceptron with four layers

the results to nodes in the subsequent layer. In this configuration, an MLP can be thought of as multiple perceptrons that are organized into layers. The first layer in the network is called the input layer, and the last layer is called the output layer; any layers in-between are called hidden layers. Commonly, the term MLP is used to describe an ANN where each node in a layer connects to all of the nodes in the subsequent layer, and nodes can only connect from one layer to the immediately subsequent layer. Other terms for this are fully connected ANN or densely connected ANN. Additionally, an MLP is a type of feed-forward ANN, i.e., signals in this network are passed from one layer to another in one direction only. The counterpart of this is the feedback ANN. In a feedback ANN, signals can flow in any direction, as can be seen in RNN. Mathematically, each node in an MLP computes the following values

$$a_i^k = b_i^k + \sum_{j=1}^{r_{k-1}} w_{ij}^k z_j^{k-1}$$
$$z_i^k = \phi(a_i^k),$$

where ϕ is a nonlinear function that gets applied to the output a_i^k of layer k, w_{ij}^k is the weight that maps the node j in layer $k-1$ to node i in layer k, b_i is the bias term for node i in layer k, z_i^k is the product sum plus bias for node i in layer k, a_i^k is the activated output of node i in layer k, and r_k is the number of nodes in layer k. This computation is similar to perceptron, excepting the function ϕ, called the *activation function*. Let n be the depth of a network, i.e., the number of layers having adjustable weights, and X be the vectors of input signals. Let W_i, b_i, and ϕ_i be the vector of weights, vector of bias terms, and the activation function for each layer, where $i \in [1, n]$. An MLP is expressing the following function

$$g(X) = \phi_n(W_n\phi_{n-1}(\cdots(W_2\phi_1(W_1X + b_1) + b_2) + \cdots) + b_n).$$

2.3.1 Training Artificial Neural Networks

MLPs and other ANNs can be used for various tasks because they are universal approximators [10]–[12]. That means a sufficiently large ANN with an arbitrary number of nodes can reasonably approximate any function $f : \mathbb{R}^M \mapsto \mathbb{R}^N$, given that appropriate weights can be found. However, the exact method for finding suitable weights is not defined. Commonly, the fitting weights are learned by ANNs through the back-propagation algorithm [46] and the gradient descent optimization technique [47]. A simple training procedure for most ANNs is given in Algorithm 1. Back-propagation is the algorithm for finding the gradients of a loss function with respect to the network's weights. A loss function measures how large

Algorithm 1 Batch gradient descent for training ANN. The learning rate η influences how much the weights and bias terms are updated in each iteration.

$\mathcal{D} \leftarrow \{(x_t, y_t) \mid t \in [1, n]\}$ ▷ Get training data with n samples
$\eta \leftarrow$ learning rate ▷ Choose a learning rate
$W \leftarrow$ random values ▷ Randomly initailize the weights
$b \leftarrow$ random values ▷ Randomly initailize the bias terms
while criteria are not met **do**
  $\Delta W \leftarrow 0$ ▷ Set the accumulated gradients w.r.t to each weight to zero
  $\Delta b \leftarrow 0$ ▷ Set the accumulated gradients w.r.t to each bias terms to zero
  for all $(x, y) \in \mathcal{D}$ **do** ▷ Go through all training data
    $\hat{y} \leftarrow \mathcal{NN}_{W,b}(x)$ ▷ Compute the neural network output
    $E \leftarrow \mathcal{L}(\hat{y}, y)$ ▷ Compute the loss value
    $\Delta W \leftarrow \Delta W + \eta \frac{\partial E}{\partial W}$ ▷ Accumulate the loss gradients w.r.t to each weight
    $\Delta b \leftarrow \Delta b + \eta \frac{\partial E}{\partial b}$ ▷ Accumulate the loss gradients w.r.t to each bias term
  end for
  $W \leftarrow W - \Delta W$ ▷ Update all the weights with the accumulated gradients
  $b \leftarrow b - \Delta b$ ▷ Update all the bias terms with the accumulated gradients
end while

the error is between the ANN's guess and the true value. One choice is the Mean Squared Error (MSE) $\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y} - y)^2$ The algorithm calculates the gradients starting from the output layer and going backward to the input layer. The following equation is used to find the gradient of the loss function $\mathcal{L}$ with respect to the weight w_{ij}^L, where L is the number of layers

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^L} = \frac{\partial \mathcal{L}}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}\frac{\partial z_i^L}{\partial w_{ij}^L}. \tag{2.4}$$

Similarly the gradient with respect to the bias term can be calculated by

$$\frac{\partial \mathcal{L}}{\partial b_i^L} = \frac{\partial \mathcal{L}}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}\frac{\partial z_i^L}{\partial b_i^L}. \tag{2.5}$$

If gradient of $\mathcal{L}$ with respect to w_{jk}^{L-1} needs to be calculated, first the gradient of $\mathcal{L}$ with respect to a_j^{L-1} is considered

$$\frac{\partial \mathcal{L}}{\partial a_j^{L-1}} = \sum_{i=1}^{r_{L-1}} \frac{\partial \mathcal{L}}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}\frac{\partial z_i^L}{\partial a_j^{L-1}}. \tag{2.6}$$

Then the following gradients can be computed

$$\frac{\partial \mathcal{L}}{\partial w_{jk}^{L-1}} = \frac{\partial \mathcal{L}}{\partial a_j^{L-1}}\frac{a_j^{L-1}}{z_j^{L-1}}\frac{z_j^{L-1}}{w_{jk}^{L-1}},$$

and

$$\frac{\partial \mathcal{L}}{\partial b_j^{L-1}} = \frac{\partial \mathcal{L}}{\partial a_j^{L-1}}\frac{a_j^{L-1}}{z_j^{L-1}}\frac{z_j^{L-1}}{b_j^{L-1}},$$

This same approach can be applied for any layer in the ANN. As can be seen from Equation 2.4, Equation 2.5, and Equation 2.6, the term $\frac{\partial \mathcal{L}}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}$ is used multiple times. By iterating backward and utilizing the chain rule, recalculations of derivatives can be avoided.

2.3.2 Activation functions

An activation function introduces non-linearity into the ANN without it, the ANN becomes a simple linear transformation and does not have the power to express complex relations between the input and the output. In the case of perceptron, the output of the node is either one or zero based on some threshold applied to the product sum. The activation function that gets applied in that case is called the Heaviside activation. The derivative of the Heaviside activation function is undefined at zero and is zero at every other point, making it unsuitable in deep ANN, which is trained using gradient-based optimization methods.

Some classic activation functions used in deep ANN include the sigmoid function

$$\phi(z) = \frac{1}{1 + e^{-z}},$$

and the hyperbolic tangent (tanh) function

$$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}.$$

Both of these activation functions suffer from the vanishing gradient problem. Recall from Section 2.3.1, training ANNs involves computing the gradients of the loss function with respect to its parameters. This process is done by applying the chain rule using the back-propagation algorithm. The chain rule is fundamentally a sequence of multiplications of intermediate derivatives. Because the derivatives of both sigmoid function and tanh function tend to be closer to zero, in deep ANNs with large numbers of layers, the gradients of the loss function become extremely close to zero. As a result, the ANN can not learn due to insignificant changes in the parameters caused by the infinitesimal gradients. One advantage the tanh function has over the sigmoid function is that its outputs are centered around zero, so its input can be highly negative or highly positive without affecting the ANN's performance.

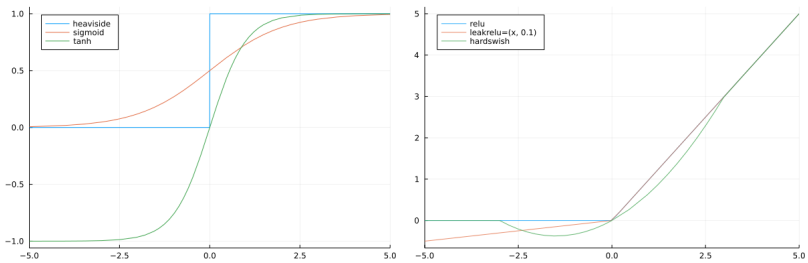

Figure 2.5: A visual comparison between the outputs of Heaviside, sigmoid, tanh, ReLU, Leaky ReLU, and hard swish activation functions

To solve the vanishing gradients problem, the ReLU activation function is used

$$\phi(z) = max(0, z).$$

One disadvantage of this activation function is the dying ReLU problem, which is when the node's output gets close to zero or is negative. When that happens, the gradients of the loss function with respect to the parameter for that node are zero, causing the node to stop learning. A function that addresses this issue is the Leaky ReLU activation

$$\phi(z) = max(\alpha z, z),$$

where α is a hyperparameter that can be chosen to define negative output values of the function. Or the hard swish activation function [48]

$$\phi(z) = \begin{cases} 0 & \text{if } z \leq -3 \\ z & \text{if } z \geq 3 \\ z * (z+3)/6 & \text{otherwise} \end{cases}$$

2.4 Physics Informed Neural Networks

Applications of deep learning methods have been achieving multiple breakthroughs, especially in computer vision and natural language processing. The performance of these models is enabled by utilizing a vast amount of data that is readily available. Nonetheless, the application of deep learning in many domains of science has not been gaining success. The cost of data collection for analyzing complex physical, biological, or engineering systems is prohibitive. Without big data, state-of-the-art deep learning techniques lack robustness and are not guaranteed to converge. Specifically, in the case of Covid-19 or any other disease, an accurate model, that can fairly estimate the trajectories of the disease at an early stage when data is scarce, is highly desirable. Raissi, Perdikaris, and Karniadakis [7] observed that prior knowledge about dynamical systems, which is captured by mechanistic models, had not been utilized for training ANNs. By utilizing this information, the solutions space that an ANN can take can be limited and the data seen by models is enriched to help them converge faster. PINNs work by using the knowledge given by mechanistic models in the form of ODEs or Partial Differential Equations (PDEs) as a regularization term for the loss function [7], [49]. With physical constraint incorporated within the loss function, the ANNs is guided to learn the parameters that give rise to a solution that obeys physical laws. Figure 2.6 shows the general schematic of PINNs.

To derive the loss function for ODEs in this framework, first the following equation is considered [49]

$$\frac{du}{dt} = f(u,t), \;\; t \in [0,1], \;\; u(0) = u_0$$

An ANN is then used to approximate the solution to this problem

$$\mathcal{NN}(t) \approx u(t).$$

Because the ANN is differentiable and assuming that $\mathcal{NN}(t)$ is the solution of the problem, then $d\mathcal{NN}(t)/dt = f((NN)(t), t)$ for all t. This condition can then be incorporated into the loss function

$$MSE = \frac{1}{N}\sum_{i}^{N}\left(\frac{d\mathcal{NN}(t_i)}{dt} - f(\mathcal{NN}(t_i), t_i)\right)^2.$$

Note that the initial condition $u(0) = u_0$ has to be satisfied. Thus a new function is defined that encodes the initial condition within itself. This function will trivially satisfies the initial condition for any possible set of parameters

$$g(t) = u_0 + t\mathcal{NN}(t).$$

The function above inherits the property of ANNs as universal approximators for any continuous function while satisfies the condition $g(0) = u_0$. At this point, the loss function turns into

$$MSE = \frac{1}{N}\sum_{i}^{N}\left(\frac{dg(t_i)}{dt} - f(g(t_i), t_i)\right)^2.$$

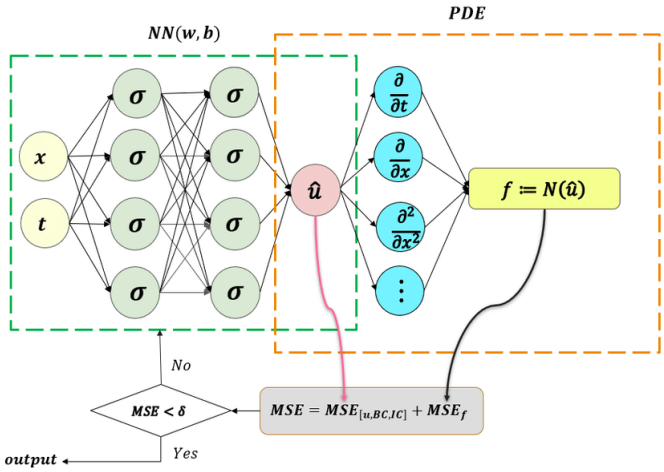

Figure 2.6: The schematic of PINNs for solving PDEs. Figure is taken from Guo, Cao, Bainian, *et al.* [50]

To derive the loss function for PDEs in this framework, the parameterized nonlinear PDEs of the general form is first considered [7]

$$u_t + \mathcal{N}[u;\lambda] = 0,\ x \in \Omega,\ t \in [0,T], \tag{2.7}$$

where $u(t,x)$ denotes the hidden solution, $\mathcal{N}[\cdot;\lambda]$ is a nonlinear differential operator parameterized by λ, and Ω is a subset of $\mathbb{R}^D$. The function $f(t,x)$ is defined to be given by the left-hand side of Equation 2.7

$$f := u_t + \mathcal{N}[u]. \tag{2.8}$$

The hidden solution $u(t,x)$ is then approximated by an ANN, denoted as $\hat{u}(t,x) = \mathcal{NN}(t,x)$. From the approximation $\hat{u}(t,x)$ and Equation 2.8, the function $\hat{f}(t,x)$ can be defined that describes the PDEs in terms of the approximation

$$\hat{f} := \hat{u}_t + \mathcal{N}[\hat{u}].$$

To train the parameters of the ANN, the following loss function is employed [7]

$$MSE = MSE_u + MSE_f,$$

where

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |\hat{u}(t_u^i, x_u^i) - u^i|^2$$

and

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |\hat{f}(t_f^i, x_f^i)|^2.$$

$\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ are the initial and boundary training data, and $\{t_f^i, x_f^i\}_{i=1}^{N_u}$ are the collocation points. The loss value MSE_u enforces the initial and boundary conditions on the ANN, while the loss value MSE_f enforces the dynamics specified by Equation 2.7.

The ANN under this framework can be training using well-known techniques in deep learning. These techniques include automatic differentiation, back-propagation, and gradient-based optimization. In the research, Raissi, Perdikaris, and Karniadakis [7] showed empirical evidence that the ANN will converge to a global minimum under the condition that the differential equations are well-posed and their solution is unique. They also showed that the model achieved high prediction accuracy with out-of-sampled data when given a sufficiently expressive ANN and a sufficient number of collocation points N_f.

2.5 Neural Ordinary Differential Equations

When modeling dynamical systems, it is generally challenging to create a model by hand because real-world data are often hard to interpret or are sampled at an irregular interval. Using Neural Ordinary Differential Equations (NeuralODEs), the system's dynamics can be learned through data-driven approaches without having explicitly specified ODEs or PDEs. NeuralODEs can also be used to replace a stack of residual blocks because they perform the same functionality while reducing the memory cost of training the model. Chen, Rubanova, Bettencourt, *et al.* made an observation that models such as residual networks (see Figure 2.7), RNN decoders, or normalizing flows build complex transformations by composing a sequence of modifications to a hidden state [8]

$$h_{t+1} = h_t + f(h_t, \theta_t)$$

where $t \in \{0 \cdots T\}$ and $h_t \in \mathbb{R}^D$. These iterative updates can be seen as an Euler discretization of a continuous transformation. In NeuralODE, the continuous dynamics of hidden units are parameterized using an ODE specified by an ANN [8]

$$\frac{dh(t)}{dt} = f(h(t), t, \theta).$$

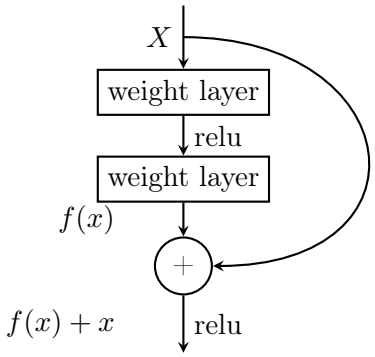

Figure 2.7: Example of a skip connection in residual network

That is starting from the input layer $h(0)$, the output layer $h(T)$ can be defined as the solution to the ODE initial value problem at time step T. Typical ANNs create a mapping from the inputs to the outputs, whereas NeuralODEs define the dynamics that can turn the inputs into the outputs. There are several benefits when doing this, such as [8]:

- **Memory efficiency** Computing the gradients of the loss function with respect to all the inputs of the ODE solver does not require back-propagation. Thus, intermediate values of the forward pass do not need to be stored. This allows for training with constant memory cost.
- **Adaptive computation** Modern ODE solver guarantees the growth of approximation error. They keep track of errors and adapt the evaluation strategy to provide the requested level of accuracy. The cost of evaluating the model can scale with the problem's complexity.
- **Scalable and invertible normalizing flows** Continuous transformations allow the change of variables formula to become easier to compute.
- **Continuous time-series models** Continuously defined dynamics can incorporate data that arrive at arbitrary times. Network models such as RNNs require the discretization of observation and emission intervals

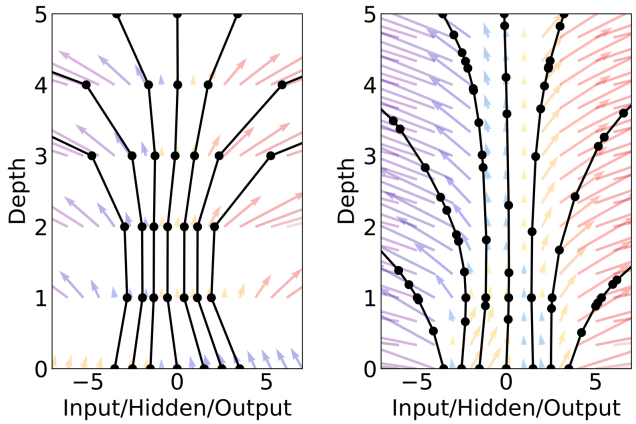

Figure 2.8: *Left*: A residual network defines discrete sequence of transformations. *Right*: A ODE network defines a vector field, which continuously transform the state. *Both*: Circles represent evaluation locations. Figure is taken from Chen, Rubanova, Bettencourt, *et al.* [8])

For a NeuralODE to be trainable with gradient descent, there must be a way to compute the gradients of the loss with respect to the parameters of the NeuralODE. The regular method for taking the loss gradients in ANNs (back-propagation) can not be applied for NeuralODEs because it incurs high memory cost and introduces additional numerical errors. Hence, Chen, Rubanova, Bettencourt, *et al.* [8] use *adjoint sensitivity analysis* to compute the gradients. The gradients are computed by solving a second, augmented ODE backward in time. This method can be applied to all ODE solvers. The approach scales linearly with the problem size, and the numerical error can be controlled explicitly.

Algorithm 2 Reverse-mode derivative of an ODE initial value problem. Algorithm is taken from Chen, Rubanova, Bettencourt, *et al.* [8])

function ODEDERIVATIVE($\theta, t_0, t_1, z(t_1), \frac{\partial L}{\partial z(t_1)}$)
  $s_0 \leftarrow \left[z(t_1), \frac{\partial L}{\partial z(t_1)}, 0_{|\theta|}\right]$ ▷ Define initial augmented state
  function AUGDYNAMICS($[z(t), a(t), \cdot], t, \theta$) ▷ Define dynamics on augmented state
    return $\left[f(z(t), t, \theta), -a(t)^T \frac{\partial f}{\partial z}, -a(t)^T \frac{\partial f}{\partial \theta}\right]$ ▷ Compute vector-Jacobian products
  end function
  $\left[z(t_0), \frac{\partial L}{\partial z(t_0)}, \frac{\partial L}{\partial \theta}\right] \leftarrow$ ODESOLVE(s_0, AUGDYNAMICS, t_1, t_0, θ) ▷ Solve ODE backward
  return $\left[\frac{\partial L}{\partial z(t_0)}, \frac{\partial L}{\partial \theta}\right]$ ▷ Return the gradients
end function

Considering the loss function L, which takes the result of an ODE solver [8]

$$L(z(t_1)) = L\left(z(t_0) + \int_{t_0}^{t_1} f(z(t), t, \theta)dt\right) = L(ODESolve(z(t_0), f, t_0, t_1, \theta)).$$

First, the gradient of the loss with respect to the hidden state $z(t)$ is considered, called the adjoint $a(t) = \partial L/\partial z(t)$. The dynamics of the adjoint are then given by another ODE [8]

$$\frac{da(t)}{dt} = -a(t)^T \frac{\partial f(z(t), t, \theta)}{\partial z}.$$

The gradient of the loss with respect to the hidden state at the initial time step can be calculated by integrating the adjoint dynamics backward in time starting from the value

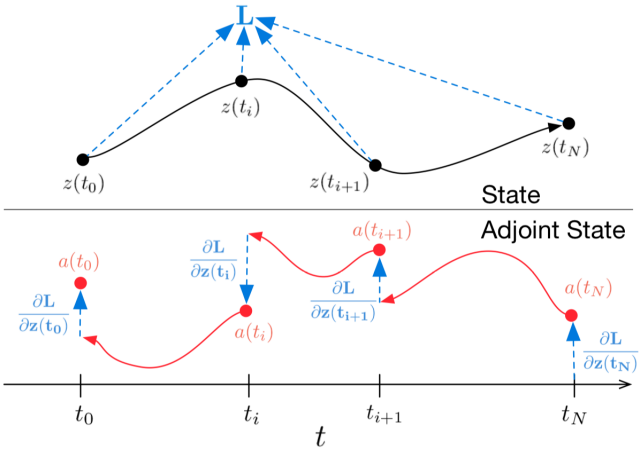

Figure 2.9: Reverse-mode differentiation of an ODE solution. An augment system contains the original state and the gradients of the loss with respect to the state is solved backwards in time. If the loss depends on the state at multiple observation times, the adjoin state is updated in the direction of the gradients of the loss with respect to each observation. Figure is taken from Chen, Rubanova, Bettencourt, *et al.* [8]

$\partial L/\partial z(t_1)$ [8]

$$\frac{dL}{dz(t_0)} = \frac{\partial L}{\partial z(t_1)} + \int_{t_1}^{t_0} -a(t)^T \frac{\partial f(z(t), t, \theta)}{\partial z} dt. \tag{2.9}$$

Computing $dL/dz(t_0)$ requires the hidden state $z(t)$ at each evaluated time step which is calculated by integrating the dynamics of the system backward in time starting from the value $z(t_1)$ [8]

$$z(t) = z(t_1) + \int_{t_1}^{t} f(z(t), t, \theta) dt. \tag{2.10}$$

Finally, the gradients of the loss function with respect to the parameters θ can be calculated using [8]

$$\frac{dL}{d\theta} = \int_{t_1}^{t_0} -a(t)^T \frac{\partial f(z(t), t, \theta)}{\partial \theta} dt. \tag{2.11}$$

The vector-Jacobian products $a(t)^T \frac{\partial f}{\partial z}$ and $a(t)^T \frac{\partial f}{\partial \theta}$ from Equation 2.9 and Equation 2.11 can be computed efficiently with automatic differentiation. The integrals in Equation 2.9, Equation 2.10, and Equation 2.11 can be computed in a single call to an ODE solver, as demonstrated in Algorithm 2.

2.6 Universal Differential Equations

Because of the scarcity of data in many scientific domains, mechanistic models are frequently employed instead of deep learning. These models often come in the form of ODEs or PDEs that describe the system's behavior over time, referred to as the system's dynamics. Mechanistic models are explainable, based on prior structural knowledge, and backed by many years of rigorous studies, but they lack the flexibility of deep learning models and usually make unrealistic assumptions about real-world phenomenons. In contrast, deep learning models are highly flexible, but they require a massive amount of data to be effective. Based on those observations, many researchers have recently presented with different methods for merging machine learning and mechanistic models.

As introduced in Section 2.4, PINNs can enforce physical constraints on ANNs through the use of PDEs/ODEs as a regularization term in the loss function. PINNs were shown to

have high performance for some scientific applications with limited data. However, it still lacks the interpretability of mechanistic models even though it can be seen that the ANN has learned real world physical characteristics. Instead of incorporating scientific knowledge into the model, NeuralODEs take a different approach by using the structure of scientific models as a basis for machine learning. While NeuralODEs can learn the continuous dynamics data, as shown in Section 2.5, the resulting models do not represent any known mechanisms. To address these issues, Rackauckas, Ma, Martensen, *et al.* [9] proposed UDEs that extend the approach taken by NeuralODEs. UDEs apply mechanistic modeling in conjunction with universal approximators where part of the differential equation embeds a universal approximator, e.g., a neural network, a random forest, or a Chebyshev expansion. This combination helps create a robust model, where well-proven terms in the mechanistic model can stay unchanged, while the terms with complex unknown interactions can be discovered by the universal approximator.

In general, a UDE model will be in the form of [9]

$$u' = f(u, t, U_\theta(u, t)),$$

where f denotes a known mechanistic model whose missing terms are defined by some universal approximator U_θ. The process for training UDEs is similar to training typical ANNs a loss function $\mathcal{L}(\theta)$, defined for the approximated solution $u_\theta(t)$ with respect to the current choice of parameters θ, is minimized. Common choices for a loss function used to train ANNs are applicable in this case, such as the MSE $\mathcal{L}(\theta) = \sum_{i=1}^{N}(u_\theta(t_i) - d_i)^2$ at discrete data points $\{(t_i, d_i)\}_{i=i}^{N}$. Then the gradients of the loss with respect to the parameters $\frac{\partial \mathcal{L}}{\partial \theta}$ are calculated for applying local gradient-based methods such as gradient descent. Methods for computing the gradients of UDEs of different types of differential equations are implemented by Rackauckas, Ma, Martensen, *et al.* [9] as a library in the Julia programming language [51], which will be utilized in this thesis.

3. Methodologies

In this thesis, the main focus was on finding an interpretable disease model for Covid-19 in Vietnam. The model was expected to have the ability to capture the complex dynamics of the disease from data at an early stage, which could help inform people about a potential outbreak of the disease. Here, UDEs was used to formulate the model, where an ANN was added on top of a classical compartmental model for infectious diseases, . Because it had been shown that multiple different factors have different effects on the dynamics of the disease, different covariates were encoded into the model to improve the predictive performance [37], [38]. Experiments with several versions of the model were conducted to evaluate the effectiveness of adding different covariates to inform the model. In this section, the data that was used, the models' formulations, the experiments that were conducted, and how the models were evaluated are described in detail.

3.1 Data

3.1.1 Covid-19 cases data

The datasets for Covid-19 at both the country level and the province level were used to train and evaluate the models. At the country level, time-series data were available for the number of confirmed cases, the number of recoveries, and the number of deaths from the disease. However, at the province level or the county level, time-series data were only available for the number of confirmed cases and the number of deaths from the disease.

To obtain Vietnam country-level time-series data, the Covid-19 datasets [1] from John Hopkins University [25] are utilized. The repository had three separate CSV files, each containing the global time-series data since January 22th 2020 for the total number of confirmed cases, the total number of recoveries, and the total number of deaths from the disease. The general structure of the files is illustrated in Table 3.1. From the three CSV files, the data for Vietnam was extracted and a single time-series dataset for the country was constructed (see Table 3.2 for an example). Because only the most recent outbreak in Vietnam (starting from 27th April 2021) was considered, the data points before 27th April 2021 were filtered out.

The data for Covid-19 cases in counties in the US were also obtained from the John Hopkins University. The repository created by the John Hopkins University [25] contained 2 separated CSV files for the total number of confirmed cases time-series and the total number of deaths time-series for counties in the US. The procedure that was applied for combining Vietnam country-level data was also used to extract Covid-19 cases data for a single county. Similar with Vietnam's data, only data for the most recent outbreak in the US was considered. 1st July 2021 was chosen to be the starting date of the most recent Covid-19 outbreak in the US, which was roughly when the fourth wave of the Covid-19 pandemic started in the US.

[1] https://github.com/CSSEGISandData/COVID-19

Province/State	Country/Region	1/22/20	1/23/20	1/24/20	⋯
⋯	⋯	⋯	⋯	⋯	⋯
—	Venezuela	0	0	0	⋯
—	Vietnam	0	2	2	⋯
⋯	⋯	⋯	⋯	⋯	⋯

Table 3.1: Each row in the dataset from John Hopkins University [25] contained the Covid-19 time-series data of a geographical location. The value at each time step could be the number of confirmed cases, the number of recoveries, or the number of deaths depending on the CSV file that was used.

date	infective	confirmed	recoveries	deaths
1/22/20	0	0	0	0
1/23/20	2	2	0	0
1/24/20	2	2	0	0
⋯	⋯	⋯	⋯	⋯

Table 3.2: The values in the *infective* column represent the number of infective individuals on that day, and they were calculated by subtracting the values in the *recoveries* column and the *deaths* column from the values in the *confirmed* column. Data for the columns besides *infective* were taken from the dataset from John Hopkins University [25]

.

Although the datasets provided by the John Hopkins University contain data at lower levels for some countries, Vietnam was not one of them. Instead, two additional sources of information were utilized, which were the Covid-19 situations dashboard made by the Vietnam General Department of Preventative Medicine [2] and the Covid-19 situations dashboard made by *vnexpress.net* [3], a local online newspaper that aggregated Covid-19 data from the government's daily announcements on Covid1-19 cases). Data from these two sources were available for the outbreak period in Vietnam starting from 27th April 2021. From the first source, the number of confirmed cases and the number of deaths from Covid-19 for the four provinces/cities, that had the highest number of confirmed cases in the countries, were downloaded. The cities/provinces were Ho Chi Minh city, Binh Duong province, Dong Nai province, and Long An province. The downloaded time-series data for each province was parsed from its original Javascript Object Notation (JSON) format and saved as a CSV file. For consistency, the structure of the time-series data for each province was similar to the one given in Table 3.2, but with the *infective* and *recoveries* columns omitted. From the second source, the number of confirmed cases for all provinces in Vietnam was obtained by accessing the underlying Application Programming Interface (API) of the website. This data was used to derive the spatiotemporal metric that was used as the input to the model (see Section 3.2). The number of confirmed cases was obtained from the second source instead of the first one because the data here was given in bulk for all the provinces (see Table 3.3) instead of as individual data files when working with the website from the Vietnam General Department of Preventative Medicine.

3.1.2 Facebook's data

Because it had been shown that population mobility can greatly reflect to spread of the disease [34], [35], [37], the public datasets from Facebook [4] were utilized to inform the

[2] https://ncov.vncdc.gov.vn/
[3] https://vnexpress.net/covid-19/covid-19-viet-nam
[4] https://dataforgood.facebook.com

date	Binh Duong	Ho Chi Minh city	Ha Noi	···
27/4	0	0	0	···
28/4	0	0	0	···
29/4	0	1	0	···
···	···	···	···	···

Table 3.3: Each row in the dataset returned by *vnexpress.net* contains the number of cases for all provinces in Vietnam on that date.

model about the spatiotemporal factors that could have an effect on the disease dynamics. Facebook's data was chosen because the company had a massive number of users with 2.89 billion users worldwide and about 71 million users in Vietnam, which took account for more than two-thirds of the country's population.

The first dataset from Facebook was the Movement Range Maps dataset [5]. This dataset contained time-series of the relative change of mobility in percentages compared to the baseline in February 2020 and the ratio of people that only stay within one area throughout the day [52] (the structure of the time-series dataset is given in Table 3.4). The values in the dataset were calculated using the anonymized and privacy-preserving mobile phone location history of Facebook's users that allowed the app to collect their location data. Other large technology companies such as Google [6] and Apple [7]also published their mobility datasets for public use, but for consistency, only the datasets from Facebook were considered. This data reflected how people in a geographic area reacted to different preventative measures from the government and was calculated for many subregions within a country. For Vietnam, this data was available for level 2 divisions from the Database of Global Administrative Area (GADM), i.e., district level.

ds	country	polygon_id	relative_change	ratio_single_tile_users
···	···	···	···	···
2021-01-01	VNM	VNM.1.10_1	0.12525	0.27042
2021-01-02	VNM	VNM.1.10_1	0.05274	0.25942
2021-01-03	VNM	VNM.1.10_1	0.18506	0.26941
···	···	···	···	···

Table 3.4: Each row in the Movement Range Map dataset from Facebook contains a date *ds*, a three-letter ISO-3166 country code *country*, a unique identifier for the geographical region *polygon_id*, the relative change in mobility compared to the baseline *relative_change*, and the proportion of people staying put *ratio_single_tile_users*.

The second dataset from Facebook was the SCI dataset [8]. This dataset measured the strength of social connectedness between two geographical areas represented by Facebook friendship ties (the structure of the dataset is given in Table 3.5). Formally the SCI between two locations i and j was calculated as

$$\text{Social Connectedness Index}_{i,j} = \frac{\text{FB connections}_{i,j}}{\text{FB users}_i * \text{FB users}_j},$$

where FB users_i and FB users_j are the numbers of Facebook users in location j and j, and $\text{FB connections}_{i,j}$ is the total number of Facebook friendship connections between people in

[5] https://dataforgood.facebook.com/dfg/tools/movement-range-maps
[6] https://www.google.com/covid19/mobility
[7] https://www.apple.com/covid19/mobility
[8] https://dataforgood.facebook.com/dfg/tools/social-connectedness-index

the two locations. In the published dataset, the index was scaled to have a minimum value of 1 and a maximum value of 1,000,000,000. The Social Connectedness Index$_{i,j}$ measured the relative probability of a Facebook friendship link between a user in location i and a user in location j. In other words, if the value is twice as large, then a user in location i is twice as likely to be a friend with a user in location j. The SCI dataset provided by Facebook has multiple CSV files, each containing the SCI between different subdivisions of many countries. The CSV file for the SCI between level 1 GADM subdivisions were downloaded to extract the data for Vietnam. For Vietnam, level 1 GADM subdivisions were defined to be the province level. To extract the data for counties in the US, the separated CSV file for the SCI between different US counties was downloaded. Each county in this CSV file is identified by its Federation Information Processing Standards (FIPS) code.

user_loc	fr_loc	scaled_sci
VNM1	VNM1	783251
VNM1	VNM10	10301
VNM1	VNM11	8237
...	...	...

Table 3.5: The SCI dataset from Facebook included every (symmetric) i to j and j to i location pair, including links of each location to itself. The *user_loc* column represents the first location, the *fr_loc* column represents the second location, and the *scaled_sci* is the scaled SCI.

3.1.3 Average population's data

The average population data for counties in the US and for provinces in Vietnam were obtained to facilitate the calculation of the Social Proximity to Cases (SPC) index. Vietnam provinces' average population data was taken from Vietnam's General Statistics Office [9]. In addition, the average population data for Vietnam's provinces was combined with with the dataset from the GADM [10] (version 2.8). This was done so that a province could be referred to using the level 1 GADM identifier for that province when calculating the SPC index. The US counties' average population data was taken from the John Hopkins University. In their time-series for the total number of deaths for the US counties [25], they included the average population of the counties from which the data was extracted, and a table that associated the county's FIPS code and its average population was created. Table 3.6 presents the general structure of the tables containing the regions' average population.

3.2 Models definitions

As mentioned in previous sections, a compartmental model for infectious diseases was utilized as the basis for the model. The basic SEIR model was chosen because of its simplicity and a low number of parameters which resulted in a simpler and more manageable learning process. Demographic effects were not included in the model because only a short outbreak period was considered, thus these effects were negligible. Moreover, additional compartments for individuals that have been vaccinated were not included because the vaccination was only available for a low percentage of the population for the period that was chosen for the model. In addition to the existing compartments in the SEIR model, three additional compartments D, C, and T were introduced for the number of total deaths, the number of newly confirmed

[9] https://gso.gov.vn
[10] https://gadm.org

ID_1	NAME_1	AVGPOPULATION
3	Ha Noi	8.2466e6
62	Vinh Phuc	1.1712e6
16	Bac Ninh	1.4191e6
...	...	...
1001.0	Autauga, Alabama, US	55869
1003.0	Baldwin, Alabama, US	223234
1005.0	Barbour, Alabama, US	24686
...	...	...

Table 3.6: Illustration of the tables containing the average population of the subregions within a country. The column *ID_1* contains the FIPS code of the counties if the table contains data for US' counties. If the table contains data for other countries, *ID_1* column will contain the level 1 GADM identifier of the location. The columns *NAME_1* and *AVGPOPULATION* contain the name of the location and the average population of that location, respectively.

cases, and the number of total confirmed cases.. These compartments were introduced so that the model could represent the available observations that were available in the real-world data. Formally, the system of ODEs that governs the extended SEIR model was defined as

$$\begin{aligned} S' &= -\frac{\beta SI}{N} \\ E' &= \frac{\beta SI}{N} - \gamma E \\ I' &= \gamma E - \lambda I \\ R' &= (1-\alpha) * \lambda * I \\ D' &= \alpha * \lambda * I \\ N' &= -\alpha * \lambda * I \\ C' &= -C + \gamma * E \\ T' &= \gamma * E \end{aligned} \tag{3.1}$$

where β is the number of people that are infected per unit time, γ is the exposure rate, λ is the rate at which infective individuals become not infective anymore, and α is the fatality rate. Unlike the definition of the SEIR model given in Section 2.2.2 where an infective person can infect βN other people per unit time, here β itself was chosen to be the number of people that can be infected by a single infective individual per unit time. Thus, the total number of newly infected people per unit time in this formulation turned into $\beta SI/N$ instead of βSI. In other words, the value of β here corresponded to $C(N)$ in the generalized SIR/SEIR model presented in Section 2.2.1 where $\beta(N) = C(N)/N$. This was done because, in the experiments, it was observed that using β as the number of people that can be infected by a single infective individual resulted in a better fit to the training data and a better out-of-sample performance of the model. Noted that existing literature [17]–[21], [37], [38], [53] had also utilized the same formulation for the transition between the S and E compartments. Based on Equation 3.1, the basic reproduction number could be obtained by using the following equation

$$\mathcal{R}_0 = \frac{\beta}{\gamma}.$$

Because it had been shown that several factors could have varying effects on the transmission rate β, and a time-dependent transmission rate $\beta(t)$ was much more representative

of Covid-19 [34], [35], [37], [38], [40], an ANN that could encode different covariates was incorporated into the system of equations so that the model could represent a time-dependent contact rate $\beta(t)$. Generally, the average contact rate was given by

$$\beta(t) = \mathcal{NN}_{\theta_1}(\mathcal{F}), \tag{3.2}$$

where $\mathcal{NN}_{\theta_1}$ is an ANN whose weights and biases are represented as θ_1, and $\mathcal{F}$ is the set of covariates that were used to inform the ANN about different effects that external factors had time-dependent contact rate $\beta(t)$. Besides β, α was chosen to be another time-dependent parameter in the system because it was observed that the fatality rate could be very high when the number of cases increased exponentially and caused an overload to healthcare facilities, then the fatality rate would decrease over time as the healthcare system adapted to the situation. The time-dependent fatality rate is given by

$$\alpha(t) = \mathcal{NN}_{\theta_2}(\frac{t}{t_{\max}}, \frac{I(t-1)}{N(t-1)}, \frac{R(t-1)}{N(t-1)}, \frac{D(t-1)}{N(t-1)}),$$

where $\mathcal{NN}_{\theta_2}$ is an ANN whose parameters are represented by θ_2, t is the currently simulated time step, $t_{\max}$ is the training duration, $I(t-1)/N(t-1)$ is the disease prevalence, $R(t-1)/N(t-1)$ is the fraction of the population who recovered from the disease, and $D(t-1)/N(t-1)$ is the fraction of the population who died of the disease. Hence, the basic system of equations in Equation 3.1 was reformulated as

$$\begin{aligned}
S' &= -\frac{\beta(t)SI}{N} \\
E' &= \frac{\beta(t)SI}{N} - \gamma E \\
I' &= \gamma E - \lambda I \\
R' &= (1-\alpha(t)) * \lambda * I \\
D' &= \alpha(t) * \lambda * I \\
C' &= \gamma * E \\
N' &= -\alpha * \lambda * I \\
C' &= -C + \gamma * E \\
T' &= \gamma * E \\
\beta(t) &= \mathcal{NN}_{\theta_1}(\mathcal{F}) \\
\alpha(t) &= \mathcal{NN}_{\theta_2}(\frac{t}{t_{\max}}, \frac{I(t-1)}{N(t-1)}, \frac{R(t-1)}{N(t-1)}, \frac{D(t-1)}{N(t-1)})
\end{aligned} \tag{3.3}$$

The system of equations in Equation 3.3 gave us an UDE [9] that can be trained in an end-to-end fashion similar to the training process for ANNs. From Equation 3.2 and Equation 3.2, the basic reproduction rate of the disease at time t was derived to be

$$\mathcal{R}_t = \frac{\beta(t)}{\gamma} = \frac{\mathcal{NN}_{\theta_1}(\mathcal{F})}{\gamma}.$$

In [9], the authors show that a small ANN with a low number of hidden layers and a low number of nodes is sufficient for an UDE to represent the complex interactions in different dynamical systems. A similar model proposed in [40] had also used a small ANN with 1 hidden layer where the layer had 10 nodes, and it had been shown to achieve a good performance on Covid-19 data with that small ANN. As a result, both the ANNs embedded in Equation 3.3 were chosen to be small ANNs where $\mathcal{NN}_{\theta_1}$ has 3 hidden layers in which each layer has 8 nodes and $\mathcal{NN}_{\theta_2}$ has 1 hidden layer that has 8 nodes. The *mish* function

[54] was used as the activation function for all hidden layers, and the output of the ANNs were transformed such that

$$\nu_i = \nu_{i,L} + (\nu_{i,U} - \nu_{i,L}) * \sigma(z_i),$$

where ν_i is a system parameter that is encoded by an ANN whose output is z_i, and $\nu_{i,L}$ and $\nu_{i,U}$ are the lower and upper bounds of the encoded parameter. The values of $\nu_{i,L}$ and $\nu_{i,U}$ were hyperparameters that can be chosen based on existing researches on the disease to help the model converge with reasonable parameters values.

Using the UDE model in the general form given by Equation 3.3, different sets of external factors that could have an effect on the time-dependent contact rate $\beta(t)$ were explored to see which covariates can improve the performance of the model. When the different sets of covariates were considered as inputs to the ANN, the only parameters of the ANN that change is the number of input features that can be taken by the ANN, while the number of hidden layers, the number of nodes, and the activation function of each layer stayed the same. In the subsequent sections, the different sets of covariates, that were chosen as input to the $\mathcal{NN}_{\theta_1}$ in Equation 3.3 are stipulated.

Informing the model with past states

The first set of covariates that were utilized as inputs for the ANN in Equation 3.3 was the states of the system in the previous time step. Concretely, the set of covariates $\mathcal{F}$ that were given as input to the ANN at time step t in this model was defined as

$$\mathcal{F}(t) = \left\{ \frac{t}{t_{\max}}, \frac{S(t-1)}{N(t-1)}, \frac{E(t-1)}{N(t-1)}, \frac{I(t-1)}{N(t-1)} \right\},$$

where t is the currently simulated time step, $t_{\max}$ is the training duration, $S(t-1)/N(t-1)$ is the proportion of the population that is susceptible, $E(t-1)/N(t-1)$ is the proportion of the population that is exposed and $I(t-1)/N(t-1)$ is the incidence rate. Here, it was assumed that I/N, E/N, and I/N were the only covariates that could have an effect on the contact rate β. In this form, the model is similar to the one that had been proposed in [40]. The other states of the system were not considered as inputs to the ANN in this version of the model because, by definition, the individuals from those states can not contribute in the transmission the disease. Thus they should have zero effect on the transmission rate $\beta(t)$. Furthermore, the fractions S/N, E/N and I/N were considered as inputs instead of the actual counts to keep the inputs to the ANN lying in the range from 0 to 1 which could help improve the model training process. Additionally, it was observed that using the fractions helped to create better results and a faster runtime. This model did not rely on external data to inform the embedded ANN and based its prediction entirely on the states of the system. As such, the model was used as the baseline for comparisons in the experiments.

Informing the model with mobility data

As noted by the authors of [40], informing the basic UDE model for Covid-19 with mobility data could improve the forecasting ability of the model. Moreover, [34], [35], [37] have showed that mobility is an important predictor for the spread of Covid-19. Therefore, the Movement Range Maps dataset from Facebook (see Section 3.1.2) was used in addition to the past system's states in the second set of covariates that were given as inputs for the ANN in Equation 3.3. In this second version of the model, the set of covariates $\mathcal{F}$ given to the ANN at time step t was defined as

$$\mathcal{F}(t) = \left\{ \frac{t}{t_{\max}}, \frac{S(t-1)}{N(t-1)}, \frac{R(t-1)}{N(t-1)}, \frac{I(t-1)}{N(t-1)}, \text{MovementRange}(t) \right\},$$

where MovementRange(t) was the relative change in movement and the ratio of people who were staying put at the area that was being modeled at time t. When performing simulations with data for Vietnam and its provinces, MovementRange(t) was calculated as the mean of the values of all the subregions within the area that was being modeled. That was done because the Movement Range Maps dataset was containing data at a much more granular level but only the country-level data and province-level data were the subjects of interests.

Informing the model with social network connections

As shown by the authors in [55], the strengths of social network connections between different regions were an important predictor of Covid-19 cases in US counties. They had defined the SPC index which is a metric defined specifically for Covid-19 that measures the spreads of the disease based on the SCI dataset from Facebook. They also noted that the correlation between the SPC index and the number of cases was especially strong when there were fewer restrictions on individuals' mobility. Thus, the SPC index was considered as an input feature to the ANN in addition to the Movement Range Maps dataset from Facebook and the past system's states. The SPC index is calculated as

$$SPC_{i,t} = \sum_{j}^{C} \text{Cases per } 10\text{k}_{j,t} \frac{SCI_{i,j}}{\sum_{h}^{C} SCI_{i,h}},$$

where $SPC_{i,t}$ is the SPC index of location i at time t, C is the number of locations, and Cases per $10\text{k}_{j,t}$ if the number of confirmed cases out of 10,000 people at location j at time t. In this third version of the model, the set of covariates $\mathcal{F}$ given to the ANN at time step t was defined as

$$\mathcal{F}(t) = \left\{ \frac{t}{t_{\max}}, \frac{S(t-1)}{N(t-1)}, \frac{R(t-1)}{N(t-1)}, \frac{I(t-1)}{N(t-1)}, \text{MovementRange}(t), \text{SPC}(t) \right\},$$

where SPC(t) was the SPC index of the area that was being modeled at time t.

3.3 Parameters estimation

An end-to-end learning mechanism was used to find the set of model parameters that produced the best fitting curves to the ground truth data. In the model learning period, only the observations for some of the compartments in the model were accessible because of the availability of data. For the modeled period, there was no observation for the states $\{S, E, I, R\}$ but only the $\{D, C, T\}$ states, which were the number of total deaths, the number of newly confirmed cases, and the number of total confirmed cases. Therefore, the model could only learn from partially-available observations where the available data for some compartments were used to supervise the learning process while other compartments were left unconstrained.

The trainable parameters were $(\gamma', \lambda', \theta_1, \theta_2)$ in which θ_1 and θ_2 were the set of parameters for the embedded ANNs, and γ' and λ' were used to determine γ and λ, such that

$$\gamma = \gamma_L + (\gamma_U - \gamma_L) * \sigma(\gamma')$$
$$\lambda = \lambda_L + (\lambda_U - \lambda_L) * \sigma(\lambda')$$

The transformation allowed the estimated values to stay within a reasonable range based on existing knowledge about Covid-19. The best fitting set of parameters was estimated by minimizing the following loss function

$$\mathcal{L}(\hat{y}, y) = \frac{1}{T} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left[e^{\zeta t} \left(\frac{\hat{y}_{i,t} - y_{i,t}}{max(y_i) - min(y_i)} \right)^2 \right] + \lambda(\|\theta_1\|_2^2 + \|\theta_2\|_2^2). \tag{3.4}$$

N is the number of compartments that were observable, T is the number of collocation points, i.e., the number of days that were used for training, $\hat{y}_{t,t}$ is the model's output for compartment i at time t when given $(\gamma', \lambda', \theta_1, \theta_2)$ as parameters, and $y_{i,t}$ is the observation for compartment i at time t. The errors between the ground truth data and the model's predictions were scaled by the difference between the maximum and minimum values of the observations for each compartment. Scaling was done so that the errors between the model's predictions and the ground truth data for different compartments could contribute equally to the final loss value even when the states of the compartments were in different scales. Moreover, the model's errors were weighted by the term $e^{\zeta t}$ with ζ being a hyperparameter that determined how fast the weights changed with respect to the time t. If ζ is negative then earlier observations will be more important than later observations, whereas if ζ is positive then later observations will be more important than earlier observations. Lastly, an L2 regularization term controlled by the hyperparameter λ was included to reduce the chance of over-fitting.

A numerical ODE solver that used the Tsitouras 5/4 Runge-Kutta method [56], implemented in the *DifferentialEquations.jl* package [57], was used to solve the model's system of ODEs on a given training time span. In the problem, the model's outputs were the states of the system of ODEs at each time step where the time steps correspond to the dates that were considered. Once the model's outputs were obtained, minimization of the loss function in Equation 3.4 was carried out using local-gradient based optimization techniques where the gradients were calculated through automatic differentiation with the *InterpolatingAdjoint* algorithm implemented in the *DifferentialEquations.jl* package. *InterpolatingAdjoint* was chosen as the algorithm for automatic differentiation because of its fast runtime speed and low memory usage comparing to other available algorithms when applied to this problem [57].

The minimization process happened in 2 stages as recommended by [9]. In the first stage, a first-order optimization algorithm was used to estimate the parameters to help them reach a reasonable parameters space. For the experiments, ADAM [58] was chosen as the optimizer in this stage. Parameters estimation with local gradient based algorithm can encounter issue with local minima. There is a high chance that the model gets stuck in a bad local minima, especially with time series data since the model can comfortably settle with predicting the time series mean. As suggested by the authors of the *DiffEqFlux.jl* package [9], one technique for reducing the chance of falling into a bad local minima is to place more weight to earlier data points which allow for fitting to earlier portions first. This can be done by choosing a negative value for the hyperparameter ζ in Equation 3.4. In addition, during the first fitting stage, the fit region could be grew iteratively allowing the model fit to earlier data points first. Specifically, the model was first trained using time series data on the time span from 0 to T' where $T' < T$. Once the training completed, the time span was increased so that later data points could be included, and the model was then trained on the new time span. This process repeated until T' equals T. Then in the second stage, we continued the training with a more complex optimization algorithm so that the model could converge faster to a good minimum. For the experiments, BFGS [59]–[62] was chosen as the optimizer for this stage. Algorithm 3 presents the general procedure that was carried out in the 2 stages optimization process.

3.4 Experiments

Several experiments were conducted to evaluate the performance of the model on out-of-sample data when different sets of covariates were chosen to inform the model. For the experiments, the different versions of the model described in Section 3.2 were first trained using the parameters optimization procedure described in Section 3.3. During the training

Algorithm 3 General training procedure for the proposed models.

$\text{maxiters}_{\text{ADAM}} \leftarrow$ max number of iterations to run ADAM optimizer
$\text{maxiters}_{\text{BFGS}} \leftarrow$ max number of iterations to run BFGS optimizer
$y \leftarrow$ real-world observations of the compartments
$T \leftarrow$ number of observations
$T' \leftarrow$ initial time span size
$k \leftarrow$ number of time steps to grow the time span
$\theta_1 \leftarrow$ randomly initialized
$\theta_2 \leftarrow$ randomly initialized
$p \leftarrow \{\gamma_0, \lambda_0, \theta_1, \theta_2\}$ ▷ Set the parameters initial values
$p_{\min} \leftarrow \{\gamma_0, \lambda_0, \theta_1, \theta_2\}$ ▷ Set the current minimizing paramters
$u \leftarrow \{S(0), E(0), I(0), R(0), D(0), N(0), C(0), T(0)\}$ ▷ Set the initial conditions
while $T' < T$ **do**
    $y' \leftarrow \{y(t) | t \in \{0, 1, \cdots, T'\}\}$ ▷ Select data on the current time span
    for $i \leftarrow 1, \text{maxiters}_{\text{ADAM}}$ **do**
        $\hat{y} \leftarrow \text{ODESolver}(u, p, (0, T'))$ ▷ Solve the initial value problem
        $p \leftarrow p - ADAM(\Delta_p \mathcal{L}(\hat{y}, y'))$ ▷ Update the parameters
        if $\mathcal{L}(\hat{y}, y') < \mathcal{L}_{\min}$ **then**
            $p_{\min} \leftarrow p$ ▷ Update the current minizing parameters
            $\mathcal{L}_{\min} \leftarrow \mathcal{L}(\hat{y}, y')$ ▷ Update the current smallest loss value
        end if
    end for
    $T' \leftarrow T' + k$ ▷ Grow the fitting region
end while
for $i \leftarrow 1, \text{maxiters}_{\text{BFGS}}$ **do**
    $\hat{y} \leftarrow \text{ODESolver}(u, p, (0, T))$ ▷ Solve the initial value problem
    $p \leftarrow p - BFGS(\Delta_p \mathcal{L}(\hat{y}, y))$ ▷ Update the parameters
    if $\mathcal{L}(\hat{y}, y) < \mathcal{L}_{\min}$ **then**
        $p_{\min} \leftarrow p$ ▷ Update the current minizing parameters
        $\mathcal{L}_{\min} \leftarrow \mathcal{L}(\hat{y}, y')$ ▷ Update the current smallest loss value
    end if
end for

process, the model was separately trained with data from each considered location. After the model training process was done, the performance of each version of the model was evaluated on out-of-sample data for 4 different future horizons: 1-week horizon, 2-week horizon, 3-week horizon, and 4-week horizon. The the evaluation results of all versions of the model were compared against each other to set if the covariates can improve the model performance.

Data at both the country level and the subdivision level was collected for a period of about 2 months (60 days) where the first 32 days of the period were used to train the model and the last 28 days of the period were used to evaluate the model out-of-sample performance. At each location, data was obtained starting from the first date where the total confirmed cases passed 500. The specific date ranges that were used at each location are given in Table 3.7. In the experiments, The initial conditions varied across different locations depending on the ground truth data. Table 3.8 presents the values that were chosen as the initial conditions across different locations. Furthermore, the initial parameters and their upper and lower bounds were initialized using the same method across different versions of the model and across different locations. Table 3.9 lists the values that were chosen as the initial parameters across different locations.

Location	First train date	Last train date	Last evaluation date
Vietnam	27th April 2021	28th May 2021	25th June 2021
Ho Chi Minh city	9th June 2021	10th July 2021	7th August 2021
Binh Duong	2nd July 2021	2nd August 2021	30th August 2021
Dong Nai	15th July 2021	15th August 2021	12th September 2021
Long An	13th July 2021	13th August 2021	10th September 2021
United States	1st July 2021	1st August 2020	29th August 2020
Los Angeles, California	1st July 2021	1st August 2020	29th August 2020
Cook, Illinois	1st July 2021	1st August 2020	29th August 2020
Harris, Texas	1st July 2021	1st August 2020	29th August 2020
Maricopa, Arizona	1st July 2021	1st August 2020	29th August 2020

Table 3.7: The date ranges that were chosen for the training process and the evaluation process of the model.

Location	$S(0)$	$E(0)$	$I(0)$	$R(0)$
Vietnam	9.75e7	25	5	2817
Ho Chi Minh city	9.22e6	173.5	34.7	360.1
Binh Duong	2.58e6	206.4	41.2	314.7
Dong Nai	3.17e6	295	59	194.8
Long An	1.71e6	217.8	43.5	300.5
United States	2.99e8	71890	14378	3.31e7
Los Angeles, California	8.78e6	2500	500	1.22e6
Cook, Illinois	4.59e6	615	123	546508
Harris, Texas	4.30e6	930	186	396430
Maricopa, Arizona	3.92e6	1325	265	549899

Table 3.8: Initial conditions at each modeled region for solving the systems of ODEs defined by each model. The initial values for the states $\{D, N, C, T\}$ were taken directly from the datasets, and they are not presented in this table. The value of $I(0)$ was assumed to be the number of new cases on the date of the first time step, and $E(0)$ was assumed to be 5 times the value of $I(0)$. Then $S(0)$ was calculated by subtracting $T(0)$ and $E(0)$ from average population. Lastly, $R(0)$ was calculated by subtracting $I(0)$ and $D(0)$ from $T(0)$.

Parameter	Value	Lower bound	Upper bound
γ	1/4	1/4	1/4
λ	1/14	1/14	1/14
θ_1	*glorot_normal initialization*	*N/A*	*N/A*
θ_2	*glorot_normal initialization*	*N/A*	*N/A*

Table 3.9: Initial parameters for solving the systems of ODEs defined by each model. The values of γ, λ, and α were chosen to match existing information about the Covid-19 Delta variant [2] where the mean incubation period was roughly 4 days, and the mean infective period was roughly 14 days.

At the country level, data for 2 countries Vietnam and the US were used to train and evaluate the performance of the model. Two versions of the model evaluated in this setting were the baseline model and the model that used Facebook's Movement Range Maps dataset. Because the SPC index was not defined at the country level, the version that used the SPC index was not considered. At the country's subdivision level, all three versions of the model were trained on Covid-19 time series data for the top 4 most populous counties in the US (Los Angeles - California, Cook County - Illinois, Harris County - Texas, and Maricopa County - Arizona) and the top 4 provinces in Vietnam that had the highest cases count during the outbreak starting from 27th April 2021 (Ho Chi Minh city, Binh Duong, Dong Nai, and Long An).

Once the data was obtained, the model was first trained using ADAM optimizer with learning rate set to 0.05, and the learning rate was exponentially decayed with the rate of 0.5 for each 1000 iterations until the minimum value of 0.00001 was reached. Using iterative growing of fit strategy, the model was first trained on a time span of size 4 which was then increased by 4 after each training session on the previous fitting time span finished. After each increment of the time span, the maximum number of iterations that ADAM was allowed to run for was increased by 500 starting from 500 for the first fitting time span. When the first training stage finished, BFGS optimizer was used for a maximum of 1000 iterations with the *initial_stepnorm* parameter set to 0.01 The hyperparameter ζ for adjusting the weighting factor for the loss function in Equation 3.4 was chosen to be -0.001 to place more importance on earlier time steps which minimized the chance of getting into a bad local minima. Finally, the hyperparameter λ was set to 0 to disable loss regularization.

In the 2 experiment settings, the choice of 500 cases was adopted from [40] in which the authors illustrated that the model could not generalize well enough when training with data before the 500 cases threshold. Noted that while the model might be sensitive to the choice of the initial conditions and the initial parameters, the effects of choosing a different set of initial conditions and initial parameters were not considered. The Covid-19 dataset, the Movement Range Maps dataset, and the derived SPC index all exhibited weekly seasonality in the data. In the Movement Range Maps dataset, the relative change in movement tended to drop to a much lower value while the stay-put ratio increased sharply in the weekend. This was expected as the data was based on human mobility where people went to work during the week and stayed rested on the weekend. With the Covid-19 cases data the SPC index, the value typically dipped in value in the weekend when fewer Covid-19 tests were made. Therefore, in the experiments, a 7-day moving average transform was applied to the Covid-19 cases data, the Movement Range Maps dataset, and the SPC index to remove the weekly seasonality exhibited in the measurements. The elimination of weekly seasonality in the data allowed the model to better capture the overall trends and helped to avoid confusing the model with highly fluctuating values. Moreover, after being transformed with a 7-day moving average, values from the Movement Range Maps dataset and the SPC index were scaled to have values ranging from 0 to 1 within the training time span. This ensured all the

input features to the ANNs had the same scale during training which would help improve the performance of the model.

3.5 Evaluation metrics

The out-of-sample performance of the models were compared using various metrics which include Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). For n observations, given the predictions $\hat{y}_i$ and the ground truth y_i, the definitions of these evaluation metrics are given as

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right|$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1} n(\hat{y}_i - y_i)^2}$$

These three metrics are commonly used for evaluating the quality of regression models. Multiple different metrics were considered because each has its advantages and disadvantages when use for evaluation. With MAE and MAPE, the final error increases linearly with the observation errors, and therefore, they are not sensitive to outliers. The value calculated using MAE is in the same unit as the data that was used as input, this makes it easier to understand and compare against other models. MAPE calculates the error in terms of percentage so the final value does not depend on the unit of the input data, this makes it useful when comparing different models that operate on different units of data. One downside of MAPE is that its value is infinite or undefined when the ground truth observation is close to zero or zero itself. Thus this should not be applied when the observed values are zero at some time steps. Unlike MAE and MAPE, RMSE places more emphasis on larger observation error, as a result, the metric is more sensitive to outliers. The value given by RMSE is also in the same unit as the input data similar to MAE. Furthermore, these metrics had also been used by other researchers to evaluate their models [6], [37], [38]. Hence, this gave us more insights into the performance of the model when compared against existing results that were applied to other countries and regions.

3.6 Software and hardware

The scripts for generating the datasets described in Section 3.1, the models defined in Section 3.2 and the training algorithm defined in Section 3.3 were implemented using the Julia programming language [51]. The implementation was written as a Julia package and was tested with Julia version 1.6.3. Julia was chosen for the implementation because of the strongly supported packages for solving differential equations and computing the gradients of those equations. The two main packages that were utilized are the *DifferentialEquations* package [57] for solving the system of ODEs defined by Equation 3.3 and the DiffEqFlux [9] package for computing the gradients and optimizing the system's parameters.

The experiments were conducted on two different hardware configurations. The first that was used for testing the models was the cloud computing instance provided by Google Collab [11]. The provided system ran on the Ubuntu 18.04 operating system and included a 2 cores Intel(R) Xeon(R) CPU with each core running at 2.30GHz, and 12Gb of memory. The

[11] https://colab.research.google.com

second hardware system that was used was a laptop ran on the Manjaro operating system with Linux kernel version 5.10.70-1-MANJARO and included a 2 cores Intel(R) Core(TM) i5-4260U CPU with each core running at 1.40GHz, and 4Gb of memory. While the models were expected to run on any operating system that was supported by Julia, it was not tested on any other operating systems besides Linux. Thus the model might not work as expected when running on a different operating system. Although the model had been tested and shown that it could run on a resource-limited system with only 4Gb memory, using a system with higher memory is recommended for the best user's experience. Because the Julia programming language utilizes Just-In-Time (JIT) compilation every time a line of code is run, the initial startup of the package can consume lots of hardware resources and crashes the program if there is not enough memory for the JIT compilation process. After the initial startup process is completed, the training process and evaluation process do not require as much hardware resources.

4. Results

In this section, the validation results obtained after performing the experimental procedure described in Section 3.4 are presented. First the model's outputs that were obtained after the model finished the training procedure were shown.

5. Discussion

6. Conclusion

References

[1] "WHO Coronavirus (COVID-19) Dashboard." (), [Online]. Available: https://covid19.who.int (visited on 09/30/2021).

[2] E. Mahase, "Delta variant: What is happening with transmission, hospital admissions, and restrictions?" *BMJ*, vol. 373, n1513, Jun. 15, 2021.

[3] "Rapid assessment of design and implementation of Government's 2nd support package for the affected by Covid-19 | UNDP in Viet Nam," UNDP. (), [Online]. Available: https://www.vn.undp.org/content/vietnam/en/home/library/Assessment2package.html (visited on 09/30/2021).

[4] I. Rahimi, F. Chen, and A. H. Gandomi, "A review on COVID-19 forecasting models," *Neural Computing & Applications*, pp. 1–11, Feb. 4, 2021.

[5] D. Adam, "Special report: The simulations driving the world's response to COVID-19," *Nature*, vol. 580, no. 7803, pp. 316–318, 7803 Apr. 2, 2020.

[6] E. L. Ray, N. Wattanachit, J. Niemi, A. H. Kanji, K. House, E. Y. Cramer, J. Bracher, A. Zheng, T. K. Yamana, X. Xiong, S. Woody, Y. Wang, L. Wang, R. L. Walraven, V. Tomar, K. Sherratt, D. Sheldon, R. C. Reiner, B. A. Prakash, D. Osthus, M. L. Li, E. C. Lee, U. Koyluoglu, P. Keskinocak, Y. Gu, Q. Gu, G. E. George, G. España, S. Corsetti, J. Chhatwal, S. Cavany, H. Biegel, M. Ben-Nun, J. Walker, R. Slayton, V. Lopez, M. Biggerstaff, M. A. Johansson, N. G. Reich, and o. b. o. t. C.-1. F. H. Consortium, "Ensemble Forecasts of Coronavirus Disease 2019 (COVID-19) in the U.S.," p. 2020.08.19.20177493, Aug. 22, 2020.

[7] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, Feb. 2019.

[8] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. "Neural Ordinary Differential Equations." (Dec. 13, 2019), [Online]. Available: http://arxiv.org/abs/1806.07366 (visited on 09/26/2021).

[9] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman. "Universal Differential Equations for Scientific Machine Learning." (Aug. 6, 2020), [Online]. Available: http://arxiv.org/abs/2001.04385 (visited on 09/11/2021).

[10] G. Cybenkot, "Approximation by superpositions of a sigmoidal function," p. 12,

[11] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[12] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[13] W. O. Kermack, A. G. McKendrick, and G. T. Walker, "A contribution to the mathematical theory of epidemics," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 115, no. 772, pp. 700–721, Aug. 1, 1927.

[14] ——, "Contributions to the mathematical theory of epidemics. II. —The problem of endemicity," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 138, no. 834, pp. 55–83, Oct. 1, 1932.

[15] ——, "Contributions to the mathematical theory of epidemics. III.—Further studies of the problem of endemicity," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 141, no. 843, pp. 94–122, Jul. 3, 1933.

[16] F. Brauer, "Compartmental Models in Epidemiology," in *Mathematical Epidemiology*, ser. Lecture Notes in Mathematics, F. Brauer, P. van den Driessche, and J. Wu, Eds., red. by J. .-M. Morel, F. Takens, and B. Teissier, vol. 1945, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 19–79.

[17] S. Zhao and H. Chen, "Modeling the epidemic dynamics and control of COVID-19 outbreak in China," *Quantitative Biology*, vol. 8, no. 1, pp. 11–19, Mar. 2020.

[18] S. He, Y. Peng, and K. Sun, "SEIR modeling of the COVID-19 and its dynamics," *Nonlinear Dynamics*, vol. 101, no. 3, pp. 1667–1680, Aug. 1, 2020.

[19] F. Ndaïrou, I. Area, J. J. Nieto, and D. F. Torres, "Mathematical modeling of COVID-19 transmission dynamics with a case study of Wuhan," *Chaos, Solitons, and Fractals*, vol. 135, p. 109 846, Jun. 2020.

[20] S. B. Bastos and D. O. Cajueiro, "Modeling and forecasting the early evolution of the Covid-19 pandemic in Brazil," *Scientific Reports*, vol. 10, no. 1, p. 19 457, Dec. 2020.

[21] K. Sarkar, S. Khajanchi, and J. J. Nieto, "Modeling and forecasting the COVID-19 pandemic in India," *Chaos, Solitons, and Fractals*, vol. 139, p. 110 049, Oct. 2020.

[22] C. C. Kerr, R. M. Stuart, D. Mistry, R. G. Abeysuriya, K. Rosenfeld, G. R. Hart, R. C. Núñez, J. A. Cohen, P. Selvaraj, B. Hagedorn, L. George, M. Jastrzębski, A. S. Izzo, G. Fowler, A. Palmer, D. Delport, N. Scott, S. L. Kelly, C. S. Bennette, B. G. Wagner, S. T. Chang, A. P. Oron, E. A. Wenger, J. Panovska-Griffiths, M. Famulare, and D. J. Klein, "Covasim: An agent-based model of COVID-19 dynamics and interventions," *PLOS Computational Biology*, vol. 17, no. 7, e1009149, Jul. 26, 2021.

[23] P. C. Silva, P. V. Batista, H. S. Lima, M. A. Alves, F. G. Guimarães, and R. C. Silva, "COVID-ABS: An agent-based model of COVID-19 epidemic to simulate health and economic effects of social distancing interventions," *Chaos, Solitons, and Fractals*, vol. 139, p. 110 088, Oct. 2020.

[24] N. Hoertel, M. Blachier, C. Blanco, M. Olfson, M. Massetti, M. S. Rico, F. Limosin, and H. Leleu, "A stochastic agent-based model of the SARS-CoV-2 epidemic in France," *Nature Medicine*, vol. 26, no. 9, pp. 1417–1421, 9 Sep. 2020.

[25] E. Dong, H. Du, and L. Gardner, "An interactive web-based dashboard to track COVID-19 in real time," *The Lancet Infectious Diseases*, vol. 20, no. 5, pp. 533–534, May 1, 2020.

[26] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.

[27] Z. Ceylan, "Estimation of COVID-19 prevalence in Italy, Spain, and France," *Science of The Total Environment*, vol. 729, p. 138 817, Aug. 10, 2020.

[28] M. H. D. M. Ribeiro, R. G. da Silva, V. C. Mariani, and L. d. S. Coelho, "Short-term forecasting COVID-19 cumulative confirmed cases: Perspectives for Brazil," *Chaos, Solitons & Fractals*, vol. 135, p. 109 853, Jun. 1, 2020.

[29] R. K. Singh, M. Rani, A. S. Bhagavathula, R. Sah, A. J. Rodriguez-Morales, H. Kalita, C. Nanda, S. Sharma, Y. D. Sharma, A. A. Rabaan, J. Rahmani, and P. Kumar, "Prediction of the COVID-19 Pandemic for the Top 15 Affected Countries: Advanced Autoregressive Integrated Moving Average (ARIMA) Model," *JMIR Public Health and Surveillance*, vol. 6, no. 2, e19115, May 13, 2020.

[30] V. K. R. Chimmula and L. Zhang, "Time series forecasting of COVID-19 transmission in Canada using LSTM networks," *Chaos, Solitons, and Fractals*, vol. 135, p. 109 864, Jun. 2020.

[31] A. Ramchandani, C. Fan, and A. Mostafavi, "DeepCOVIDNet: An Interpretable Deep Learning Model for Predictive Surveillance of COVID-19 Using Heterogeneous Features and Their Interactions," *IEEE Access*, vol. 8, pp. 159 915–159 930, 2020.

[32] F. Shahid, A. Zameer, and M. Muneeb, "Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM," *Chaos, Solitons, and Fractals*, vol. 140, p. 110 212, Nov. 2020.

[33] K. Roosa and G. Chowell, "Assessing parameter identifiability in compartmental dynamic models using a computational approach: Application to infectious disease transmission models," *Theoretical Biology and Medical Modelling*, vol. 16, no. 1, p. 1, Jan. 14, 2019.

[34] R. Li, S. Pei, B. Chen, Y. Song, T. Zhang, W. Yang, and J. Shaman, "Substantial undocumented infection facilitates the rapid dissemination of novel coronavirus (SARS-CoV-2)," *Science*, vol. 368, no. 6490, pp. 489–493, May 1, 2020.

[35] S. Chang, E. Pierson, P. W. Koh, J. Gerardin, B. Redbird, D. Grusky, and J. Leskovec, "Mobility network models of COVID-19 explain inequities and inform reopening," *Nature*, vol. 589, no. 7840, pp. 82–87, 7840 Jan. 2021.

[36] K. A. Schneider, G. A. Ngwa, M. Schwehm, L. Eichner, and M. Eichner, "The COVID-19 pandemic preparedness simulation tool: CovidSIM," *BMC Infectious Diseases*, vol. 20, p. 859, Nov. 19, 2020.

[37] IHME COVID-19 Forecasting Team, "Modeling COVID-19 scenarios for the United States," *Nature Medicine*, vol. 27, no. 1, pp. 94–105, Jan. 2021.

[38] S. O. Arık, C.-L. Li, J. Yoon, R. Sinha, A. Epshteyn, V. Menon, S. Singh, L. Zhang, N. Yoder, M. Nikoltchev, H. Nakhost, E. Kanal, and T. Pfister, "Interpretable Sequence Learning for COVID-19 Forecasting," p. 49,

[39] S. Y. Jung, H. Jo, H. Son, and H. J. Hwang, "Real-World Implications of a Rapidly Responsive COVID-19 Spread Model with Time-Dependent Parameters via Deep Learning: Model Development and Validation," *Journal of Medical Internet Research*, vol. 22, no. 9, e19907, Sep. 9, 2020.

[40] R. Dandekar, C. Rackauckas, and G. Barbastathis, "A Machine Learning-Aided Global Diagnostic and Comparative Tool to Assess Effect of Quarantine Control in COVID-19 Spread," *Patterns*, vol. 1, no. 9, p. 100 145, Dec. 11, 2020.

[41] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[42] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[43] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec. 1, 1989.

[44] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[45] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm." (Dec. 5, 2017), [Online]. Available: http://arxiv.org/abs/1712.01815 (visited on 10/04/2021).

[46] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 6088 Oct. 1986.

[47] S. Ruder. "An overview of gradient descent optimization algorithms." (Jun. 15, 2017), [Online]. Available: http://arxiv.org/abs/1609.04747 (visited on 10/05/2021).

[48] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam. "Searching for MobileNetV3." (Nov. 20, 2019), [Online]. Available: http://arxiv.org/abs/1905.02244 (visited on 10/09/2021).

[49] I. Lagaris, A. Likas, and D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, Sep. 1998.

[50] Y. Guo, X. Cao, L. Bainian, and M. Gao, "Solving Partial Differential Equations Using Deep Learning and Physical Constraints," *Applied Sciences*, vol. 10, p. 5917, Aug. 26, 2020.

[51] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman. "Julia: A fast dynamic language for technical computing." (2012).

[52] "Protecting privacy in Facebook mobility data during the COVID-19 response," Facebook Research. (Jun. 3, 2020), [Online]. Available: https://research.fb.com/blog/2020/06/protecting-privacy-in-facebook-mobility-data-during-the-covid-19-response/ (visited on 10/23/2021).

[53] Q. Deng, "Dynamics and Development of the COVID-19 Epidemic in the United States: A Compartmental Model Enhanced With Deep Learning Techniques," *Journal of Medical Internet Research*, vol. 22, no. 8, e21173, Aug. 21, 2020.

[54] D. Misra. "Mish: A Self Regularized Non-Monotonic Activation Function." (Aug. 13, 2020), [Online]. Available: http://arxiv.org/abs/1908.08681 (visited on 12/06/2021).

[55] T. Kuchler, D. Russel, and J. Stroebel, "The Geographic Spread of COVID-19 Correlates with the Structure of Social Networks as Measured by Facebook," National Bureau of Economic Research, Working Paper 26990, Apr. 2020.

[56] C. Tsitouras, "Runge–Kutta pairs of order 5(4) satisfying only the first column simplifying assumption," *Computers & Mathematics with Applications*, vol. 62, no. 2, pp. 770–775, Jul. 2011.

[57] C. Rackauckas and Q. Nie, "Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia," *Journal of Open Research Software*, vol. 5, no. 1, 2017.

[58] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization." (Jan. 29, 2017), [Online]. Available: http://arxiv.org/abs/1412.6980 (visited on 10/31/2021).

[59] C. G. BROYDEN, "The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations," *IMA Journal of Applied Mathematics*, vol. 6, no. 1, pp. 76–90, Mar. 1, 1970.

[60] R. Fletcher, "A new approach to variable metric algorithms," *The Computer Journal*, vol. 13, no. 3, pp. 317–322, Jan. 1, 1970.

[61] D. Goldfarb, "A family of variable-metric methods derived by variational means," *Mathematics of Computation*, vol. 24, no. 109, pp. 23–26, 1970.

[62] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Mathematics of Computation*, vol. 24, no. 111, pp. 647–656, 1970.

Glossary

ANN Artificial Neural Network. 1, 2, 6–9, 13–23, 28–30, 35

API Application Programming Interface. 1, 24

ARIMA Autoregressive Integrated Moving Average. 1, 6, 7

Bi-LSTM Bidirectional Long Short Term Memory. 1, 7

CNN Convolutional Neural Network. 1, 13

CSV Comma Separated Values. vi, 1, 23, 24, 26

FIPS Federation Information Processing Standards. 1, 26

GADM Database of Global Administrative Area. 1, 25, 26

GAN Generative Adversarial Network. 1, 13

GDP Gross Domestic Product. 1, 8

GPU Graphics Processing Unit. 1, 13

GRU Gated Recurrent Unit. 1, 7

ICU Intensive Care Unit. 1, 4

JIT Just-In-Time. 1, 36

JSON Javascript Object Notation. 1, 24

LSTM Long Short Term Memory. 1, 6, 7, 13

MAE Mean Absolute Error. 1, 35

MAPE Mean Absolute Percentage Error. 1, 35

MLP Multi-Layer Perceptron. 1, 13, 14

MSE Mean Squared Error. 1, 15, 22

NeuralODE Neural Ordinary Differential Equation. 1, 19, 20, 22

NPI Non-Pharmaceutical Intervention. 1, 8

ODE Ordinary Differential Equation. vi, 1, 2, 9, 10, 17, 19–21, 27, 31, 33–35

PDE Partial Differential Equation. 1, 17–19, 21

PINN Physics-Informed Neural Network. 1, 8, 17, 21

ReLU Rectified Linear Unit. v, 1, 16

RMSE Root Mean Squared Error. 1, 35

RNN Recurrent Neural Network. 1, 7, 14, 19

SCI Social Connectedness Index. vi, 1, 25, 26, 30

SEIR Susceptible-Exposed-Infective-Removed. 1, 4, 7, 8, 26, 27

SINDy Sparse Identification of Nonlinear Dynamical System. 1

SIR Susceptible-Infective-Removed. 1, 4, 5, 7, 9–12, 27

SPC Social Proximity to Cases. 1, 26, 30, 34

UDE Universal Differential Equation. 1, 9, 22, 23, 28, 29

US United States. 1, 5, 6, 8, 23, 26, 34