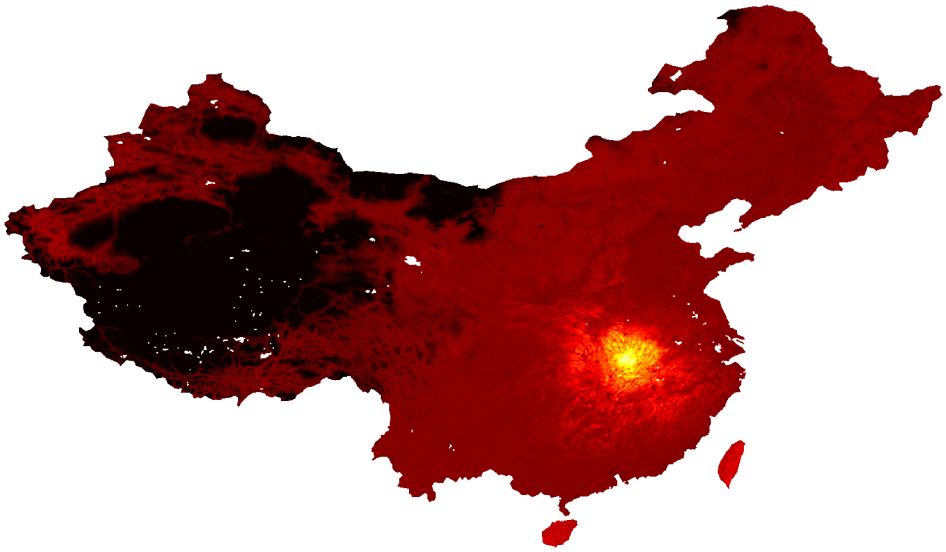
GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Georg-August-Universität Göttingen
Göttingen, Summer Semester 2020

Fine-Scale Spatial Predictions of COVID-19 Cases in China using GIS Data and Deep Learning Algorithms



12-Weeks Thesis as part of the Bachelor of Science in Economics degree at the University of Göttingen

Submitted on: 30.09.2020

By: Davit Svanidze

From: Tbilisi, Georgia

Supervisors: Dr. Benjamin Säfken and Christoph Weisser

External supervisor: Dr. Andre Python

Acknowledgements

The writing process of this thesis has been a long journey. Many great people have accompanied me over the course of this period and I would like to thank them all from my heart.

Firstly, I would like to thank my supervisors, Dr. Benjamin Säfken, Christoph Weisser and Dr. Andre Python, whose expertise and advice have enabled me to understand the details of scientific work. They provided me with all of the resources I needed to write my thesis and have always found time for me in spite of their numerous tasks. Additionally, the supervisors motivated and encouraged me through difficulties during the writing process. Besides aiding me to overcome these challenges, this also raised my self-confidence. This journey was an invaluable academic experience and I will always remember the difference between great and bad scientific works when I think of Dr. Andre Python drawing the distinction between Einstein's book and a manual of a washing machine. I have been very fortunate to have made the acquaintance of such great supervisors and scientists.

In addition, I would like to thank my family and friends. They were supporting me throughout the course of this undertaking as best as they could. My loved ones took over a lot of my responsibilities so that I would have more time to work on my thesis. I would like to thank my future wife, who accompanied me on all the ups and downs of my journey. She empowered me with her love to get through difficult times and I was blessed to share my accomplishments with her. She did everything so that I could concentrate on my thesis. Her healthy, delicious food, cooked for me with love, gave me a lot of strength and energy throughout my endeavour. I would like to thank my friend Levan Veshapidze, who gave me the opportunity to meet Christoph Weisser and thus all my supervisors. Levan always supported me regardless of the time of day, the scope of the task and personal time constraints. Having such a great friend is invaluable.

Finally, I would like to thank my body. It has allowed me to get through this whole time without pain, even though I had to remain sitting for hours on end over the course of several weeks and persevere with little sleep.

Contents

List of figures

List of tables

List of acronyms

ACCESS travel time to the nearest city with a population of more than 50,000.

Adam adaptive moment estimation.

AI artificial intelligence.

ANN artificial neural network.

BN Batch Normalization.

CNN convolutional neural network.

COVID-19 coronavirus disease in 2019.

CSV comma-separated values.

DL deep learning.

GD gradient descent.

IR COVID-19 incidence rate per 100,000 population (Jan-Mar 2020).

JHU Johns Hopkins University.

LAT latitude.

LIR log COVID-19 incidence rate per 100,000 population (Jan-Mar 2020).

LONG longitude.

LTC log COVID-19 total confirmed cases (Jan-Mar 2020).

M million.

MAE mean absolute error.

ML machine learning.

MLP multilayer perceptron.

MLP0 multilayer perceptron without a hidden layer.

MLP1 multilayer perceptron with one hidden layer.

MLP2 multilayer perceptron with two hidden layers.

MSE mean squared error.

MSLE mean squared logarithmic error.

NLP natural language processing.

NN neural network.

OS operating system.

PET potential evapotranspiration.

POP population size.

RAM random-access memory.

ReLU rectified linear unit.

RMSE root mean squared error.

RMSProp root mean square propagation.

RNN recurrent neural network.

SGD stochastic gradient descent.

SLOO spatial leave-one-out.

TanH hyperbolic tangent.

TC COVID-19 total confirmed cases (Jan-Mar 2020).

URBAN urbanity.

VAPOP Voronoi adjusted population size.

WACCESS travel time from a location in China to Wuhan.

WHO World Health Organization.

XGBoost Extreme Gradient Boosting.

1 Introduction

It was hardly conceivable in the beginning of 2019 that in such a globalised world, flight, travel and commercial activity would be restricted; schools and universities would work remotely; and jobs with home office would become the standard. However, all of the above has become a reality since the COVID-19 outbreak in Wuhan (Hubei province), China at the end of 2019.

COVID-19 is often referred to with different names in newspapers and journals. These names are COVID-19, coronavirus and SARS-CoV-2. COVID-19 stands for the coronavirus disease in 2019 which is caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) (Yuen et al., 2020), exhibiting human transmission (Bai et al., 2020). The virus has spread to other locations worldwide and the World Health Organization (WHO) declared COVID-19 a pandemic on March 11, 2020. The pandemic has caused one of the most severe economic shocks since World War II and the costs for the global economy are estimated to be between $30 and $100 billion (Paules et al., 2020).

There are several sources of COVID-19 data in China (Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE), 2020; Pengpai News agency, 2020; Xu et al., 2020). However, the provided data are at the province or district level. A thorough examination of the literature identified only one study Python et al. (2020) which makes fine-scale predictions (with a spatial resolution of 5km) in China using statistical methods and GIS data from Xu et al. (2020).

In the last decade, artificial intelligence (AI) and its subfield machine learning (ML) became very popular and is set to be a technology in high demand for the next decade according to Forbes (Marr, 2020). However, the breakthrough in the field of AI came after neural network (NN) algorithms began to learn very complex non-linear functions. These algorithms stem from the new field deep learning (DL), which is a subfield of machine learning.

Deep learning has the capability to learn and understand very complex non-linear relationships in the data (Inglese et al., 2017). Therefore, it is being used in a variety of disciplines, such as medicine (Babaian et al., 1991; Shen et al., 2017), economics (Fishman et al., 1991; Chang et al., 2019), image analysis (Fukushima et al., 1983; Litjens et al.,

2017), landscape classification (Brown et al., 1998; Buscombe & Ritchie, 2018), climate forecasting (Drummond et al., 1998; Scher, 2018) and spatial analysis (Yuan et al., 2018; Ma et al., 2015; Mubin et al., 2019). He et al. (2015) built a deep learning model which has even surpassed human performance in image classification on the ImageNet dataset.

This study uses Geographic Information System (GIS) data and deep learning algorithms — multilayer perceptron (MLP) (Ramchoun et al., 2016) — to make fine-scale spatial predictions of COVID-19 confirmed cases in China from January to March 2020. In addition to fine-scale confirmed cases, the incidence rate of COVID-19 is predicted to calculate the risk of COVID-19 within 5km grid-cells. Additionally, the deep learning algorithm's performance is compared with XGBoost, which is the most popular method in machine learning competitions hosted by the site Kaggle (Chen & Guestrin, 2016). To make the results, workflow, data, code, useful sources and development tools publicly available for policymakers, researchers and students, the study is hosted on github.com. Additionally, this study uses tools designed to make it easily and quickly reproducible across different computers, servers and operating systems.

Section 2 provides information about the academic knowledge regarding COVID-19 and important factors contributing to COVID-19 transmission. The variables, utilised data for model training, evaluation and predictions are described in section 3. Deep learning and machine learning models, detailed decisions for hyperparameter tuning and the workflow are described in section 4. The results are presented in section 5, with a subsequent discussion in section 6. Section 7 concludes.

2 Literature review

The literature on COVID-19 considers several factors which might affect the spread of the virus. Liu et al. (2020) show the importance of meteorological factors on the number of COVID-19 cases. They used ambient temperature, diurnal temperature range and absolute humidity. They included the migration scale index as a control variable. The authors demonstrate that absolute humidity is significantly associated with a declining number of cases in Chinese cities. Government interventions were controlled for, because they have effects on transmission, and the study concludes that a climate with low temperature,

mild diurnal temperature range and low humidity favours COVID-19 transmission.

The author hypothesised that one infection case could be sufficient for an outbreak of COVID-19. However, Kucharski et al. (2020) offer a contrasting view. They found that a single case introduced to a new location would not necessarily lead to an outbreak, even if the reproduction number is as high as it was in Wuhan in early January 2020. New chains of transmission become more fragile because of high individual-level variation in transmission. Therefore, it is less likely that a single infection case can induce an outbreak. The authors found that if a location has similar transmission potential as Wuhan at the beginning of January 2020 with at least four independently introduced COVID-19 cases, the chance that the virus establishes a foothold in the population is more than 50%.

Carlson et al. (2020) criticise the explanation of COVID-19's spread using only ecological tools, species distributional models, and variables like temperature and humidity. The study states that the results of this type of research have a significant impact on policymakers' decisions and can be misinterpreted (wilfully or not), e.g. countries lift social distancing policies as the temperature rises in spring and summer. Meanwhile, Qian et al. (2020) show that almost all outbreaks of COVID-19 occurred in an indoor environment. The variables humidity and temperature might have some effects on COVID-19's spread, e.g. through better conditions for transmission, but as most of the transmission is through humans, it can spread in regions with varying climates.

There are multiple factors which influence the risk of human transmission of a virus. Wesolowski et al. (2017) discuss the importance of seasonality on human movements and on outbreaks of different viruses. People tend to travel more to celebrate Christmas, New Year and holidays with their families or friends, which increases the risk of rapid spreading. The authors use mobile phone data of 14M to 40M users in each country and show that traffic on rural and urban routes is significantly higher in December and January than in other months, indicating a higher chance of transmission from one location to another. Additionally, Metcalf et al. (2009) have investigated the pre-vaccination dynamics of six childhood infections in Copenhagen. They concluded that term-time forcing is a contributing component to the seasonal dynamics of many childhood infections.

Governments went into lockdown to mitigate the spread of COVID-19. The spread of COVID-19 is determined by the reproduction number (R_0), which defines the mean

number of secondary cases generated by one primary case (Anderson et al., 2020). Salje et al. (2020) calculated in their study that lockdown decreased the reproduction number by 77% from 2.90 to 0.67 in France. Kraemer et al. (2020) emphasise the importance of drastic control measures implemented in China which substantially mitigated the spread of COVID-19. The authors argue that human mobility was a very important factor in China and the biggest part ($R^2 = 0.89$) of the COVID-19 outbreak could be explained by human movement from Wuhan to other locations. They visualise the impact of early stage travel restrictions and the significant reduction of COVID-19 spread in China. However, they stress that restrictions on human mobility would not have been effective if the virus were already widespread. Other researchers investigated the effects of non-pharmaceutical interventions on COVID-19 in 11 European countries (Flaxman et al., 2020). Although large-scale non-pharmaceutical interventions vary between countries, they include border and school closures, and social distancing (e.g. ban of large gatherings). Additionally, they include measures to isolate symptomatic individuals and their contacts, and finally large-scale lockdowns of populations with all but essential travel banned. The authors estimate that a total of 3.5M deaths were to be prevented through these interventions in Europe from the pandemic's start to early May. Although all these policies exhibited positive effects, lockdowns have the most significant effect with an average reduction of 81% in the reproduction number. Prem et al. (2020) show that social distancing measures in China until April were the most effective. This policy decreased the median number of infections by more than 92% (interquartile range 66-97) in mid-2020. The authors predict that a premature and sudden lifting of restrictions could lead to an earlier secondary peak.

Although the use of climate and urban parameters for predicting COVID-19 cases is disputed in the literature, Pirouz et al. (2020) use daily average temperature, relative humidity, and wind speed as climate parameters to investigate whether climate parameters have a delayed effect on the number of total confirmed cases of COVID-19. Their study is conducted in three regions of Italy: Lombardy, Veneto and Emilia-Romagna. Only the climate parameters humidity and wind speed have a significant effect (10% significance level) on the number of total confirmed cases of COVID-19, but their effects exhibit a lag of six to seven days, which could be due to the delayed results of COVID-19 tests. Additionally, the authors use trend analysis and show that population density could affect the number of daily confirmed cases of COVID-19.

Liu (2020) uses variables specific to cities to predict the number of confirmed cases in Chinese cities. The variables are: distance to Wuhan, subway (total length of built urban metro lines), urban area, population density, wastewater (annual quantity of wastewater discharged), garbage (residential garbage connected and transported), greenspace (per capita public recreational green space), daily highest temperature and capital city (whether a city is a capital). The author predicted the number of confirmed COVID-19 cases in Wuhan accurately, with the predictions deviating only slightly from the public value in February 2020. However, the most influential and significant factor is the distance to Wuhan with a negative effect on the number of confirmed COVID-19 cases in China. The significance of other variables varies depending on the model. The variables urban area and capital city have negative coefficients, but the coefficient of the variable capital city is not statistically significant. The variable subway has a significant coefficient, but it is almost zero. The study identifies two variables, wastewater and garbage, which have positive and significant coefficients. However, the author cannot explain their significant positive effect on the number of total confirmed cases of COVID-19. The last variable is greenspace, which has a negative coefficient, but it is not significant.

Deep learning was successfully utilised in the research on COVID-19 to automatically detect cases (Narin et al., 2020; Apostolopoulos & Mpesiana, 2020; Ozturk et al., 2020), to screen pneumonia caused by COVID-19 (Butt et al., 2020), and to predict and analyse positive cases of COVID-19 in India on the state level (Arora et al., 2020). Deep learning was used to make fine-scale predictions of housing prices (Yao et al., 2018) and to map fine-scale land cover (Jia et al., 2019). Nevertheless, other studies using a similar methodology to predict and analyse COVID-19 in China could not be identified during the course of this study. This provides ample opportunity to use findings from similar studies and to address the gap in the literature on COVID-19, deep learning, and fine-scale spatial mapping.

3 Data

This study uses COVID-19 Geographical Information System (GIS) data (Chang, 2006, p. 1) for China from January to March 2020 from Python et al. (2020), which are referred to as an augmented dataset. The dataset is composed of positive-only COVID-19 cases

along with background data (also-called pseudo-absence data). In the augmented dataset, zero cases originate from background data which are generated in locations where there the probability of having positive cases is assumed to be zero since the locations are remote from populated areas. The general framework used to generate background data is further described in Barbet-Massin et al. (2012). The original data were collected by Xu et al. (2020).

The data have three different formats: fine-scale (5km grid-cell) raster data (Chang, 2006, p. 75) in GeoTIFF format (Ritter & Ruth, 1997), geographic shapes of Chinese provinces with aggregated data at the province level in ESRI Shapefile format (ESRI, 1998) and spatio-temporal data in CSV format (Shafranovich, 2005). COVID-19 total confirmed cases exhibited exponential growth until control measures were implemented (Hsiang et al., 2020). Therefore, the temporal component in the data could explain large differences of COVID-19 total confirmed cases between locations. However, in comparison to spatial data, spatio-temporal data requires more complex models and computational resources. Additionally, the fine-scale data are available for this study only at spatial resolution, which enables only spatial predictions of the fine-scale COVID-19 incidence rate and the number of total confirmed cases. In order to develop the model for predicting the fine-scale COVID-19 incidence rate and total confirmed cases at the spatial dimension, which is the goal of the study, the spatio-temporal CSV-format data are aggregated by month (January, February, March 2020), hereby solely retaining the spatial dimension. The spatial data have 2,321 rows.

This study utilises nine variables from the spatial data: (1) LONG, (2) LAT, (3) ACCESS, (4) PET, (5) POP, (6) URBAN, (7) WACCESS, (8) TC and (9) VAPOP. The variables from (1) to (7) are used as features. The features (1) LONG and (2) LAT are geographical coordinates (Chang, 2006, p. 19) of the Chinese cities. According to Zhang et al. (2020), the geographic coordinates bear relevance for the speed of COVID-19's spread. The cities in the east (higher longitude) tended to record a COVID-19 case earlier than the cities in the west (lower longitude). The cities in the north (higher latitude) were more likely to report the first case later than those in the south (lower latitude). Additionally, the cities in China are mainly located in the east (higher longitude), where the potential spread is higher. The feature (3) ACCESS is travel time to the nearest city with a population of

more than 50,000 (Weiss et al., 2018), and (7) WACCESS is travel time from a location in China to Wuhan (computed from (Weiss et al., 2018)). According to Kraemer et al. (2020), about 90% of the COVID-19 outbreak can be explained by human movement. Also, Liu (2020) reported the variable distance to Wuhan as the most important factor to explain the COVID-19 outbreak in China. Since COVID-19 is characterised by human transmission, the cities with lower travel times to other cities with a population exceeding 50,000 could have higher incidence rates and more total confirmed cases of the virus. Additionally, the travel time to other cities can capture control and social distancing measures in an observed city and in neighbouring cities. The stricter the control measures, the longer the travel time (Prem et al., 2020). The feature (5) POP represents population size provided by WorldPop (Tatem, 2017). According to Tian et al. (2020), COVID-19 arrived sooner in cities with larger populations and with more travellers from Wuhan. Additionally, the large population size of a location increases the size of the susceptible population (Maier & Brockmann, 2020). The urbanity of a location in China is represented by (6) URBAN from Global Urban Footprint data (Esch et al., 2017). Cities with high urbanity tend to demonstrate lower population density and more potential contacts among people. Therefore, cities with high urbanity should have a higher number of total confirmed cases (Liu, 2020; Pirouz et al., 2020). The feature (4) PET represents humidity by potential evapotranspiration, which measures the potential amount of evaporation. According to Liu et al. (2020), relative humidity is positively correlated with the number of total confirmed cases of COVID-19 in China. Humidity could facilitate the transmission of the virus and make the population more vulnerable.

The variables (8) COVID-19 total confirmed cases (Jan-Mar 2020) (TC) and (9) VAPOP - adjusted population size using Voronoi polygons (Burrough et al., 2015, pp. 160) are used to normalise (8) TC by population size and to calculate the COVID-19 incidence rate per 100,000 population (Jan-Mar 2020) (IR) in Equation 3.1. The incidence rate is a more robust measure than the number of total confirmed cases because it is expected that cities with a larger population size have more of the latter. Therefore, it is difficult to determine whether TC is high because of the population size or if there actually is a higher risk of COVID-19 transmission. However, there are significantly more cities with low values of IR and the value of IR is much higher in Wuhan than in other cities, which makes the distribution of IR positively skewed. To make the distribution of IR less skewed, LIR in

Equation 3.2 is calculated. The term *+1* is very important in the logarithm calculation as IR values can be zero and the logarithm of zero is not defined. Additionally, it has a beneficial property when IR equals zero. In this case, the transformed value also remains zero $\ln(0+1) = \ln(1) = 0$. The target variable will be LIR and predictions can be easily transformed backwards to IR and TC.

$$IR = \frac{TC}{VAPOP} \times 100,000 \tag{3.1}$$

$$LIR = \ln(IR + 1) \tag{3.2}$$

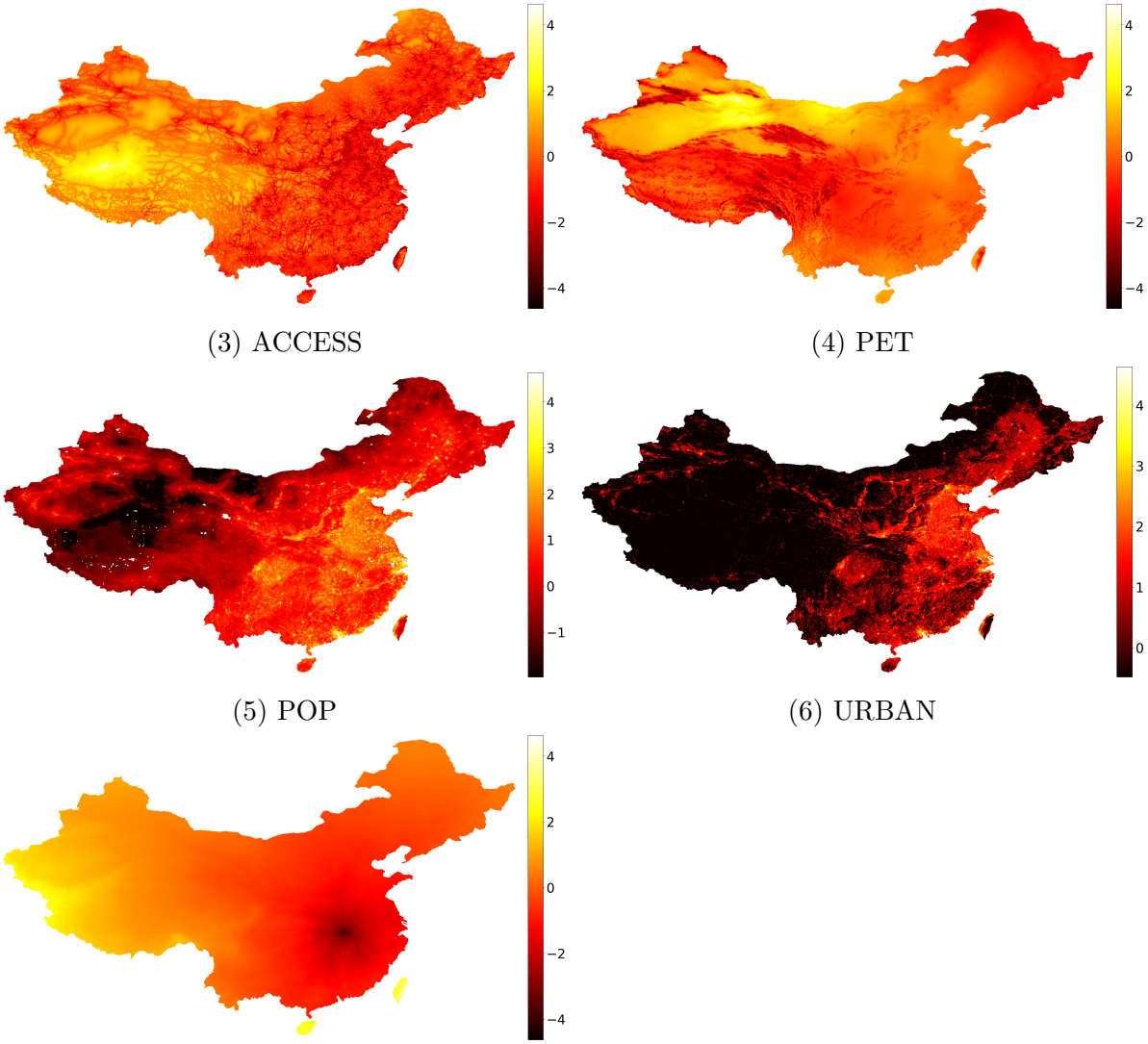
Values of the variable (8) TC in the data are positive-only cases of COVID-19. Additionally, the variable has many missing values (NA). Although there are no data points in the data with zero COVID-19 cases, there is a high probability that this is the case for locations which are remote from populated areas. This study assumes and generates zero COVID-19 cases (TC = 0) for all locations which have higher travel time (ACCESS) than 95% of the data points (the highest 5 percentile) to cities with a population exceeding 50,000, or in locations where the population size (POP) is lower than 95% of the data points (the lowest 5 percentile). In fact, all these locations had missing values of TC and they were replaced by zeros. Therefore, IR is also zero in these locations.

The missing data points have been cleaned from the spatial data and the last variable is (8) LIR. The cleaned data are summarised in Table 3.1. There are 812 data points without missing values. Features (3) - (7) are normalised by the quantile normalisation technique (Bolstad et al., 2003). The feature (1) LONG is normalised by dividing by the highest possible absolute value of longitude (180) and the feature (2) LAT is normalised by dividing by the highest possible absolute value of latitude (90). All absolute minimum and maximum values are in the range of 0 and 5, which improves model stability (Bengio, 2012).

Table 3.1: Summary statistics for the cleaned data. The features are: geographic coordinates (1) longitude (LONG) and (2) latitude (LAT), (3) travel time to the nearest city of with a population exceeding 50,000 (ACCESS), (4) potential evapotranspiration which accounts for the role of humidity (PET), (5) population size (POP), (6) urbanity (URBAN) and (7) travel time to Wuhan (WACCESS). The target variable is (8) log COVID-19 incidence rate per 100,000 population (Jan-Mar 2020) (LIR)

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
count	812	812	812	812	812	812	812	812
mean	0.60	0.37	−0.48	0.12	0.67	0.79	−0.43	0.61
std	0.07	0.07	1.41	0.78	1.58	1.16	1.11	0.77
min	0.43	0.20	−3.49	−2.36	−1.97	−0.48	−3.68	0.00
25%	0.56	0.33	−1.49	−0.40	−0.22	−0.48	−1.13	0.06
50%	0.61	0.38	−0.73	0.17	0.91	0.80	−0.72	0.34
75%	0.65	0.42	0.30	0.58	1.76	1.58	0.31	0.83
max	0.74	0.55	3.44	2.79	4.63	4.34	2.69	4.89

The data in Table 3.1 are used for model training, validation and testing, but for fine-scale predictions, features with the same spatial resolution are needed. The fine-scale data of the features (3) ACCESS, (4) PET, (5) POP, (6) URBAN, (7) WACCESS are presented in continuous rasters in Figure 3.1. The features (1) LONG and (2) LAT are coordinates of each grid-cell and therefore, they are extracted from a raster (e.g. (3) ACCESS). There are approximately 500,000 grid-cells in the continuous raster covering China, therefore about 500,000 sets of feature values and the same number of grid-cell values to predict. All fine-scale feature values are normalised in the same way as the features in Table 3.1, thus exhibiting similar distributions.



(3) ACCESS (4) PET

(5) POP (6) URBAN

(7) WACCESS

Figure 3.1: Maps of fine-scale features. The features are (3) travel time to the nearest city with a population exceeding 50,000 (ACCESS), (4) humidity represented by potential evapotranspiration which measures the potential amount of evaporation (PET), (5) population size (POP), (6) urbanity (URBAN) and (7) travel time from a location in China to Wuhan (WACCESS)

Finally, the province-level data of the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE) (2020) - the most commonly reported COVID-19 data in the media and research — are used to facilitate a comparison with the fine-scale predicted values of TC aggregated at the province level. Since the model initially makes fine-scale predictions of LIR, they are transformed backwards to IR and then multiplied by non-normalised fine-scale values of POP to calculate corresponding predicted values of TC. To aggregate the fine-scale predictions of TC at the province level in China, ESRI shapefiles are used.

Machine learning engineers mostly utilised the 80/20 rule for data splitting, which means 80% of data will be in the training set and 20% of data in the test set. However, this rule has changed with 60/20/20 method, which means 60% of data is used for the training set, 20% for the validation set and 20% for the test set. The validation set is often called a development or dev set in other studies, because it is used to evaluate model performance during training. The validation set helps researchers to quickly assess model underfitting or overfitting problems which are explained in section 4.4. Additionally, model evaluation is more robust than with only two sets (training and test), because in this setting the test set represents unseen data for the model and if the loss of the test set is also low, then it can be assumed that the model can generalise well. The splitting process of data should be random, which protects it from being biased towards any characteristic of the data. Generally, it is acceptable that the distribution of the training set is different from the validation and test sets (Ng et al., 2018d). However, the validation and test sets should have similar distributions so model performance can be properly evaluated. If there are many variables with values between 0 and 1 in the validation set and with values between 100 and 1,000 in the test, the model will have performed well on the validation set, but will have very poor performance on the test set. Additionally, it should not be forgotten that the validation and test sets only serve model evaluation. If data have a million points than it is better to use a larger portion of the data for model training, because 10,000 data points in the validation and test sets would be enough to assess model performance. Therefore, a more appropriate split would be, for instance, 98% training set, 1% validation set and 1% test set. The data of this study have only 812 data points, thus the 60/20/20 rule is applied. This study uses the Python library scikit-learn (Pedregosa et al., 2011) to split the data randomly.

4 Method

4.1 Research workflow

The workflow (Figure 4.1) begins with getting theoretical (epidemiological) knowledge on the risk factors of COVID-19. GIS data are needed to make fine-scale spatial predictions of COVID-19 cases in China. The data should be cleaned and prepared to have the correct

format. Additionally, variables should be normalised to have similar ranges. After the data is prepared, the model development can begin. This study uses a deep learning algorithm (MLP) as the main method and a machine learning algorithm (XGBoost) as the benchmark for its deep learning counterpart. Both algorithms, especially MLP, have multiple hyperparameters. This study shows in detail how the MLP model is developed as the main method. After the model is developed, its hyperparameters and accuracy should be optimised. Different models are optimised and evaluated. In the next step, the model that performs best on the unseen data is selected. After the optimal model is selected, it is used to make fine-scale spatial predictions of COVID-19 cases in China, which has been the goal of the research. The fine-scale spatial prediction can then be aggregated to the province level and compared with other datasets.

Figure 4.1: Research Workflow. The research workflow visualises steps which are conducted from theory to predictions

The research workflow to train, evaluate, test and predict the MLP models and make the predictions requires high computational power and several days on a personal computer. Therefore, a cloud server with 32 processor cores and 128 gigabyte random-access memory (RAM) is used.

However, running the code on various hardware with different operating systems (OS) causes many problems. The first problem is that the Python programming language does not have a good package dependency management system, with the issue becoming more severe when packages on different OS are installed. Another problem is transferring code from one system to another quickly and tracking changes and development of the project. This study attaches a high value to easy and fast replication, and to the consistency of the results independent of hardware and OS.

This study uses several tools to facilitate the tracking of its development as well as the replication of the results. It is also possible to easily run the code on different computers and servers with the help of those tools. The version control system Git (Wilson, 2006) and its hosting service github.com provides access to the project code, and enables easy replication of the project by others. Besides tracking the project's development, it enables the conduction of more experiments because it needs only one command to recover any state of the project without any losses. If the number of participants in the project increases, Git facilitates collaboration. This study also uses Git and Github to transfer files much faster than standard file transfer tools and track changes between the cloud server and the local machine.

It is important to run Python code in different environments with as little effort as possible to save time and invest it in research. Therefore, this study uses Conda package management and Docker containers (Merkel, 2014). Conda runs in a lightweight Docker container and this enables all users to manage Python packages to isolate this project and its requirements from other projects, to set the OS manually in the container and to make the project independent of the current OS on the machine.

To summarise, users need to only install Git and Docker on any computer or server (if they are not already installed on the machine), and then they can completely replicate all results with a few commands in the present and in the future. Additionally, this workflow helps to save a lot of time and avoids OS-specific issues by running the same code on a local machine and on the cloud server during project development.

4.2 Algorithms

Deep learning

Deep learning uses many different architectures of neural networks. The most popular is the artificial neural network (ANN) which is able to learn very complex non-linear functions (Yang et al., 2019). ANN with at least one hidden layer is called multilayer perceptron (MLP). It is commonly used for regression and classification problems in various fields. (Ramchoun et al., 2016). MLP is an extension of Feed-Forward Neural Network (Feed-Forward NN) architecture (Bishop, 1995, p. ix). A classical Feed-Forward NN has only two layers: input and output. MLP has hidden layers in addition to input and output, which enables it to learn significantly more complex functions than a classical Feed-Forward NN (Bishop, 1995, p. 123). However, if the data have too many features (e.g. 1,000), MLP's computational performance will decrease significantly. Many features in an input layer also need many hidden units in hidden layers to be able to learn patterns. For example, a greyscale 64x64 pixel image would have 4,096 features (the number of pixels). Therefore, a convolutional neural network (CNN) uses convolutions to increase computational efficiency of neural networks. It is mostly used for image classification problems (Krizhevsky et al., 2012). MLP and CNN have difficulties to predict sequence data (e.g. time-series), because they do not provide functionality to consider previous or next sequences of data for current prediction. Therefore, a recurrent neural network (RNN) is commonly used for time-series data or Natural Language Processing (NLP). The COVID-19 data provides only seven input features, no images and time-series data, thus MLP is the most suitable architecture.

An MLP model has multiple hyperparameters which determine its architecture and performance. The values of the hyperparameters depend on the data. Although there are papers with practical advice for hyperparameter values such as Bengio (2012), there is no algorithm or simple recipe to ascertain the optimal values of hyperparameters. This study shows in detail how the MLP model is developed, optimised and evaluated.

Machine learning

Extreme Gradient Boosting (XGBoost) is a machine learning algorithm which has gained its popularity by winning numerous machine learning competitions (Chen & Guestrin, 2016). The competitions have been hosted by the site Kaggle. Additionally, every winning team in the top-10 used XGBoost algorithm in KDDCup (the annual Data Mining and Knowledge Discovery competition) in 2015, which made XGBoost even more popular. This study will use it as a benchmark for the MLP model. Friedman (2001) developed the first gradient boosting algorithm. For a brief explanation of the gradient boosting algorithm, it is easier to take the classification problem which is shown in Figure 4.2.

The boxes 1, 2 and 3 (Figure 4.2) are called weak learners (Friedman, 2001), which are defined as models whose performance is at least slightly better than random guessing (Saraswat, 2019). The gradient boosting algorithm is a sequential process and it starts with Box 1 to define the decision boundary D1. However, three plus signs are misclassified. Then the loss function is calculated. To minimise the loss function, the new weak learner Box 2 is added, which tries to improve the classification errors of Box 1. However, the decision boundary D2 does not classify two minus signs correctly. To minimise the loss function and improve the classification errors of Box 2, the new weak learner Box 3 is added. However, the decision boundary D3 also has some classification errors. Finally, all weak learners are weighted and used to make the strong learner Box 4, which classifies all plus and minus signs correctly. The very next model capitalises on the error of the previous model and tries to reduce it (Saraswat, 2019). This sequential process and the depth of strong learners look like a tree and it is also called gradient tree boosting.

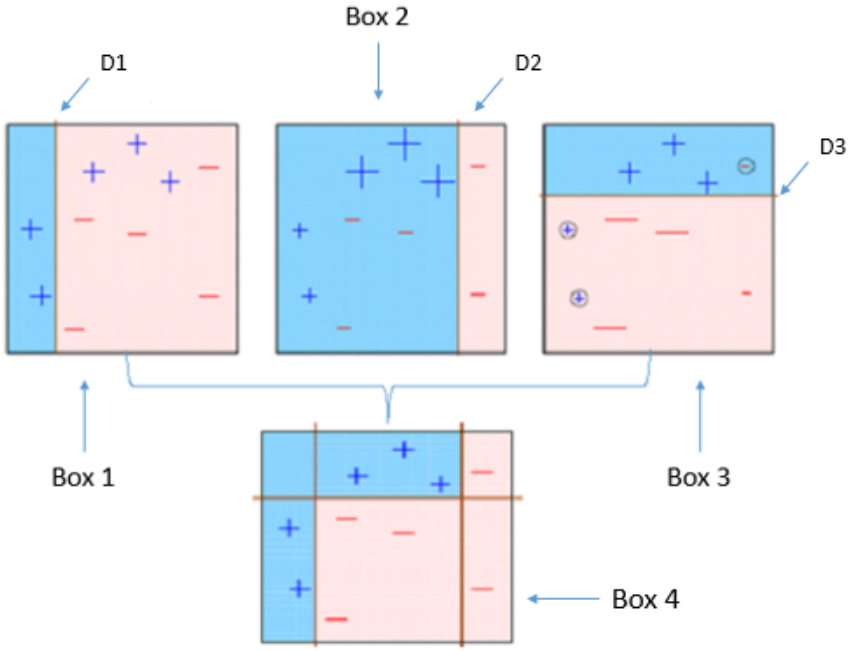


Figure 4.2: Classification of plus and minus signs using the gradient boosting algorithm. Boxes 1, 2 and 3 represent the weak learners and box 4 represents the strong learner. D1, D2 and D3 represent decision boundaries for the weak learners. The weak learners together build the strong learner (Saraswat, 2019)

Chen & Guestrin (2016) developed the end-to-end tree boosting system XGBoost. XGBoost can be used for classification and regression problems which are implemented as two separate methods: XGBClassifier and XGBRegressor (XGBoost, 2018). It improves the performance of previous gradient boosting algorithms in several ways. Firstly, it is implemented and optimised for different programming languages. Secondly, the training time is much lower. Additionally, it automatically uses L1 and L2 regularisation techniques and tries to avoid the overfitting problem. It does not need data scaling or data cleaning, because it handles both automatically. The algorithm uses computation power optimally and enables parallelisation. Additionally, it can be scaled easily to process billions of data examples and does not need many parameters to be tuned. It is these features that make it so popular for machine learning competitions.

XGBoost will be configured with hyperparameters according to best practices (Brownlee, 2016, p. 81). The number of trees will be set to 100, maximum tree depth to 3, learning rate to 0.1 and subsample to 1 (ratio of training instances (Chen & Guestrin, 2016)).

4.3 Model development

Problem type

There are two main types of problems in machine learning: classification and regression. It is easy to distinguish between them because the problem type depends on the target variable. If the target variable is categorical (e.g. yes/no) then it is a classification problem. If the target variable is continuous then it is a regression problem. If the target variable is ordinal, it can be handled as a special case of classification (Frank & Hall, 2001). The problem type has a significant impact on other decisions for the model (neural network structure, loss function, optimisation algorithms, activation functions, etc). This study uses LIR as a target variable, which is continuous and therefore this is a regression problem.

Loss function

The choice of a loss function depends on the problem type. This study has a regression problem and there are three main loss functions for it. These three loss functions are mean squared error (MSE) loss, mean squared logarithmic error (MSLE) loss and mean absolute error (MAE) loss.

MSE loss is the widespread loss function. It is often used as a default loss function in different machine learning libraries. It is calculated as the average of squared differences between true target values and predicted values. The formula in Equation 4.1 shows how MSE loss is calculated, where N is the number of data points, y_i is the i-th true target value and $\hat{y}_i$ is the i-th predicted value. MSE loss always has positive values because of squared differences. The best possible value is 0.0. As all errors are squared, the model is disproportionately penalised for producing larger errors.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4.1}$$

There are regression problems with a target variable with widely dispersed values, but it is not needed to penalise the model as severely as the MSE loss when predicting large values. MSLE loss is a better choice in this situation (Brownlee, 2018, p. 60). In addition

to the MSE loss calculation steps, the MSLE loss makes an additional step to scale true and predicted target variable values by logarithm. The term *+1* is used in logarithmic form to avoid mathematical calculation problems if y_i or $\hat{y}_i$ equals zero. The value of $\ln(0)$ is not mathematically defined, but $\ln(0+1) = 0$ is. According to Brownlee (2018, p. 60), it has the effect of relaxing the penalising effect of large differences in large predicted values of the target variable, and as a loss measure it may be more appropriate if the model is directly predicting unscaled quantities.

$$MSLE = \frac{1}{N}\sum_{i=1}^{N}(\ln(y_i + 1) - \ln(\hat{y}_i + 1))^2 \tag{4.2}$$

The last prevalent loss function for regression problems is the MAE loss. If the distribution of the target variable is approximately Gaussian but it has outliers, then MAE is the appropriate loss function to use in this case (Brownlee, 2018, p. 62). It is more robust to outliers. The formula in Equation 4.3 shows how MAE loss is calculated. MAE loss is the sum of absolute differences between true and predicted target variable values. Like MSLE, it performs better than MSE when the values of a target variable are not scaled.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{4.3}$$

This study uses MSE loss, because the target variable (LIR) values are scaled. Therefore, it does not have large outliers. It is important that relatively larger values are predicted correctly because it is very interesting to identify locations in China with a COVID-19 outbreak.

Number of hidden layers

There is no method or rule to know how many hidden layers are necessary in advance. Woods & Bowyer (1997) show with their theoretical research that any complex and non-linear function can be estimated with a single hidden layer. Additionally, Mollalo et al. (2019) use MLP models with one and two hidden layers, respectively, to make predictions for spatial data. Their results show that MLP with one hidden layer performs equally or better than MLP with two hidden layers measured by root mean squared error (RMSE) (lower RMSE is better; training: $0.33 < 0.34$, validation: $0.35 = 0.35$, test: $0.35 < 0.36$).

Each hidden layer passes their calculated values to the next one, thus the next hidden layer can learn more complex non-linear functions than the previous ones. This study will use MLP models without a hidden layer, with one hidden layer and with two hidden layers. An increasing number of hidden layers usually causes an overfitting problem and is computationally more expensive. According to Brownlee (2018, p. 23) an MLP model with too many layers will likely be unable to learn relationships in the training dataset, getting stuck during the optimisation process.

Number of hidden units

Each hidden layer consists of hidden units. Methodological research shows that a model with a larger number of hidden units per hidden layer performs better than a model with less hidden units (Bengio, 2012). Therefore, it is advantageous to try a large number (100+) of hidden units and evaluate model performance. If the model exhibits bias, more hidden units can be added. If overfitting occurs, the number of hidden units should be decreased or the model should be simplified. Larochelle et al. (2009) explored if model performance changes when the number of units are distributed differently across hidden layers. They found that the same size for all hidden layers worked generally better or the same as using a decreasing size (pyramid like) or an increasing size (upside down pyramid). However, they mentioned that these results could be conditional on the data, thus it is better to try different sizes if model performance is not satisfactory. Bengio (2012) recommends to have more hidden units than input units (features) in the first hidden layer than vice versa, because it showed better performance in most tasks he worked on. However, it should be considered that an increasing number of units leads to higher computational costs.

In this study, an initial model with 128 hidden units will be used and these will be increased or decreased by evaluating model performance for different numbers of hidden units.

Activation function

Hidden units have usually non-linear activation functions. It is possible to use a linear activation function, but the neural network will not be able to learn complex functions

(Brownlee, 2018, p. 141). Calculation of the activation of a hidden unit is a two-step process. Firstly, a linear summation of all inputs of the hidden unit from the previous layer is calculated using weights and biases of the hidden unit, and then the sum is transformed by a non-linear activation function for this hidden layer. The first step is shown in Equation 4.4 and the second step in Equation 4.5.

$$z_i^{[l]} = {W_i^{[l]}}^T a^{[l-1]} + b_i^{[l]} \tag{4.4}$$

$$a_i^{[l]} = g^{[l]}(z_i^{[l]}) \tag{4.5}$$

The variable $z_i^{[l]}$ describes the linear sum for the hidden unit i in the hidden layer l of the inputs from the previous hidden layers' $(l-1)$ hidden units $a^{[l-1]}$ using weights and biases of the current hidden units. In the next step, the linear sum $z_i^{[l]}$ is transformed by the non-linear activation function g of the hidden layer l, so that the value $a_i^{[l]}$ of the hidden unit i in the hidden layer l is calculated.

There are three commonly used activation functions: the sigmoid function, the hyperbolic tangent function (TanH) and the rectified linear unit (ReLU). To note, ReLU means a unit which uses the rectified linear function, but in this study ReLU is also used as a synonym or an abbreviation of the rectified linear function. The main activation functions' formulas are shown in Equation 4.6, 4.7 and 4.8.

$$Sigmoid(z) = \frac{1}{1+e^{-z}} \tag{4.6}$$

$$TanH(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{4.7}$$

$$ReLU(z) = max(0, z) \tag{4.8}$$

These activation functions have different shapes and intervals (Figure 4.3). The sigmoid and TanH activation functions have an S-shaped curve, but their intervals are different. Sigmoid values are between 0 and 1 and TanH values are between -1 and 1. The general problem with both functions is that they saturate (Ng et al., 2018e). If z has large values, sigmoid and TanH function values snap to 1.0 and vice versa, if z has small values, sigmoid and TanH function values snap to 0 or -1 (Brownlee, 2018, p. 143). Both activation

functions are only sensitive to changes (higher derivative) when z is near 0. Therefore, optimisation algorithms need more training time and calculations to reach the optima. However, they are often unable to achieve the best possible performance.

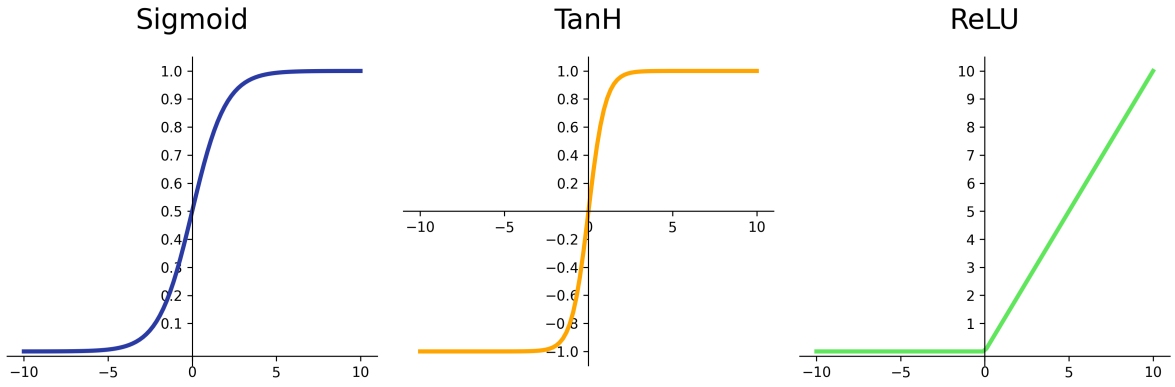


Figure 4.3: Activation functions

ReLU overcomes the saturation problem characterising sigmoid and TanH. It is a non-linear function, but it is very sensitive to all z values when $z > 0$, because it is linear in this interval. It allows the deep neural network to learn very complex functions. Additionally, it is very cheap for computers to calculate the function's value and the function's derivative. Its discovery may easily be considered one of the few milestones in the deep learning revolution (Brownlee, 2018, p. 143). However, ReLU also has limitations. The most important limitation is the so called "dying" ReLU. It happens if the unit learns a large negative bias term and therefore, z is always negative. This means that a hidden unit with this problem will always output an activation value of 0.0 (Brownlee, 2018, p. 150). There are several extensions of ReLU to solve this problem (Brownlee, 2018, p. 150), but usually, this does not occur.

ReLU has many advantages over sigmoid and TanH. Therefore, it is used as a default activation function in most neural networks for hidden units, even though sigmoid and TanH had been used as default activation functions for several decades (Ng et al., 2018a). However, this study uses ReLU also in the output layer, because the default linear activation function for regression problems returns positive and negative values, but the target variable of this study has only positive values. Therefore, this study will use ReLU in the hidden and output layers and apply recommended techniques from the literature to improve MLP performance which uses ReLU (Brownlee, 2018, p. 147). One of the

techniques is to use *He* initialisation of weights (Brownlee, 2018, p. 149). This should make MLP training faster and easier.

Regularisation

Regularisation is used to decrease variance and avoid the overfitting problem. There are three main regularisation techniques: L1, L2 and and dropout. All of them decrease model complexity through constraining model weights. The models with an overfitting problem have higher weight values or a higher number of weights (Brownlee, 2018, p. 247). L1 regularisation is rarely used, but both L1 and L2 regularisation techniques (Tikhonov, 1943) constrain weight values by adding a regularisation term in the loss function. If the model has weight values that exceed their respective optima, this induces a higher loss value. Therefore, the regularised model learns to decrease the weight values until the optimal weights and loss are achieved.

Unlike L1 and L2 regularisation, the dropout regularisation technique constrains the number of weights (Hinton et al., 2012). It does not add any regularisation term in the loss function. The dropout regularisation randomly drops hidden units of a hidden layer with a certain rate on each pass. The rate is called the dropout rate and is in the range from 0 to 1. A dropout rate of zero means that no hidden unit is dropped and it is equivalent to not using dropout regularisation. If the dropout rate equals 1, all hidden units in the specific hidden layer are dropped. Therefore, if dropout regularisation is used, the dropout rate is always between 0 and 1. Hidden units cannot continue to rely on their counterparts and they adjust their weights. This regularisation technique has been widely adopted, because it not only regularises a model, but also makes it more robust (Phaisangittisagul, 2016). Dropout regularisation contributes to the robustness of the model, because many simpler neural networks are tried by randomly dropping hidden units.

This study will use the dropout regularisation technique to avoid the overfitting problem and to increase model robustness. The recommended dropout rates are between 0.5 and 0.8 (Garbin et al., 2020). This study will try these dropout rates as defaults, but other dropout rates will be also explored from 0.1 to 0.9.

Batch Normalization

Ioffe & Szegedy (2015) introduced the Batch Normalization (BN) technique in 2015. BN uses the logic of input data normalisation, which has been proven to accelerate training and make models more stable. BN normalises z values (Equation 4.4) in hidden layers. These values are then transformed with activation functions (Equation 4.5). BN is commonly used in multiple types of neural networks because of its advantages. It substantially decreases the training time, improves model performance and makes models more stable. Additionally, it has a slight regularisation effect and decreases the impact of weight initialisation on model performance. The authors tested BN performance for a classification problem and showed that the model with BN achieved the same accuracy as the best published results at the time by training steps that were 14 times fewer in number. Additionally, their model outperforms the original model significantly. Since BN offers important advantages, it will also be utilised in this study.

4.4 Model optimisation

Gradient optimisation algorithm, learning rate, batch size and epochs

Gradient optimisation algorithms are used for the so-called backpropagation (Rumelhart et al., 1986) in neural networks to adjust the weights of hidden units and to minimise the loss function. It is useful to observe a backpropagation step when weights and biases are updated in order to understand the differences between optimisation algorithms.

$$W_i^{[l]} := W_i^{[l]} - \alpha \, dW_i^{[l]} \tag{4.9}$$

$$b_i^{[l]} := b_i^{[l]} - \alpha \, db_i^{[l]} \tag{4.10}$$

Equation 4.9 shows how the weights of the hidden unit i in the hidden layer l are updated. Equation 4.10 shows the same for biases. The sign $:=$ stands for mathematical value assignment, so that newly calculated values are assigned to weights and biases. The variable α is the learning rate, and $dW_i^{[l]}$ and $db_i^{[l]}$ stand for the derivatives of weights

and biases of the hidden unit i in the hidden layer l. The derivative has an advantageous property that if the current weights are lower than the optimal values and therefore, the loss is high, then the derivative would be negative. Hence, the term $-\alpha\, dW_i^{[l]}$ in Equation 4.9 would be positive. Therefore, the value of $W_i^{[l]}$ will increase, moving towards the optimal value. If the current value of $W_i^{[l]}$ is higher than the optimal value and therefore, the loss is high, then the derivative $dW_i^{[l]}$ would be positive and the term $-\alpha\, dW_i^{[l]}$ negative. Consequently, the new value $W_i^{[l]}$ will decrease, and thus move towards the optimal value. The values of $b_i^{[l]}$ are adjusted with the same logic.

The learning rate α is a hyperparameter and it has a significant impact on the optimisation process and the training time. If α is too low, then this will scale down the values of the derivatives too much and therefore, the training needs a very long time to achieve the optimal weights and biases. Additionally, it could also stop the training before the optimal values are achieved, because the derivatives' values are small when they are near the optimal value and if the hyperparameter α is also very low, then each update will be approximately zero. On the other hand, if the value of α is too high, then it can increase the derivative to the degree that the optimal values of the weights and biases are consistently overshot, thus optimal values will be never achieved. Both problems can be observed when the training loss does not decrease or it increases greatly with more training. There is no common value for a learning rate addressing all problems. However, machine learning researchers usually use $\alpha < 1$ and they start with values 0.1 or 0.01. Then they observe the loss changes and adjust the alpha value incrementally.

Optimisation algorithms are used to calculate the derivative terms $dW_i^{[l]}$ and $db_i^{[l]}$. The most popular algorithm is gradient descent (GD). GD has different versions known as stochastic gradient descent (SGD) and mini-batch gradient descent. GD is also known as batch gradient descent. The difference between these three versions is the batch size for which derivatives are calculated. Batch size describes how many training examples go through a neural network in each iteration. For GD, loss and derivative calculations are done after all training examples go through a neural network simultaneously (batch size is equal to the number of training examples). SGD does the same calculations after each training example goes through the neural network (batch size is equal to one). Mini-batch GD performs the calculations after a fixed number of training examples go through the

neural network (batch size is equal to 32, 64 or 128 e.g.).

Therefore, the batch size determines which type of gradient descents are used. All of them have advantages and disadvantages. Figure 4.4 shows different optimisation paths. The left path has frequent oscillations and the right one is smoother. The literature shows that if the batch size is equal to one, then the optimisation path looks like the left image with more oscillations. When the batch size increases, it becomes smoother like the right image. The smoother the optimisation path the faster the training process. However, if the data contains many observations, then it needs very high computational power that is often unavailable. Additionally, Goodfellow et al. (2016) propose not to use the batch size of all training examples, because it does not generalise as well as the models with smaller batch sizes. Therefore, researchers usually divide all training examples in mini-batches with a fixed number of training examples 32, 64, 96, 128 or something else. Therefore, mini-batch GD is not smooth and fast as GD, but it does not need as much computational power and is faster and smoother than SGD. Additionally, it can generalise better than GD. Furthermore, all of these three types of GDs (in spite of different batch sizes) are often called SGDs. Hence, SGD with a batch size of 32 is the same as mini-batch GD.

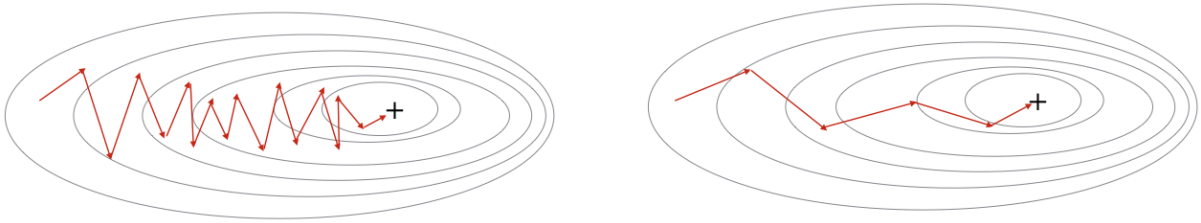


Figure 4.4: Oscillations during neural network optimisation (Ng et al., 2018c). The crosses represent the optima. The optimisation path is more noisy with a small batch size (left) than with a large batch size (right)

Machine learning researchers discovered other algorithms to make oscillations as smooth as possible, thus decreasing training time and achieving better optima. There are three approaches: SGD with momentum, root mean square propagation (RMSProp) and adaptive moment estimation (Adam). These algorithms differ only by calculating and scaling the derivative terms $dW_i^{[l]}$ and $db_i^{[l]}$. Rumelhart et al. (1986) proposed SGD with momentum, where they use the technique called exponentially weighted averages (Crowder & Wiel, 2014) to smooth oscillations. Hinton (2012) explains that RMSProp uses root mean square to calculate and scale derivatives. Kingma & Ba (2014) created Adam which

puts together SGD with momentum and RMSProp algorithms. It has been enjoying increased popularity as an optimisation algorithm (Kingma & Ba, 2015).

Additionally, optimisation performance depends significantly on the number of epochs. An epoch refers to a single pass through the full training set examples (Nielsen, 2015). If the learning rate is low, then it needs more epochs to train. The number of epochs should be empirically explored for each problem. Often, the number of epochs is set to a high value and an early stopping algorithm is used to stop the training process before the maximum number of epochs is reached if the validation loss does not improve further (Prechelt, 1998).

This study will use the Adam optimisation algorithm which has achieved top results in multiple studies (Kingma & Ba, 2015). Although the default learning rate for Adam is 0.001, this study begins with a higher learning rate of 0.01, with the value being halved iteratively until 0.0003125 is reached. The initial batch size will be 64 and then other batch sizes of 32, 96 and 128 will be explored. Finally, epochs will be set to 10,000 and an early stopping algorithm will be used.

Problem of underfitting and overfitting

Underfitting occurs when a statistical model is too simple to adequately capture the underlying structure of the data (Goodfellow et al., 2016, pp. 109). It is shown in Figure 4.5 (left). In contrast, overfitting occurs when a statistical model is too complex and tailored too closely to a particular set of data, and may therefore lack external validity or fail to predict future observations reliably (LEXICO, 2020). It is shown in Figure 4.5 (right). The desirable model is a balance between underfitting and overfitting, which is shown in Figure 4.5 (center). It fits the data well, but it is not too complex and not very specific to this data, thus generalising better. This, as well as other studies try to find the model that is “just right”.

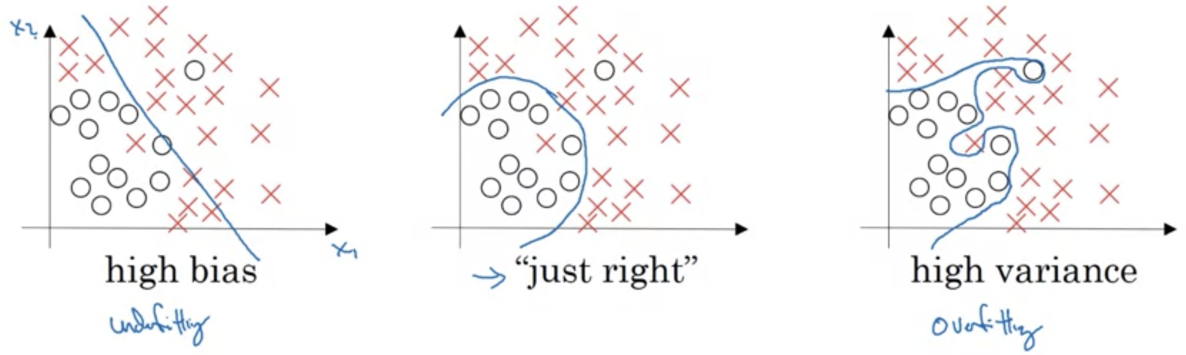


Figure 4.5: Underfitting and overfitting problems (Ng et al., 2018b)

It seems easy to recognise underfitting and overfitting problems in two-dimensional graphs. However, in practice, models use multiple variables and therefore, they are high-dimensional, being difficult to visualise or imagine. Ng et al. (2018b) propose to use training and validation sets to understand underfitting and overfitting problems. If the model exhibits a similar but high MSE on both training and validation sets, then the model is more likely to have an underfitting problem. If the model has a low MSE on the training set, but a much higher MSE on the validation set, then it is more likely to have an overfitting problem as the model cannot generalise well. Finally, if the model performs well and has a similar and low MSE on both training and validation sets, then it is more likely the model that is "just right".

To avoid underfitting and overfitting problems and find the optimal model, this study uses training and validation sets. Model performance will be evaluated on both sets and then analysed. Although the underfitting problem can be easily solved by increasing model complexity, it is not so easy to avoid the overfitting problem with neural networks. It either needs more data or the model should be simplified.

Evaluation metrics

Evaluation metrics differ from each other depending on the problem type. As described above, this study has a regression problem and therefore, only regression evaluation metrics are considered. The widespread regression evaluation metrics are mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE) and R-squared

(R^2) (Ulyanov et al., 2020).

RMSE is equal to the square root of MSE ($\sqrt{MSE}$). Therefore, if MSE increases, RMSE is increased too and vice versa. The MSE calculation is shown in Equation 4.1. MSE and RMSE are very sensitive to large errors and both are absolute measures. It is difficult to interpret RMSE properly, but MSE shows mean squared differences between true and predicted values of the target variable. Decreasing MSE and RMSE means increasing model accuracy.

In contrast to MSE and RMSE, MAE is more robust to outliers because it is the mean of absolute differences instead of squared differences. The calculation is shown in Equation 4.3. If data have outliers to be smoothed in the evaluation process, then it is more viable to use MAE. As with MSE and RMSE, if MAE decreases, model accuracy increases.

However, MSE, RMSE and MAE are all absolute measures. Therefore, their values depend much on the scale of a target variable and it is very difficult to compare two models with differently scaled target variables. Although the lowest value of MSE, RMSE and MAE is zero, and RMSE or MAE with the value 0.5 seem low, this would not be the case if the values of target variables were between 0.4 and 1.0. Therefore, the widely accepted relative measure R^2 is used. It is calculated in Equation 4.11. It can be interpreted as the proportion of the variance in the target variable which can be explained by the features. The equation in the numerator describes the sum of squared errors of the model predictions and its counterpart in the denominator describes the sum of squared errors of the baseline model, where the predictions are always equal to the mean of the target variable. The best possible model can explain all of the variance in the target and therefore, the numerator would be approximately 0, and R^2 would be equal to 1. If the model is not better than the baseline model, R^2 would be 0. However, if the fitted model is even worse than a baseline model, R^2 has negative values. The disadvantage of R^2 is that it always increases with a increasing number of features.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2} \tag{4.11}$$

This study reports RMSE, MAE and R^2 metrics. These metrics are widely reported in machine learning literature and can also be used to compare models in different studies.

Since large errors are particularly undesirable in this study, RMSE metric is used to select the model.

Benchmark

Although it is very easy to calculate MSE, RMSE, MAE and R^2 of a model, it is impossible to know which values of these metrics are achievable with the best possible model using the current data. It is not possible to know in the pre-modelling phase what the best possible model or its performance could be. However, if other researchers have already reported their model performance with the same data, it could be used as a benchmark. Additionally, a simple model such as MLP without a hidden layer can be fitted to the data and the performance can be used as a baseline.

The GIS data of this study have not been published during its preparation, so there is no model performance that has already been reported. Therefore, this study uses a very simple neural network without hidden layers as the baseline model and compares two MLP models with one and two hidden layers with it, respectively. Additionally, the model performance will be compared with the performance of XGBoost (Chen & Guestrin, 2016).

However, there are no labelled data for the 5km fine-scale predictions of COVID-19 cases. To assess the quality of the fine-scale predictions, the aggregated data of Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE) (2020) from January to March 2020 will be used and compared with the aggregated fine-scale predictions of this study at the province level. JHU data are a major reference for national and subnational data on COVID-19 (Python et al., 2020). JHU provides data consistent with the daily data from the Chinese Centre for Disease Control and Prevention, and WHO situation reports (Dong et al., 2020).

Structured hyperparameter tuning

MLP has a range of hyperparameters and they may interact in nonlinear ways. There are sometimes hundreds or thousands of different combinations of hyperparameters. The search process for optimal hyperparameters for the best performing model on the data is called hyperparameter optimisation or hyperparameter tuning (Brownlee, 2020). Search space is the volume which is to be searched. Each dimension of the volume represents a

hyperparameter and each point represents one model configuration. It would be a highly time-consuming task to try each model configuration manually, track their performance and compare them with each other to find the optimal hyperparameters. There are two algorithms to tune hyperparameters automatically, namely grid search and random search (Bengio, 2012).

Grid search defines a search space, as the name suggests, as a grid of hyperparameter values and evaluates each model configuration. Since all model configurations are independent of each other, the hyperparameter tuning process is trivially parallelisable. This makes the process of hyperparameter tuning much faster depending on computation resources. Random search is a slight variation on grid search which only evaluates a random sample of points on the grid instead of searching over the entire grid (Zheng, 2015) which makes it cheaper computationally. It was developed by Bengio (2012) and they demonstrated that the model configurations found with the random search algorithm perform as well as the grid search.

This study will use the random search algorithm for hyperparameter tuning. O'Malley et al. (2019) developed Keras-Tuner, an extension of the Keras library, which combines random search algorithms with Keras MLP models and enables the parallelisation of the hyperparameter tuning process.

5 Results

5.1 Model selection

MLP and XGBoost models have been separately tuned, trained and evaluated. The best models were selected according to the RMSE metric and all these model performance metrics (RMSE, MAE and R^2) are shown in Table 5.1, 5.2 and 5.3. MLP has three different models: without a hidden layer (MLP0), with one hidden layer (MLP1) and with two hidden layers (MLP2). The fourth model is XGBoost (XGBOOST). The models MLP0 and XGBOOST are used as benchmarks for MLP1 and MLP2 models. In Table 5.1, the best model for training, validation and test sets can be identified with bold characters.

Table 5.1: RMSE metric of models for training, validation and test sets

	training	validation	test
MLP0	0.70	0.68	0.66
MLP1	0.63	0.62	**0.63**
MLP2	0.68	0.63	0.66
XGBOOST	**0.46**	**0.58**	0.64

Table 5.2: MAE metric of models for training, validation and test sets

	training	validation	test
MLP0	0.45	0.45	0.45
MLP1	0.39	0.40	0.40
MLP2	0.41	0.42	0.43
XGBOOST	**0.28**	**0.37**	**0.39**

Table 5.3: R^2 metric of models for training, validation and test sets

	training	validation	test
MLP0	0.227	0.111	0.242
MLP1	0.361	0.279	**0.321**
MLP2	0.271	0.239	0.255
XGBOOST	**0.657**	**0.354**	0.288

The model XGBOOST performs the best on the training and validation sets, and the MLP1 on the test set according to the RMSE metric in Table 5.1. The results of the metric R^2 in Table 5.3 are consistent with Table 5.1. Both tables show that on the test set, MLP1 and MLP2 outperform MLP0, but only MLP1 outperforms XGBOOST. However, XGBOOST shows the best performance according to the MAE metric in Table 5.2 on all sets (training, validation and test). The target variable LIR is already scaled using the natural logarithm function and the magnitude of differences is increasingly important for the target variable. Therefore, RMSE is a more important metric for the model selection than MAE.

The best model for fine-scale predictions is selected by the performance on the test set. Therefore, the neural network model with one hidden layer MLP1 is the best model according to RMSE (0.63). According to R^2 (0.320), MLP1 explains approximately 32% of the total variance in the test set. The model architecture is shown in Figure 5.1.

The model has one input layer with seven input units (LONG, LAT, ACCESS, PET, POP, URBAN, and WACCESS), one hidden layer with 64 hidden units and one output layer with one output unit, namely log COVID-19 incidence rate per 100,000 population (Jan-Mar 2020) (LIR). The hidden and output layers consist of the linear sum (z_i) of weights and biases of the inputs from the previous layer, the Batch Normalization layer (b_i) and ReLU activation (a_i). The index i defines the i-th unit in the layer. Additionally, the hidden layer has a dropout layer after ReLU activation, but it is used only for training and therefore, it is not shown in Figure 5.1. The optimal hyperparameters are: 64 hidden units, a 0.8 dropout rate, a 0.005 learning rate, a batch size of 96 and 800 epochs. The weight initialisation is a random process and to replicate the model, the random seed should be set to 1184. All steps needed to reproduce the model are already defined in Jupyter notebook.

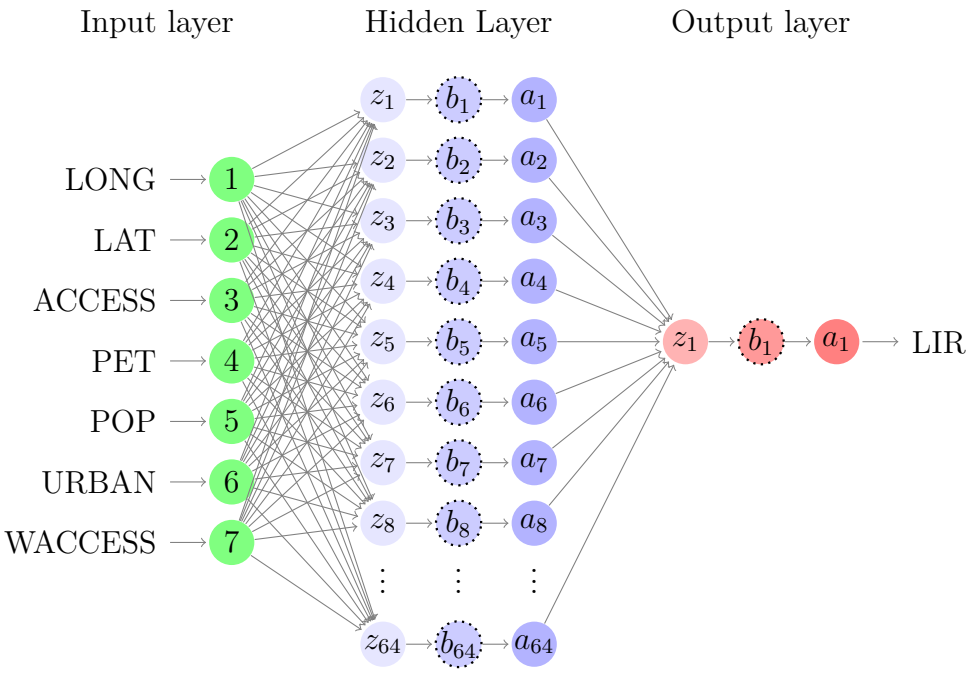


Figure 5.1: MLP architecture. z_i is a linear sum of inputs from the previous layer with weights and biases for the i-th unit. b_i defines the Batch Normalization layer for the i-th unit. a_i calculates ReLU activation of b_i. The model outputs the predicted log COVID-19 incidence rate per 100,000 population (Jan-Mar 2020) (LIR)

5.2 Fine-scale predictions

As the best performing model, MLP1 is used to make fine-scale predictions of the log COVID-19 incidence rate per 100,000 population (Jan-Mar 2020) (LIR) and then transform

the values to the predictions of the COVID-19 incidence rate per 100,000 population (Jan-Mar 2020) (IR) and COVID-19 total confirmed cases (Jan-Mar 2020) (TC). However, the maximum predicted values of IR and TC are significantly higher than the other predicted values because of the Wuhan outbreak. The distribution of predicted LIR is smoother than of predicted IR, which makes the visualisation of values easier. Therefore, the predicted values for log COVID-19 total confirmed cases (Jan-Mar 2020) (LTC) are calculated. For log transformation of TC as IR, the *+1* term is needed, because both predicted variables have zeros as values. The summary statistics for all four variables are shown in Table 5.4. All variables have 548,255 values. The predictions have positively skewed distributions because most of the values are under 1.1 and around 0. This is shown using the 75% quantile in Table 5.4. Additionally, another indicator of a positively skewed distribution is that mean values are higher than median values (50% quantile). Most of the values of IR are higher than TC, because IR is a predicted number of COVID-19 cases per 100,000 individuals, but the population on pixel level has mean, median and maximum values of 2,570 individuals, 235 individuals and 725,643 individuals, respectively.

Table 5.4: Summary statistics for predicted fine-scale values

	IR	LIR	TC	LTC
count	548255	548255	548255	548255
mean	0.87	0.54	0.03	0.03
std	1.12	0.37	0.40	0.08
min	0.00	0.00	0.00	0.00
25%	0.34	0.30	0.00	0.00
50%	0.92	0.65	0.00	0.00
75%	1.05	0.72	0.02	0.02
max	37.73	3.66	116.21	4.76

The fine-scale predicted values of IR are visualised using a continuous raster in Figure 5.2. The white pixels inside China are missing values. The map shows that the highest values are in the area surrounding Wuhan. The lowest predicted values of IR are in Northwest and Southwest China, which is visible in the fine-scale map of predicted LIR in Figure 5.3. This is correlated with population per pixel, because the population in these areas is almost zero. The highest value of IR (37.73) is located in Wuhan, which is shown in Figure 5.2.

The highest predicted value (116.21) of TC is located in Wuhan. The predicted fine-scale values of TC have a positively skewed distribution and therefore, most of the pixels on the map would be dark. Figure 5.4 shows the predicted values for log COVID-19 total confirmed cases (Jan-Mar 2020) (LTC). This enables us to observe other high predicted values of TC in areas outside Wuhan. Higher predicted values of TC and LTC are located close to Wuhan, in Central China and close to Central China.

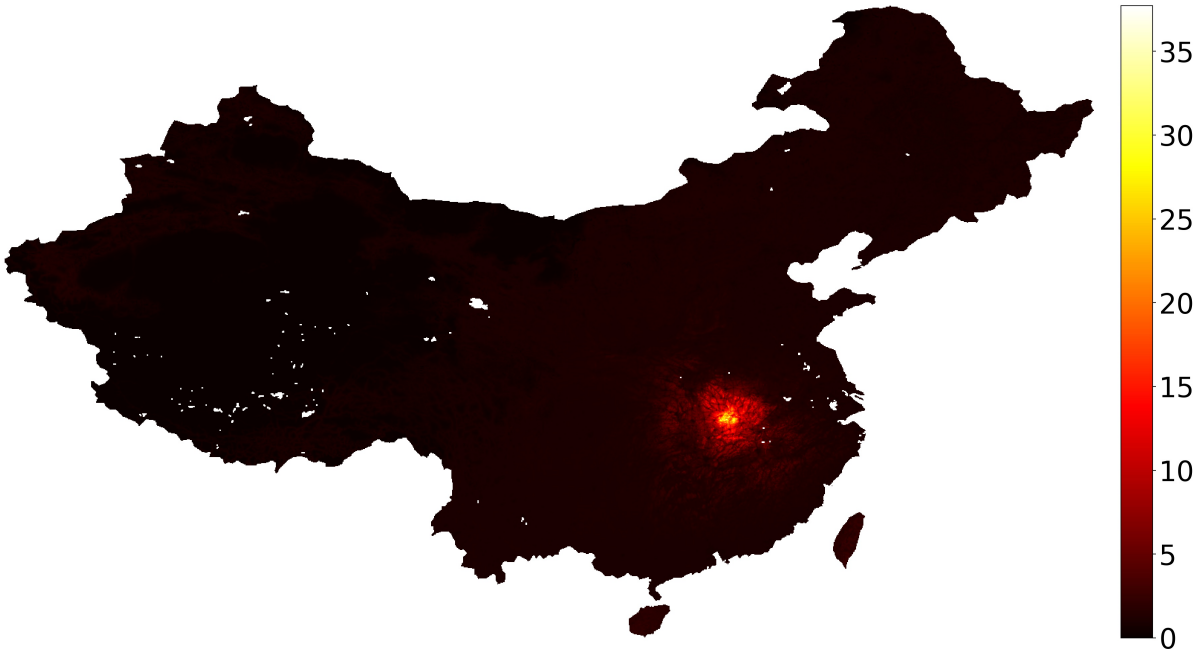


Figure 5.2: Map of fine-scale predicted values of COVID-19 incidence rate per 100,000 population (Jan-Mar 2020) (IR)

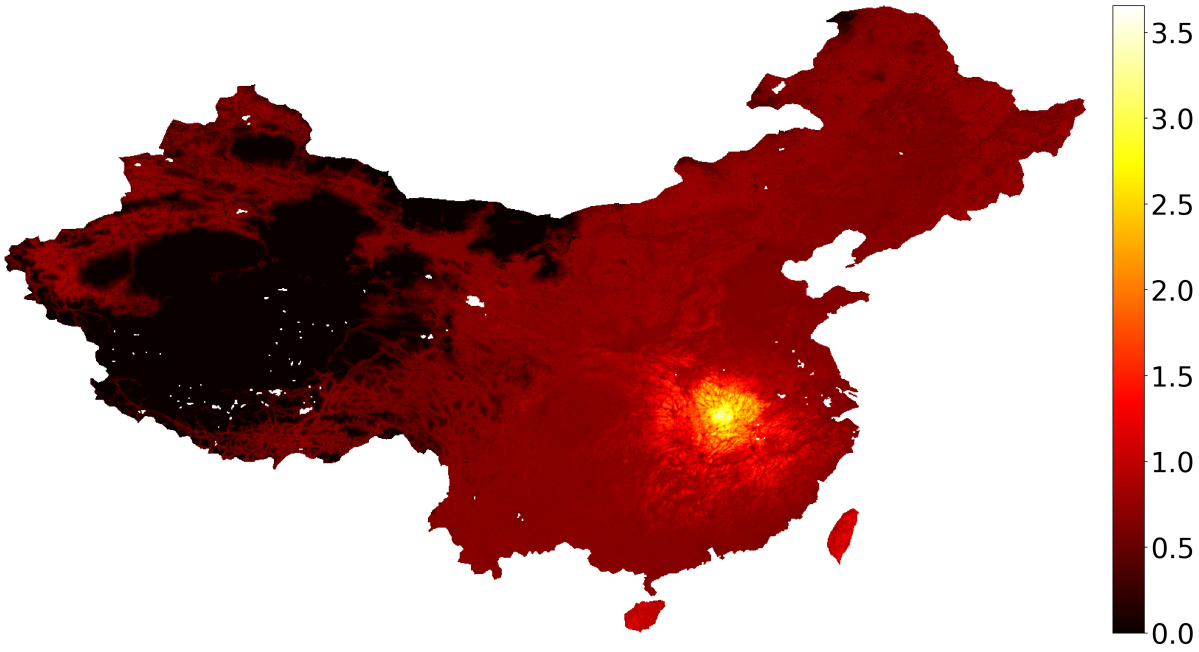


Figure 5.3: Map of fine-scale predicted values of log COVID-19 incidence rate per 100,000 population (Jan-Mar 2020) (LIR)

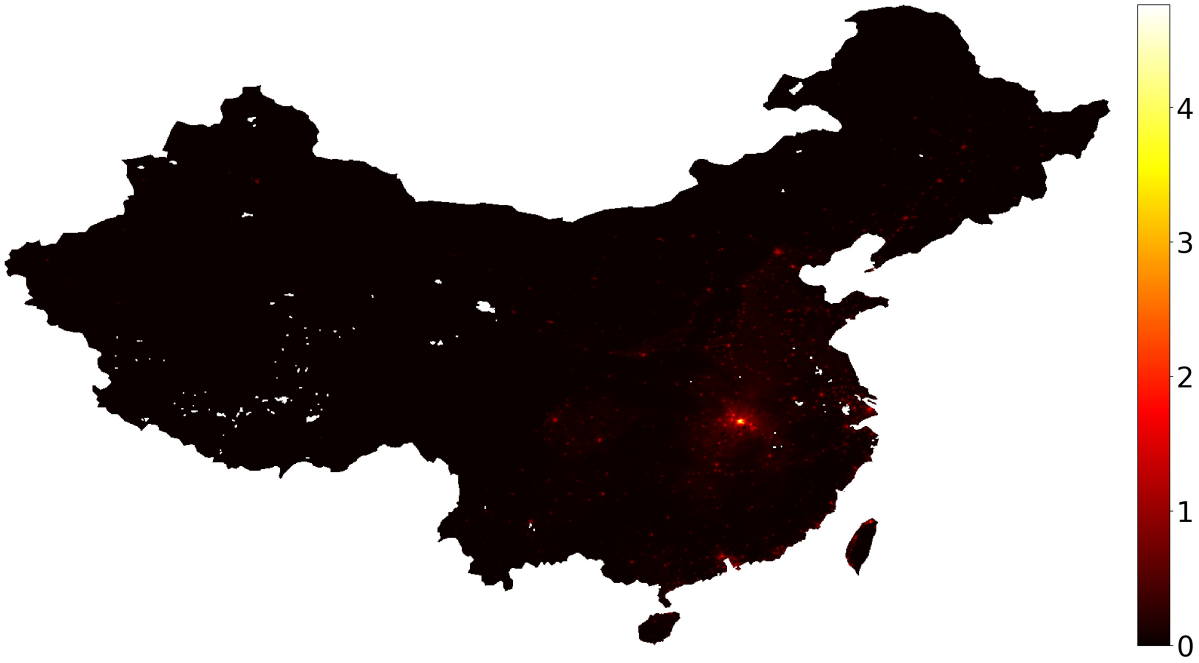


Figure 5.4: Map of fine-scale predicted values of log COVID-19 total confirmed cases (Jan-Mar 2020) (LTC)

5.3 Province-level comparison of aggregated total cases with JHU data

There are no fine-scale true values of TC to compare with the predicted values of TC. This study uses JHU province-level data of total confirmed cases from January to March 2020 to compare them with the province-level aggregated fine-scale predictions of TC. Table A1.1 summarises JHU province-level values and province-level aggregated fine-scale predictions of TC. Additionally, the table shows raw and percentage differences between the predictions and JHU data. The largest difference between both variables is in Hubei province (-61936.05). This means that the model underestimates TC in Hubei in comparison to JHU data. However, the model overestimates aggregated TC in Hunan province (836.00). The model also predicts 363.77 cases in Inner Mongolia, but according to JHU data from January to March 2020, TC was zero. Therefore, the percentage difference between predicted TC and TC reported by JHU in Inner Mongolia is 100% (Table A1.1). On the other hand, predicted TC in Hong Kong is 65.08, but according to JHU data, there were 714 COVID-19 cases between January and March 2020. Therefore, the percentage difference is approximately -1000%. RMSE, MAE and R^2 metrics of the predictions with the reference JHU data at the province level are summarised in Table 5.5. The first row (*all*) of the table shows all metrics for all provinces together and the second row (*excepthubei*) shows all metrics for all provinces except Hubei. RMSE for all provinces (10787.34) is approximately 30 times higher than its counterpart for all provinces except Hubei (354.75). MAE for all provinces (2156.18) is approximately 7 times higher than for all provinces except Hubei (288.06). The reason for larger differences between RMSEs than MAEs is that RMSE penalises larger differences more than MAE. According to R^2, the model explains close to 13% of the total variance in JHU data for all provinces. However, the model explains approximately 23% of the total variance in JHU data for all provinces except Hubei.

Table 5.5: RMSE, MAE and R^2 metrics of JHU data and predictions for all provinces and for all provinces except Hubei

	RMSE	MAE	R^2
all	10 787.34	2156.18	0.128
excepthubei	354.75	288.06	0.235

Figure 5.5 shows the map of differences between the predictions of COVID-19 cases and JHU data at the province level. As the variable *differences* of Table A1.1 shows, differences between the predictions and JHU data ($prediction - JHU$) are between -1,000 and 1,000 except in Hubei province. Therefore, to visualise different values across the provinces smoothly, the minimum and maximum values of the map are constrained to -1,000 and 1000. The map shows that the model significantly underestimates aggregated TC in Hubei province (14) and overestimates it in Hunan province (15). The model overestimates values in most provinces close to Hubei (14), but there are also provinces Chongqing (3) and Zhejiang (33), where the aggregated predicted values are lower than reported by JHU. It is visible that the model overestimates TC in most provinces, even if they are not close to Hubei (14) (e.g. Xinjiang (30) and Inner Mongolia (16)). The differences between predicted values and JHU reported cases are expectedly low in Xizang (31) and Qinghai (22) because the population density is very low in these provinces and therefore, both predicted cases and JHU reported cases are minimal.

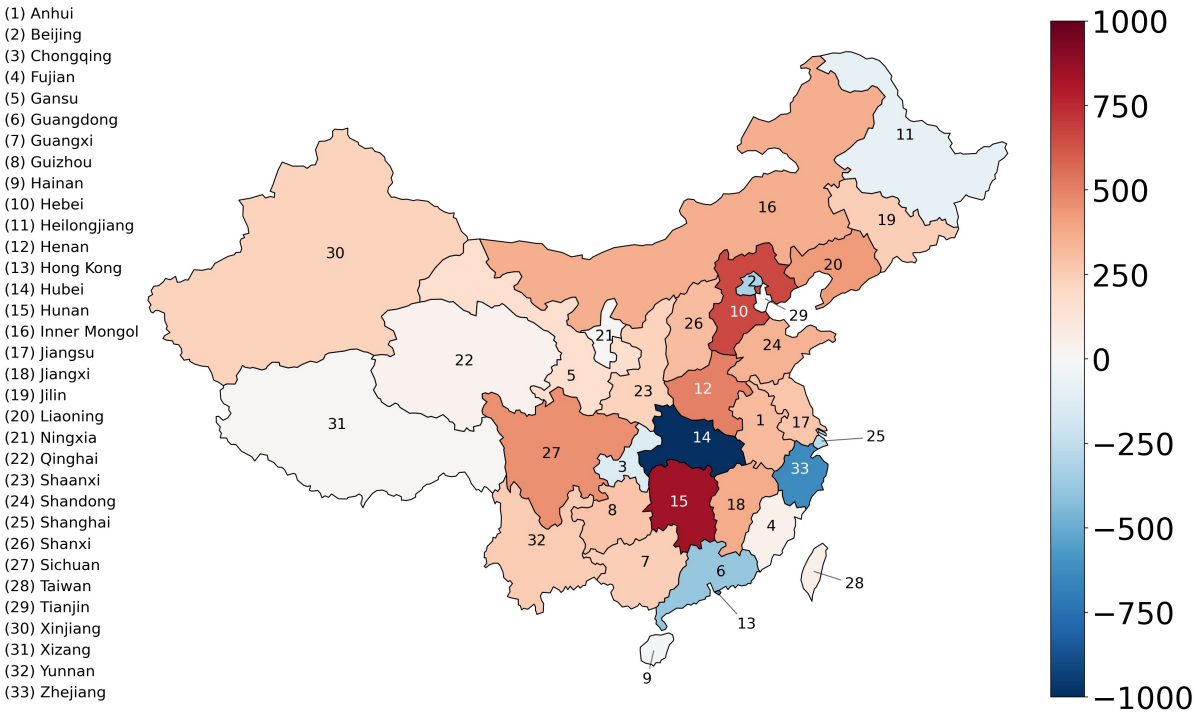


Figure 5.5: Differences between aggregated predictions of total COVID-19 cases and JHU data at the province level. All provinces have values between -1,000 and 1,000 except Hubei. To show the distribution of values across provinces smoothly, the maximum and minimum values are constrained to -1,000 and 1,000

6 Discussion

6.1 Summary of results

This study predicts fine-scale COVID-19 cases in China from January to March 2020 using GIS data and deep learning algorithms. In addition to total confirmed cases, the fine-scale incidence rate of COVID-19 per 100,000 population was predicted in China. The fine-scale total confirmed cases were aggregated to the province level and compared with JHU data, which show similar patterns across provinces with differences in absolute values. To make the predictions, the deep learning algorithm multilayer perceptron (MLP) was tuned, trained, tested and compared with the machine learning algorithm XGBoost. Additionally, this study describes how MLP hyperparameters are tuned step by step and how they are selected, which is a complex process. The results of this study are easily reproducible. The code is publicly accessible on Github, the project development is tracked by Git (Wilson, 2006) and the code can run locally without issues concerning Python dependencies and system requirements by using Conda package management and Docker containers (Merkel, 2014). Therefore, anyone who might be interested in the implementation of deep learning algorithms, working with GIS data and making fine-scale predictions can use the project, make changes and share them with others. Additionally, policymakers and researchers who are interested in COVID-19 cases at high spatial resolutions can use the fine-scale predictions of COVID-19 cases and also aggregate them at district, province or other levels to compare them with other datasets.

6.2 Scientific context and implications

As expected, hyperparameter tuning and the training process of neural networks is time-consuming and computationally intensive (Bengio, 2012). Hyperparameters were tuned and MLP was trained on 8- and 32-core machines. However, the speedup with the 32-core machine was approximately 2x and not 4x, which is in accordance with Bengio (2012). The machine learning model XGBoost is a widely adopted model in data science competitions because participants have very little time to solve the problem and XGBoost provides high performance, computational optimisations and has less hyperparameters to tune (Chen

& Guestrin, 2016). In this study, XGBoost expectedly took only 5-10 seconds to train the model and the MLP models took approximately 10-15 minutes to train for 800-1,000 epochs (Windeatt, 2008). Additionally, finding the optimal hyperparameters of MLP with one hidden layer and MLP with two hidden layers took approximately 48 hours and 72 hours for 1,000 trials with 3 executions per trial (O'Malley et al., 2019). However, MLP with one hidden layer (0.63 RMSE) could generalise better on the test set than XGBoost (0.64 RMSE) according to Table 5.1. Woods & Bowyer (1997) proposes that MLP with one hidden layer and a high enough number of hidden units could learn any complex non-linear function. It was expected before the beginning of the study that MLP with two hidden layers could perform better than MLP with one hidden layer. However, the results of this study in Table 5.1 and 5.3 confirm that MLP with one hidden layer can outperform MLP with two hidden layers. This also shows that if the number of hidden layers increases, it will not necessarily lead to better model performance. However, there are studies which show that MLP with two hidden layers generalises better than MLP with one hidden layer (Thomas et al., 2017), but these differences could depend on the problem type (regression vs. classification), the data structure and the hyperparameter tuning process.

Fine-scale predictions of this study are in accordance with the fine-scale predictions of Python et al. (2020). However, Python et al. (2020) makes predictions from January to February 2020 and this study performs predictions on the period from January to March 2020. Liu (2020) showed in the study that the most important factor in explaining the number of total confirmed cases of COVID-19 in China is the access to Wuhan. Figure 5.3 and 5.4 also show that the higher predicted incidence rates and predicted confirmed cases of COVID-19 are closer to Wuhan. Generally, the model predicts more cases in the east part of China, where population density is higher and cities are larger, thus decreasing the time needed to access cities with a population exceeding 50,000. Qian et al. (2020) show that almost all outbreaks of COVID-19 occurred in indoor environments and Kraemer et al. (2020) calculated that approximately 89% ($R^2 = 0.890$) of the COVID-19 outbreak could be explained by human movement from Wuhan to other locations. It would be expected that people in cities or urban areas have a higher risk to be infected on average as they come in contact with more people in stores and in public transport, etc. In rural areas, people tend to make less use of commercial outlets and do not come in contact

with as many people. It was expected that ecological factors like humidity (potential evapotranspiration) does not have a significant impact on the COVID-19 outbreak (Carlson et al., 2020). However, it was hypothesised that the MLP model with its complex pattern recognition capability could find a relationship between the variables. Although it is not possible to explain direct relationships between variables in the MLP model, the fine-scale distribution of evapotranspiration in China exhibits its highest and lowest values in the western part of China. However, in both types of areas, the number of total confirmed cases and the incidence rate of COVID-19 are predicted to be low.

The predictions of fine-scale confirmed cases of COVID-19 were aggregated at the province level and compared with JHU data provided by Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE) (2020). The differences are mapped in Figure 5.5. There are provinces where the model overestimates or underestimates the number of total confirmed cases aggregated at the province level. The largest difference is in Hubei province (Table A1.1), where Wuhan — the origin of COVID-19 — is located with the highest number of confirmed cases in China. As Liu (2020) identified, the distance to Wuhan is a significant feature for the prediction of COVID-19 cases. However, neighbouring regions are often characterised by spatial autocorrelation (Getis, 2008) and therefore, their values are likely to be similar. Since the beginning of the COVID-19 outbreak in Wuhan, the observed confirmed cases in JHU data for Hubei are at least 50 times higher than in other provinces. It is expected that the model will underestimate the total confirmed cases for COVID-19 in Hubei. Moreover, the significantly high value in Hubei will increase the mean values in neighbouring provinces which did not have many COVID-19 cases. Although the provinces are characterised by spatial autocorrelation, it is not enough to reverse the effect of the high value in Hubei province.

It is expected that that provinces or grid-cells with higher control and social distancing measures will have a lower number of confirmed cases (Kraemer et al., 2020; Prem et al., 2020). It is important to know whether control measures were implemented before or after the outbreak in the location. In Wuhan, they were implemented after the outbreak and there were already many confirmed cases of COVID-19. Therefore, it would be very helpful for the model to know when the outbreak in a specific location has begun. COVID-19 cases grow exponentially in time (Verelst et al., 2020). Hence, having had more time

until March 2020, it is expected that locations where the outbreak began before control measures were introduced will have a higher number of total confirmed cases. Additional features in the model could be control measures, their implementation dates and the number of days from the outbreak to March 2020. It is expected that a model with these additional features could predict the high number of TC in Wuhan (Hubei province) more precisely. However, these data are not available at a fine-scale level and would be very costly and time consuming to collect. There is a trade-off between model accuracy and costs for data collection.

6.3 Limitations and future research

Although the data of Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE) (2020) are used as province-level reference data, the model is trained for fine-scale predictions with the data of Xu et al. (2020) which provides city-level GIS data of COVID-19 in China. Python et al. (2020) show that the data of JHU and Xu et al. (2020) have discrepancies in the number of total confirmed cases of COVID-19 at the province level. The best example is the Inner Mongolia province where JHU data have zero total confirmed cases of COVID-19 and Xu et al. (2020) have more than 300 from January to February 2020, according to Python et al. (2020). Therefore, the trained model predicts more than 300 total confirmed cases in Inner Mongolia from January to March 2020, while JHU data have zero total confirmed cases. It is difficult to verify if JHU data have the true numbers of total confirmed cases of COVID-19 at the province level in China. It would be expected that there are at least some confirmed cases of COVID-19 in Inner Mongolia because the province has a population exceeding 20M. However, provinces with high discrepancies in total confirmed cases of COVID-19 can be explored in further research. Furthermore, correction indices for the provinces can be calculated, facilitating more realistic comparisons of the results with JHU data.

Neural networks' performance improves when the size of the dataset increases (Hestness et al., 2017). This study utilises a dataset with 812 observations, which is considered as small. The model MLP with one hidden layer (MLP1) has an RMSE of 0.63 on the test set, which is the lowest RMSE in comparison to other models. Nevertheless, Figure 6.1 shows that MLP1 mostly predicts values close to 1 and the error increases when

true values increase. It is difficult for deep learning algorithms to recognise complex dependencies between variables with a small dataset. However, it was not possible to have more data points in this study. More data can be collected in further research, increasing the accuracy of the model. Another limitation of MLP or neural networks in general is their interpretability (Preece, 2018). It is difficult in neural networks to understand which feature has the highest impact, how features affect model performance, and which relationship features and a target variable have.

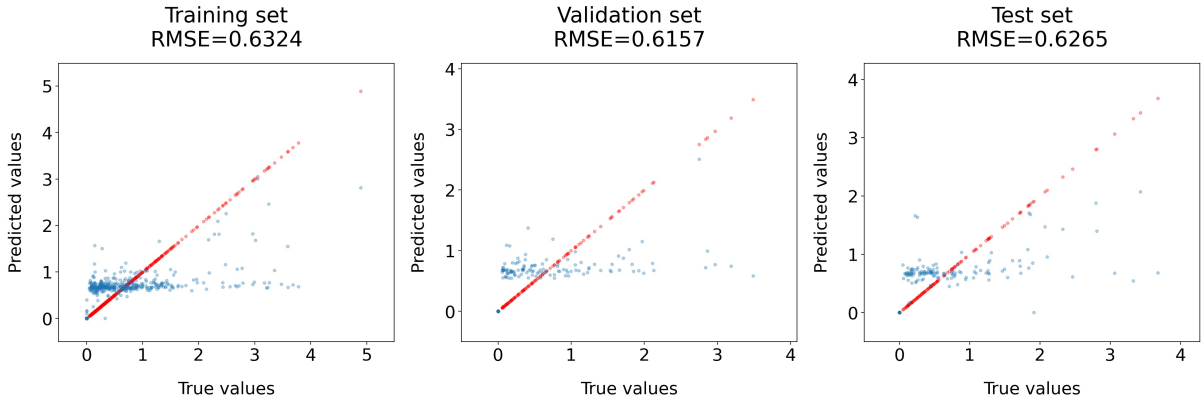


Figure 6.1: RMSE metrics with true/true (red dots) vs true/predictions (blue dots) plot

This study randomly split the data in one training, one validation and one test set, which is a widely accepted practice in the deep learning field (Bengio, 2012). However, this could induce spatial proximity for data points from different sets. Therefore, it would be easier for a model to predict data points in a test set which are spatially close to data points of training and validation sets. Hence, the test set performance is overoptimistic (Schratz et al., 2018) and it is not true out-of-sample performance in the geographical sense. There is a cross-validation method called spatial leave-one-out (SLOO) cross-validation designed to address the spatial autocorrelation problem and overoptimistic results (Le Rest et al., 2014). However, to tune hyperparameters in machine learning and deep learning, it does not suffice to use SLOO, as nested spatial cross-validation is necessary (Schratz et al., 2018; Jin, 2018). This process is highly time-consuming and computationally intensive because many models should be trained, validated and tested. China has 33 provinces and each province should be a test set for spatial nested cross-validation. This makes for 33 different training and test set splits. However, to tune model hyperparameters, each training set must be split into training and validation sets, and the validation set should contain a distinct province from the training set for every iteration. Therefore, for

each training and test split, 32 training, validation and test splits are needed. This study used 1,000 trials and 3 executions per trial to tune hyperparameters. This would mean 3,168,000 models ($33 \times 32 \times 1,000 \times 3$) for training, validation and testing (Jin, 2018). This study does not have sufficient computational and temporal resources to run spatial nested cross-validation, but it could be a topic for further research. It would be very interesting to compare current results with the results of more sophisticated out-of-sample predictions.

7 Conclusion

This study provides fine-scale spatial predictions of total confirmed cases and incidence rates of COVID-19 in China. For predictions, GIS data and MLP (deep learning algorithm) are used. Additionally, XGBoost, which is the most popular algorithm in machine learning competitions hosted by the site Kaggle, was trained and compared with MLP. The MLP model with one hidden layer (0.63 RMSE) outperformed MLP without a hidden layer (0.66), with two hidden layers (0.66) and XGBoost (0.64) on the test set. Fine-scale predictions were aggregated and compared with JHU data at the province level. The comparison shows that the model identifies the distribution of TC values in China among provinces, but with differences in absolute values.

This study could be valuable for researchers and policymakers who would like to understand the COVID-19 pattern at a fine-scale level in China. Additionally, predicted cases at a fine spatial scale (5 km resolution) can be aggregated into larger spatial units, such as regular polygons (e.g. pixels, triangles, hexagonal shapes) or irregular polygons (e.g. municipalities, districts, provinces) according to the users' needs. Researchers could use them as supplementary results to other data sources to compare and evaluate their results.

This study and project would be a very good start for researchers, students and generally, all who have an interest in working with GIS data and/or with deep learning and machine learning algorithms. Additionally, this project uses Conda, Git, Github, Docker containers and other techniques to easily and precisely reproduce all results. This workflow drastically decreased the risk of data and code loss, and the time and effort for transitioning across different hardware and operating systems. Additionally, project development is tracked

by Git since its initiation and this facilitates the observation of its evolution for those who might be interested. This study gives an opportunity to readers to understand the advantages of this workflow and easily adopt it in their studies. Additionally, there are few studies which publicly provide their complete code and data, and are intended from the beginning to be reproduced effortlessly in a few minutes.

References

Anderson, R. M., Heesterbeek, H., Klinkenberg, D., & Hollingsworth, T. D. (2020). How will country-based mitigation measures influence the course of the covid-19 epidemic? *The Lancet*, *395*(10228), 931–934. doi: 10.1016/s0140-6736(20)30567-5

Apostolopoulos, I. D., & Mpesiana, T. A. (2020). Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks. *Physical and Engineering Sciences in Medicine*, 1.

Arora, P., Kumar, H., & Panigrahi, B. K. (2020). Prediction and analysis of covid-19 positive cases using deep learning models: A descriptive case study of india. *Chaos, Solitons & Fractals*, *139*, 110017.

Babaian, R. J., Miyashita, H., Von Eschenbach, A. C., Evans, R. B., & Ramirez, E. I. (1991). Early detection program for prostate cancer: results and identification of high-risk patient population. *Urology*, *37*(3), 193–197.

Bai, Y., Yao, L., Wei, T., Tian, F., Jin, D.-Y., Chen, L., & Wang, M. (2020). Presumed asymptomatic carrier transmission of covid-19. *Jama*, *323*(14), 1406. doi: 10.1001/jama.2020.2565

Barbet-Massin, M., Jiguet, F., Albert, C. H., & Thuiller, W. (2012). Selecting pseudo-absences for species distribution models: how, where and how many? *Methods in ecology and evolution*, *3*(2), 327–338.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *Lecture Notes in Computer Science Neural Networks: Tricks of the Trade*, 437–478. doi: 10.1007/978-3-642-35289-8_26

Bishop, C. M. (1995). *Neural networks for pattern recognition* (Vol. 1111). Oxford university press.

Bolstad, B. M., Irizarry, R. A., Åstrand, M., & Speed, T. P. (2003). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, *19*(2), 185–193.

Brown, D., Lusch, D., & Duda, K. (1998). Supervised classification of glaciated landscape types using digital elevation data. *Geomorphology*, *21*, 3–4.

Brownlee, J. (2016). *Xgboost with python: Gradient boosted trees with xgboost and scikit-learn*. Machine Learning Mastery. Retrieved from https://books.google.de/books?id=HgmqDwAAQBAJ

Brownlee, J. (2018). *Better deep learning: Train faster, reduce overfitting, and make better predictions*. Machine Learning Mastery.

Brownlee, J. (2020). *Hyperparameter optimization with random search and grid search*. Retrieved from https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/

Burrough, P. A., McDonnell, R., McDonnell, R. A., & Lloyd, C. D. (2015). *Principles of geographical information systems*. Oxford university press.

Buscombe, D., & Ritchie, A. C. (2018). Landscape classification with deep neural networks. *Geosciences*, *8*(7), 244.

Butt, C., Gill, J., Chun, D., & Babu, B. A. (2020). Deep learning system to screen coronavirus disease 2019 pneumonia. *Applied Intelligence*, 1.

Carlson, C. J., Chipperfield, J. D., Benito, B. M., Telford, R. J., & O'Hara, R. B. (2020). Species distribution models are inappropriate for covid-19. *Nature Ecology & Evolution*. doi: 10.1038/s41559-020-1212-8

Chang, K. (2006). *Introduction to geographic information systems*. McGraw-Hill Higher Education Boston.

Chang, W. J., Chen, L. B., Hsu, C. H., Lin, C. P., & Yang, T. C. (2019). A deep learning-based intelligent medicine recognition system for chronic patients. *IEEE Access*, *7*, 44441–44458.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/2939672.2939785 doi: 10.1145/2939672.2939785

Crowder, S. V., & Wiel, S. A. (2014). Exponentially weighted moving average (ewma) control chart. *Wiley StatsRef: Statistics Reference Online*. doi: 10.1002/9781118445112.stat04041

Dong, E., Du, H., & Gardner, L. (2020). An interactive web-based dashboard to track covid-19 in real time. *The Lancet infectious diseases*, *20*(5), 533–534.

Drummond, S., Joshi, A., & Sudduth, K. A. (1998). Application of neural networks: precision farming. In *1998 ieee international joint conference on neural networks proceedings. ieee world congress on computational intelligence (cat. no. 98ch36227)* (Vol. 1, pp. 211–215).

Esch, T., Heldens, W., Hirner, A., Keil, M., Marconcini, M., Roth, A., ... Strano, E. (2017). Breaking new ground in mapping human settlements from space–the global urban footprint. *ISPRS Journal of Photogrammetry and Remote Sensing*, *134*, 30–42.

ESRI. (1998). *ESRI Shapefile Technical Description.* Available at https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf. Accessed on 06.09.2020.

Fishman, M. B., Barr, D. S., & Loick, W. J. (1991). Using neural nets in market analysis. *Technical Analysis of Stocks and Commodities*, *9*(4), 18–22.

Flaxman, S., Mishra, S., Gandy, A., Unwin, H. J. T., Mellan, T. A., Coupland, H., ... Eaton, J. W. (2020). Estimating the effects of non-pharmaceutical interventions on covid-19 in europe. *Nature*. doi: 10.1038/s41586-020-2405-7

Frank, E., & Hall, M. (2001). A simple approach to ordinal classification. In *European conference on machine learning* (pp. 145–156).

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Fukushima, K., Miyake, S., & Ito, T. (1983). Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE transactions on systems, man, and cybernetics*(5), 826–834.

Garbin, C., Zhu, X., & Marques, O. (2020). Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia Tools and Applications*, *79*(19-20), 12777–12815. doi: 10.1007/s11042-019-08453-9

Getis, A. (2008). A history of the concept of spatial autocorrelation: A geographer's perspective. *Geographical analysis*, *40*(3), 297–309.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. (http://www.deeplearningbook.org)

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the ieee international conference on computer vision* (pp. 1026–1034).

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., ... Zhou, Y. (2017). Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*.

Hinton, G. E. (2012). *Neural networks for machine learning.* Retrieved from http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hsiang, S., Allen, D., Annan-Phan, S., Bell, K., Bolliger, I., Chong, T., ... Wu, T. (2020). The effect of large-scale anti-contagion policies on the covid-19 pandemic. *Nature*, *584*(7820), 262–267. Retrieved from https://doi.org/10.1038/s41586-020-2404-8 doi: 10.1038/s41586-020-2404-8

Inglese, P., McKenzie, J. S., Mroz, A., Kinross, J., Veselkov, K., Holmes, E., ... Glen, R. C. (2017). Deep learning and 3d-desi imaging reveal the hidden metabolic heterogeneity of cancer. *Chemical science*, *8*(5), 3500–3511.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jia, Y., Ge, Y., Chen, Y., Li, S., Heuvelink, G., & Ling, F. (2019). Super-resolution land cover mapping based on the convolutional neural network. *Remote Sensing*, *11*(15), 1815.

Jin, W. (2018). *Nested cross validation explained.* Retrieved from https://weina.me/nested-cross-validation/

Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE). (2020). *Coronavirus data at province level via github.* Available at https://github.com/CSSEGISandData/COVID-19/blob/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_global.csv. Accessed on 05.09.2020.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *CoRR*, *abs/1412.6980*.

Kraemer, M. U. G., Yang, C.-H., Gutierrez, B., Wu, C.-H., Klein, B., Pigott, D. M., ... Hanage, W. P. (2020). The effect of human mobility and control measures on the covid-19 epidemic in china. *Science*, *368*(6490), 493–497. doi: 10.1126/science.abb4218

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Kucharski, A. J., Russell, T. W., Diamond, C., Liu, Y., Edmunds, J., Funk, S., ... Munday, J. D. (2020). Early dynamics of transmission and control of covid-19: a mathematical modelling study. *The Lancet Infectious Diseases*, *20*(5), 553–558. doi: 10.1016/s1473-3099(20)30144-4

Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *J. Mach. Learn. Res.*, *10*, 1–40.

Le Rest, K., Pinaud, D., Monestiez, P., Chadoeuf, J., & Bretagnolle, V. (2014). Spatial leave-one-out cross-validation for variable selection in the presence of spatial autocorrelation. *Global ecology and biogeography*, *23*(7), 811–820.

LEXICO. (2020). *Overfitting: Definition of overfitting by oxford dictionary on lexico.com also meaning of overfitting.* Lexico Dictionaries. Retrieved from https://www.lexico.com/definition/overfitting

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical*

image analysis, *42*, 60–88.

Liu, J., Zhou, J., Yao, J., Zhang, X., Li, L., Xu, X., ... Niu, T. (2020). Impact of meteorological factors on the covid-19 transmission: A multi-city study in china. *Science of The Total Environment*, *726*, 138513. doi: 10.1016/j.scitotenv.2020.138513

Liu, L. (2020). Emerging study on the transmission of the novel coronavirus (covid-19) from urban perspective: Evidence from china. *Cities*, *103*, 102759. doi: 10.1016/j.cities.2020.102759

Ma, X., Yu, H., Wang, Y., & Wang, Y. (2015). Large-scale transportation network congestion evolution prediction using deep learning theory. *PloS one*, *10*(3), e0119044.

Maier, B. F., & Brockmann, D. (2020). Effective containment explains subexponential growth in recent confirmed covid-19 cases in china. *Science*, *368*(6492), 742–746.

Marr, B. (2020). *These 25 technology trends will define the next decade.* Forbes Magazine. Retrieved from https://www.forbes.com/sites/bernardmarr/2020/04/20/these-25-technology-trends-will-define-the-next-decade/

Merkel, D. (2014). Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, *2014*(239).

Metcalf, C. J. E., Bjørnstad, O. N., Grenfell, B. T., & Andreasen, V. (2009). Seasonality and comparative dynamics of six childhood infections in pre-vaccination copenhagen. *Proceedings of the Royal Society B: Biological Sciences*, *276*(1676), 4111–4118. doi: 10.1098/rspb.2009.1058

Mollalo, A., Mao, L., Rashidi, P., & Glass, G. E. (2019). A gis-based artificial neural network model for spatial distribution of tuberculosis across the continental united states. *International Journal of Environmental Research and Public Health*, *16*(1), 157. doi: 10.3390/ijerph16010157

Mubin, N. A., Nadarajoo, E., Shafri, H. Z. M., & Hamedianfar, A. (2019). Young and mature oil palm tree detection and counting using convolutional neural network deep learning method. *International Journal of Remote Sensing*, *40*(19), 7500–7515.

Narin, A., Kaya, C., & Pamuk, Z. (2020). Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. *arXiv preprint arXiv:2003.10849*.

Ng, A., Katanforoosh, K., & Mourri, Y. B. (2018a). *Activation functions.* Available at https://www.coursera.org/lecture/neural-networks-deep-learning/activation-functions-4dDC1. Accessed on 12.09.2020.

Ng, A., Katanforoosh, K., & Mourri, Y. B. (2018b). *Bias / Variance.* Available at https://www.coursera.org/learn/deep-neural-network/lecture/ZhclI/bias-variance. Accessed on 12.09.2020.

Ng, A., Katanforoosh, K., & Mourri, Y. B. (2018c). *Optimization algorithms.* Available at https://www.coursera.org/learn/deep-neural-network/home/week/2. Accessed on 12.09.2020.

Ng, A., Katanforoosh, K., & Mourri, Y. B. (2018d). *Train/dev/test distributions.* Available at https://www.coursera.org/learn/machine-learning-projects/lecture/78P8f/train-dev-test-distributions. Accessed on 12.09.2020.

Ng, A., Katanforoosh, K., & Mourri, Y. B. (2018e). *Vanishing / Exploding gradients.* Available at https://www.coursera.org/lecture/deep-neural-network/vanishing-exploding-gradients-C9iQO. Accessed on 12.09.2020.

Nielsen, M. A. (2015). *Neural networks and deep learning* (Vol. 2018). Determination press San Francisco, CA.

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., & etc. (2019). *Keras Tuner.* https://github.com/keras-team/keras-tuner.

Ozturk, T., Talo, M., Yildirim, E. A., Baloglu, U. B., Yildirim, O., & Acharya, U. R. (2020). Automated detection of covid-19 cases using deep neural networks with x-ray images. *Computers in Biology and Medicine*, 103792.

Paules, C. I., Marston, H. D., & Fauci, A. S. (2020). Coronavirus infections—more than just the common cold. *Jama*, *323*(8), 707. doi: 10.1001/jama.2020.0757

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pengpai News agency. (2020). *The Paper & Sixth Tone data.* Available at https://www.thepaper.cn/. Accessed on 05.09.2020.

Phaisangittisagul, E. (2016). An analysis of the regularization between l2 and dropout in single hidden layer neural network. *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS).* doi: 10.1109/isms.2016.14

Pirouz, B., Haghshenas, S. S., Pirouz, B., Haghshenas, S. S., & Piro, P. (2020). Development of an assessment method for investigating the impact of climate and urban parameters in confirmed cases of covid-19: A new challenge in sustainable development. *International Journal of Environmental Research and Public Health*, *17*(8), 2801. doi: 10.3390/ijerph17082801

Prechelt, L. (1998). Early stopping-but when? In *Neural networks: Tricks of the trade* (pp. 55–69). Springer.

Preece, A. (2018). Asking 'why'in ai: Explainability of intelligent systems–perspectives and challenges. *Intelligent Systems in Accounting, Finance and Management*, *25*(2), 63–72.

Prem, K., Liu, Y., Russell, T. W., Kucharski, A. J., Eggo, R. M., Davies, N., ... Clifford, S. (2020). The effect of control strategies to reduce social mixing on outcomes of the covid-19 epidemic in wuhan, china: a modelling study. *The Lancet Public Health*, *5*(5). doi: 10.1016/s2468-2667(20)30073-6

Python, A., Bender, A., Blangiardo, M., Illian, J. B., Lin, Y., Liu, B., ... Yin, J. (2020). A downscaling approach to compare covid-19 count data from databases aggregated at different spatial scales. *Available at SSRN 3627252*.

Qian, H., Miao, T., LIU, L., Zheng, X., Luo, D., & Li, Y. (2020). Indoor transmission of sars-cov-2. *medRxiv*. Retrieved from https://www.medrxiv.org/content/early/2020/04/07/2020.04.04.20053058 doi: 10.1101/2020.04.04.20053058

Ramchoun, H., Amine, M., Idrissi, J., Ghanou, Y., & Ettaouil, M. (2016). Multilayer perceptron: Architecture optimization and training. *International Journal of Interactive Multimedia and Artificial Intelligence*, *4*(1), 26. doi: 10.9781/ijimai.2016.415

Ritter, N., & Ruth, M. (1997). The geotiff data interchange standard for raster geographic images. *International Journal of Remote Sensing*, *18*(7), 1637–1647.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. doi: 10.1038/323533a0

Salje, H., Kiem, C. T., Lefrancq, N., Courtejoie, N., Bosetti, P., Paireau, J., ... Dubost, C.-L. (2020). Estimating the burden of sars-cov-2 in france. *Science*. doi: 10.1126/science.abc3517

Saraswat, M. (2019). *Beginners tutorial on xgboost and parameter tuning in r.* Retrieved from https://www.hackerearth.com/blog/developers/beginners-tutorial-on-xgboost-parameter-tuning-r/

Scher, S. (2018). Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning. *Geophysical Research Letters*, *45*(22), 12–616.

Schratz, P., Muenchow, J., Iturritxa, E., Richter, J., & Brenning, A. (2018). Performance evaluation and hyperparameter tuning of statistical and machine-learning models using spatial data. *arXiv preprint arXiv:1803.11266*.

Shafranovich, Y. (2005). *Common Format and MIME Type for Comma-Separated Values (CSV) Files.* Available at https://www.hjp.at/doc/rfc/rfc4180.html. Accessed on 06.09.2020.

Shen, D., Wu, G., & Suk, H.-I. (2017). Deep learning in medical image analysis. *Annual review of biomedical engineering*, *19*, 221–248.

Tatem, A. J. (2017). Worldpop, open data for spatial demography. *Scientific data*, *4*(1), 1–4.

Thomas, A. J., Petridis, M., Walters, S. D., Gheytassi, S. M., & Morgan, R. E. (2017). Two hidden layers are usually better than one. In *International conference on engineering applications of neural networks* (pp. 279–290).

Tian, H., Liu, Y., Li, Y., Wu, C.-H., Chen, B., Kraemer, M. U. G., ... Dye, C. (2020). An investigation of transmission control measures during the first 50 days of the covid-19 epidemic in china. *Science*, *368*(6491), 638–642. Retrieved from https://science.sciencemag.org/content/368/6491/638 doi: 10.1126/science.abb6105

Tikhonov, A. N. (1943). On the stability of inverse problems. In *Dokl. akad. nauk sssr* (Vol. 39, pp. 195–198).

Ulyanov, D., Guschin, A., Trofimov, M., Altukhov, D., & Michailidis, M. (2020). *Regression metrics review i - metrics optimization.* National Research University Higher School of Economics. Retrieved from https://www.coursera.org/lecture/competitive-data-science/regression-metrics-review-i-UWhYf

Verelst, F., Kuylen, E., & Beutels, P. (2020). Indications for healthcare surge capacity in european countries facing an exponential increase in coronavirus disease (covid-19) cases, march 2020. *Eurosurveillance*, *25*(13), 2000323.

Weiss, D., Nelson, A., Gibson, H., Temperley, W., Peedell, S., Lieber, A., ... Fullman, N. (2018). A global map of travel time to cities to assess inequalities in accessibility in 2015. *Nature*, *553*(7688), 333.

Wesolowski, A., Erbach-Schoenberg, E. Z., Tatem, A. J., Lourenço, C., Viboud, C., Charu, V., ... Buckee, C. O. (2017). Multinational patterns of seasonal asymmetry in human movement influence infectious disease dynamics. *Nature Communications*, *8*(1). doi: 10.1038/s41467-017-02064-4

Wilson, G. (2006). Software carpentry: Getting scientists to write better code by making them more productive. *Computing in Science & Engineering*.

Windeatt, T. (2008). Ensemble mlp classifier design. In *Computational intelligence paradigms* (pp. 133–147). Springer.

Woods, K., & Bowyer, K. W. (1997). Generating roc curves for artificial neural networks. *IEEE Transactions on Medical Imaging*, *16*(3), 329–337. doi: 10.1109/42.585767

XGBoost. (2018). *Xgboost documentation - python api reference.* Retrieved from https://xgboost.readthedocs.io/en/latest/python/python_api.html

Xu, B., Gutierrez, B., Mekaru, S., Sewalk, K., Goodwin, L., Loskill, A., ... Kraemer, M. U. G. (2020). *Epidemiological data from the COVID-19 outbreak, real-time case information.* Available at https://github.com/beoutbreakprepared/nCoV2019. Accessed on 05.09.2020.

Yang, C., Jiang, W., & Guo, Z. (2019). Time series data classification based on dual path cnn-rnn cascade network. *IEEE Access*, *7*, 155304–155312. doi: 10.1109/access.2019.2949287

Yao, Y., Zhang, J., Hong, Y., Liang, H., & He, J. (2018). Mapping fine-scale urban housing prices by fusing remotely sensed imagery and social media data. *Transactions in GIS*, *22*(2), 561–581.

Yuan, Z., Zhou, X., & Yang, T. (2018). Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining* (pp. 984–992).

Yuen, K.-S., Ye, Z. W., Fung, S.-Y., Chan, C.-P., & Jin, D.-Y. (2020). Sars-cov-2 and covid-19: The most important research questions. *Cell & Bioscience*, *10*(1). doi: 10.1186/s13578-020-00404-4

Zhang, Y., Zhang, A., & Wang, J. (2020). Exploring the roles of high-speed train, air and coach services in the spread of covid-19 in china. *Transport Policy*, *94*, 34 - 42. Retrieved from http://www.sciencedirect.com/science/article/pii/S0967070X20304273 doi: https://doi.org/10.1016/j.tranpol.2020.05.012

Zheng, A. (2015). *How to evaluate machine learning models: Hyperparameter tuning.* Retrieved from https://web.archive.org/web/20160701182750/http://blog.dato.com/how-to-evaluate-machine-learning-models-part-4-hyperparameter-tuning

Appendix

A1 JHU data and predicted total cases

Table A1.1: Summary table of province-level aggregated predictions and JHU data from January to March 2020. The table additionally shows absolute and percentage differences between predictions and JHU data with the variables *difference* and *difference (%)*. Rows are sorted by the values of the variable *difference*

admin	province	prediction	JHU	difference	difference (%)
China	Hunan	1854.00	1018	836.00	45.09
China	Hebei	983.93	321	662.93	67.38
China	Henan	1781.74	1276	505.74	28.38
China	Sichuan	1006.41	550	456.41	45.35
China	Liaoning	566.11	139	427.11	75.45
China	Jiangxi	1310.15	937	373.15	28.48
China	Inner Mongol	363.77	0	363.77	100.00
China	Shandong	1121.03	774	347.03	30.96
China	Anhui	1309.56	990	319.56	24.40
China	Shanxi	445.63	136	309.63	69.48
China	Guizhou	432.14	146	286.14	66.21
China	Jiangsu	925.91	646	279.91	30.23
China	Yunnan	439.63	182	257.63	58.60
China	Guangxi	500.98	254	246.98	49.30
China	Jilin	343.04	98	245.04	71.43
China	Shaanxi	485.85	253	232.85	47.93
China	Xinjiang	303.84	76	227.84	74.99
China	Gansu	307.66	138	169.66	55.14
Taiwan	Taiwan	380.61	322	58.61	15.40
China	Fujian	392.35	343	49.35	12.58
China	Qinghai	61.58	18	43.58	70.77
China	Xizang	16.10	1	15.10	93.79
China	Ningxia	84.23	75	9.23	10.96
China	Tianjin	170.26	174	−3.74	−2.20
China	Hainan	142.09	168	−25.91	−18.23
China	Heilongjiang	403.32	484	−80.68	−20.00
China	Chongqing	444.21	579	−134.79	−30.35
China	Shanghai	239.18	509	−269.82	−112.81
China	Beijing	260.20	580	−319.80	−122.90
China	Guangdong	1108.80	1494	−385.20	−34.74
China	Zhejiang	631.10	1257	−625.90	−99.17
Hong Kong	Hong Kong	65.08	714	−648.92	−997.03
China	Hubei	5864.95	67801	−61 936.05	−1056.04

Statement of authorship

"I hereby declare that I wrote this present thesis paper independently, without assistance from external parties, and without use of other resources than those indicated. All information that are taken from other publications or sources in text or in meaning are duly acknowledged in the text."

Göttingen, 30.09.2020 Davit Svanidze