VIETNAMESE - GERMAN UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Frankfurt University Of Applied Sciences
Faculty 2: Computer Science and Engineering

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

Full name: Vo Le Tung
Matriculation number: 1276244

Fist supervisor: Assoc. Prof. Huynh Trung Hieu
Second supervisor: Assoc. Prof. Nguyen Tuan Duc

BACHELOR THESIS

Submitted in partial fulfillment of the requirements for
the Bachelor of Science degree in Computer Science
at the Vietnamese - German University

6th January 2022, Binh Duong, Vietnam

Abstract

The Covid-19 disease that emerged in 2019 had evolved at a fast pace and affected millions of lives both medically and economically. In the efforts to learn more about the disease and help to inform governments on policies making process, researchers had proposed various modeling techniques that can help with forecasting the evolution of the disease and estimating the different effects of government interventions in slowing down the spread of the virus. Nonetheless, not many studies had been conducted with data for Vietnam. We identified a gap in the knowledge about the dynamics of Covid-19 in Vietnam as well as the lack of study on the effects of government interventions on containing disease outbreaks. In addition, the recent failure in suppressing an outbreak in Vietnam after a long period of successfully keeping a low number of infections had proven that there was more to learn about Covid-19, and a model that was specifically tailored for the data availability in Vietnam could be an important tool for such task.

We implemented an infectious disease model for Covid-19 that integrated machine learning techniques and compartmental modeling for assessing the situations of the disease and for forecasting future progression. Additionally, we tried to improve the model forecast performance by incorporating mobility data and social network connections data as covariates in the model since the method has been proven to be effective by state-of-the-art model for Covid-19. We then analyzed and compared the performance of the model when different covariates were used to study whether they could improve the model forecast performance. By design, the model was explainable as it demonstrated how the different compartments developed, and domain-specific insights could be gained from the model because it was based on well-proven epidemiological knowledge. The explainability of the model is one desirable characteristic that ensures the model's credibility to epidemiologists and instills confidence in the end-users of the model. The model can be applied for different geographic resolutions, and we demonstrated it for Vietnam, the United States, provinces in Vietnam, and counties in the United States. We showed that the model was capable of learning valuable insights about Covid-19 and was able to make forecasts with high accuracy in many cases.

Declaration

I, Vo Le Tung, hereby declare that this bachelor thesis is a product of my own work, unless otherwise stated. I further declare that the thesis has not been previously or concurrently submitted for evaluation at any other institutions.

6th January 2022
Signature

Acknowledgement

I would like to show my gratitude to Prof. Huynh Trung Hieu for having guided me through the process of completing this thesis. His inputs have been invaluable and have helped me learn a lot more about the topic of the thesis. Additionally, I would like to thank my family, my friends, and all the faculty members for their support during my time at the Vietnamese - German University.

Contents

List of Figures

List of Tables

List of Algorithms

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. A systematic review of these methods were given in [4]. 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 Ordinary Differential Equations (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 choice 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 high quality data is highly available. 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.

To help the model better reflect the government intervention in China, where infective individuals were placed in quarantine immediately after they were tested positive for the SARS-NCoV-2 virus, a new compartmental model was proposed which considered three different classifications of the infective individuals: unquarantined infective, quarantined infective, and confirmed infective [17]. It was demonstrated to be effective when applied to data from thirty other countries and suggested to be a helpful tool for quantifying parameters and variables concerning the effects of quarantine or confirmation method. Similarly, multiple researches had modified the classic SIR and SEIR model with additional compartments such as hospitalizations and asymptomatic infections and applied it to data for Hubei province [18] and Wuhan [19]. Additionally, similar approaches for compartmental modelling had been applied to countries and areas that had been heavily affected by the disease such as Brazil [20], and India [21].

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, which 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 *agent* 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. Multiple agent-base models had been developed to help simulated the evolution of Covid-19, and they typically relied on information about the population size, age structure, transmission networks for different locations such as households, schools, workplaces, and long-term care facilities. Using these models, researchers had been simulating different

interventions scenarios in many locations such as King County in the US [22], Brazil [23], and France [24]. 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 can capture the temporal dependency structure of a time series. The ARIMA model had shown to have high accuracy when applied to data for Italy, Spain, France [27], and in 15 different countries [28]. In addition, the ARIMA model had been used in combination with other machine learning approaches to further improve its accuracy [29]. 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 in which 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. Long Short Term Memory (LSTM), the commonly utilized network for time series forecast, had been used to predict future transmission in Canada, where the complex disease dynamics had been successful learned [30]. Additionally, the Bidirectional Long Short Term Memory (Bi-LSTM) architecture had been shown to achieve the best overall predictive performance for Covid-19 when compared against the ARIMA statistical model, LSTM, and Gated Recurrent Unit (GRU) [31]. 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. 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. As a result, a new interpretable ANN architecture was proposed that incorporates multivariate spatial time series data to forecast the range of increase in COVID-19 infected cases in US counties [32]. The model

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

can utilize a wide range of heterogeneous features 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. Having an explainable model is extremely important for healthcare and public authorities to derive meaningful analyses that aid in the planning process.

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. For example, city-to-city movement data in China had been incorporated with a SIR model to simulate to effect of undocumented infective individuals [34]. Moreover, a fine-grained mobility network of the population's hourly movements had been integrated with a simple SEIR model by weighting the parameters with the hourly movements data, and the model was used to accurately simulate hourly infections in the US [35]. This model 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, 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. Different covariates had been used to model the transitions between compartments in the SEIR as a stepwise process to eliminate the assumption that time-delay in those transitions is exponentially distributed [36]. Similarly, covariates had also been used to informed how the transmission rate in the SEIR model changes over time under different Non-Pharmaceutical Intervention (NPI) in the US [37]. Furthermore, one of the most accurate model for US state-level forecasts, compared to other models submitted for the USs ensemble model [6], had been a compartmental model that incorporated weight matrices similar to ANNs to encode the covariates [38].

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.

Multiple ANNs had been used to learn how the parameters of classical SEIR model change over time [39]. This hybrid model was shown to be effective in capturing the disease dynamics when trained using Physics-Informed Neural Network (PINN) [7] with data for South Korea and several different cities within the country. Similarly, an ANN that learned the strength of quarantine had been incorporated in a modified SIR model, and the model was shown to be able to learn the correlation between the quarantine strength and the spread of the disease in 70 countries [40]

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, and the most basic model was described in [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].

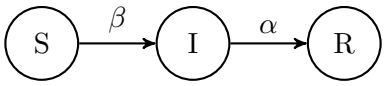

Figure 2.1: Graph of transitions between each compartment in the SIR model

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 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]

$$\begin{aligned} S' &= -\beta SI \\ I' &= \beta SI - \alpha I \\ R' &= \alpha I \end{aligned} \tag{2.1}$$

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

$$\mathcal{R}_0 = \frac{\beta S(0)}{\alpha}. \tag{2.2}$$

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]

$$ln\frac{S(0)}{S_\infty} = \mathcal{R}_0 \left[1 - \frac{S_\infty}{K}\right]. \tag{2.3}$$

Equation 2.3 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]

$$I' = (\beta K - \alpha)I, \tag{2.4}$$

and the initial growth rate is given by

$$r = \beta K - \alpha = \alpha(\mathcal{R}_0 - 1). \tag{2.5}$$

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

$$\beta = \frac{r + \alpha}{K}. \tag{2.6}$$

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]

$$I_{max} = S(0) + I(0) - \frac{\alpha}{\beta} lnS(0) - \frac{\alpha}{\beta} + \frac{\alpha}{\beta} ln\frac{\alpha}{\beta} \tag{2.7}$$

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

$$\beta(N) = \frac{C(N)}{N}, \tag{2.8}$$

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

$$\begin{aligned} S' &= -\beta(N)SI \\ I' &= \beta(N)SI - \alpha I \\ R' &= f\alpha I \\ N' &= -(1-f)\alpha I \end{aligned} \tag{2.9}$$

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]

$$\mathcal{R}_0 = \frac{K\beta(K)}{\alpha}. \tag{2.10}$$

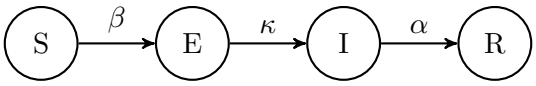

Figure 2.2: Graph of transitions between each compartment in the SEIR model

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]

$$\mathcal{R}^* = \frac{S\beta(N)}{\alpha}. \tag{2.11}$$

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

$$\mathcal{R}_0\left[1-\frac{S_\infty}{K}\right] \leq ln\frac{S(0)}{S_\infty} \leq \frac{\beta(0)(K-S_\infty)}{\alpha K} \tag{2.12}$$

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]

$$\begin{aligned} S' &= -\beta(N)SI \\ E' &= \beta(N)SI - \kappa E \\ I' &= \kappa E - \alpha I \\ R' &= f\alpha I \\ N' &= -(1-f)\alpha I \end{aligned} \tag{2.13}$$

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 2.10.

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

$$ln\frac{S(0)}{S_\infty} \geq \mathcal{R}_0\left[1-\frac{S_\infty}{K}\right] \tag{2.14}$$

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]

$$\begin{aligned} S' &= -\beta(N)S(I+\epsilon E) \\ E' &= \beta(N)S(I+\epsilon E) - \kappa E \\ I' &= \kappa E - \alpha I \\ R' &= f\alpha I \\ N' &= -(1-f)\alpha I \end{aligned} \tag{2.15}$$

where the basic reproduction number is calculated with

$$\mathcal{R}_0 = \frac{K\beta(K)}{\alpha} + \epsilon\frac{K\beta(K)}{\kappa}, \tag{2.16}$$

and the final size relation is the following inequality

$$ln\frac{S(0)}{S_\infty} \geq \mathcal{R}_0\left[1-\frac{S_\infty}{K}\right]-\frac{\epsilon\beta(K)}{\kappa}I_0. \tag{2.17}$$

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*.

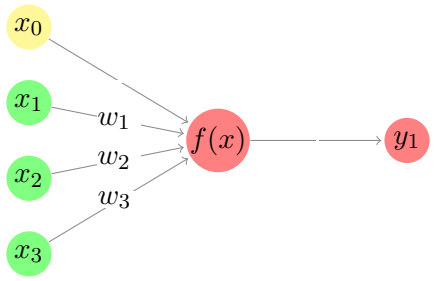

Figure 2.3: Graph representation of a perceptron described in Equation 2.18. x_0 is the bias b, and x_{1-3} are the input signals.

The idea for a computational model based on the brain was first proposed in 1943 [41], in which the authors 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, the concept of perceptrons, which is now considered the first ANN, was introduced [42]. Mathematically, a perceptron is expressed as

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.18}$$

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].

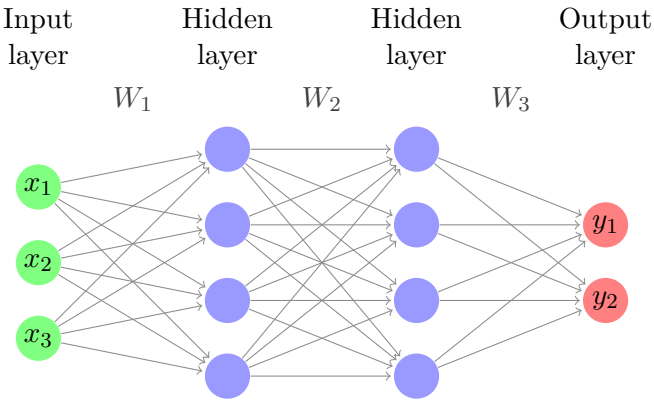

Figure 2.4: Graph representation of a multi-layer perceptron with four layers

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 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 Recurrent Neural Network (RNN). Mathematically, each node in an MLP computes the following values

$$\begin{aligned} a_i^k &= b_i^k + \sum_{j=1}^{r_{k-1}} w_{ij}^k z_j^{k-1} \\ z_i^k &= \phi(a_i^k), \end{aligned} \tag{2.19}$$

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

$$g(X) = \phi_n(W_n\phi_{n-1}(\cdots(W_2\phi_1(W_1X + b_1) + b_2) + \cdots) + b_n). \tag{2.20}$$

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

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

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 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

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^L} = \frac{\partial \mathcal{L}}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}\frac{\partial z_i^L}{\partial w_{ij}^L}. \tag{2.21}$$

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

$$\frac{\partial \mathcal{L}}{\partial b_i^L} = \frac{\partial \mathcal{L}}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}\frac{\partial z_i^L}{\partial b_i^L}. \tag{2.22}$$

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

$$\frac{\partial \mathcal{L}}{\partial a_j^{L-1}} = \sum_{i=1}^{r_{L-1}} \frac{\partial \mathcal{L}}{\partial a_i^L}\frac{\partial a_i^L}{\partial z_i^L}\frac{\partial z_i^L}{\partial a_j^{L-1}}. \tag{2.23}$$

Then the following gradients can be computed

$$\frac{\partial \mathcal{L}}{\partial w_{jk}^{L-1}} = \frac{\partial \mathcal{L}}{\partial a_j^{L-1}}\frac{a_j^{L-1}}{z_j^{L-1}}\frac{z_j^{L-1}}{w_{jk}^{L-1}}, \tag{2.24}$$

and

$$\frac{\partial \mathcal{L}}{\partial b_j^{L-1}} = \frac{\partial \mathcal{L}}{\partial a_j^{L-1}}\frac{a_j^{L-1}}{z_j^{L-1}}\frac{z_j^{L-1}}{b_j^{L-1}}, \tag{2.25}$$

This same approach can be applied for any layer in the ANN. As can be seen from Equation 2.21, Equation 2.22, and Equation 2.23, 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

$$\phi(z) = \frac{1}{1 + e^{-z}}, \tag{2.26}$$

and the hyperbolic tangent (tanh) function

$$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}. \tag{2.27}$$

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.

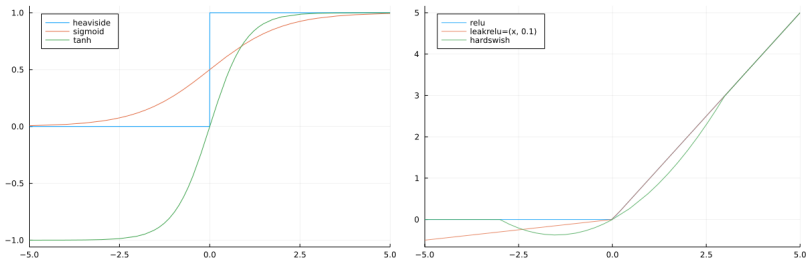

Figure 2.5: A visual comparison between the outputs of Heaviside, sigmoid, tanh, ReLU, Leaky ReLU, and hard swish activation functions

To solve the vanishing gradients problem, the Rectified Linear Unit (ReLU) activation function is used

$$\phi(z) = max(0, z). \tag{2.28}$$

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

$$\phi(z) = max(\alpha z, z), \tag{2.29}$$

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

$$\phi(z) = \begin{cases} 0 & \text{if } z \leq -3 \\ z & \text{if } z \geq 3 \\ z * (z+3)/6 & \text{otherwise} \end{cases} \tag{2.30}$$

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. Prior knowledge about dynamical systems, which is captured by mechanistic models, had not been utilized for training ANNs [7]. 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]

$$\frac{du}{dt} = f(u,t), \;\; t \in [0,1], \;\; u(0) = u_0 \tag{2.31}$$

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

$$\mathcal{NN}(t) \approx u(t). \tag{2.32}$$

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

$$MSE = \frac{1}{N}\sum_{i}^{N}\left(\frac{d\mathcal{NN}(t_i)}{dt} - f(\mathcal{NN}(t_i), t_i)\right)^2. \tag{2.33}$$

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

$$g(t) = u_0 + t\mathcal{NN}(t). \tag{2.34}$$

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

$$MSE = \frac{1}{N}\sum_{i}^{N}\left(\frac{dg(t_i)}{dt} - f(g(t_i), t_i)\right)^2. \tag{2.35}$$

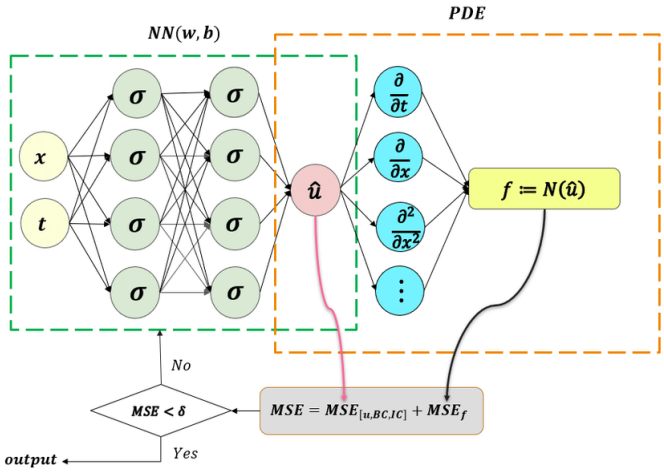

Figure 2.6: The schematic of PINNs for solving PDEs. Figure is taken from [50]

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

$$u_t + \mathcal{N}[u;\lambda] = 0,\ x \in \Omega,\ t \in [0,T], \tag{2.36}$$

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 2.36

$$f := u_t + \mathcal{N}[u]. \tag{2.37}$$

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 2.37, the function $\hat{f}(t,x)$ can be defined that describes the PDEs in terms of the approximation

$$\hat{f} := \hat{u}_t + \mathcal{N}[\hat{u}]. \tag{2.38}$$

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

$$MSE = MSE_u + MSE_f, \tag{2.39}$$

where

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |\hat{u}(t_u^i, x_u^i) - u^i|^2 \tag{2.40}$$

and

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |\hat{f}(t_f^i, x_f^i)|^2. \tag{2.41}$$

$\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ 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 2.36.

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. Empirical evidence showed that the ANN will converge to a global minimum under the condition that the differential equations are well-posed and their solution is unique [7]. 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. 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]

$$h_{t+1} = h_t + f(h_t, \theta_t) \tag{2.42}$$

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]

$$\frac{dh(t)}{dt} = f(h(t), t, \theta). \tag{2.43}$$

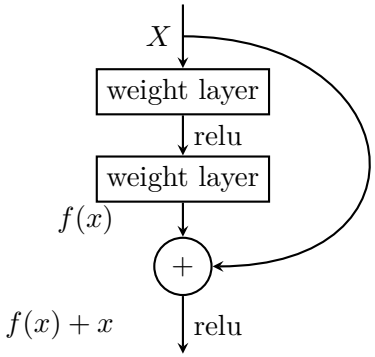

Figure 2.7: Example of a skip connection in residual network

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

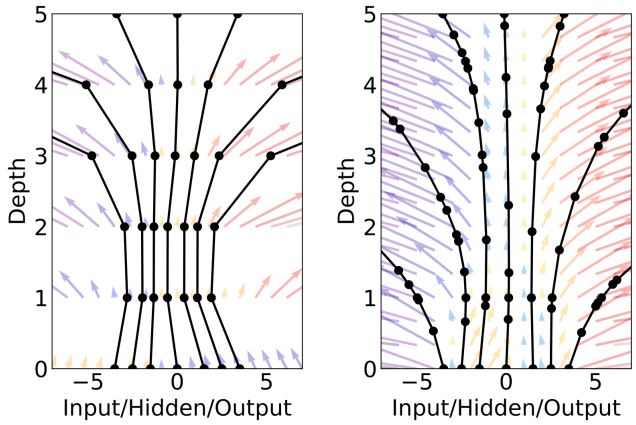

Figure 2.8: *Left*: A residual network defines discrete sequence of transformations. *Right*: A ODE network defines a vector field, which continuously transform the state. *Both*: Circles represent evaluation locations. Figure is taken from [8])

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, *adjoint sensitivity analysis* is used to compute the gradients [8]. 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 [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]

$$L(z(t_1)) = L\left(z(t_0) + \int_{t_0}^{t_1} f(z(t), t, \theta)dt\right) = L(ODESolve(z(t_0), f, t_0, t_1, \theta)). \tag{2.44}$$

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]

$$\frac{da(t)}{dt} = -a(t)^T \frac{\partial f(z(t), t, \theta)}{\partial z}. \tag{2.45}$$

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

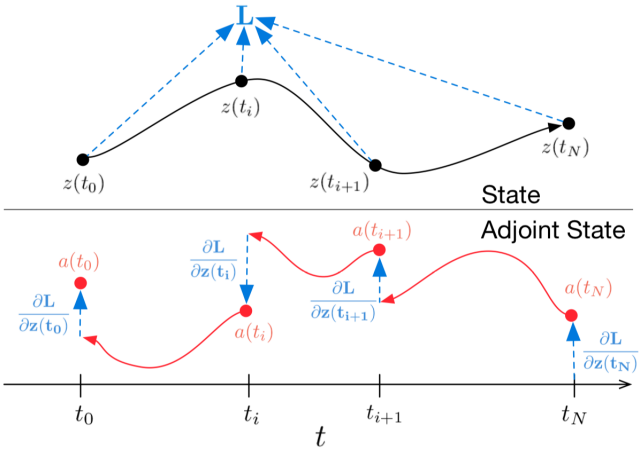

Figure 2.9: Reverse-mode differentiation of an ODE solution. An augment system contains the original state and the gradients of the loss with respect to the state is solved backwards in time. If the loss depends on the state at multiple observation times, the adjoin state is updated in the direction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

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

$$\frac{dL}{dz(t_0)} = \frac{\partial L}{\partial z(t_1)} + \int_{t_1}^{t_0} -a(t)^T \frac{\partial f(z(t), t, \theta)}{\partial z} dt. \tag{2.46}$$

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]

$$z(t) = z(t_1) + \int_{t_1}^{t} f(z(t), t, \theta) dt. \tag{2.47}$$

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

$$\frac{dL}{d\theta} = \int_{t_1}^{t_0} -a(t)^T \frac{\partial f(z(t), t, \theta)}{\partial \theta} dt. \tag{2.48}$$

The vector-Jacobian products $a(t)^T \frac{\partial f}{\partial z}$ and $a(t)^T \frac{\partial f}{\partial \theta}$ from Equation 2.46 and Equation 2.48 can be computed efficiently with automatic differentiation. The integrals in Equation 2.46, Equation 2.47, and Equation 2.48 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, Universal Differential Equation (UDE) was proposed which extended the approach taken by NeuralODEs [9]. 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]

$$u' = f(u, t, U_\theta(u, t)), \tag{2.49}$$

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 as a library in the Julia programming language [51], which will be utilized in this thesis [9].

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 to inform people about a potential outbreak of the disease. Here, UDEs [9] 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 and 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 dataset [1] from John Hopkins University [25] were utilized. The dataset had three separate Comma Separated Values (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 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.

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

[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: Structure of the CSV file obtained from the John Hopkins University [25]. Each row in the dataset 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: Structure of the table that contained the time series for a specify location. 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]

.

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 Covid-19 cases). Data from these two sources were available for the outbreak period in Vietnam starting from . From the first source, the number of confirmed cases and the number of deaths from Covid-19 for four cities/provinces 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 for Covid-19 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 model about the spatiotemporal factors that could have an effect on the disease dynamics.

[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.

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). This reflected how people in a geographic area reacted to different preventative measures from the government and was calculated for many subregions within a country. 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. The dataset from Facebook was available for Vietnam's level 2 divisions defined in the Database of Global Administrative Area (GADM), i.e., district level. For the US, this data was available at the county resolution.

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 Social Connectedness Index (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

$$\text{Social Connectedness Index}_{i,j} = \frac{\text{FB connections}_{i,j}}{\text{FB users}_i * \text{FB users}_j}, \tag{3.1}$$

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. With data 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 was 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, defined in Section 3.2. 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.

[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.

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

$$\begin{aligned} S' &= -\frac{\beta SI}{N} \\ E' &= \frac{\beta SI}{N} - \gamma E \\ I' &= \gamma E - \lambda I \\ R' &= (1-\alpha) * \lambda * I \\ D' &= \alpha * \lambda * I \\ N' &= -\alpha * \lambda * I \\ C' &= -C + \gamma * E \\ T' &= \gamma * E \end{aligned} \tag{3.2}$$

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. We noted that existing literature had also utilized the same formulation for the transition between the S and E compartments [17]–[21], [37], [38], [53]. Based on Equation 3.2, the basic reproduction number could be obtained by using the following equation [16]

$$\mathcal{R}_0 = \frac{\beta}{\gamma}. \tag{3.3}$$

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

$$\beta(t) = \mathcal{NN}_{\theta_1}(\mathcal{F}), \tag{3.4}$$

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 passed a certain threshold 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

$$\alpha(t) = \mathcal{NN}_{\theta_2}(\frac{t}{t_{\max}}, \frac{I(t-1)}{N(t-1)}, \frac{R(t-1)}{N(t-1)}, \frac{D(t-1)}{N(t-1)}), \tag{3.5}$$

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 3.2 can be written as

$$\begin{aligned}
S' &= -\frac{\beta(t)SI}{N} \\
E' &= \frac{\beta(t)SI}{N} - \gamma E \\
I' &= \gamma E - \lambda I \\
R' &= (1-\alpha(t)) * \lambda * I \\
D' &= \alpha(t) * \lambda * I \\
C' &= \gamma * E \\
N' &= -\alpha * \lambda * I \\
C' &= -C + \gamma * E \\
T' &= \gamma * E \\
\beta(t) &= \mathcal{NN}_{\theta_1}(\mathcal{F}) \\
\alpha(t) &= \mathcal{NN}_{\theta_2}(\frac{t}{t_{\max}}, \frac{I(t-1)}{N(t-1)}, \frac{R(t-1)}{N(t-1)}, \frac{D(t-1)}{N(t-1)})
\end{aligned} \tag{3.6}$$

The system of equations in Equation 3.6 gave us an UDE [9] that can be trained in an end-to-end fashion similar to the training process for ANNs. From Equation 3.3 and Equation 3.4, the basic reproduction rate of the disease at time t was derived to be

$$\mathcal{R}_t = \frac{\beta(t)}{\gamma} = \frac{\mathcal{NN}_{\theta_1}(\mathcal{F})}{\gamma}. \tag{3.7}$$

It had been shown 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 [9]. Additionally, a similar model that used a small ANN with 1 hidden layer, where the layer had 10 nodes, had been shown to achieve a good performance on Covid-19 data [40]. As a result, both the ANNs embedded in Equation 3.6 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

$$\nu_i = \nu_{i,L} + (\nu_{i,U} - \nu_{i,L}) * \sigma(z_i), \tag{3.8}$$

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 3.6, 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 3.6 are stipulated.

Informing the model with past states

The first set of covariates that were utilized as inputs for the ANN in Equation 3.6 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

$$\mathcal{F}(t) = \left\{ \frac{t}{t_{\max}}, \frac{S(t-1)}{N(t-1)}, \frac{E(t-1)}{N(t-1)}, \frac{I(t-1)}{N(t-1)} \right\}, \tag{3.9}$$

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 version 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, many have showed that mobility is an important predictor for the spread of Covid-19 [34], [35], [37]. 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 3.6. In this second version of the model, the set of covariates $\mathcal{F}$ given to the ANN at time step t was defined as

$$\mathcal{F}(t) = \left\{ \frac{t}{t_{\max}}, \frac{S(t-1)}{N(t-1)}, \frac{E(t-1)}{N(t-1)}, \frac{I(t-1)}{N(t-1)}, \text{MovementRange}(t) \right\}, \tag{3.10}$$

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 the US, 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

$$SPC_{i,t} = \sum_{j}^{C} \text{Cases per } 10\text{k}_{j,t} \frac{SCI_{i,j}}{\sum_{h}^{C} SCI_{i,h}}, \tag{3.11}$$

where $SPC_{i,t}$ is the SPC index of location i at time t, C is the number of locations, and Cases per $10\text{k}_{j,t}$ is 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

$$\mathcal{F}(t) = \left\{ \frac{t}{t_{\max}}, \frac{S(t-1)}{N(t-1)}, \frac{E(t-1)}{N(t-1)}, \frac{I(t-1)}{N(t-1)}, \text{MovementRange}(t), \text{SPC}(t) \right\}, \tag{3.12}$$

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 training 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

$$\begin{aligned} \gamma &= \gamma_L + (\gamma_U - \gamma_L) * \sigma(\gamma'), \\ \lambda &= \lambda_L + (\lambda_U - \lambda_L) * \sigma(\lambda'). \end{aligned} \tag{3.13}$$

The transformations 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

$$\mathcal{L}(\hat{y}, y) = \frac{1}{T} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \left[e^{\zeta t} \left(\frac{\hat{y}_{i,t} - y_{i,t}}{max(y_i) - min(y_i)} \right)^2 \right] + \frac{\lambda}{2T} (\|\theta_1\|_2^2 + \|\theta_2\|_2^2), \tag{3.14}$$

where N is the number of observable compartments, 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. The scaling also helped to prevent stability issues that could be encountered with stiff ODE [56]. 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 [57], implemented in the *DifferentialEquations.jl* package [58], 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 3.14 was carried out using local-gradient 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 [58]. Additionally, *InterpolatingAdjoint* algorithm was chosen because the method for computing the gradients through solving an augmented ODE backwards in time might not work when the ODE was stiff [56]. Even though there was no method for determining whether the model was a stiff ODE, choosing *InterpolatingAdjoint* lower the chance of miscalculating the gradients.

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. 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. In the experiments, ADAM [59] was chosen as the optimizer for the first stage, and BFGS [60]–[63] was chosen as the optimizer for the second stage. Algorithm 3 presents the general procedure that was carried out in the 2 stages optimization process. Parameters estimation with local gradient based algorithm could encounter issue with local minima, and there was a high chance that the model got stuck in a bad local minima.

This was especially true with time series data since the model could 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 was to place more weight to earlier data points which allowed for fitting to earlier portions first. This can be done by choosing a negative value for the hyperparameter ζ in Equation 3.14. 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 could first be 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, the model was then trained on this new time span, and the process repeated until T' equals T.

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

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 trained using the parameters optimization procedure described in Section 3.3. During the training 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 evaluation results of all versions of the model were then compared against each other to see 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 two and a half months (76 days) where the first 48 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. The period of 48 days was chosen to train the model because it was found that the model yielded the best predictions when trained with data for 48 days while still kept the training duration relatively low. The others training periods that had been validated against were 32 days and 60 days. With 32 days, the model could fit well to the data from the training period and provided useful insights about the disease but it failed to extrapolate and had the low accuracy when tested on out-of-sample data. While with 60 days, it took much longer to finish training the model but the accuracy did not significantly improves comparing to training with 48 days. At each location, data was obtained starting from the first date where the total confirmed cases passed 500 and the total number of deaths passed 10. 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 values 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. Note 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.

Location	First train date	Last train date	Last evaluation date
Vietnam	27th April 2021	13th June 2021	11th July 2021
Ho Chi Minh city	9th June 2021	26th July 2021	23rd August 2021
Binh Duong	2nd July 2021	18th August 2021	15th September 2021
Dong Nai	15th July 2021	31st August 2021	28th September 2021
Long An	13th July 2021	29th August 2021	26th September 2021
United States	1st July 2021	17th August 2021	14th September 2021
Los Angeles, California	1st July 2021	17th August 2021	14th September 2021
Cook, Illinois	1st July 2021	17th August 2021	14th September 2021
Harris, Texas	1st July 2021	17th August 2021	14th September 2021
Maricopa, Arizona	1st July 2021	17th August 2021	14th September 2021

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

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 counts during the outbreak starting from 27th April 2021 (Ho Chi Minh city, Binh Duong, Dong Nai, and Long An).

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
β	*N/A*	0.05	1.67
γ	1/4	1/4	1/4
λ	1/14	1/14	1/14
α	*N/A*	0.005	0.05
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 and their boundaries. The values of γ and λ were chosen as fixed values 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. For fixed parameters, the lower bounds and the upper bounds are set to be equal

Once the data was obtained, the model was first trained using ADAM optimizer with learning rate set to 0.05 which 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. At each time span in the first stage, the ADAM optimizer was configured to run for the maximum of 1000 iterations. 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 3.14 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 0.00001 to help reduce the chance of over-fitting.

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 majority of people would go to work during the week and stayed rested on the weekend. With the Covid-19 cases data the SPC index, the values typically dipped in the weekend most because fewer Covid-19 tests

were made. Therefore, in the experiments, a 7-day moving average transform was applied to these datasets 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 to improve the performance of the model. Because issues with bad local minima could occur when learning from time series data, we trained the model at each location for 5 times and selected the set of weights that produced the minimum training loss among the samples.

3.5 Evaluation metrics

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

$$\begin{aligned} MAE &= \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \\ MAPE &= \frac{100}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \\ RMSE &= \sqrt{\frac{1}{n}\sum_{i=1} n(\hat{y}_i - y_i)^2} \end{aligned} \tag{3.15}$$

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. Hence, using the same metrics gave us more insights into the performance of the model when compared against existing results that were obtained from other models for the same countries or regions [6], [37], [38].

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 1.7.0. 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 [58] for solving the system of ODEs defined by Equation 3.6 and the *DiffEqFlux* [9] package for computing the gradients and estimating 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 compute instance provided by Google Cloud [11]. The system ran on the Ubuntu 20.04 operating system and was configured using the *n2-standard-8* machine type provided by Google [12]. The second hardware system was a laptop running on the Manjaro operating system with Linux kernel version *5.10.70-1-MANJARO* and included a 2-core *Intel(R) Core(TM) i5-4260U* CPU with each core running at 1.40GHz, and the system has 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 system 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 if the script is running along with other processes. Because the Julia programming language utilizes Just-In-Time (JIT) compilation on the first time a block of code is run to create optimized native code, the initial startup of the package can consume lots of hardware resources and crashes 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 resource.

3.6.1 Package overview

The core of the package is the function that implements the system of ODEs given by Equation 3.2. With Julia and the *DifferentialEquations* [13] package, Equation 3.2 can be implemented as a simple function given by Figure B.1. Using the function shown in Figure B.1, proxy functions on top of it can be implemented to make changes to the values of the parameters before using them to compute the derivatives. Several structs are defined to help to create these proxies for the different versions of the model, and their general structure is illustrated in Figure B.3. By utilizing the Julia feature that allows us to make an object of a struct callable like any function, a proxy function around the function shown in Figure B.1 can be created that makes use of the data held by the struct in Figure B.3. The definition of such proxy function is shown in Figure B.4. To solve the system of ODE given by the function in Figure B.4, the function *solve* together with the type *ODEProblem* from the *DifferentialEquations* package are used. The general usage of the function is given by Figure 3.1. Because the *solve* function has several arguments that can be chosen, a callable struct is defined to hold these arguments and provide an object that can be invoked with the minimal number of arguments. The definition of this struct is given by Figure B.5.

The package utilizes the function *sciml_train* from the *DiffEqFlux* [14] package to train the model. This function defines all the necessary steps for estimating the set of parameters that minimizes an arbitrary loss function using a specific optimizer. The general usage of the function is illustrated by Figure 3.2. Because the *sciml_train* function only works with loss functions that accept the system parameters as the only argument, we define a callable struct that contains all the other data that is required when calculating the loss in addition to the system parameters. Figure B.6 shows the definition of this helper struct.

Most of the core functions and structs in this package are parametric over its parameters or fields to allow for a wide range of data types that can be used with them while not

[11] https://cloud.google.com/compute
[12] https://cloud.google.com/compute/docs/compute-optimized-machines
[13] https://github.com/SciML/DifferentialEquations.jl
[14] https://github.com/SciML/DiffEqFlux.jl

```
model = SEIRDFbMobility2( /* ... */ )
u0 = [ /* ... */ ]
tspan = (0, 47)
tsteps = tspan[1]:1:tspan[2]
problem = ODEProblem(model, u0, tspan)
solution = solve(problem, Tsit5(), saveat=tsteps)
```

Figure 3.1: A simple usage of the *solve* function from the *DifferentialEquations* package. Here we are solving the system of ODEs on the time span from 0 to 47 and save the system state at an interval of 1 time step. *model* is an object of the struct *SEIRDFbMobility2* shown in Figure B.3, *u0* is the system initial state, *tspan* is the modeled period, and *tsteps* is the time steps where the system state will be saved for output stored by *solution*. Additionally, the first argument to the constructor of the type *ODEProblem* could take any callable object that can be invoked with the arguments *(du,u,p,t)*, where *du* is a vector that will be modified in-place to contain the system derivatives at each time step, u is a vector contains the current state of the system, p is a vector contains all the system parameters, and t is the value of the current time step.

.

```
function loss(params)
    /* ... */
end
params = [ /* ... */ ]
opt = ADAM(lr=0.01)
res = DiffEqFlux.sciml_train(loss, params, opt; maxiters=500)
```

Figure 3.2: A simple usage of the *sciml_train* function from the *DiffEqFlux* package. Here, we are finding the set of parameters that can minimize the loss function defined by *loss*, which takes a set of parameters and returns a scalar loss value calculated using the given parameters. Furthermore, in this example, the ADAM optimizer is used for 500 iterations, and the variable *params* holds that initial set of parameters.

sacrificing performance. Using parametric type helps to create *type-stable* functions where the type of any variables used by a function is known at compile time. As a result, Julia will produce more performance native code as there is no runtime type checking and no heap allocation caused by polymorphic variables.

Besides the core functionalities described above, the developed package also contains structs for representing time series data and functions for data collecting and preprocessing. Functions that collect and process the datasets used in the experiments of the thesis are grouped into modules based on where the datasets originated from. The module *FacebookData* contains functions for extracting data from the Movement Range Maps dataset and the SCI dataset. Moreover, this module also contains the function for calculating the SPC index from the SCI index. The module *JHUCSSEData* contains functions for extracting Covid-19 cases data and for getting US counties population from the datasets from the John Hopkins University. The module *PopulationData* contains functions for combining Vietnam's population data from the Vietnam General Statistics Office with the administrative levels data from the GADM. The module *VnExpressData* contains functions for downloading Covid-19 data from VnExpress [15] and the module *VnCdcData* contains functions for parsing the data

[15] https://vnexpress.net

from Vietnam General Department of Preventative Medicine [16]. All the data collecting and processing modules depended on the package *DataDeps* [17] for managing data dependencies. The *DataDeps* package simplified the process of downloading static files from servers into a local machine and provided the program the paths to all the downloaded files.

Finally, using the package, several experiments are set up based on the configurations given in Section 3.4, and the functions for setting up these experiments are included along with the package. The implementation the loss function defined by Equation 3.14 is included among these functions, and its definition is shown in Figure B.2. Moreover, the experiments can be run in parallel when training with data for multiple different locations by using the threads that are available in the system. For easy access to the training and evaluation of the experiments, the package *ArgParse* [18] is used to provide a command-line interface for running the experiments with predefined locations.

[16] https://ncov.vncdc.gov.vn
[17] https://github.com/oxinabox/DataDeps.jl
[18] https://github.com/carlobaldassi/ArgParse.jl

4. Results

This section presents the results obtained from the different versions of the model after having trained with the procedure described in Section 3.4. From the results, it was found that all three versions of the model were capable of following the trends of the disease when applied to the considered locations, except for Ho Chi Minh city and Vietnam. Moreover, the comparative analysis between the different versions revealed that the model's performance on out-of-sample data did not improve when the chosen covariates were included to inform the model.

4.1 Model predictions errors for Vietnam and the United States

When applied to data for the considered countries, namely Vietnam and the United States, both the baseline model and the second version show consistent results in predicting the trends of the disease. Both versions were able to produce predictions that closely followed the actual numbers for shorter forecast horizons, while with longer horizons the errors quickly accumulated. These results demonstrated the model's inability to capture the long-term dynamics of the disease even when mobility data was incorporated. Moreover, it concluded that there was not a clear improvement in the model's performance when data from Facebook's Movement Range Maps dataset for the considered locations was included in the model. In the case of Vietnam, both versions were able to extrapolate from the training period and predict the number of deaths with good accuracy for up to 21 days. Furthermore, they were able to closely follow the trend in the number of cumulative cases for up to 14 days after the training period. However, both failed to predict the sharp growth in the number of new cases as can be seen in Figure A.2. Figure A.2 also shows that when trained with data for the US, both versions produced similar results when predicting the trend of the disease. One noticeable difference between the results obtained from both versions was in the forecasted number of cases where the baseline model underestimated the trend while the second version overestimated it.

According to Table 4.1, when predicting the number of deaths for the first three forecast horizons, the baseline model showed better results on data for the US whereas the second version of the model performed better on data for Vietnam. With a forecast horizon of 28 days, the second version of the model achieved lower errors on both datasets.

When predicting the number of new cases, Table 4.2 shows that the second version of the model had better performance on Vietnam's dataset when predicting on all four different forecast horizons. Other cases where the baseline model performed better were with the 7-day predictions and 28-day predictions for the US. As previously mentioned, both versions failed to produce good predictions for the number of new cases for Vietnam with the errors calculated using MAPE of 25% for 7-day prediction and up to 50% for 28-day predictions.

Lastly, comparing the predictions errors for the number of cumulative cases of the two versions revealed that the baseline model made better predictions for the US for all forecast horizons. In contrast, the predictions for Vietnam made by the second version were better

Days	Loc.	Baseline			2nd. Version		
		MAE	MAPE	RMSE	MAE	MAPE	RMSE
7	VN	2.409	4.033	2.470	**2.356**	**3.943**	**2.420**
	US	**727.905**	**0.116**	**834.818**	826.050	0.132	949.121
14	VN	2.222	3.549	2.321	**2.221**	**3.540**	**2.310**
	US	**1689.681**	**0.267**	**2022.840**	1871.675	0.296	2225.024
21	VN	1.743	2.697	1.969	**1.676**	**2.609**	**1.929**
	US	**2986.879**	**0.467**	**3662.566**	3155.909	0.494	3803.063
28	VN	3.530	4.318	5.408	**3.301**	**4.062**	**5.074**
	US	4512.893	0.697	5576.519	**4495.416**	**0.695**	**5401.567**

Table 4.1: Out-of-sample errors of the model's predictions on the number of deaths for Vietnam and the United States. The lowest errors for each evaluation metrics at each location are highlighted.

Days	Loc.	Baseline			2nd. Version		
		MAE	MAPE	RMSE	MAE	MAPE	RMSE
7	VN	96.166	27.928	106.205	**88.005**	**25.491**	**97.969**
	US	**2006.080**	**1.343**	**2415.311**	2051.455	1.370	2976.705
14	VN	110.478	32.456	117.440	**99.469**	**29.140**	**106.410**
	US	6775.296	4.326	8842.444	**6004.020**	**3.849**	**7582.764**
21	VN	174.686	41.840	207.332	**161.454**	**38.390**	**194.470**
	US	12426.818	7.766	16039.493	**10781.952**	**6.974**	**14765.447**
28	VN	357.254	52.338	502.876	**342.409**	**49.266**	**490.063**
	US	16943.424	**10.771**	**21371.281**	**16516.025**	10.859	21514.390

Table 4.2: Out-of-sample errors of the model's predictions on the number of new cases for Vietnam and the United States. The lowest errors for each evaluation metrics at each location are highlighted.

Days	Loc	Baseline			2nd. Version		
		MAE	MAPE	RMSE	MAE	MAPE	RMSE
7	VN	**234.690**	**2.022**	**312.078**	241.284	2.081	313.870
	US	**39553.840**	**0.106**	**39745.361**	61863.751	0.166	62287.404
14	VN	672.504	5.098	831.731	**643.697**	**4.894**	**785.383**
	US	**30561.907**	**0.081**	**33712.852**	88894.016	0.233	94595.820
21	VN	1285.683	8.495	1645.257	**1206.696**	**7.994**	**1533.632**
	US	**71921.579**	**0.184**	**97776.403**	132684.773	0.341	151150.555
28	VN	2650.472	14.012	3786.242	**2512.913**	**13.276**	**3605.530**
	US	**137006.424**	**0.342**	**189405.400**	210325.947	0.529	259678.415

Table 4.3: Out-of-sample errors of the model's predictions on the number of cumulative cases for Vietnam and the United States. The lowest errors for each evaluation metrics at each location are highlighted.

across different horizons except for the 7-day horizon where it performed slightly worse than the baseline model.

Overall, we can see that the baseline model performed better in some instances while in other instances, the second version performed better. In cases where one model produced better predictions than the other, the difference in predictions errors between the models was not significant, and the differences in predictions errors can be attributed to the stochastic

training process. Hence, based on these results, mobility data did not have a positive effect on the model's predictions.

4.1.1 Reproduction number and fatality rate

Looking at Figure 4.1, both versions of the model learned the same decreasing trend in the effective reproduction number and the fatality rate in Vietnam. The reproduction number at the start of the simulated period reached as high as 4.5 with the baseline model and as high as 3.5 with the second version. Both versions learned that the effective reproduction number went below 1 at around the 20th time step. Additionally, the fatality rate predicted by both versions were different at the start but then followed a decreasing trend and was highest at 0.008

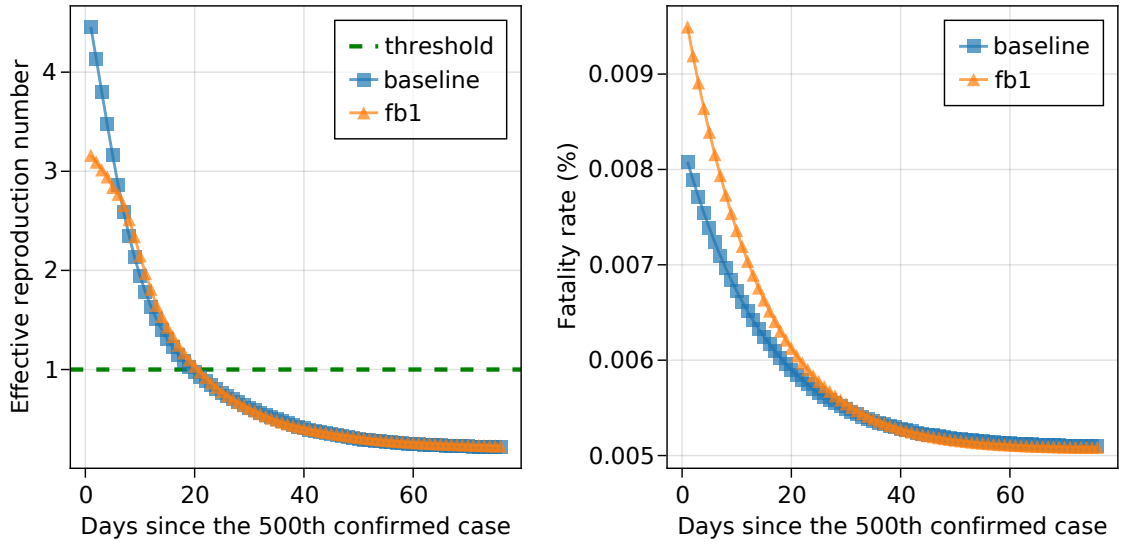

Figure 4.1: The effective reproduction number (left) and the fatality rate (right) for Vietnam learned by different versions of the model

Considering data for the US, both versions learned the same trend of the fatality rate. It can be observed that the fatality rate was at its highest of around 4.5% at the 0th time steps and decreased exponentially as time went on. Then at around the 60th time step, the fatality rate was predicted to start to slowly increase. For the effective reproduction number, the baseline model predicted the highest value of 1 at the 0th time step followed by a gradual decrease until the end of the simulation. On the other hand, the second version predicted a higher effective reproduction number of around 1.2 at the 0th time step which then decreased for the next 20 time steps before it started to rise for 10 time steps and finally died out.

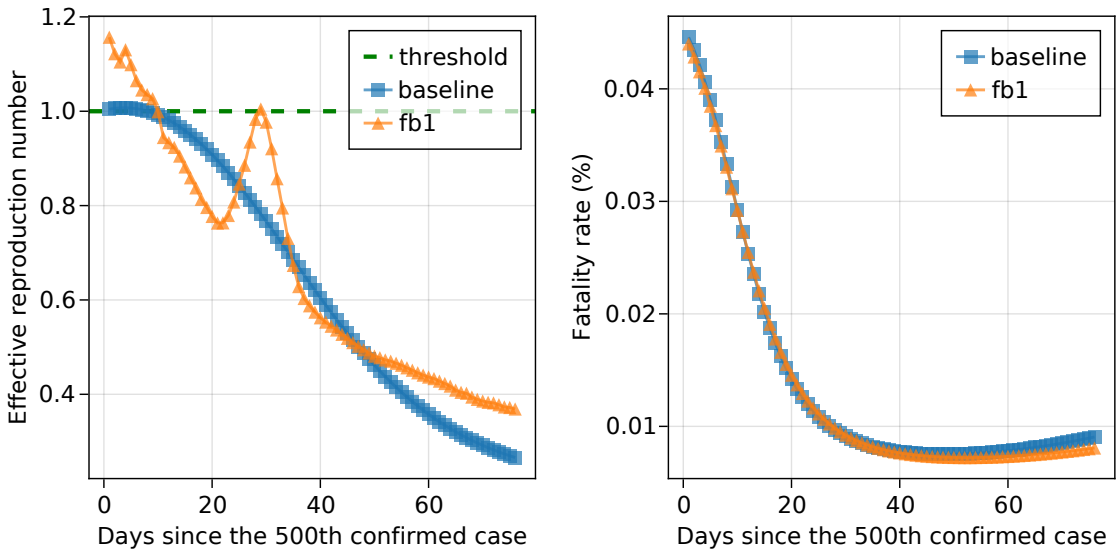

Figure 4.2: The effective reproduction number (left) and the fatality rate (right) for the United States learned by different versions of the model

4.2 Model predictions errors for counties in the United States

When applied to data for different counties in the US, all three versions of the model also showed consistent results and predicted similar trends across all locations. Similar to the results obtained from the experiments with country-level data, all versions were able to simultaneously predict the number of cases in different categories with high accuracy for up to 14 days. After the 14-day mark, the predictions errors started to increase as more uncertainties were accumulated when the dynamics were extrapolated further into the future. Additionally, the results once again indicated that our method of encoding data from Facebook into the model did not improve the long-term predicting performance of the model.

When predicting the number of deaths of the disease, Table 4.4 shows that the baseline model significantly outperformed the other two versions with data for Cook (Illinois) and Los Angeles (California) on all forecast horizons. Here, the predictions errors of the baseline model were less than half of the predictions errors from the other two versions across all metrics. On the other hand, the third version of the model achieved the lowest error when predicting for Harris (Texas) on all horizons. While the third version had the best performance across all metrics when predicting for Maricopa (Arizona) on 7-day and 14-day horizons, at the 21-day horizon, its performance started to diminish. With a 21-day horizon, the baseline model's predictions for Maricopa (Arizona) had the lowest RMSE and with the other metrics closely followed that of the third version. Then with a 28-day horizon, the baseline model completely outperformed the third version across all metrics.

On the predictions for the number of new cases, Table 4.5 shows results similar to the performance of the different versions when predicting the number of deaths. Here, the baseline model significantly outperformed the others across all metrics when predicting the number of new cases for Cook (Illinois) and Los Angeles (California) for all forecast horizons. When predicting the number of new cases for Harris (Texas), the third version had the best performance on the 7-day horizon, however, with 14-day and 21-day horizons, the third version only had the lowest errors with the MAE and RMSE metrics, whereas the best MAPEs for those horizons were achieved by the baseline model. The baseline model additionally had the lowest MAE and MAPE when predicting for Harris (Texas) on the

28-day horizon, while the lowest RMSE was produced by the third version. Furthermore, the third version of the model had the lowest errors across all metrics when predicting for Maricopa (Arizona) for the 7-day and 21-day horizons. With the 14-day predictions for Maricopa (Arizona), the second version of the model had the lowest predictions error with RMSE while the third version of the model had the lowest predictions error with MAE and MAPE. Lastly, with the 28-days predictions for Maricopa (Arizona), the second version had the lowest errors across all metrics.

When predicting the number of cumulative cases, Table 4.6 shows that for the 7-day forecast horizon, the baseline model performed best with data for Los Angeles (California), while the third version had the lowest predictions errors for all the other locations. For the 14-day and 21-day forecast horizons, the third version had the lowest predictions errors for Harris (Texas) and Maricopa (Arizona), while the baseline had the lowest predictions errors for Cook (Illinois) and Los Angeles (California). Finally, for the 28-day forecast horizon, the baseline only achieved the best performance on data for Harris (Texas) and the third version performed best for all the other locations.

Overall, the results show that there was not a significant difference in the long-term forecasting performance across the three different versions. In addition, it was observed from the errors tables that the model's performance was largely dependent on the chosen location where the baseline model performed generally better with data for Cook (Illinois) and Los Angeles (California), and the third version performed generally better with data for Harris (Texas) and Maricopa (Arizona).

Days	Loc.	Baseline			2nd. Version			3rd. Version		
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
7	Cook, IL	**5.847**	**0.055**	**6.291**	18.390	0.173	19.257	21.309	0.200	22.031
	Harris, TX	46.021	0.666	50.000	38.705	0.560	43.120	**32.571**	**0.471**	**36.267**
	Los Angeless, CA	**28.781**	**0.115**	**31.633**	51.605	0.206	54.813	50.488	0.202	53.633
	Maricopa, AZ	39.739	0.374	40.227	38.180	0.360	38.746	**36.632**	**0.345**	**37.215**
14	Cook, IL	**13.181**	**0.123**	**15.800**	31.772	0.297	35.375	34.551	0.323	37.832
	Harris, TX	95.339	1.355	111.180	87.944	1.249	104.920	**74.100**	**1.052**	**88.397**
	Los Angeless, CA	**54.539**	**0.217**	**61.814**	87.975	0.350	97.176	86.045	0.343	95.029
	Maricopa, AZ	55.439	0.519	58.393	54.897	0.514	58.269	**53.215**	**0.498**	**56.632**
21	Cook, IL	**21.380**	**0.199**	**25.491**	45.720	0.426	51.581	48.620	0.453	54.215
	Harris, TX	158.343	2.205	189.162	152.570	2.123	185.661	**127.313**	**1.772**	**154.368**
	Los Angeless, CA	**82.513**	**0.327**	**95.410**	126.976	0.503	143.413	123.921	0.491	139.857
	Maricopa, AZ	76.472	0.711	**83.882**	77.590	0.721	86.019	**75.672**	**0.703**	84.128
28	Cook, IL	**29.629**	**0.275**	**35.088**	59.121	0.549	66.872	62.355	0.579	69.979
	Harris, TX	227.124	3.099	272.682	225.190	3.070	274.928	**184.261**	**2.513**	**223.061**
	Los Angeless, CA	**117.110**	**0.461**	**138.626**	171.721	0.677	197.699	167.210	0.659	192.284
	Maricopa, AZ	**101.192**	**0.933**	**114.101**	104.663	0.965	119.408	102.385	0.944	117.066

Table 4.4: Out-of-sample errors of the model's predictions on the number of deaths for the counties in the United States. The lowest errors for each evaluation metrics at each location are highlighted.

Days	Loc.	Baseline			2nd. Version			3rd. Version		
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
7	Cook, IL	**22.545**	**2.377**	**28.432**	55.668	6.041	66.778	41.352	4.467	55.134
	Harris, TX	1022.438	30.850	1193.354	1093.142	32.772	1280.746	**797.754**	**26.782**	**865.731**
	Los Angeless, CA	**981.461**	**20.945**	**1140.886**	1126.805	24.141	1297.286	1082.929	23.181	1248.996
	Maricopa, AZ	95.129	4.730	99.906	19.956	0.987	23.867	**19.382**	**0.968**	**21.617**
14	Cook, IL	**35.341**	**3.517**	**45.173**	90.017	9.156	101.192	59.667	6.104	68.594
	Harris, TX	890.257	**28.652**	1093.096	984.436	30.810	1205.333	**790.239**	30.445	**874.964**
	Los Angeless, CA	**698.901**	**15.302**	**950.789**	802.626	17.610	1085.485	768.943	16.918	1038.125
	Maricopa, AZ	136.129	6.883	172.143	69.022	3.427	**94.846**	**62.939**	**3.176**	95.737
21	Cook, IL	**70.168**	**7.131**	**92.661**	136.784	14.489	177.086	96.828	10.550	144.579
	Harris, TX	715.160	**23.718**	932.114	963.073	32.888	1132.567	**671.354**	26.827	**775.119**
	Los Angeless, CA	**477.179**	**10.682**	**776.682**	557.786	12.785	887.881	584.712	14.463	857.916
	Maricopa, AZ	139.674	6.888	169.314	92.801	4.357	129.381	**76.112**	**3.609**	**116.774**
28	Cook, IL	**83.696**	**8.105**	**119.396**	184.792	19.135	230.863	139.144	14.665	190.726
	Harris, TX	**605.536**	**20.796**	824.160	982.211	35.537	1123.076	689.374	28.746	**779.813**
	Los Angeless, CA	**412.908**	**11.617**	**686.484**	498.917	14.897	794.109	546.785	17.824	784.312
	Maricopa, AZ	217.947	11.510	271.885	**104.125**	**5.223**	**134.620**	109.935	5.689	149.951

Table 4.5: Out-of-sample errors of the model's predictions on the number of new cases for the counties in the United States. The lowest errors for each evaluation metrics at each location are highlighted.

Days	Loc.	Baseline			2nd. Version			3rd. Version		
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
7	Cook, IL	678.757	0.117	679.641	750.786	0.130	761.605	**595.890**	**0.103**	**604.919**
	Harris, TX	2429.798	0.520	3193.800	2629.532	0.562	3482.363	**1535.176**	**0.333**	**1736.348**
	Los Angeless, CA	**2351.443**	**0.172**	**2920.659**	2898.771	0.212	3633.093	2545.519	0.186	3107.429
	Maricopa, AZ	1491.513	0.242	1510.229	974.924	0.159	975.073	**410.099**	**0.067**	**412.091**
14	Cook, IL	**612.953**	**0.105**	**621.265**	1129.956	0.194	1212.272	874.273	0.150	928.652
	Harris, TX	5883.932	1.224	6993.098	6824.524	1.419	8209.270	**2209.289**	**0.465**	**2506.827**
	Los Angeless, CA	**5141.081**	**0.371**	**5985.620**	6376.008	0.460	7440.514	5501.848	0.397	6381.884
	Maricopa, AZ	2026.365	0.325	2112.334	963.663	0.155	969.336	**480.406**	**0.077**	**492.111**
21	Cook, IL	**465.474**	**0.080**	**519.420**	1659.534	0.282	1879.133	1219.233	0.207	1359.485
	Harris, TX	7611.514	1.553	8583.826	9913.906	2.017	11519.766	**1664.322**	**0.348**	**2097.628**
	Los Angeless, CA	**6242.229**	**0.447**	**6903.758**	7690.419	0.551	8509.198	6313.624	0.452	6943.113
	Maricopa, AZ	2549.187	0.402	2705.335	767.923	0.123	826.909	**396.265**	**0.063**	**425.505**
28	Cook, IL	**486.965**	**0.083**	**548.673**	2493.195	0.420	3000.083	1869.379	0.315	2260.015
	Harris, TX	8785.001	1.760	9652.091	13119.518	2.612	15167.205	**2479.698**	**0.499**	**3145.165**
	Los Angeless, CA	6715.207	0.478	7232.734	7974.478	0.568	8596.755	**6094.396**	**0.435**	**6615.172**
	Maricopa, AZ	3327.537	0.517	3705.984	670.792	0.107	754.504	**452.858**	**0.071**	**533.121**

Table 4.6: Out-of-sample errors of the model's predictions on the number of cumulative cases for the counties in the United States. The lowest errors for each evaluation metrics at each location are highlighted.

4.2.1 Reproduction number and fatality rate

Considering Figure 4.3, all three versions were able to learn the same trend for the effective reproduction number where it decreased sharply to below 1 for the first 15 time steps and then sharply increased for a short period of 10 time steps before gradually decreased until the end of the simulation. At the first few time steps, both the baseline model and the third version predicted a very low value for the effective reproduction number before rapidly increasing to a maximum value. The reason for this behavior was because in the dataset for Cook (Illinois) the number of new cases at the beginning of the chosen period was on the decrease for a few days before it increased again. Looking at Figure A.4, it can be seen that both the baseline model and the third version closely follow the declining number of new cases for the first few days, resulting in the low effective reproduction number in the beginning. On the other hand, the second version did not have a good fit for this period, hence the effective reproduction number given by the second version was different from the other two versions. Moreover, the max effective reproduction number predicted by the second version was higher at around 2.5 whereas the other versions had the max effective reproduction number at around 2. With regards to the fatality rate, both the baseline model and the third version learned a similar trend where the fatality rate started at around 0.045 and exponentially decreased as time went on. Because of the high effective reproduction number that was predicted at the beginning of the simulation, the fatality rate predicted by the second version was lower for the first 20 time steps compared to the baseline model and the third version. For the period after the first 20 time steps, the fatality rate predicted by the second version followed a decreasing trend similar to the fatality rate predicted by the other two models.

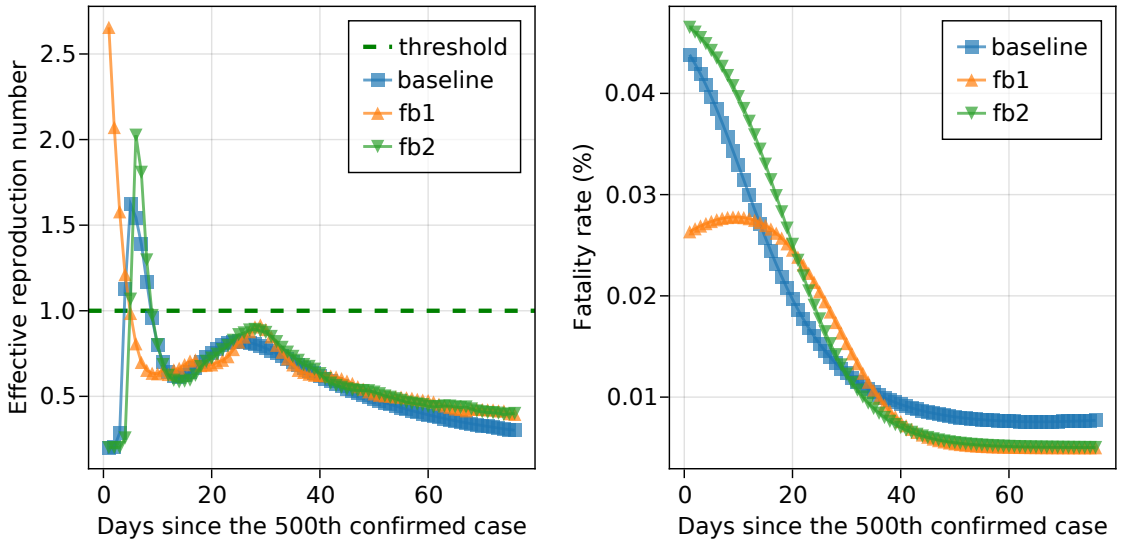

Figure 4.3: The effective reproduction number (left) and the fatality rate (right) for Cook (Illinois), learned by different versions of the model

For the county Harris (Texas), according to Figure 4.4, all three versions predicted a similar trend in both the effective reproduction number and the fatality rate. For the first few time steps, all three versions predicted a low effective reproduction number because of similar reasons to the case with the dataset for Cook (Illinois), as illustrated in Figure A.3. Both the baseline model and the third version predicted the maximum reproduction number of around 2.5 while the second version predicted a much higher value of 4.5. After reaching the maximum effective reproduction number, all versions predicted a dropped in value to

below 1 at around the 15th time step followed by a sharp increase to above 1 at the 25th time step before the effective reproduction number gradually decreased for the rest of the simulated period. This trend of the effective reproduction number was similarly observed when trained with data for Cook (Illinois). When predicting the fatality rate, all three versions showed similar results with the initial fatality rate at the 0th time step being the highest at around 0.045, and the fatality rate dropped exponentially after the initial time step.

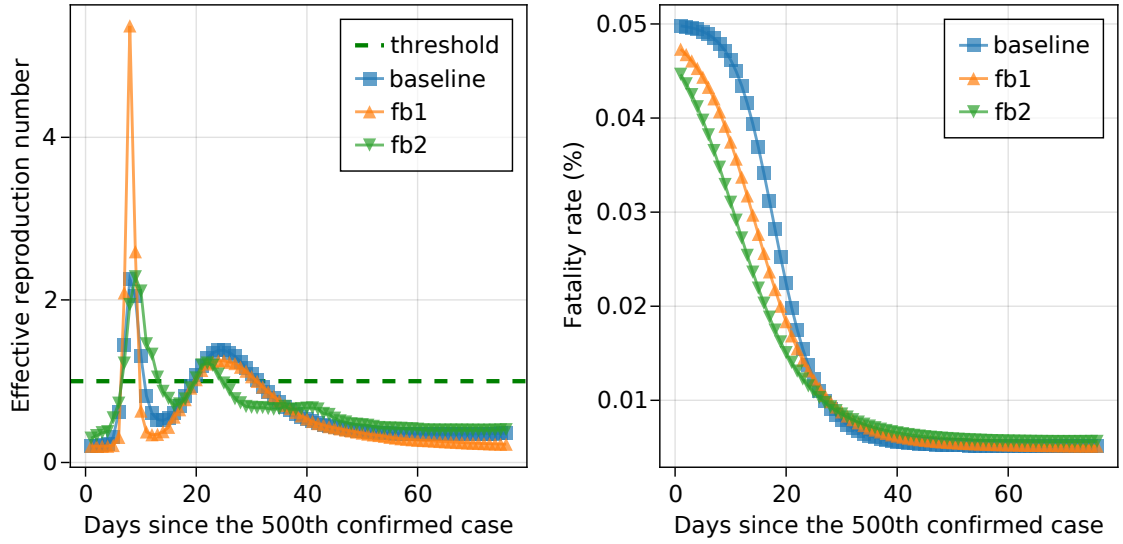

Figure 4.4: The effective reproduction number (left) and the fatality rate (right) for Harris (Texas) learned by different versions of the model

As illustrated in Figure 4.5, after being trained with data for Los Angeles (California), the baseline model predicted the max effective reproduction number to be around 1.5 whereas both the second version and the third version predicted a max value of around 1.25. Because of the initial decrease in the number of cases in the data for Los Angeles (California), the baseline model predicted a very low effective reproduction number for the first few time steps before predicting a sharp rise. This behavior was also observed with the second version, although the initial predicted values were not as low as in the case with the baseline model. However, in the third version, this low initial effective reproduction number was not predicted, and as a result, the third version's predictions for the number of new cases for Los Angeles (California) in the first few time steps did not decrease as sharply, as shown in Figure A.3. After the initial max predicted value, all three versions showed a similar trend for the effective reproduction number that gradually decreased until the end of the simulation and passed the 1 boundary at around the 20th time step. Due to the low initial predicted effective reproduction number, the max fatality rate of 0.04 predicted by the baseline model was much higher than the value predicted by the other two versions, which was around 0.025. Here, the predicted fatality rate in all three versions also followed an exponentially decreasing trend.

For Maricopa (Arizona), all the versions predicted similar trends and values for both the effective reproduction number and the fatality rate. The predicted effective production number was at its maximum at the 0th time step with a value of around 1.25 which gradually drop off as time went on. According to Figure 4.6, the predicted effective reproduction number of all three versions crossed below 1 at around the 10th time steps. For the fatality rate, the maximum value of 0.05 was predicted for the first few time steps before it exponentially decreased and stabilized until the end of the simulation.

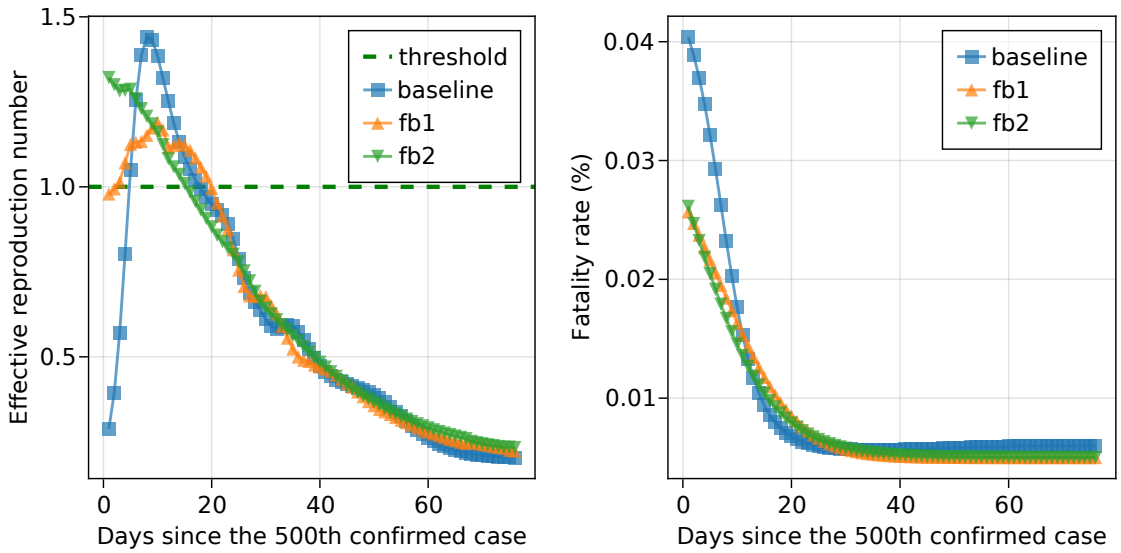

Figure 4.5: The effective reproduction number (left) and the fatality rate (right) for Los Angeles (California) learned by different versions of the model

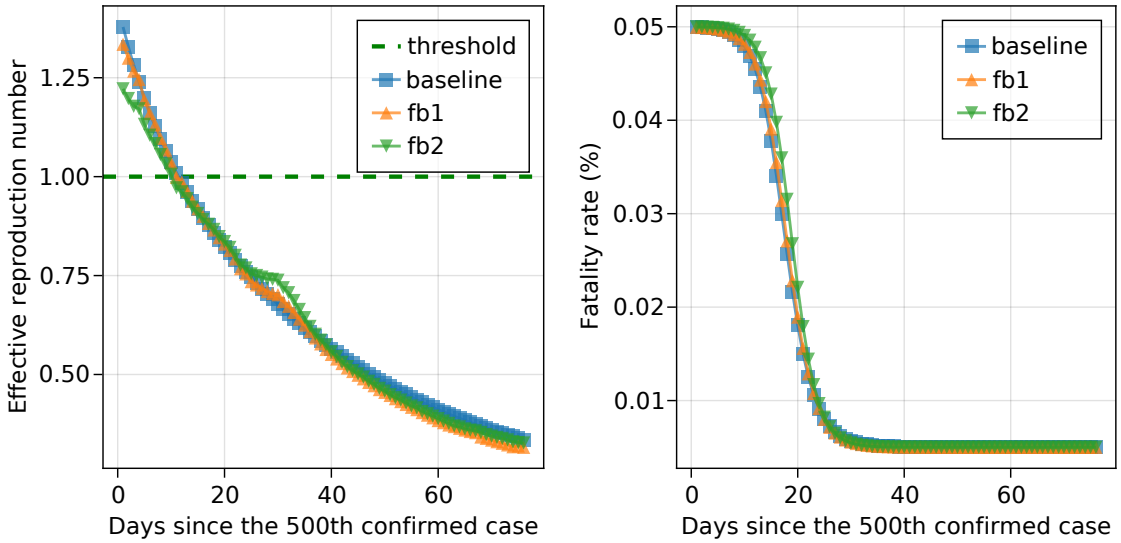

Figure 4.6: The effective reproduction number (left) and the fatality rate (right) for Maricopa (Arizona) learned by different versions of the model

4.3 Model predictions errors for provinces in Vietnam

Finally, we evaluated the performance of the different versions of the model when applied to data for different provinces in Vietnam. Comparable to the previous results obtained when experimenting with country-level data and US counties data, all versions were capable of predicting future cases for all compartments with relatively high accuracy at most locations except for Ho Chi Minh city. With Vietnam provinces data, it was found that all the versions performed best for up to 14 days before the accuracy started to fall off. Moreover, the results also suggested that the covariates that we had chosen did not have a positive effect on the long-term forecasting ability of the model.

Regarding predictions errors for the number of deaths in Table 4.7, across all metrics

for the 7-day forecast horizon, the baseline model achieved the lowest predictions errors for three provinces Binh Duong, Dong Nai, and Ho Chi Minh city while the second version performed best on data for Long An. On all the considered error metrics, for the 14-day, 21-day, and 28-day forecast horizons, the baseline model performed best on data for Dong Nai and Ho Chi Minh city whereas the second version made the best predictions for Binh Duong and Long An. When predicting the number of deaths for Ho Chi Minh city, all three versions did not make good predictions where the MAPE value was lowest at 35% for the 7-day forecast and highest at 46% for the 28-day forecast. Furthermore, the baseline model and the third version also made bad predictions for the number of deaths for Long An where the lowest MAPE was 21% for the 7-day forecast and the highest MAPE was 45% for the 28-day forecast.

Looking at Table 4.8, it can be seen that across all metrics for the 7-day forecast horizon, the baseline model had the lowest predictions errors for Ho Chi Minh city, the second version had the lowest predictions errors for Binh Duong, and the third version performed best for Dong Nai and Long An. Moving on to the 14-day forecast horizon, across all metrics, the baseline model now achieved the lowest predictions errors for Dong Nai and Ho Chi Minh city, the second version had the best performance for Binh Duong, and the third version had the best performance for Long An. Next with the 21-day forecast horizon, the baseline model continued to have the best performance across all metrics for Dong Nai and Ho Chi Minh city. While the second version had the lowest MAE for Binh Duong city, the predictions with the lowest MAPE and lowest RMSE were made by the third version. In addition, the third version also had the lowest predictions errors for Long An across all metrics. With the 28-day forecast horizon, the baseline model had the lowest predictions errors for Dong Nai across all metrics, however, for Ho Chi Minh city, the model only achieved the lowest MAE and MAPE while the second version had the lowest RMSE. Finally, the third version performed best for Binh Duong and Long An across all metrics. Except for the baseline model's predictions for Dong Nai, all other predictions for the number of new cases for provinces in Vietnam had relatively high errors, especially with larger forecast horizons. Here predictions' accuracy deteriorated when predicting with a horizon larger than 14 days where the MAPE value was higher than 20% across all locations.

On the predictions for the number of cumulative cases, Table 4.9 shows that for Binh Duong, the baseline model had the lowest predictions errors across all metrics for the 7-day horizon, the second version had the lowest predictions errors across all metrics for the 14-day and 21-day horizon, and the third version had the lowest predictions errors across all metrics for the 28-day horizon. For Dong Nai, the third version performed best across all metrics for the 7-day and 14-day horizon, however, with the 21-day horizon, the third version only had the lowest MAPE while the lowest MAE and lowest RMSE were held by the baseline model. In addition, the baseline model also had the best performance across all metrics for Dong Nai with the 28-day horizon. For Ho Chi Minh city, the third version made the best predictions across all metrics for the 7-day horizon, while the baseline model was best across all metrics for the 14-day, 21-day, and 28-day horizons. Lastly, for Long An, the third version had the best performance across all metrics and horizons.

Overall, all three versions had a good out-of-sample performance for the number of deaths and the number of cumulative when extrapolated from the training period. Nonetheless, the model predictions errors for the number of new cases were still high, and in the case of Ho Chi Minh city, as shown in Figure A.10, all versions completely failed to capture the trend of the new cases. Similar to the experiments with US counties, it was likely that the model's performance was dependent on the characteristics of the locations that were modeled, and the encoded covariates provided little to no help in improving the model out-of-sample performance.

Days	Loc.	Baseline			2nd. Version			3rd. Version		
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
7	Binh Duong	**95.426**	**5.812**	**98.695**	100.100	6.106	102.736	121.271	7.385	125.535
	Dong Nai	**70.096**	**16.939**	**70.288**	94.131	22.727	94.283	77.339	18.677	77.493
	Ho Chi Minh city	**969.400**	**35.439**	**991.778**	976.721	35.690	999.832	1015.894	37.060	1042.086
	Long An	94.571	33.788	95.339	**8.248**	**2.931**	**8.550**	61.189	21.732	63.484
14	Binh Duong	139.067	7.795	147.809	**138.019**	**7.764**	**144.724**	176.025	9.869	186.921
	Dong Nai	**59.087**	**13.976**	**60.270**	87.745	20.673	88.071	70.580	16.639	71.003
	Ho Chi Minh city	**1403.959**	**38.084**	**1489.430**	1423.473	38.555	1512.527	1515.313	40.813	1619.890
	Long An	115.136	38.395	117.520	**11.775**	**3.890**	**12.432**	92.512	30.445	99.238
21	Binh Duong	177.847	9.335	190.719	**168.792**	**8.910**	**177.966**	223.925	11.759	239.736
	Dong Nai	**55.633**	**12.716**	**56.685**	87.775	19.906	87.995	72.183	16.345	72.517
	Ho Chi Minh city	**1891.890**	**40.052**	**2062.596**	1926.393	40.695	2104.055	2095.582	43.845	2309.499
	Long An	134.235	42.112	138.380	**14.507**	**4.507**	**15.372**	124.800	38.414	136.510
28	Binh Duong	203.904	10.232	217.207	**185.288**	**9.373**	**193.737**	257.546	12.925	274.293
	Dong Nai	**52.644**	**11.686**	**53.744**	87.106	19.115	87.288	74.530	16.270	74.884
	Ho Chi Minh city	**2387.155**	**41.454**	**2637.652**	2434.133	42.186	2693.007	2701.371	46.196	3022.084
	Long An	151.608	45.047	157.341	**16.404**	**4.841**	**17.303**	157.136	45.593	173.810

Table 4.7: Out-of-sample errors of the model's predictions on the number of deaths for the provinces in Vietnam. The lowest errors for each evaluation metrics at each location are highlighted.

Days	Loc.	Baseline			2nd. Version			3rd. Version		
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
7	Binh Duong	258.212	8.744	299.194	**116.056**	**4.086**	**164.898**	296.040	9.960	330.132
	Dong Nai	130.533	17.341	135.187	209.730	27.792	214.491	**39.442**	**5.266**	**49.236**
	Ho Chi Minh city	**139.151**	**3.560**	**163.295**	376.279	9.647	394.165	705.167	18.114	723.858
	Long An	189.861	33.532	200.279	188.479	33.290	198.773	**157.530**	**27.707**	**168.869**
14	Binh Duong	508.485	16.627	597.647	**205.217**	**6.720**	**267.533**	545.052	17.835	626.841
	Dong Nai	**84.028**	**11.122**	**103.395**	179.143	23.323	186.383	133.415	16.778	169.034
	Ho Chi Minh city	**379.416**	**9.804**	**466.377**	600.225	15.498	651.818	989.250	25.554	1040.354
	Long An	199.785	36.232	211.350	197.566	35.802	209.171	**163.229**	**29.231**	**176.954**
21	Binh Duong	514.164	25.439	596.107	**500.765**	32.829	695.279	518.767	**24.589**	**594.628**
	Dong Nai	**63.199**	**8.373**	**85.472**	173.204	22.846	178.936	174.459	22.671	202.875
	Ho Chi Minh city	**742.775**	**17.905**	**960.595**	902.403	22.027	1040.717	1338.583	32.902	1468.122
	Long An	153.175	30.224	176.074	150.502	29.573	173.932	**112.808**	**20.656**	**144.752**
28	Binh Duong	650.869	59.526	741.151	787.829	87.134	1021.467	**623.044**	**54.074**	**696.459**
	Dong Nai	**51.988**	**7.033**	**74.758**	177.252	24.773	181.688	179.379	24.798	200.903
	Ho Chi Minh city	**1297.598**	**27.166**	1713.583	1388.426	29.725	**1697.844**	1863.602	40.654	2149.100
	Long An	118.454	24.197	152.753	116.804	23.893	150.918	**98.185**	**21.637**	**128.612**

Table 4.8: Out-of-sample errors of the model's predictions on the number of new cases for the provinces in Vietnam. The lowest errors for each evaluation metrics at each location are highlighted.

Days	Loc.	Baseline			2nd. Version			3rd. Version		
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
7	Binh Duong	**543.123**	**0.300**	**669.448**	1080.846	0.605	1092.888	2716.160	1.539	2807.759
	Dong Nai	925.781	2.478	968.738	1151.564	3.071	1239.187	**171.280**	**0.473**	**191.766**
	Ho Chi Minh city	2880.532	2.432	2897.982	**1146.135**	**0.998**	**1337.867**	1530.084	1.231	1929.637
	Long An	1838.836	8.607	1885.619	558.175	2.541	694.931	**358.230**	**1.733**	**408.240**
14	Binh Duong	2672.053	1.341	3614.244	**834.983**	**0.450**	**894.195**	2511.522	1.337	2752.084
	Dong Nai	1216.773	3.022	1267.355	1809.578	4.441	1961.363	**538.527**	**1.284**	**749.337**
	Ho Chi Minh city	**2038.919**	**1.623**	**2269.591**	2365.506	1.693	2975.292	5096.246	3.529	6552.386
	Long An	2689.645	11.316	2851.355	1400.049	5.699	1684.345	**791.856**	**3.278**	**942.779**
21	Binh Duong	3647.509	1.764	4430.807	**1576.284**	**0.770**	**2335.074**	2776.520	1.406	3008.196
	Dong Nai	**1278.627**	2.994	**1314.002**	2364.315	5.378	2573.613	1282.437	**2.779**	1737.311
	Ho Chi Minh city	**3863.159**	**2.489**	**5067.508**	5495.625	3.348	7444.975	10170.384	6.199	13131.423
	Long An	3228.181	12.685	3406.408	1927.496	7.360	2204.703	**1084.391**	**4.180**	**1233.620**
28	Binh Duong	3212.031	1.538	3992.435	4506.591	2.075	7143.902	**2646.462**	**1.308**	**2953.921**
	Dong Nai	**1307.494**	**2.909**	**1334.511**	2948.022	6.268	3243.385	2048.294	4.150	2650.432
	Ho Chi Minh city	**8875.541**	**4.700**	**13108.589**	11039.327	5.797	15550.166	17695.850	9.442	23398.714
	Long An	3549.776	13.298	3714.577	2237.386	8.175	2480.566	**1193.084**	**4.410**	**1312.184**

Table 4.9: Out-of-sample errors of the model's predictions on the number of cumulative cases for the provinces in Vietnam. The lowest errors for each evaluation metrics at each location are highlighted.

4.3.1 Reproduction number and fatality rate

As can be seen in Figure 4.7, for Binh Duong, all three models predicted a similar trend for the fatality rate that started at a maximum value of around 0.035 at the 0th time step and decreased gradually as time went one. With the effective reproduction number, at the 0th time step, the baseline model predicted a maximum value of 2.5, the second version predicted a maximum value of 2.0, and the third version predicted a maximum value of 4.0. After the initial time steps, all versions predicted a sharp decline in the effective reproduction number where the value dropped below 1 at around the 10th time step. From the baseline model, we can see that after the effective reproduction number dropped below 1, it plateaued for 10 time steps before declining once again and stabilizing. In the case of the second version and the third version, we see a slight increase in the effective reproduction number after it dropped below 1. This increase in the effective reproduction number lasted for around 10 time steps before it started to decrease and stabilize.

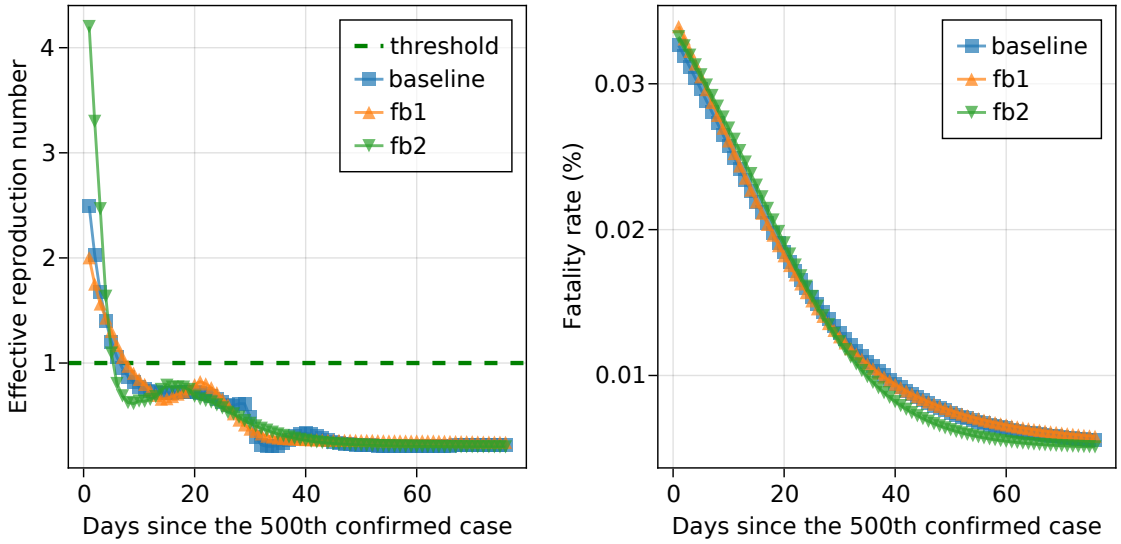

Figure 4.7: The effective reproduction number (left) and the fatality rate (right) for Binh Duong learned by different versions of the model

Looking at Figure 4.8, it can be seen that all three versions predicted a similar trend for the fatality rate for Dong Nai. In all versions, the fatality rate started at its maximum at the 0th time step and gradually decreased until the end of the simulation. With the baseline model and the third version, the maximum fatality rate was predicted to be around 3, while the second version predicted a higher fatality rate of 3.5. For the effective reproduction number, all three models also showed similar trends with slight differences in the first 20 time steps. With the baseline model and the second version, the effective reproduction number started at its maximum value of 4.5 and dropped drastically to below 1 at around the 5th time step. After that, the effective reproduction number increased for about 5 time steps to a value slightly above one before starting to decrease and stabilize. With the third version, the effective reproduction number at time step 0 was the same as in the case with the other two versions, however, the value sharply dropped to below 1 at around the 10th time step and started to stabilize without experiencing an increase.

For Ho Chi Minh city, Figure 4.9 showed that all versions predicted the same trend for both the fatality rate and the effective reproduction number. The predicted effective reproduction number from all three versions fluctuated between 1.25 and 1.5 in the first 25 time steps before decreasing exponentially and stabilizing at around 0.25 until the end of

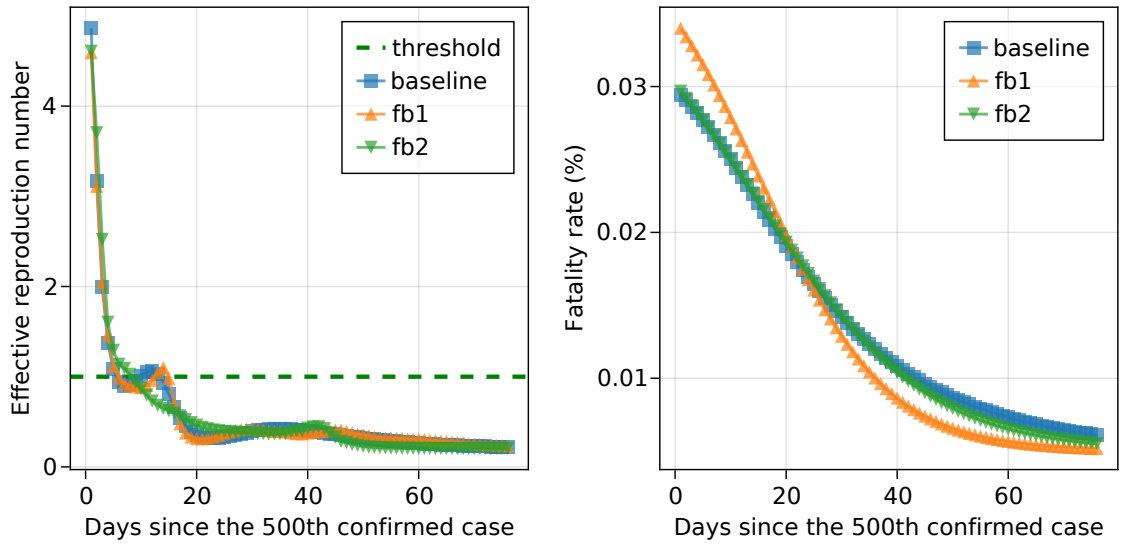

Figure 4.8: The effective reproduction number (left) and the fatality rate (right) for Dong Nai learned by different versions of the model

the simulation. When predicting the fatality rate, all three versions showed an increasing trend that started from around 0.005 at the 0th time step and increased to around 0.04. With the third version, the maximum predicted fatality rate was just below 0.04 while the other two versions predicted a higher value of around 4.5.

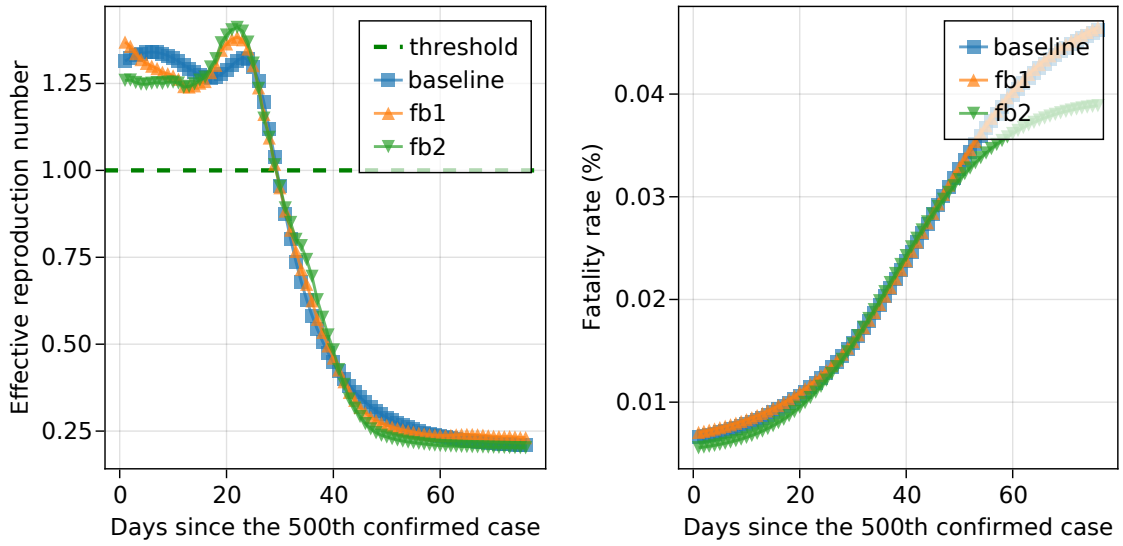

Figure 4.9: The effective reproduction number (left) and the fatality rate (right) for Ho Chi Minh city learned by different versions of the model

Finally, for Long An, Figure 4.10 shows that each version predicted a different value and trend for the fatality rate. The baseline model predicted a stagnated fatality rate that was around 0.03 and did not change much over time. With the second version, the fatality rate started at a higher maximum of 0.04 at the 0th time step and gradually decreased to a lower minimum that was slightly above 0.01. In contrast, the third version predicted a minimum fatality rate of around 0.01 at the 0th time step that gradually increased to a maximum of 0.04. Because of these differences, the resulting dynamics for the deaths compartment

from these versions were also highly different from each other. Looking at Figure A.7 and Figure A.9, we can see that the second version had a much better fit to the training data and can extrapolate with high accuracy, while both the baseline model and the third version fitted poorly to training data and overestimated the number of deaths when extrapolated into the future. For the effective reproduction number, all versions predicted a low initial value that surged to a maximum after 5 time steps. With the baseline model, a maximum effective reproduction number of around 2.5 was predicted, and once the max value was reached, the effective reproduction decreased exponentially and crossed the 1 threshold at around the 20th time steps. On the other hand, both the second version and the third version predicted a maximum effective reproduction number of around 4 where the value predicted by the second version was slightly lower than 4 and the value predicted by the third version was slightly higher than 4. Once the maximum was reached, the predicted effective reproduction number in both versions dropped significantly and crossed the 1 threshold at around the 15th time step. Finally, the predicted effective reproduction number raised slightly and reached the 1 threshold at around the 20th time step before finally decreasing for the rest of the simulated period.

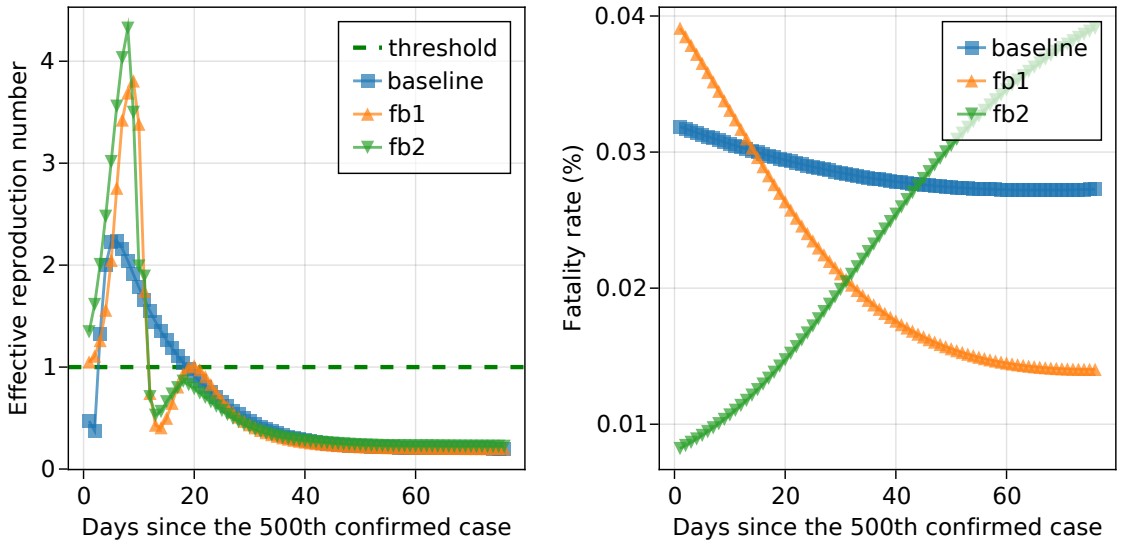

Figure 4.10: The effective reproduction number (left) and the fatality rate (right) for Long An learned by different versions of the model

5. Discussion

Regarding Figure A.1, Figure A.3, Figure A.4, Figure A.7, Figure A.8, and the data presented in Chapter 4, it is observed that the model could fit closely to data for the training period and provided useful insight into the evolution of the disease, which could not be done by a basic SEIR model [40]. Moreover, while we did not study the effects that the covariates had on the model's parameters in the second and third versions, an in-depth analysis could be done to gain additional insights into which factors played an important role in slowing down to spread of the virus. However, the model could not fit well to data where high fluctuations were presented, which can be seen in the model's fit for Harris (Texas), Dong Nai, Long An, and Binh Duong. Even though we had tried to eliminate high-frequency changes in the data by using a 7-day moving average, high variations in the number of new cases from day to day still existed due to various reasons such as how the data was collected, the presence of asymptomatic cases, underreporting, the number of tests that were performed, etc.

When extrapolated beyond the training period, the model had low errors on the out-of-sample data for most of the considered locations and showed that it can capture the evolution of the disease. In instances where the model could forecast the trend of the disease, it was observed that the model had the highest accuracy when making short-term forecasts, and the accuracy deteriorated as the forecast horizon expanded. This was expected as many real-world phenomenons could invalidate the assumptions made by the model during training, and the further we extrapolated from the training period, the more the errors resulted from these assumptions accumulated. Two cases where the model completely failed to forecast the trend of the disease were with data for Ho Chi Minh city and Vietnam. In these two instances, we can see that the data used for training the model, illustrated by Figure A.1 and Figure A.8, indicated a monotonically decreasing trend in the spread of the disease before the sharp increase was presented in the testing data. Because this sudden change in the disease dynamics was not exhibited in the training data for both cases, the model failed to make an accurate forecast of when such changes would happen. This result was aligned with existing studies [38]. Furthermore, it can be seen that the predictions made for locations where high fluctuations were presented were not as good as the predictions made for locations where the numbers followed smoother curves. This result indicated the model's shortcomings in learning and generalizing from noisy data.

Considering the benefits of encoding mobility data and social network data into the model, our results demonstrated that there was no improvement in the model accuracy for both short-term and long-term forecasts when these covariates were encoded. This result contrasted existing findings on mobility data and SPC index being important predictors for the future number of cases [34], [35], [38], [55]. We hypothesized several reasons to explain this discrepancy. Firstly, the mobility data that we used was not as granular and complex as those used in existing researches because we only considered datasets that were available for Vietnam. Secondly, even though the SPC had been shown to have a high correlation with the early number of cases, it became a less important predictor when the disease was prolonged and when heavy government restrictions on mobility were in place. Thirdly, the underlying SEIR model could be too simple and introduce strong model bias that hindered

the ability to learn the true interactions between the covariates and the spread of the disease.

5.1 Model's convergence and generalizability

Although the model could converge on a good minimum, this property was not guaranteed, and the model would occasionally get stuck in bad local minima. This was a known problem with fitting UDE [9], and fitting to time series data in general. To mitigate the issue, we implemented various techniques that helped to reduce the chance of encountering bad local minima including iteratively growing the fit and placing more weight on errors for earlier data points, as suggested by the library authors. However, these techniques did not guarantee to completely eliminate the existing problems.

Another property that could prevent the model from converging was with the *stiffness* of the ODE [56]. Stiff ODEs are ODEs that can not be solved by explicit methods. This property prevented us from calculating the gradients by solving an augmented ODE backward in time as proposed in the NeuralODE paper [8]. Additionally, even when the gradients could be calculated, various problems with the model's stability and the miscalculation of the gradients might still exist, which additionally needed to be addressed. Because there was not a clear definition for stiff ODEs and there was no method for determining whether a system was stiff, we circumvented this by relying on multiple suggestions from the existing research on this issue when implementing our model [56]. Firstly, the *mish* activation function [54] was used to avoid known problems with saturating activations, e.g, *tanh* and *sigmoid*. Secondly, we used the *InterpolatingAdjoint* method implemented in the *DiffEqFlux.jl* package for calculating the gradients instead of relying the the *BacksolveAdjoint* method which computed the gradients by solving an augmented ODE backward in time. Thirdly, we scaled the model's output with min-max scaling when calculating the loss value, which was shown to help to reduce instability when calculating the gradients.

Because we trained the model separately on data for each of the considered locations, each location would have a separated set of ANN weights and system parameters. This meant that the model could only forecast cases for the location that it had been trained on. We did not perform extensive testing on the model's performance on data outside of the location that it had been trained on, however, we believed that with the current training method, the model was not capable of predicting for different locations once it had been trained with only one location. Furthermore, because we were modeling real-world scenarios that were subjected to changes, the model's performance might change dramatically as the forecasting time span moves further away from the training time span.

5.2 Limitations

Issues with ground truth data Like all data-driven methods, the quality of our model depended on the quality of the collected data. It had been noted that the method for collecting Covid-19 data varied across different locations, and the collected data might suffer detection and measurement biases. Thus the performance of the model could change drastically based on the location.

Inability to capture rapid trend changes As shown in the result, our model failed to capture the disease dynamics when the number of cases became very flat or very sharp. In some cases, such trends occurred due to modifications to the number of cases in the report, in other cases, they might be caused by other covariates that we did not consider.

Bad local minima While we implemented different techniques for lowering the chance of getting into bad local minima, the problem still existed. One method that we tried that

was not mentioned in previous sections was to use a different loss function [64], but the obtained results were unsatisfactory.

Compartmental model bias Like all compartmental models, our model made many assumptions such that real-world scenarios can be approximated with feasible computational power. Hence, many aspects of the disease were simplified, one of which was the disease transmission process. Moreover, because we wanted to limit the complexity in the model training process, we did not consider additional compartments that better reflected our current knowledge about Covid-19 such as asymptomatic infections, reinfections, vaccinations, and quarantine control measures.

Covariates bias Our two versions of the model were informed using the datasets from Facebook which may introduce biases. We chose these datasets because of the large number of Facebook users in Vietnam, however, these datasets might not be representative of the whole population and introduced certain biases into the model predictions.

The changing dynamics of Covid-19 The dynamics of the disease constantly changes due to a multitude of factors, and our knowledge of these dynamics will improve over time. Thus the results produced by our model are not final, and changes to the model that incorporate new findings are needed to improve its robustness.

Neural network interpretability Even though the model produced domain-specific encoding that can be understood by experts, the ANN was still a black-box algorithm that was not interpretable. This property may be undesirable if we want to apply the model in situations where full transparency is required.

6. Conclusion

In this thesis, we implemented an infectious disease model that was enhanced by incorporating an ANN into the SEIR model to eliminate the compartmental model basic assumptions about the contact rate and the fatality rate being static. We further tried to improve the forecast performance of the model by using mobility data and social network data as input features to the embed ANN. The model utilized the UDE approach [9] to learn from real-world data and make predictions backed by domain-specific knowledge. We showed that the model could capture the trend for the evolution of the disease in many of the tested locations and provide epidemiological significant insights about the disease. Our experiments with encoding mobility data and social network data as covariates into the model revealed that these data points could not improve the forecast performance of the model. However, these results were not conclusive since we had a small sample size and the existing issues with training the model on real-world data might restrict our ability to observe the improvements brought by these covariates.

Even though the UDE method had shown promising results on simulated data, our application of the approach on real-world Covid-19 data revealed many difficulties. These issues were circumvented with different techniques suggested by existing researches [9], [56]. Having said that, not all issues were alleviated and many additional enhancements can be made to improve the model performance. Firstly, additional compartments can be used to represents the characteristics of Covid-19 such as asymptomatic infection, vaccinations, etc. Secondly, additional time-varying covariates can be considered to better inform the model, especially in cases when the disease dynamics change rapidly. Thirdly, new improvements to training UDEs should be considered to make a better guarantee of model convergence. Finally, using the model in combination with recent advancements in methods for identifying unknown dynamical systems, such as Sparse Identification of Nonlinear Dynamical System (SINDy) [65], can help to extract the underlying governing equation from the embed ANN and provide a fully interpretable model.

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. Jastrzbski, 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] 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.

[29] 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.

[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] 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.

[32] 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.

[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. Ark, 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] S. Kim, W. Ji, S. Deng, Y. Ma, and C. Rackauckas, "Stiff neural ordinary differential equations," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 9, p. 093 122, Sep. 20, 2021.

[57] C. Tsitouras, "RungeKutta 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.

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

[59] 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).

[60] 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.

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

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

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

[64] R. Vortmeyer-Kley, P. Nieters, and G. Pipa, "A trajectory-based loss function to learn missing terms in bifurcating dynamical systems," *Scientific Reports*, vol. 11, no. 1, p. 20 394, 1 Oct. 14, 2021.

[65] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, Apr. 12, 2016.

A. Model predictions

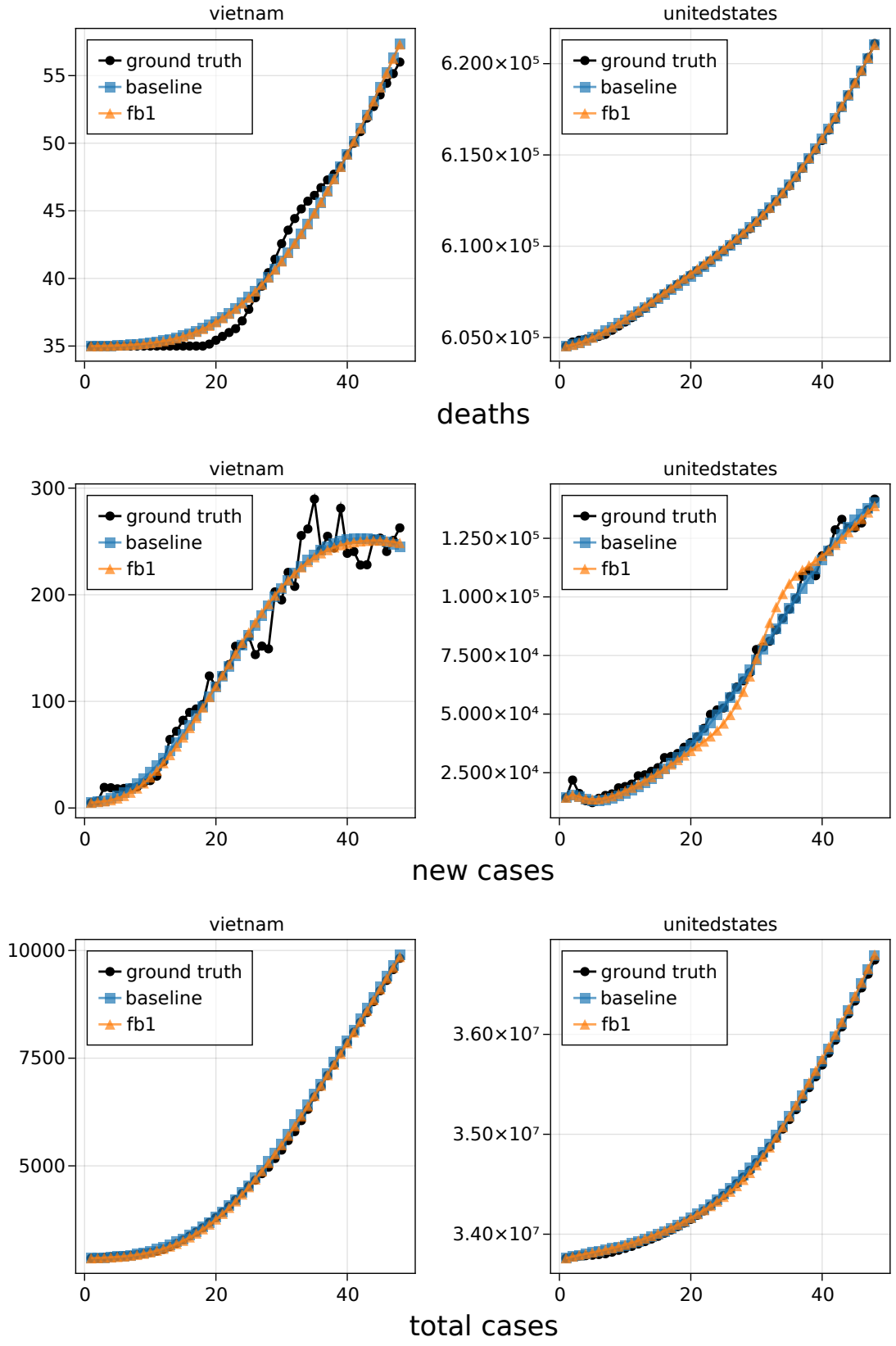

Figure A.1: Predictions made by all versions of the model for the training period after having trained with data for Vietnam (left) and the US (right). Each row contains the predictions for a compartment for each of the considered countries. Here the second version is denoted as *fb1*

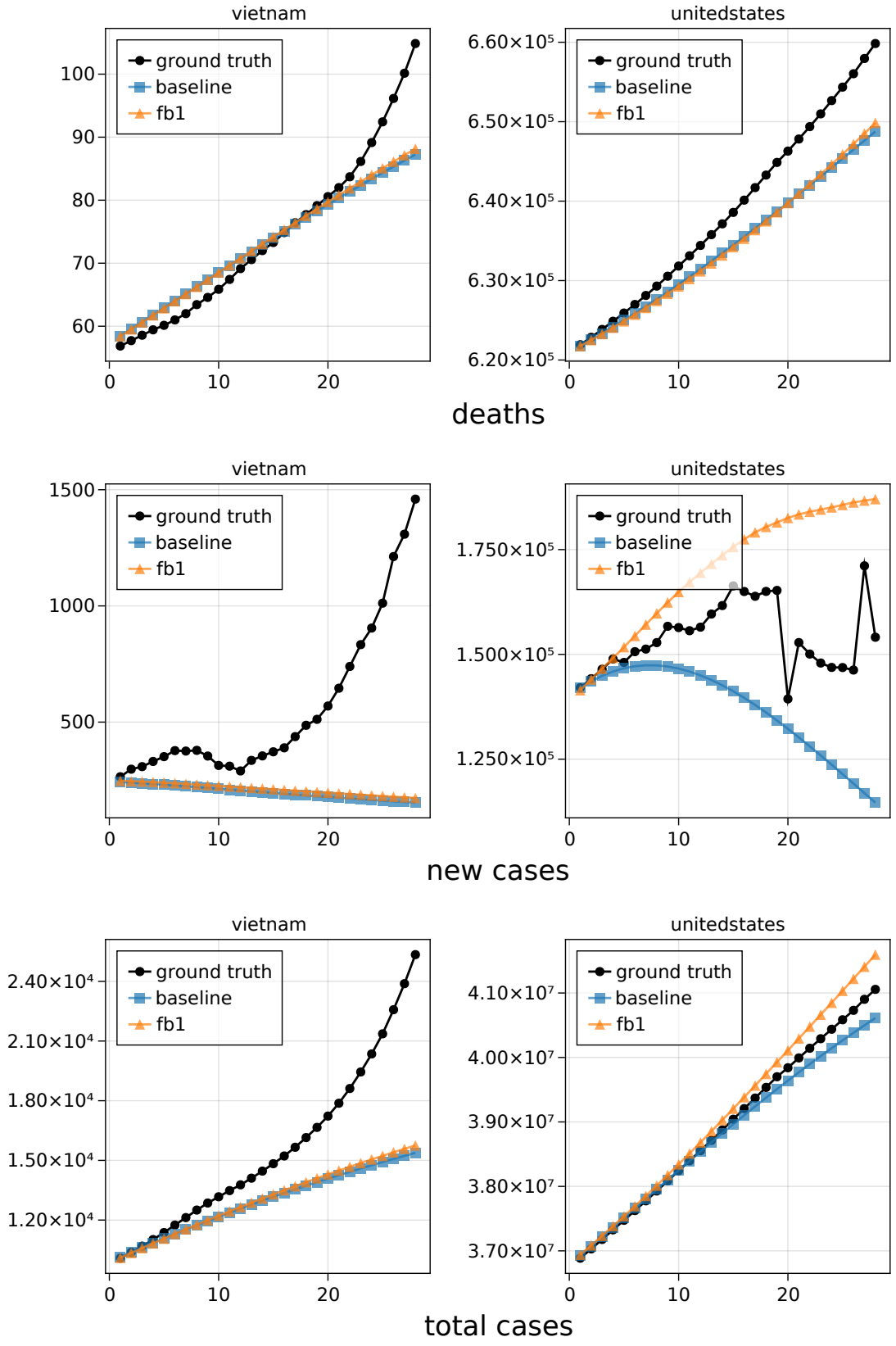

Figure A.2: Predictions made by all versions of the model for the testing period after having trained with data for Vietnam (left) and the US (right). Each row contains the predictions for a compartment for each of the considered countries. Here the second version is denoted as *fb1*

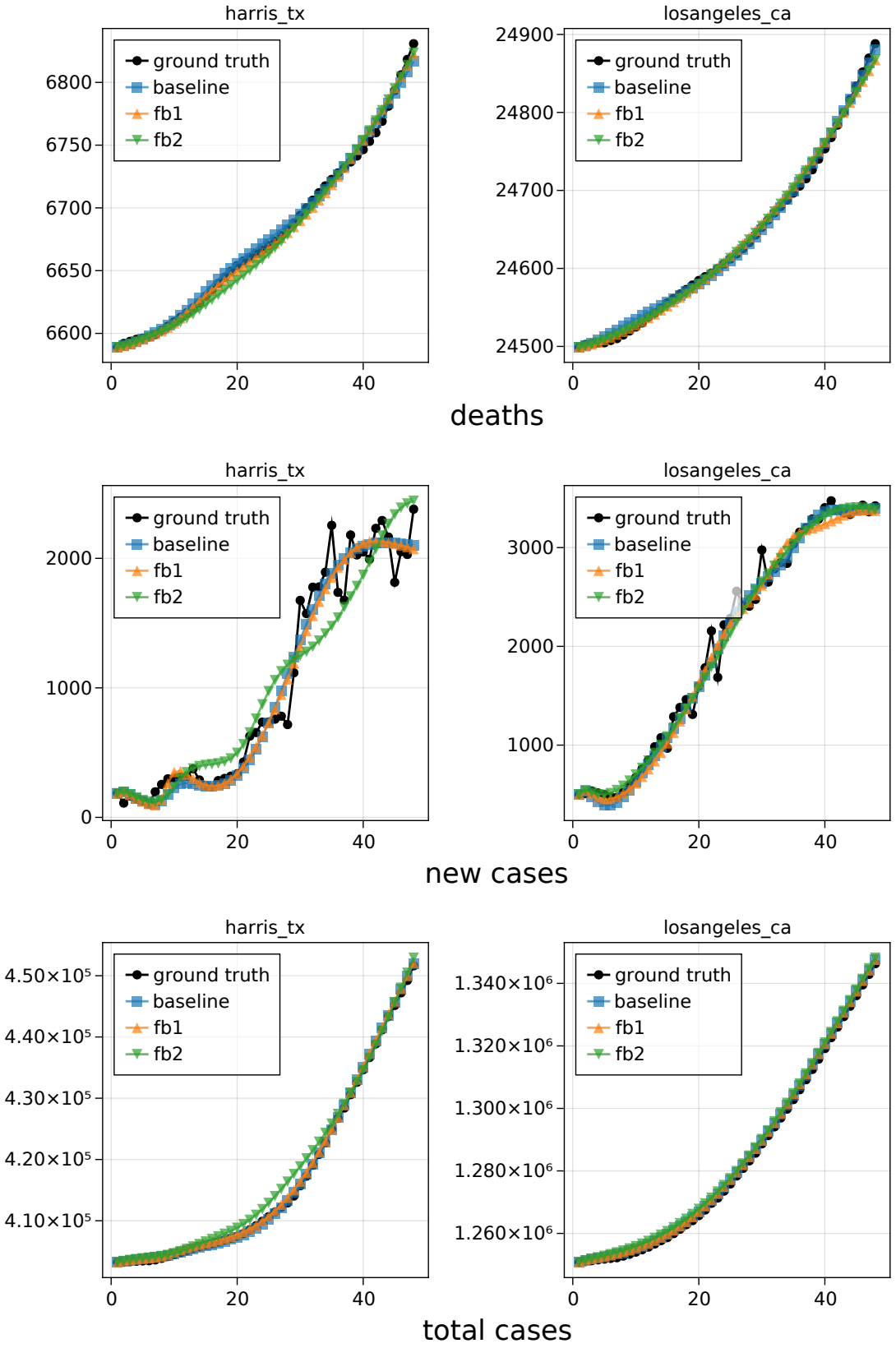

Figure A.3: Predictions made by all versions of the model for the training period after having trained with data for Harris - Texas (left) and Los Angeles - California (right). Each row contains the predictions for a compartment for each of the considered counties. Here the second version is denoted as *fb1* and the third version is denoted as *fb2*

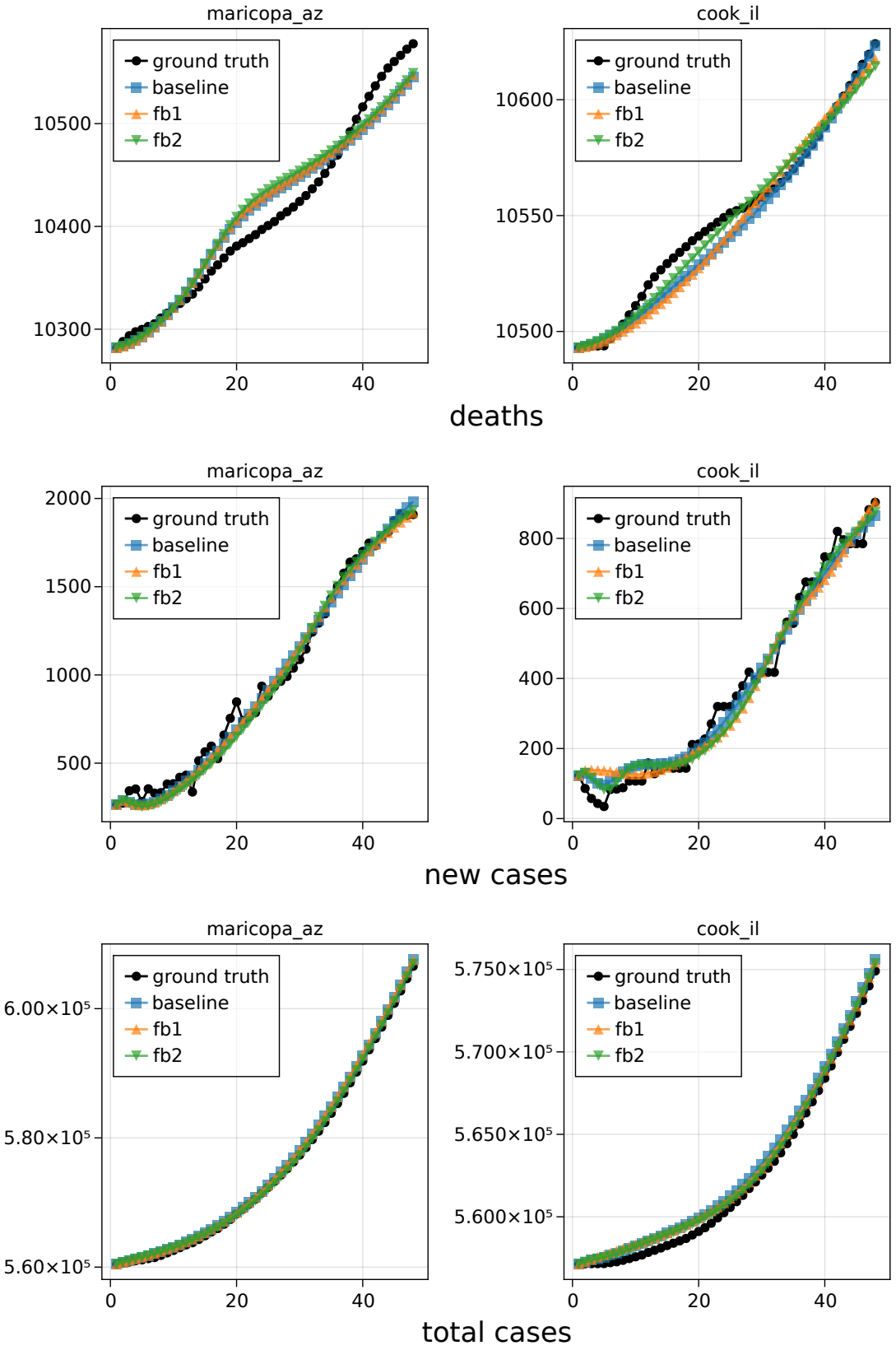

Figure A.4: Predictions made by all versions of the model for the training period after having trained with data for Maricopa - Arizona (left) and Cook - Illinois (right). Each row contains the predictions for a compartment for each of the considered counties. Here the second version is denoted as *fb1* and the third version is denoted as *fb2*

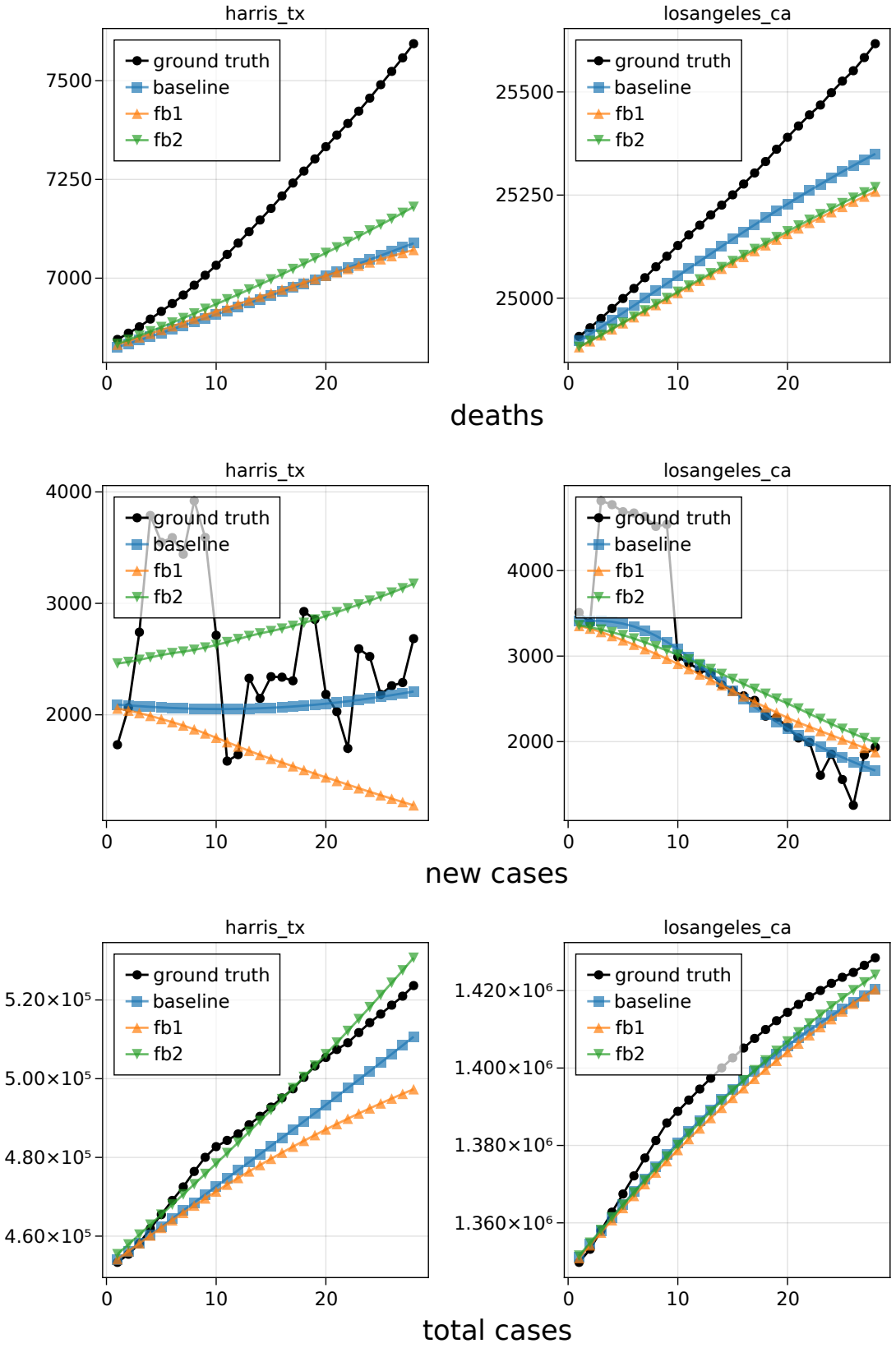

Figure A.5: Predictions made by all versions of the model for the testing period after having trained with data for Harris - Texas (left) and Los Angeles - California (right). Each row contains the predictions for a compartment for each of the considered counties. Here the second version is denoted as *fb1* and the third version is denoted as *fb2*

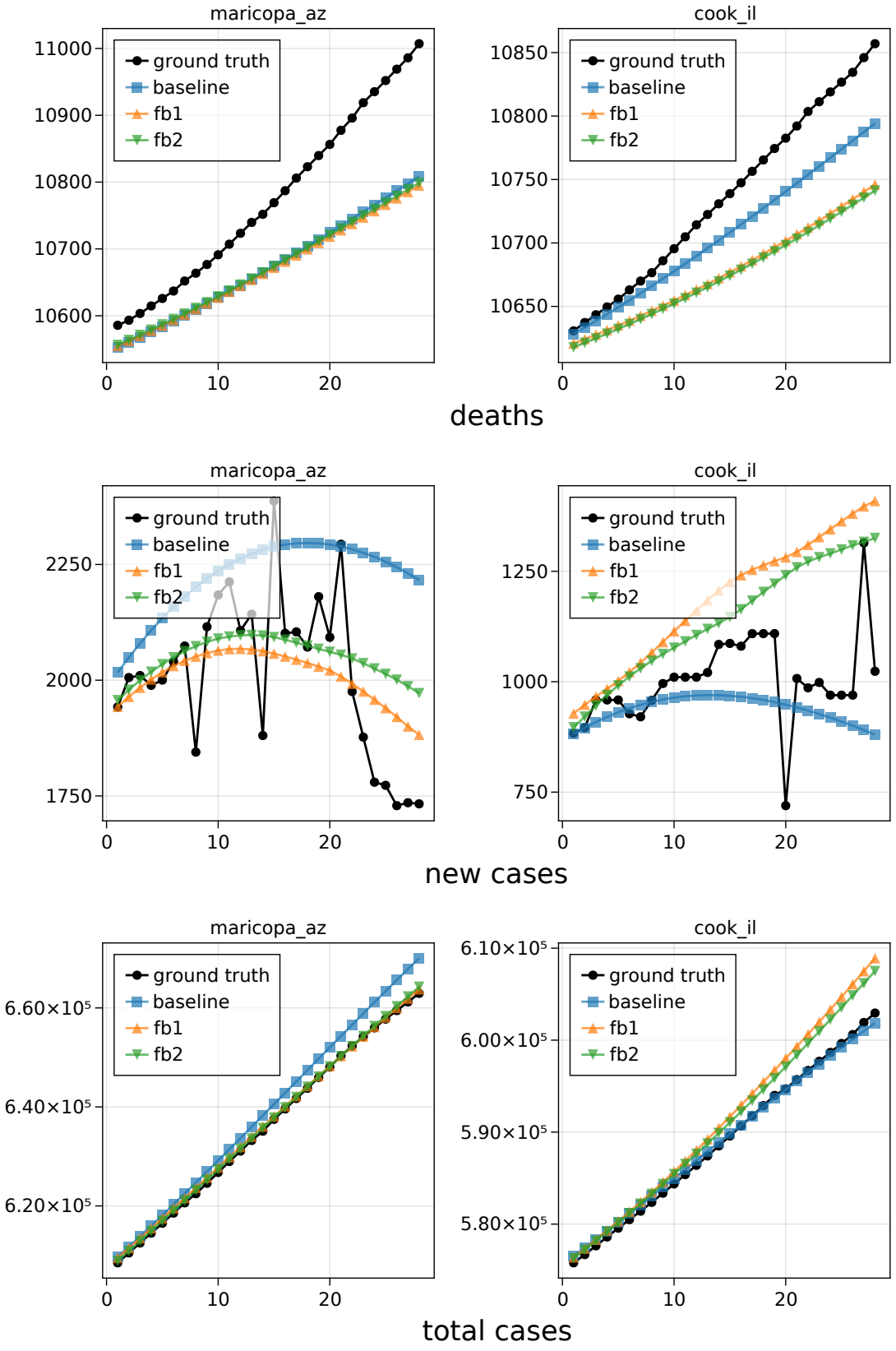

Figure A.6: Predictions made by all versions of the model for the testing period after having trained with data for Maricopa - Arizona (left) and Cook - Illinois (right). Each row contains the predictions for a compartment for each of the considered counties. Here the second version is denoted as *fb1* and the third version is denoted as *fb2*

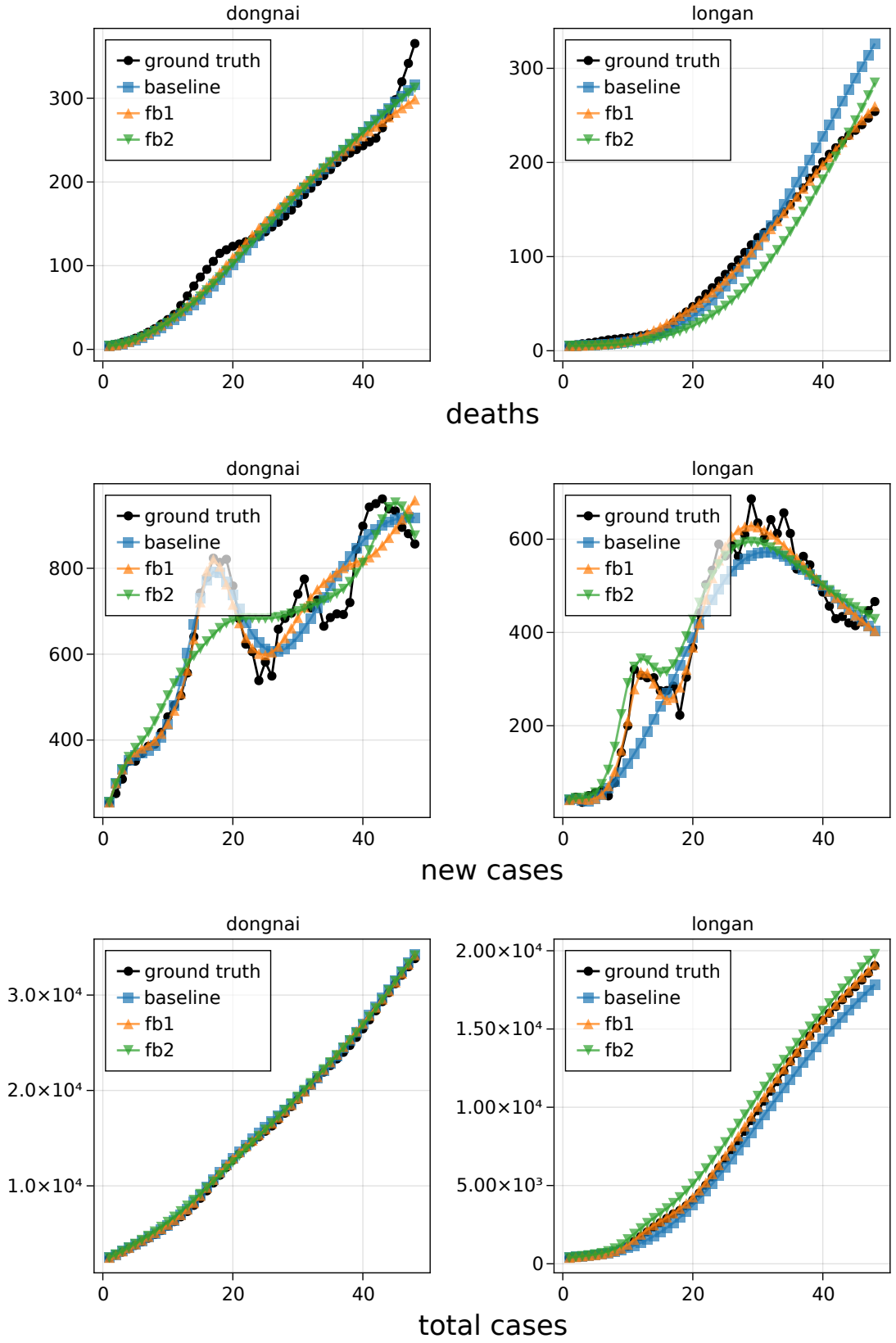

Figure A.7: Predictions made by all versions of the model for the training period after having trained with data for Dong Nai (left) and Long An (right). Each row contains the predictions for a compartment for each of the considered provinces. Here the second version is denoted as *fb1* and the third version is denoted as *fb2*

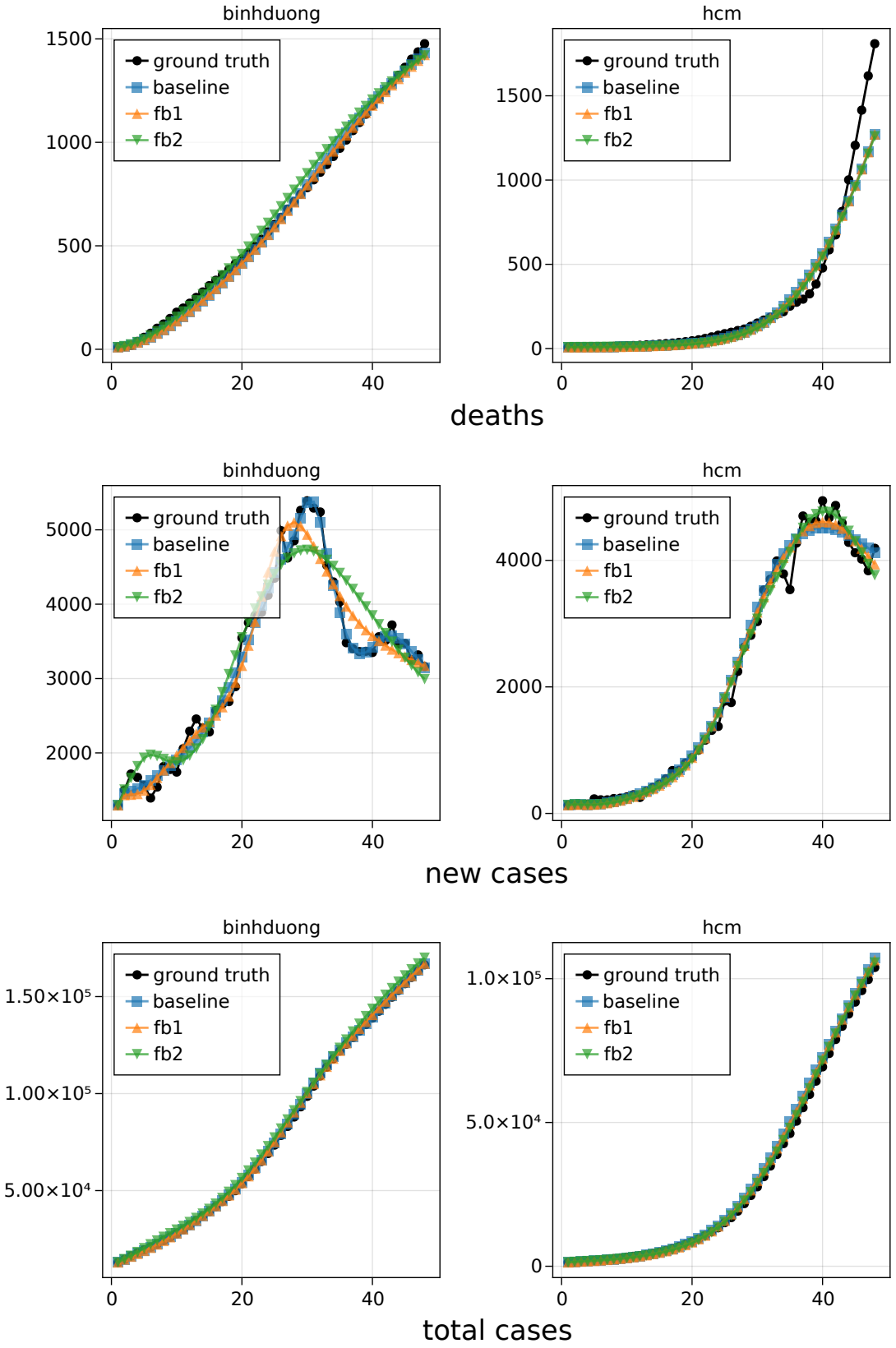

Figure A.8: Predictions made by all versions of the model for the training period after having trained with data for Binh Duong (left) and Ho Chi Minh city (right). Each row contains the predictions for a compartment for each of the considered provinces. Here the second version is denoted as *fb1* and the third version is denoted as *fb2*

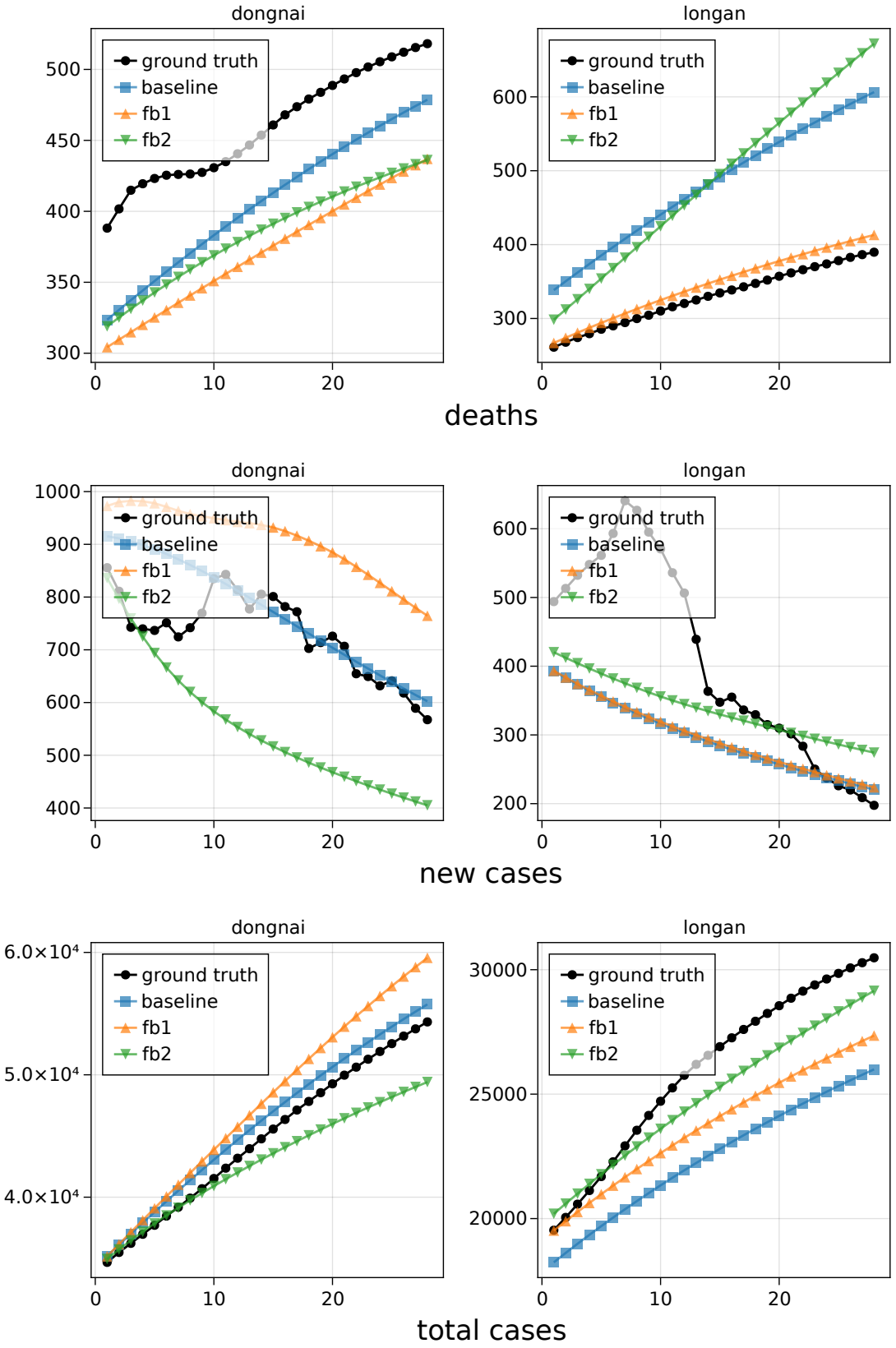

Figure A.9: Predictions made by all versions of the model for the testing period after having trained with data for Dong Nai (left) and Long An (right). Each row contains the predictions for a compartment for each of the considered provinces. Here the second version is denoted as *fb1* and the third version is denoted as *fb2*

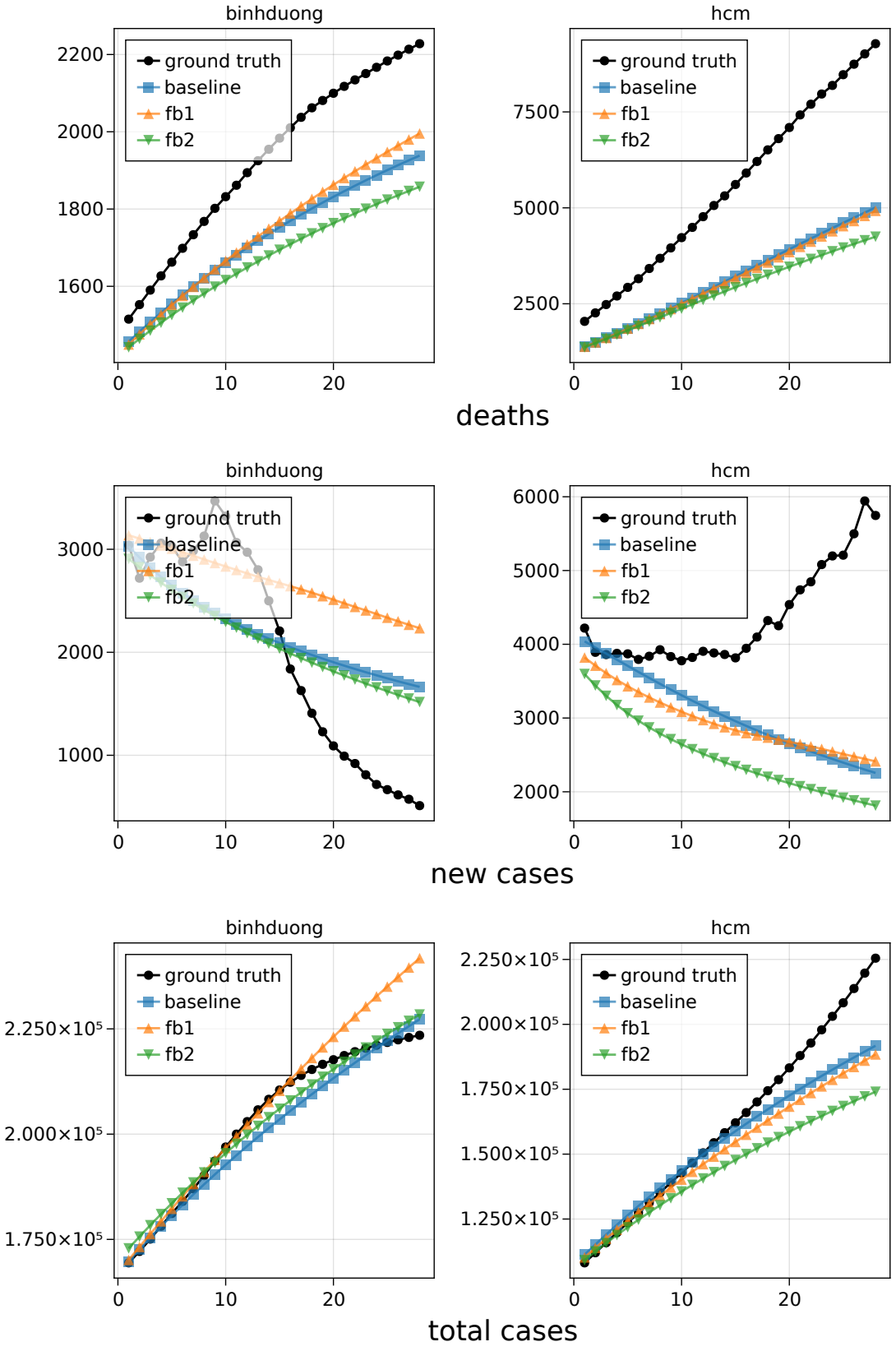

Figure A.10: Predictions made by all versions of the model for the testing period after having trained with data for Binh Duong (left) and Ho Chi Minh city (right). Each row contains the predictions for a compartment for each of the considered provinces. Here the second version is denoted as *fb1* and the third version is denoted as *fb2*

B. Package core implementations

```julia
function SEIRD!(du, u, p, t)
    @inbounds begin
        S, E, I, _, _, N, C, _ = u
        β, γ, λ, α = p
        du[1] = -β * S * I / N
        du[2] = β * S * I / N - γ * E
        du[3] = γ * E - λ * I
        du[4] = (1 - α) * λ * I
        du[5] = α * λ * I
        du[6] = -α * λ * I
        du[7] = -C + γ * E
        du[8] = γ * E
    end
    return nothing
end
```

Figure B.1: The function that defines the underlying SEIR model, where du is a vector that will be modified in-place to contain the system derivatives at each time step, u is a vector contains the current state of the system, p is a vector contains all the system parameters, and t is the value of the current time step.

```
function experiment_loss_sse(min::AbstractVector{R},
                             max::AbstractVector{R},
                             ζ::R) where {R<:Real}
    scale = max .- min
    lossfn = function (ŷ::AbstractArray{R}, y, tsteps) where {R<:Real}
        s = zero(R)
        sz = size(y)
        @inbounds for j in 1:sz[2], i in 1:sz[1]
            s += ((ŷ[i, j] - y[i, j]) / scale[i])^2 / sz[2] * exp(ζ * tsteps[j])
        end
        return s
    end
    return lossfn
end

min = vec(minimum(train_dataset.data; dims=2))
max = vec(maximum(train_dataset.data; dims=2))
lossfn_inner = experiment_loss_sse(min, max, loss_time_weighting)

lossfn = function (ŷ, y, params, tsteps)
    pnamed = namedparams(model, params)
    return lossfn_inner(ŷ, y, tsteps) +
            loss_regularization / (2 * size(y, 2)) *
            (sum(abs2, pnamed.01) + sum(abs2, pnamed.02))
end
```

Figure B.2: The implementation for the loss function defined by Equation 3.14. Here, *lossfn_inner* is a closure that is used for calculating the loss value with min-max scaling and time weighting, and *lossfn* is a closure that is used for calculating the loss value regularized with the ANN parameters. These two steps are separated because how the ANN parameters are arranged within system parameters *params* can be different among different versions of the model.

```
struct SEIRDFbMobility2{ANN1<:FastChain,
                        ANN2<:FastChain,
                        T<:Real,
                        DS<:AbstractMatrix{T}} <: AbstractCovidModel
    β_ann::ANN1
    α_ann::ANN2
    β_ann_paramlength::Int
    α_ann_paramlength::Int
    β_bounds::Tuple{T,T}
    γ_bounds::Tuple{T,T}
    λ_bounds::Tuple{T,T}
    α_bounds::Tuple{T,T}
    population::T
    time_scale::T
    movement_range_data::DS
    social_proximity_data::DS

    function SEIRDFbMobility2(
        β_bounds::Tuple{T,T},
        γ_bounds::Tuple{T,T},
        λ_bounds::Tuple{T,T},
        α_bounds::Tuple{T,T},
        population::T,
        time_scale::T,
        movement_range_data::DS,
        social_proximity_data::DS,
    ) where {T<:Real,DS<:AbstractMatrix{T}}
        β_ann = FastChain(
            StaticDense(7, 8, mish, initW = Flux.glorot_normal),
            StaticDense(8, 8, mish, initW = Flux.glorot_normal),
            StaticDense(8, 8, mish, initW = Flux.glorot_normal),
            StaticDense(8, 1,
                        x -> boxconst(x, β_bounds),
                        initW = Flux.glorot_normal),
        )
        α_ann = FastChain(
            StaticDense(4, 8, mish, initW = Flux.glorot_normal),
            StaticDense(8, 1,
                        x -> boxconst(x, α_bounds),
                        initW = Flux.glorot_normal),
        )
        return new{typeof(β_ann),typeof(α_ann),T,DS}(
            β_ann, α_ann,
            DiffEqFlux.paramlength(β_ann),
            DiffEqFlux.paramlength(α_ann),
            β_bounds, γ_bounds, λ_bounds, α_bounds,
            population, time_scale,
            movement_range_data,
            social_proximity_data,
        )
    end
end
```

Figure B.3: The third version of the model represented by a Julia struct that holds all the necessary objects and data for modelling the disease a location. *β_ann* and *α_ann* are the object that represent the ANN, and *β_ann_paramlength* and *α_ann_paramlength* are the number of parameters used by each of the ANN, respectively. *β_bounds*, *γ_bounds*, *λ_bounds*, and *α_bounds* are the lower and upper bounds constraints of the parameters. *population* is the population of the modelled location, *time_scale* is length of the modeled time span, *movement_range_data* is a matrix holding the Movement Range Maps data, and *social_proximity_data* is a matrix holding the SPC indices. The two other versions were defined with similar structures, but the baseline version did not contain the fields for the covariates and the second version did not contain the field for the SPC index.

```julia
function (model::SEIRDFbMobility2)(du, u, p, t)
    @inbounds begin
        time_idx = Int(floor(t + 1))
        # states and params
        S, E, I, R, D, N, _, _ = u
        pnamed = namedparams(model, p)
        # infection rate depends on time, susceptible, and infected
        β = first(
            model.β_ann(
                SVector{7}(
                    t / model.time_scale,
                    S / N, E / N, I / N,
                    model.movement_range_data[1, time_idx],
                    model.movement_range_data[2, time_idx],
                    model.social_proximity_data[1, time_idx],
                ),
                pnamed.θ1,
            ),
        )
        α = first(
            model.α_ann(SVector{4}(t / model.time_scale,
                                   I / N, R / N, D / N),
                        pnamed.θ2)
        )
        SEIRD!(du, u, SVector{4}(β, pnamed.γ, pnamed.λ, α), t)
    end
    return nothing
end
```

Figure B.4: The definition of a proxy function around the function shown in Figure B.1 that utilized the ability to make a function call on an object in Julia. Here, *model* is the object that is invoked, *du* is a vector that will be modified in-place to contain the system derivatives at each time step, u is a vector contains the current state of the system, p is a vector contains all the system parameters, and t is the value of the current time step. Unlike in Figure B.1 where p is the parameters for the SEIR model, in this function, p is the parameters of the UDE which included the weights and biases of the ANN.

```
struct Predictor{
    P<:SciMLBase.DEProblem,
    SO<:SciMLBase.DEAlgorithm,
    SE<:SciMLBase.AbstractSensitivityAlgorithm,
}
    problem::P
    solver::SO
    sensealg::SE
    abstol::Float64
    reltol::Float64
    save_idxs::Vector{Int}

    function Predictor(problem::SciMLBase.DEProblem, save_idxs::Vector{Int})
        solver = Tsit5()
        sensealg = InterpolatingAdjoint(; autojacvec=ReverseDiffVJP(true))
        return new{typeof(problem),typeof(solver),typeof(sensealg)}(
            problem, solver, sensealg, 1e-5, 1e-5, save_idxs
        )
    end
end

function (p::Predictor)(params, tspan, saveat)
    problem = remake(p.problem; p=params, tspan=tspan)
    return solve(problem, p.solver;
                 saveat=saveat,
                 sensealg=p.sensealg,
                 abstol=p.abstol,
                 reltol=p.reltol,
                 save_idxs=p.save_idxs)
end
```

Figure B.5: Predictor is a callable struct that contains the necessary arguments that get passed to the *solve* function from the *DifferentialEquations* package. Here, *problem* is an object defining the differential equation to be solved, *solver* is the algorithm used to numerically solve the problem, *sensealg* is the algorithm used for calculating the system gradients, *abstol* and *reltol* are the tolerances for the numerical errors used when solving the system, and *save_idxs* is the list of indices for the compartments that will be saved. An object of the type *Predictor* can be invoked with three arguments, where p is the invoked object, *params* is the system parameters, *tspan* is the predicted time period, and *saveat* is the time steps which will be saved by the solver.

```
struct Loss{Reg,Metric,Predict,DataCycle}
    metric::Metric
    predict::Predict
    datacycle::DataCycle

    function Loss{false}(metric, predict, dataset::TimeseriesDataset,
                         batchsize=length(dataset.tsteps))
        dataloader = TimeseriesDataLoader(dataset, batchsize)
        datacycle = Iterators.Stateful(Iterators.cycle(dataloader))
        return new{false,typeof(metric),typeof(predict),typeof(datacycle)}(
            metric, predict, datacycle
        )
    end

    function Loss{true}(metric, predict, dataset::TimeseriesDataset,
                        batchsize=length(dataset.tsteps))
        dataloader = TimeseriesDataLoader(dataset, batchsize)
        datacycle = Iterators.Stateful(Iterators.cycle(dataloader))
        return new{true,typeof(metric),typeof(predict),typeof(datacycle)}(
            metric, predict, datacycle
        )
    end
end

function (l::Loss{false,Metric,Predict,DataCycle})(params)
where {Metric<:Function,Predict<:Predictor,DataCycle<:Iterators.Stateful}
    data, tspan, tsteps = popfirst!(l.datacycle)
    sol = l.predict(params, tspan, tsteps)
    if sol.retcode != :Success
        return Inf
    end
    pred = @view sol[:, :]
    if size(pred) != size(data)
        return Inf
    end
    return l.metric(pred, data)
end

function (l::Loss{true,Metric,Predict,DataCycle})(params)
where {Metric<:Function,Predict<:Predictor,DataCycle<:Iterators.Stateful}
    data, tspan, tsteps = popfirst!(l.datacycle)
    sol = l.predict(params, tspan, tsteps)
    if sol.retcode != :Success
        return Inf
    end
    pred = @view sol[:, :]
    if size(pred) != size(data)
        return Inf
    end
    return l.metric(pred, data, params, tsteps)
end
```

Figure B.6: A callable struct that encapsulates that logic for calculating the model loss, where *metric* is a function for calculating the differences between the predicted value and the ground truth, *predict* is a function for getting the mode output when given the arguments *(params,tspan,saveat)* as described in Figure B.5, and *datacycle* is an object for cycling through the ground truth data. An object of type *Loss* can be invoked to calculate the model loss by passing the system parameters as the only argument, and the loss can be calculated with or without regularization depending on whether the value of the first parametric type is *true* or *false*.

Glossary

ANN Artificial Neural Network. 1, 2, 5–7, 11–21, 26–29, 33, 56–58

API Application Programming Interface. 1, 22

ARIMA Autoregressive Integrated Moving Average. 1, 5

Bi-LSTM Bidirectional Long Short Term Memory. 1, 5

CNN Convolutional Neural Network. 1, 11

CSV Comma Separated Values. 1, 21, 22, 24

FIPS Federation Information Processing Standards. 1, 24

GADM Database of Global Administrative Area. 1, 23, 24, 35

GAN Generative Adversarial Network. 1, 12

GDP Gross Domestic Product. 1, 6

GPU Graphics Processing Unit. 1, 11

GRU Gated Recurrent Unit. 1, 5

ICU Intensive Care Unit. 1, 4

JIT Just-In-Time. 1, 34

JSON Javascript Object Notation. 1, 22

LSTM Long Short Term Memory. 1, 5, 12

MAE Mean Absolute Error. 1, 33, 40, 41, 48

MAPE Mean Absolute Percentage Error. 1, 33, 37, 40, 41, 48

MLP Multi-Layer Perceptron. 1, 12

MSE Mean Squared Error. 1, 13, 20

NeuralODE Neural Ordinary Differential Equation. 1, 17, 18, 20, 56

NPI Non-Pharmaceutical Intervention. 1, 6

ODE Ordinary Differential Equation. 1, 2, 7, 8, 15, 17–19, 25, 29, 34, 56

PDE Partial Differential Equation. 1, 15–17, 19

PINN Physics-Informed Neural Network. 1, 7, 15, 19

ReLU Rectified Linear Unit. 1, 14

RMSE Root Mean Squared Error. 1, 33, 40, 41, 48

RNN Recurrent Neural Network. 1, 12, 17

SCI Social Connectedness Index. 1, 23, 24, 28, 35

SEIR Susceptible-Exposed-Infective-Removed. 1, 4, 6, 7, 25, 55, 58

SINDy Sparse Identification of Nonlinear Dynamical System. 1, 58

SIR Susceptible-Infective-Removed. 1, 4, 6–10, 25

SPC Social Proximity to Cases. 1, 24, 28, 31–33, 35, 55

UDE Universal Differential Equation. 1, 20, 21, 26, 27, 56, 58

US United States. 1, 5, 6, 21, 23, 24, 28, 31, 35, 37, 39, 40, 47, 48