Hybrid Deep Learning with Stacked Dilated Causal Convolutions for Health Forecasting using Multivariate Time-series Data

by

Brandon Mossop

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE FACULTY OF GRADUATE STUDIES
OF LAKEHEAD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE (SPECIALIZATION IN ARTIFICIAL INTELLIGENCE)

Hybrid Deep Learning with Stacked Dilated Causal Convolutions for Health Forecasting using Multivariate Time-series Data

by

Brandon Mossop

Supervisory Committee

Dr. Quazi Abidur Rahman,
Supervisor
(*Department of Computer Science, Trent University, Peterborough, Ontario, Canada*)

Dr. Zubair Fadlullah,
Internal Examiner
(*Department of Computer Science, Lakehead University, Thunder Bay, Ontario, Canada*)

Dr. Farhana Zulkernine,
External Examiner
(*School of Computing, Queen's University, Kingston, Ontario, Canada*)

ABSTRACT

Health forecasting using time-series data facilitates preventive medicine and healthcare interventions by predicting future health events. This thesis introduces a novel hybrid deep-learning architecture for health forecasting that combines the Stacked-dilated-causal Convolutional Neural Network and Bidirectional Long Short-Term Memory (SCNN-BiLSTM). Stacked-dilated-causal Convolutional Neural Networks provide full history-coverage of the input window while maintaining the causal structure such that each output in a temporal sequence depends on all previous elements. Two use-case scenarios were studied to examine the effectiveness of the proposed SCNN-BiLSTM architecture: (1) hospital admission forecasting for mental health patients and (2) infectious disease forecasting.

In hospital admission forecasting, the number of admissions for mental health patients at the Thunder Bay Regional Health Sciences Centre was predicted using multivariate time-series data. In the one-step forecast, the CNN-BiLSTM hybrid model outperformed various statistical and neural network techniques. Consequently, this hybrid model involving a standard CNN was compared with the proposed SCNN-BiLSTM to determine if having full history-coverage improved forecasting performance for long-term forecasting. This experiment revealed that the SCNN-BiLSTM outperformed the standard CNN-BiLSTM hybrid model for multi-step forecasting.

The infectious disease experiment utilized COVID-19 data and Google mobility data in Ontario, Canada, to predict the spread of new daily COVID-19 cases for a long forecasting horizon of 28 days. Various configurations of Convolutional Neural Networks (CNN) with recurrent neural networks (RNN) were tested to determine whether: (1) the full history coverage provided by the SCNN performed better than a standard CNN (2) the multivariate or univariate approach has superior performance; (3) the LSTM, Bidirectional LSTM (BiLSTM) or the gated recurrent unit (GRU) was the optimal RNN for the hybrid model. The experiments revealed that SCNN outperformed standard CNN, the multivariate approach was superior to the univariate even in the presence of an incomplete dataset, and BiLSTM was the optimal RNN. These results all support the effectiveness of the proposed multivariate SCNN-BiLSTM.

Experiments conducted in this thesis using two use cases demonstrate that the proposed SCNN-BiLSTM deep-learning architecture may potentially be utilized to train generalizable hybrid models that are effective in multivariate health forecasting.

ACKNOWLEDGEMENTS

I want to express my deepest gratitude to my supervisor, Dr. Quazi Abidur Rahman, for shaping and directing my research and development throughout my studies as a master's student. Dr. Rahman's constant support and feedback have been invaluable, and without his help, none of my academic work would have been possible. I want to thank him for letting me pursue research topics that I find exciting.

I will forever be grateful for the love and support of my family: my mother, father, and brother. They have always motivated me to do my best, even when life is tough.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Health forecasting is the prediction of future health events. A health event is a physical or psychological condition that affects an individual or a broader population.

As the global population continues to grow, there is an increasing need for adequate health care. Health forecasting is a vital tool in meeting the resource demands of future health care needs. By predicting health factors, health care management can better prepare for a range of likely future outcomes.

Researchers previously developed deep learning models to conduct univariate and multivariate health forecasting. However, providing full history coverage of incomplete multivariate temporal sequences while maintaining the causal structure through a hybrid deep learning architecture has not been addressed.

This dissertation extends previous research on multivariate hybrid neural networks by introducing the combination of the Stacked-dilated-causal Convolutional Neural Network (SCNN) and the Bidirectional Long Short-Term Memory (BiLSTM) to create the SCNN-BiLSTM architecture. The main innovation that the proposed architecture provides is full history coverage of the input values while maintaining the causal structure. The stacked-dilated-causal approach allows for efficient casual coverage of the input window without requiring an infeasible number of convolutional layers.

The two critical health forecasting problems this dissertation considers as use-cases to develop such an architecture are: hospital admissions forecasting and infectious disease forecasting. These pressing problems have been addressed in the literature with statistical, mathematical, and machine learning techniques. However, comparatively little work has been done with hybrid neural network methods that consider multivariate data, provide full history-coverage, and forecast long horizons.

This chapter will briefly introduce forecasting and some historical background. The principles of forecasting models and time series data will follow. The organization of the thesis will close the chapter.

1.1 Background and Brief History of Forecasting

A forecast is any statement about the future. The range of questions one might ask about the future is practically endless. We all must perform forecasts continuously, such as: ‘will it rain tomorrow?’; ‘will the cost of gasoline increase next week?’; ‘is my current job right for my entire career, or are there better-suited positions for my talents and interests?’; and so on. Beyond our personal forecasts, anticipating the future is vital for facilitating decisions and formulating planning, policy, and strategy in many industries, such as finance, education, and healthcare.

A forecast can simply be an opinion or guess about the future since a forecast is not necessarily a true statement. However, forecasts of this kind are unreliable and subject to inherent biases [1]. Making statements about the future is easy, but producing competent forecasts is a much harder task. Therefore, forecasts based on systematic methods are preferred since they consolidate knowledge in progressive research, aid in explaining their own failures, and are reproducible. Only systematic forecasting methods will be considered in this thesis.

The term *forecasting* was coined by Robert FitzRoy in 1859 when he sought to predict future weather [2]. FitzRoy wanted a system that could predict storms at sea to reduce the number of sailors dying. He believed that many lives could be saved by forewarning future storms. His forecasts were not always accurate; nevertheless, many lives were saved at sea due to his storm warnings.

Today, weather forecasting has improved considerably. The forecasts are based on physical theories of meteorology and use vast amounts of data [3]. However, weather forecasting is still often incorrect, showing the inherent difficulty of the problem.

Besides weather forecasting, economic, primarily financial, forecasting has been heavily studied. Research on economic forecasting has led to better statistical methods to predict the future value of economic variables and explain economic relationships. However, economic forecasting, like weather forecasting, continues to miss predicting important events, such as the failure of most economists to predict the 2008 Recession [4].

When the performance of a forecasting model deviates by a greater magnitude than could be reasonably expected based on past results, it is known as a *systematic forecast failure* [5]. When we produce forecasting models for the real world we need to ensure that systematic forecast failure does not occur or the model is useless for decision making.

1.2 Models and Time Series Data

A systematic forecast based on rules is produced by models estimated from past data or time series. The models must include reliable historical time series factors that are relevant to the variable being forecast, otherwise the model is considered *misspecified*. Time series data differs significantly from the independent and identically distributed (iid) data typically used for statistical learning tasks.

1.2.1 Time Series Data

A time series is a set of historical observations ordered by the passing of time. The frequency (hourly, daily, weekly, etc.) of the time series determines how far into the future can be reasonably forecast. For example, it is unreasonable to forecast years in advance when the time series frequency is hourly. The forecasting model is considered univariate or multivariate depending on the number of variables used to train it. A univariate model trains with a single time series, while a multivariate model has two or more variables.

Time series data differs significantly from independent and identically distributed (iid) data. The samples from an iid dataset are assumed to be independent of each other. For example, consider rolling two dice. Each time the two dice are rolled, the sum of both is recorded. This record is an *event* or sample from the probability distribution of all the possible outcomes. In this case, we know that the total number of outcomes is $6 \times 6 = 36$. Each sample from the distribution is random since it does not depend on previous samples. In other words, future outcomes do not depend on past outcomes. This is an example of an iid dataset. This kind of data is not useful when forecasting. Instead, we rely upon time series data where observations or samples are related to past observations, known as auto-correlation. In this way, the temporal structure of the time series data is an important source of information to extract when forecasting.

Another significant way forecasting differs from the dice example is that we tend not to know the complete set of possible future outcomes when forecasting. This makes the task of forecasting future values even more challenging.

Ideally, each input variable of a forecasting model could be manipulated while all other factors are fixed to measure the effect of each input on the output (forecast). However, most time series data is non-experimental (not from a randomized controlled trial), so confounding factors could exist. Therefore, the models must learn from correlated relationships between the included variables and past (lagged) values of each time series variable.

The time series data may be missing values for specific observations, making the dataset incomplete. There are various techniques to impute the missing data. This thesis will consider incomplete data and a forecasting method to infer the values (see Chapter 5).

1.2.2 Patterns in Time Series Data

Auto-correlations in time series produce various temporal patterns. These patterns can be classified as seasonal short-term movements, cyclical short-term movements, and long-term movements or trends [6]. Collectively, the temporal patterns are known as the *signal* of the time series data.

The *trend* refers to the overall movement of the time series during a time interval. The movement can be positive (increasing), negative (decreasing) or stable (constant). The trend can shift in a time series, and special attention must be paid to this possibility when forecasting.

A seasonal pattern is a temporal fluctuation that repeats in a period of less than a year. Seasonality always occurs at a constant time interval. For example, if more ice cream is always sold during the summer, this is a seasonal pattern.

A cyclic pattern is similar to seasonality, except the repeating fluctuation does not occur at a fixed and constant time interval. A classic example is the business cycle (the movement of gross domestic product), which can last several years, but future movements are unknown.

Noise is the last component that causes variations in time series data. Therefore, each time series value y_t at time t is

$$y_t = signal + noise \tag{1.1}$$

where the *noise* is assumed to be described by the Gaussian distribution, because it is composed of many unrelated factors.

A successful forecasting model aims to capture the signal while excluding the random or irregular fluctuations (noise), as the noise component is unpredictable and erratic.

1.2.3 Forecasting Models

A future event can be forecast regardless of the predictability of the event. Forecasting rules are derived by extrapolating from past information. Successful forecasting models require that:

1. There are regularities to be captured from historical data (time series).
2. The regularities are informative of the future.
3. The proposed model captures such regularities, yet excludes distorting non-regularities (noise).

The first two requirements relate to the time series data. The last requirement is about the forecasting model. We must ensure that the model is in the correct form to capture the regularities in the time series. For example, if the time series data is a nonlinear process, the forecasting model must be nonlinear.

In this thesis, statistical and machine learning forecasting models are considered. Statistical models have been developed that are well-established in the forecasting literature. Meanwhile, machine learning approaches are only recently being applied to forecasting problems [7]. These models will be discussed in detail in Chapter 3.

Model misspecification is when a model incorrectly omits predictive features and unknowingly includes irrelevant features. Model misspecification does not necessarily result in wrong forecasts. For example, the Ptolemaic system wrongly assumed that features like a planet's retrograde distance are predictive of the movement of that planet. Yet, the Ptolemaic system could predict a planet's positions with a reasonable degree of accuracy. However, model misspecification complicates calculating the likely magnitudes of forecast errors and usually increases the variance of the resulting forecast errors.

1.3 Thesis Organization

A literature survey is presented in Chapter 2. Specifically, previous work on infectious disease forecasting and hospital admissions forecasting will be discussed from a multivariate hybrid neural network perspective.

In chapter 3, forecasting theory and methods will be explained. This includes classical models from statistics and neural network techniques. Then, two techniques for hyperparameter tuning forecasting models will be discussed: grid search and evolutionary neural architecture search. The evolutionary neural architecture search approach requires background material on the principles of genetic algorithms, which will be covered in detail. Finally, the proposed SCNN-BiLSTM architecture is introduced. The details of each component of the hybrid method will be explained in detail. Special attention will be paid to the Stacked-dilated-causal Convolutional approach that the SCNN-BiLSTM utilizes to achieve full history coverage while maintaining causal structure efficiently.

Chapters 4 and 5 will present the experiments using the SCNN-BiLSTM architecture in two health forecasting use cases to determine if this novel hybrid approach is suitable for health forecasting problems.

The first use case for health forecasting with the SCNN-BiLSTM is presented in Chapter 4. It involves forecasting the weekly number of admissions to adult mental health services at the Thunder Bay Regional Sciences Centre. The first experiment establishes that hybrid models are suitable for one-step forecasting, as the CNN-BiLSTM outperformed statistical and neural network methods. Next, long-range forecasting is examined to determine if multivariate models are better than univariate models and if the stacked-dilated-causal Convolutional (SCNN) approach outperforms a standard CNN in a hybrid architecture. The experiments revealed that the multivariate approach is superior to the univariate and that the SCNN outperformed the CNN. Finally, the evolutionary neural architecture search was again compared with the grid search to find the optimal configuration of the SCNN-BiLSTM.

The second use case (Chapter 5) is infectious disease forecasting. The infectious disease forecast in the experiments is the daily spread of COVID-19 across Ontario. The experimental results establish that the Stacked-dilated-causal Convolutional approach is better than a standard CNN layers in a hybrid architecture, the multivariate approach outperformed univariate models, and the BiLSTM was the best performing RNN compared to the LSTM and GRU (gated recurrent unit). The multivariate dataset in the experiments is unique in infectious disease forecasting, as it has incomplete temporal data that must first be inferred before forecasting is possible. This scenario is more realistic since public data for each variable is often not updated at the same rate. Finally, the evolutionary neural architecture search is compared with the grid search to find the optimal configuration of the SCNN-BiLSTM.

Finally, chapter 6 will review the major contributions of this thesis and discuss some of the future work that will be conducted to advance the research completed here.

Chapter 2

Related Work

This chapter will cover previous research that is related to this thesis. Additionally, the principles and background of health forecasting are discussed.

Health forecasting is a tool that facilities preventive medicine and healthcare interventions by predicting future health events. Furthermore, forecasting future health events allows health service providers to reduce and manage future risks and demands [8].

For health forecasting to be a helpful tool, reliable data collection and robust forecasting models are required. A reliable health forecast is essential for health service delivery since it enhances preventive healthcare and services, alerts management of expected future patient overflows, and reduces the costs of supplies and staff redundancy.

There are four main principles of health forecasting:

1. The measure of uncertainty.
2. The nature of data aggregation.
3. The horizon of health forecasting.
4. The focus.

There are multiple sources of uncertainty in forecasting. First, there is uncertainty in the data. The data may be incomplete or imperfect. Then, there is uncertainty when estimating the parameters of a forecasting model. When considering neural networks, which are the focus of this thesis, the parameters are the weights that connect the network nodes. These weights are estimated with the back-propagation technique. These estimates will always intrinsically contain uncertainty. Finally, the error term (noise) of the forecasting model has uncertainty. Each aspect of uncertainty needs to be considered when producing health forecasts.

The nature of data aggregation refers to population data compared to individual time series data. In general, forecasting population-level data is an easier task as more robust and general data is available. The alternative is forecasting a particular individual's ailment, usually referred to as prognosis.

There is no single technique for health forecasting, and various methods have been studied. Likewise, no forecast horizon has been established to match a health forecasting method. Therefore, each health forecast problem is a unique context that requires experimentation to establish a superior technique for the required forecasting horizon [8].

The models must be able to produce the desired forecasting horizon. The horizon will depend on the difficulty of the problem, the time series frequency, and what is considered useful. For example, a one-step daily forecast will not be useful if long-term planning is required to manage health care resources.

The focus refers to the specific health problem being addressed. The two problems this thesis addresses are infectious disease forecasting and hospital admissions forecasting.

2.1 Infectious Disease Forecasting

An infectious disease is an organism, such as a virus or bacterium, invading another host organism. Many infectious organisms are harmless and even beneficial, such as the bacteria in our digestive system that aids digestion. However, pathogenic infections harm their host and cause disease, spreading from the host to a susceptible individual [9].

Infectious disease forecasting is the prediction of the future spread of an infectious disease in a population. The relation of infectious disease forecasting to public health is direct and substantial [10]. Public health is greatly affected by the spread of infectious diseases throughout the population. Infectious diseases place a heavy burden on health care services and resources.

Accurate infectious disease forecasting aids in preparing for surge capacity and hospital resource management by anticipating staffing needs and resource usage, potentially guiding the allocation and deployment of human resources and treatment inventory. Additionally, infectious disease forecasts can guide community mitigation strategies, such as which public restrictions would reduce the spread of a disease during a pandemic [10].

Infectious disease forecasting has traditionally been approached with mathematical modelling techniques, such as agent-based and compartmental models. Agent-based approaches model the spread of a disease as a collection of individual agents (people) interacting with each other and the environment. These models have been studied to determine the impacts of different interventions on the spread of the disease. The agent-based approaches are challenging to model accurately because how a population will behave is difficult to predict. The modelling also requires massive computational resources.

Compartmental models divide a population into categories (compartments). The compartments reflect the state of the individual based on the disease. In the classic susceptible-infectious-recovered (SIR) model, each individual is placed in one of the three susceptible, infectious, or recovered compartments. A susceptible individual has not been infected and is a potential target for spreading an infectious disease. An infected individual currently has the disease and can spread it to a susceptible individual. A recovered person has previously been infected and is now considered immune from the infectious disease. The SIR model contains differential equations describing each compartment's rate of change. This simple model has been rigorously studied, which is an advantage. It can also be easily modified to more accurately model infectious diseases [11]. However, the model's simplicity is also often a shortcoming, as the model usually leaves out essential assumptions about the spread of an infectious disease.

The COVID-19 pandemic highlighted the importance of accurately predicting the future spread of an infectious disease. It is difficult to overstate the impact of the COVID-19 pandemic over the past two years. This has ignited a burst of research to accurately forecast the spread of COVID-19 cases. A recent literature search in the NIH LitCovid online database revealed 2843 peer-reviewed publications on COVID-19 forecasting [12]. Traditional approaches to infectious disease forecasting, such as compartmental and agent-based models, were again popular in this current COVID-19 pandemic [13]. However, mechanistic modelling requires reliable epidemiological data, which has not been consistently available throughout this pandemic [14]. Additionally, mechanistic models only take advantage of a small amount of available data on a pandemic. For example, the classic SIR model is unable to incorporate mobility data directly. The COVID-19 infection is spread by close contact [15], so this data could increase the accuracy of the forecast of new cases. These limitations have led to pandemic forecasting with data-driven models.

A data-driven model does not rely on assumptions about a virus's biological characteristics. Instead, these models will learn directly from the data. A popular data-driven approach is the artificial neural network. Indeed, neural networks have been widely used for COVID-19 forecasting [16].

Depending on the task, many popular neural network configurations exist. For example, the convolutional neural network has been successfully used for computer vision tasks [17]. Neural network configurations can be combined into hybrid models to leverage the strengths of each design.

In time series forecasting, the effectiveness of hybrid neural network models, combining the CNN with the RNN, has been investigated for various problems, such as gold price prediction [18], web traffic forecasting [19], and forecasting tool wear in manufacturing [20]. The main idea behind the CNN-RNN hybrid model is to use the ability of the CNN to extract informative features from the raw time series and then feed these more complex features to a RNN, which can learn long-term dependencies.

Previous research on the CNN-RNN has been performed in the context of COVID-19 forecasting. In Zain et al. [21], a CNN-LSTM model improved forecasting performance compared to a CNN or LSTM alone for the number of global confirmed COVID-19 cases. This model was univariate (global confirmed COVID-19 cases), with a 7-day forecast horizon. Xu et al. [22] developed a CNN-LSTM hybrid model, CNN model, and LSTM model to predict a 14-step forecast of the cumulative cases for Brazil, India, and Russia. The models were multivariate with features derived from government policies such as face-covering requirements, public gathering sizes, closure of public transportation, and stay-at-home orders. However, this study found that the LSTM alone outperformed the CNN and hybrid CNN-LSTM. Widiputra [23] developed a multivariate CNN-LSTM optimized with genetic algorithms for predicting mobility measurements during the COVID-19 pandemic. The GA-optimized CNN-LSTM model performed better than stand-alone CNN and LSTM models. In another study [24], a multivariate CNN-LSTM performed better than the ARIMA and LSTM models in forecasting the number of new cases in each province of India. In Verma et al. [25], an LSTM, stacked LSTM, ED-LSTM, BiLSTM, CNN, and CNN-LSTM were trained to predict the daily confined cases for 7, 14, and 21 days in India. Every model in this study was univariate. The stacked LSTM and hybrid CNN-LSTM performed better than the other six models.

2.2 Hospital Admissions Forecasting

As the global population continues to grow, there is an increased demand for healthcare services. Hospital admission forecasting is vital to an effective and robust healthcare system. An optimal balance between healthcare resources such as doctors, nurses, beds, medical equipment, etc. and patient admission (demand) is necessary to enhance healthcare processes. Accurate forecasting of patient admission can significantly reduce the associated costs in supplies and staff redundancy.

Hospital admission prediction has been extensively studied in the literature, but mainly from a feature-based approach. In [26], a feature-based approach with logistic regression was employed. Several key features were included that were supported by past studies.

A common feature-based approach is with neural networks. However, the relative superiority of machine learning approaches compared to traditional methods like logistic regression has not been proven.

A study developed a multi-layer artificial neural network to predict a peak in the number of patients with respiratory disease in the emergency department [27]. A similar study focused on predicting the number of admissions [28]. Another study [29] developed three separate multi-layer artificial neural networks for each subgroup of children aged 0 to 4, 5 to 14, and the entire study population. The study aimed to evaluate the possible impact of meteorological factors and air pollution on the number of children admitted for asthma. A significant shortcoming of feature-based approaches is neglecting temporal patterns in the data.

An essential part of any hospital is its inpatient wards, which admit patients to stay in the hospital while under treatment. When no inpatient beds are available to new inpatients, overcrowding in that ward will occur.

Forecasting hospital admissions has been highly studied, but hybrid hospital admission forecasting has not seen the same attention. Only one study follows the hybrid forecasting approach in the first use case of this thesis (see Chapter 4). This work applied a hybrid autoregressive integrated moving average (ARIMA) and the nonlinear autoregressive neural network (NARNN) models to forecast new admission inpatients at a hospital in Chongqing, China [30]. The individual components of the hybrid model were each compared to the hybrid model to determine if the hybrid approach was superior. The study found that only for the monthly time series data did the combination ARIMA-NARNN outperform the separate ARIMA and NARNN models. Indeed, for the daily data, the NARNN model was superior to the ARIMA and the ARIMA-NARNN models.

2.3 Summary

This chapter covered essential health forecasting background material. Health forecasting has four main principles: (1) The measure of uncertainty; (2) The nature of time series data aggregation; (3) the forecasting horizon; (4) The focus. Each of these principles must be considered when conducting a health forecast.

Previous health forecasting research related to this thesis was reviewed. Health forecasting problems from infectious disease and hospital admissions forecasting were considered from a multivariate hybrid neural network perspective. No previous work has proposed methods long-range forecasting using multivariate data with full history coverage that maintains the causal structure in the temporal sequences.

This thesis builds on previous work on hybrid neural network models for health forecasting by offering the following improvements: (1) Stacked-dilated-causal CNN layers to provide full history coverage that maintains the causal structure of the time series ; (2) Incorporation of multivariate data; (3) Automatic inference of incomplete multivariate data; (4) Multi-headed architecture, so a separate CNN and RNN process each time-series; (5) Longer-range forecast horizon.

Chapter 3 will cover basic forecasting theory and forecasting methods. The primary focus of this chapter will be presenting the novel SCNN-BiLSTM architecture, which will be explained in detail.

Chapter 3

Forecasting Theory, Methods, and the Proposed Architecture

To judge the skill of a forecast, we need to compare and rank the performance of various methods. This chapter begins with a discussion on properly evaluating a forecast so we can rank each forecast from best to worst.

Then, the statistical and neural network methods used to test the skill of the proposed hybrid neural network architecture, the SCNN-BiLSTM, are explained.

This is followed by explaining two hyperparameter methods: grid search and evolutionary neural architecture search. Evolutionary neural architecture search is an intelligent search for the optimal configuration of a neural network. These two methods are compared in the use cases to determine which one provides a more optimal design for the SCNN-BiLSTM.

Finally, the proposed health forecasting hybrid architecture is presented. The SCNN-BiLSTM has three main neural network components: stacked-dilated-causal convolution, bidirectional LSTM, and fully connected layers. Each component will be explained in mathematical detail.

3.1 Evaluating Forecasts

A forecast should be judged based on the quality of the decision it facilitates. This often requires considering the differential *cost* of a decision. For example, a forecast that systematically underestimates the departure time of a flight is not useful because the cost of missing a flight is higher than being early for most people.

Often the context-specific decisions that result from a forecast are difficult or impossible to consider completely. In this case, an objective forecasting metric is required. The metric selected for both use-cases was the Mean Average Percentage Error (MAPE) which is

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \times 100 \tag{3.1}$$

where A_t are the actual values, F_t are the forecasts, n is the number of steps in the forecast.

A MAPE score of zero percent indicates that the forecast was perfect. As the MAPE score increases, the performance of the forecast decreases.

With an objective metric, such as the MAPE score, it is possible to compare the forecasts produced by a model. Then the skill of each model can be ranked from worst (highest MAPE score) to best (lowest MAPE score).

A MAPE score is a standard metric to evaluate forecasting performance. However, if any actual values are zero, the MAPE score will not work (division by zero). For both use cases, there are no values that are zero, so this limitation was not an issue.

In time series forecasting, the gold standard to validate the performance of a model is the walk-forward method. This approach entails training the model on data from the beginning of the time series to a certain point T. Then the model forecasts h time steps into the future. The value of h is known as the forecasting horizon. If the forecast consists of the following five steps, it is called a five-step forecasting horizon. Then the model is input with the actual values for the next h steps and forecasts the following h values. This process is repeated for the selected testing time interval, and the MAPE scores of each forecast are combined to determine the complete MAPE score for the walk-forward time range.

The walk-forward method ensures that the model forecasts a wide range of values so that the chance a model was simply lucky forecasting a particular set of values is reduced.

3.2 Statistical Methods

Classical methods for time series forecasting are generally focused on linear relationships between historical time series observations and future values. The classical methods also tend to be univariate, meaning a single time series variable is used to train the models. One of the most popular classical statistical methods is the Autoregressive Integrated Moving Average Method.

3.2.1 Autoregressive Integrated Moving Average Method

The Autoregressive Integrated Moving Average (ARIMA) model forecasts values as a linear function of past observations and the residuals. The residuals are the difference between the actual observations and the forecast.

The ARIMA model consists of three parameters: p, d, q. The parameter p represents the number of time lags included in the model. The parameter d is the degree of differencing, it describes the number of times the time series will be differenced to make it stationary (variance and mean are constant over time). Differencing involves subtracting the previous value from present value for the entire time series. The parameter q is the order of the moving average term. The $ARIMA$ model is a combination of the AR, I, and MA models. The model is a general representation of each of the component models, so if a parameter of the model is assigned zero, then that part of the model is not included. The notation for the ARIMA is: $ARIMA(p, d, q)$.

3.2.2 Seasonal Autoregressive Integrated Moving Average Method

The Seasonal ARIMA or SARIMA is an extension of ARIMA. It is useful for time series that exhibit seasonality. Seasonality is a repeating pattern that occurs over a fixed time interval. SARIMA adds four more parameters to the ARIMA model. These parameters are P for the seasonal autoregressive order, D for the seasonal difference order, Q for the seasonal moving average order, and m for the number of time steps for a single seasonal period. An m of 12 for monthly data suggest a yearly seasonal cycle. The notation for a SARIMA is expressed as: $SARIMA(p, d, q)(P, D, Q)_m$.

3.3 Neural Network Methods

Neural networks simulate a simplified mathematical model of biological neurons, with neurons or nodes connected to each other by weights that control the amount of information that passes between the nodes.

The two essential characteristics of how neural networks learn from data are the incremental, layer-by-layer way increasingly complex representations are developed and the fact that these intermediate incremental representations are learned jointly.

The time series data requires a specific technique to prepare for neural networks, known as the sliding window method.

3.3.1 Sliding Window Method

A time series dataset must first be transformed to a supervised learning problem prior to training any of the neural network forecasting methods. A supervised learning method requires that the data be a set of samples that has an input, X, and output, y. A supervised learning method fits a function that maps the input to the output during training that approximates the actual mapping. For univariate time series with a one-step forecast, the input is prior time steps, and the output is the observation at the next time step. The use of prior time steps to predict the next time step is called the sliding window method. The order between the observations is preserved when creating the windowed dataset, which is essential when learning the temporal structure of the time series. The number of previous steps is called the window width or size of the lag. The sliding window method allows any time series problem to transformed into a supervised learning problem.

3.3.2 Multi-layer Perceptron

The multi-layer perceptron (MLP) or fully connected neural network is a network of nodes organized in layers. The nodes in each layer are connected to every node from the previous layer (fully connected). The nodes compute the weighted sum of each weight that is connected to it. The weighted sum is then subjected to an activation function. The nonlinear activation function allows the model to learn nonlinear relationships from the data. In time series forecasting the time series from real-world data is often nonlinear, which allows deep learning methods to capture patterns in time series data. The result of the activation function is an input to the next layer of nodes. The final output of the neural network is the forecast as a continuous value.

3.3.3 Convolutional Neural Network

The Convolutional Neural Network (CNN) was originally designed for image data. It operates directly on the pixels of the raw image data to extract relevant features. This eliminates the need for manual feature extraction.

The CNN is able to automatically extract features by using the convolution operation, which consists of a filter moving around the image and performing the element-wise multiplication of the filter values with the pixel values, then summing the result. The filters are typically connected to more layers of fully-connected nodes, like the architecture of the fully connected neural network.

In the context of time series forecasting, a sequence of observations can be considered a one-dimensional *image*. The CNN can learn the most important parts of the sequence that aid in better forecasting results.

3.3.4 Long Short-Term Memory Network

The Long Short-Term Memory Network (LSTM) is a type of Recurrent Neural Network (RNN). RNNs allow for the temporal structure of the time series to be explicitly modelled. This is achieved by retaining results of past inputs by feeding the past outputs into the next time step.

The LSTM has a series of gates that allow it to remember long-term dependencies in a time series. The cells of the LSTM are updated with the input gate. The information from the previous hidden state is combined with the input at the current time step at the input gate. LSTMs also have a forget gate. The forget gate decides what information to ignore and what information from the time series is important and should be remembered. The output gate decides what the next hidden state should be for the next cell of the LSTM. The current input and previous hidden state are combined with the sigmoid function. The output from the sigmoid function is then subjected to the *tanh* activation function. These operations allow the LSTM to decide what information the hidden state should retain.

3.3.5 Bidirectional Long Short-Term Memory Network

The Bidirectional LSTM or BiLSTM consists of two layers of LSTM nodes. The first layer will consider the time series forward in time, while the second layer of the LSTM will consider the observations in reverse (backward) in time. In this way, the LSTM layers attempt to capture time-dependent regularities forward and backward in time. This approach will capture more time dependencies in the time series than a regular (forward) LSTM layer of nodes.

3.4 Hyperparameter Tuning

Statistical and neural network methods require hyperparameter tuning, which is determining the optimal values of parameters that are not learned from data. Two techniques were used for hyperparameter tuning: grid search and evolutionary neural architecture search.

3.4.1 Grid Search

A grid search uses the permutations of each hyperparameter selected and tests the performance of the resulting configuration. Grid search is not an intelligent search, it simply tests each configuration from a list of all possible configurations.

3.4.2 Evolutionary Neural Architecture Search

Determining the optimal hyperparameters of a model is a challenging task. The search space is massive, and the time required to search even a small part is considerable. The optimal hyperparameters of a model are often determined by domain expertise gained by trial-and-error experimentation. To combat this, techniques from evolutionary computation have been used for automatic and efficient hyperparameter search. When considering neural networks, one such technique is evolutionary neural architecture search.

Evolutionary neural architecture search utilizes techniques from genetic algorithms to automatically determine the best hyperparameters for a neural network [31] [32].

The techniques of genetic algorithms consist of the genetic operators of selection, crossover, and mutation, which are applied to a population of individuals [33]. Each individual consists of a chromosome representing an array of values; each value is a *gene*. Each individual is a candidate solution to a specific task. In the case of this thesis, the tasks are to forecast the number of daily COVID-19 cases and to forecast the weekly number of hospital admissions.

The selection operator determines which individuals from the current generation will be parents to the next generation of individuals. There are various selection methods, but each one aims to select the *best* individuals. The best individuals are determined based on the *fitness* of the individual. The fitness value is computed based on a specific task. The best individuals in this thesis produce the *lowest* mean average percentage error score when forecasting.

The crossover (or recombination) operator creates children (or offspring) from the selected individuals. This is typically performed by using two individuals as the parents and swapping parts of their chromosomes to create two new chromosomes representing the children.

The mutation operator will randomly alter the chromosome of a child to promote diversity in the population. The chance of mutation is usually very rare.

In the following subsections, the specific methods chosen for selection, crossover, and mutation in the evolutionary neural architecture search for each use-case experiment are discussed in detail. But first, the way a chromosome is represented will be explained.

Chromosome Representation

There are two ways to represent the chromosome of an individual: integer-coded or real-coded. An integer-coded chromosome consists of *genes* that are whole numbers. However, an integer-based individual requires that each gene has the same range of values. In evolutionary neural architecture search, each gene of the chromosome represents a part of the specific configuration of the hybrid neural network. Unless each gene has the same range of values, the integer-based solution will not be feasible.

In response to this, there is the real-coded chromosome, which consists of real-valued genes. The real-values allow each gene to represent its own range of values. This approach is used for the architecture search. The representation of a real-coded chromosome for the optimal architecture search of the proposed hybrid model (described next section) is shown in Table 3.1.

The chromosome has 9 genes (see Gene Index column in Table 3.1). Each gene represents a part of the specific architecture of the hybrid neural network. Since the genes are real numbers, the values need to be converted to a whole number prior to becoming hyperparameters to the hybrid neural network. The range of values of each gene is restricted to a lower and upper bound. These limits guarantee the values of the gene always stay within a bounded range.

Beginning at gene index 0, this part of the chromosome represents the window

Table 3.1: Bounded Limits for each Gene of an Individual

Gene Index	**Hyperparameter**	**Lower Bound**	**Upper Bound**
0	Window Length	5	20
1	Horizon Length	1	12
2	Number of RNN Nodes	5	100
3	Maximum Pooling Size	1	2
4	Fully Connected Layer 1	5	200
5	Fully Connected Layer 2	-5	100
6	Fully Connected Layer 3	-5	100
7	Fully Connected Layer 4	-5	100
8	Number of Convolutional Filters	5	60

length. The window length is the number of lagged time steps for the input. The next gene is the horizon length, which is the number of time steps to be forecast during training. The number of RNN nodes is represented by gene 2. Gene 3 is the maximum pool size. The maximum pooling occurs after the convolutional layer(s). It can have a value of either 1 or 2. The following four genes are for the number of nodes in the fully connected layers. A value of less than 1 means that layer and subsequent layers are not included. For example, if genes 4, 5, 6, 7 were 23, 40, -3, and 22, this would mean that the hybrid neural network has two fully connected layers with 23 and 40 nodes, respectively. The positive lower bound of gene 4 ensures that there will be at least one fully connected layer. Finally, the number of convolutional filters is represented by gene 8. The chromosomes have a total of $16 \times 12 \times 96 \times 2 \times 196 \times 106 \times 106 \times 106 \times 56 = 4.819 \times 10^{14}$ possible configurations, or approximately 500 thousand billion. This means that if it took 1 second to train and test a configuration, it would take over 15 million years for each possible configuration.

The use of a real-coded chromosome restricts the methods available for crossover and mutation. The specific architecture that each chromosome represents is used to produce a forecast. The forecast is what is evaluated as the fitness of each chromosome.

Fitness Value

In order to select the best individuals at each generation, there must be a method to evaluate the individuals. This is captured with the fitness value. As mentioned previously, the MAPE score will be the metric to evaluate the forecasting results.

Therefore, the MAPE score will represent the fitness value.

Selection

The selection operator determines which individuals from the current generation will be the parents of the next generation. In contrast to the crossover and mutation operators, the selection method does not depend on the chromosome representation. The tournament selection method was chosen for this thesis in combination with elitism. [34].

Tournament selection

Tournament selection requires a tournament size value, n. When selecting the parents of the next generation, a subgroup of n individuals are randomly selected from the current generation. These n individuals are compared to each other, and the individual with the best fitness is selected as a parent.

The larger the tournament size, n, the greater the probability that the best individuals will be part of the tournament. When the best individuals are part of more tournaments the most fit individuals of the current population will become parents more often, on average. A larger tournament size will also tend to reduce the variability or diversity of the population.

Tournament selection does not require a fitness value to determine which individuals are selected as parents. The only requirement is a method to compare any two individuals to determine which is better. In this thesis the fitness value is the MAPE score from forecasting with the architecture that each individual represents. A lower MAPE score is considered a better fitness value.

The tournament selection method was combined with *elitism* to select the population at the next generation [35].

Elitism

Generally, the average fitness of the population increases with each generation. However, the best individuals may be lost in the next generation. This is the result of applying the genetic operators on each individual. In many cases the loss is only temporary, and the best individuals will be reintroduced in later generations.

To avoid the potential time wasted on rediscovering good individuals, the best n individuals from each generation can be automatically selected to be part of the next generation, without applying crossover or mutation. This concept is known as *elitism*. Elitism was combined with tournament selection to select each new generation.

Crossover

The crossover methods applicable to integer-based chromosomes are not suitable for a real-coded chromosomes. Therefore, a specialized crossover method, unique to real-coded individuals is required. For this purpose, the simulated binary crossover (SBX) method was chosen [36].

The SBX method imitates the properties of the single-point crossover technique that is common with integer-coded chromosomes. One of these proprieties is that the children's chromosomes are the average of their parents. The SBX is expressed as

$$c_{1,k} = \frac{1}{2}[(1-\beta_k)p_{1,k} + (1+\beta_k)p_{2,k}] \tag{3.2}$$

$$c_{2,k} = \frac{1}{2}[(1+\beta_k)p_{1,k} + (1-\beta_k)p_{2,k}] \tag{3.3}$$

where $c_{i,k}$ is the i^{th} child with the k^{th} dimension, $p_{i,k}$ is the selected parent and β_k is a random number referred to as the $spread\ factor$.

This formula has several notable properties. As mentioned previously, the average of the two children is equal to their parents, regardless of the value of β. This is how SBX imitates single-point crossover. When the value of β is 1, the children are effectively clones of the parents. If the value of β is less than 1, the children's chromosomes will be closer to each other than the parents were. Conversely, when the value of β is greater than 1, the children will be farther apart from each other than the parents were.

In the SBX method, the similarity between children and parents is preserved with the random distribution of the β value. The probability of β is higher for values around 1, where the children are similar to the parents. To achieve this, the value of β is computed with u, which is a random number sampled uniformly from $[0,1]$. Once the value of u is selected β is computed as

$$\beta(u) = \begin{cases} (2u)^{\frac{1}{\eta+1}} & \text{if } u \leq 0.5 \\ \frac{1}{[2(1-u)]^{\frac{1}{\eta+1}}} & \text{otherwise} \end{cases} \tag{3.4}$$

where η is the crowding factor, which is set as a constant representing the distribution index. Children will be more similar to the parents with larger values of η.

Mutation

Similar to the crossover operator, the mutation operator must be suitable for a real-coded genetic algorithm. Therefore, the polynomial bounded mutation operator was chosen [36]. The polynomial bounded mutation operator determines the probability distribution with a polynomial function, rather than a Gaussian function. This operator requires arguments for the upper and lower boundaries of the search space. The polynomial bounded mutation operator is expressed as

$$c_k = p_k + (p_k^u - p_k^l)\delta_k \tag{3.5}$$

where c_k is the child, p_k is the parent, k is the dimension being considered, p_k^u is the upper bound on the parent, p_k^l is the lower bound, and δ_k is the variation which is

$$\delta_k = \begin{cases} (2r_k)^{\frac{1}{n_m+1}} - 1 & \text{if } r_k < 0.5 \\ 1 - [2(1-r_k)]^{\frac{1}{n_m+1}} & \text{otherwise} \end{cases} \tag{3.6}$$

where r_k is a random number sampled from a uniform distribution between $[0, 1]$ and n_m is the mutation distribution index.

3.5 Proposed Hybrid SCNN-BiLSTM Architecture for Health Forecasting

The proposed hybrid neural network is composed of three main neural network architectures: a stacked-dilated-causal convolutional neural network, a bidirectional LSTM, and a fully connected neural network. Each component will be discussed in detail.

As discussed in Chapter 2, there has been previous research on hybrid models in health forecasting. This thesis extends previous research on the CNN-RNN hybrid model for health forecasting in the following ways:

1. Stacked-dilated-causal CNN to provide full history coverage that maintains the causal structure of a time series.

2. Incorporates multivariate data.

3. Automatically infers incomplete multivariate data.
4. Multi-headed architecture, so a separate CNN and RNN process each time-series.
5. Longer-range forecast horizon.

The proposed Stacked-dilated-causal CNN-BiLSTM or SCNN-BiLSTM architecture incorporates all of these enhancements to accurately forecast health data, which will be displayed in the two use cases (see Chapter 4 and 5) .

Next, Each component of the SCNN-BiLSTM will be explained in mathematical detail.

3.5.1 Convolutional Neural Networks for Time Series

Convolutional neural networks are usually used in computer vision. This involves applying a two-dimensional convolutional filter over a two dimensional image. In forecasting, a one-dimensional convolutional filter is applied to time series.

One-Dimensional Convolution

The convolution operation is defined as the dot product between a kernel matrix $\mathbf{k} \in \mathbb{R}^{h \times d}$ and the input vector $\mathbf{x}$. The kernel $\mathbf{k}$ has length h and width d. In this study, the width d of $\mathbf{k}$ is 1, since each input is one-dimensional. In the case of non-causal convolution the input vector is $\mathbf{x}_{i:i+h+(s-1)}$, which is the concatenation of h continuous time steps, initiated from the i^{th} step, with stride s. Formally, $\mathbf{x}$ can be expressed as

$$\mathbf{x}_{i:i+h+(s-1)} = \mathbf{x}_i \oplus \mathbf{x}_{i+s} \oplus ... \oplus \mathbf{x}_{i+h+(s-1)} \tag{3.7}$$

where $\oplus$ is the direct sum operator. The dot product between $\mathbf{k}$ and $\mathbf{x}$ is

$$y(w) = \begin{cases} \sum_{i=1}^{h} \mathbf{k}_i \mathbf{x}_{w+i} & \text{if } w\text{=1} \\ \sum_{i=1}^{h} \mathbf{k}_i \mathbf{x}_{w+i+(s-1)} & \text{otherwise} \end{cases} \tag{3.8}$$

where w is the width of the input (the number of lagged values) and y is the output of the dot product. If the output y is dependent on future inputs $\mathbf{x}$, then such systems are called non-causal. For example, if w is 5, h is 3, and s is 1 then,

$$y(1) = k(1)x(1) + k(2)x(2) + k(3)x(3)$$
$$y(2) = k(1)x(2) + k(2)x(3) + k(3)x(4)$$
$$y(3) = k(1)x(3) + k(2)x(4) + k(3)x(5)$$

Notice that the output y(1) is dependent on future inputs x(2) and x(3). The output length is not equal to the length of the input. This corresponds to *valid* padding in the deep learning library *Keras* [37], which was used in this study. To make the output length equal with the input there is *same* or *causal* padding.

In causal convolution, the output is not dependent on future inputs. The input vector becomes

$$\mathbf{x}_{i:i-h+(s-1)} = \mathbf{x}_i \oplus \mathbf{x}_{i-s} \oplus ... \oplus \mathbf{x}_{i-h+(s-1)} \tag{3.9}$$

The causal dot product of input $\mathbf{x}$ and kernel $\mathbf{k}$ is now computed as

$$y(w) = \begin{cases} \sum_{i=1}^{h} \mathbf{k}_i \mathbf{x}_{w-i} & \text{if } w = h-1 \\ \sum_{i=1}^{h} \mathbf{k}_i \mathbf{x}_{w-i+(s-1)} & \text{otherwise} \end{cases} \tag{3.10}$$

If w is set to 5, h is 3, and s is 1 (as in the previous example) then,

$$y(3) = k(1)x(3) + k(2)x(2) + k(3)x(1)$$
$$y(4) = k(1)x(4) + k(2)x(3) + k(3)x(2)$$
$$y(5) = k(1)x(5) + k(2)x(4) + k(3)x(3)$$

In this example the output y is independent of future inputs of x, and hence represents causal convolution.

In addition to the dot product between $\mathbf{k}$ and $\mathbf{x}$, a bias b is added. The result $y(w_i) + b$ is input to the Rectified Linear Unit (ReLU) which is

$$a_i = max(0, y(w_i) + b) \tag{3.11}$$

where a_i is the activation value for the feature map. It is worth noting that a_i is a scalar value, representing a single activation value in a feature map. The complete feature map is then

$$\mathbf{a}_j = [a_1, a_2, ..., a_{w-h+1}] \tag{3.12}$$

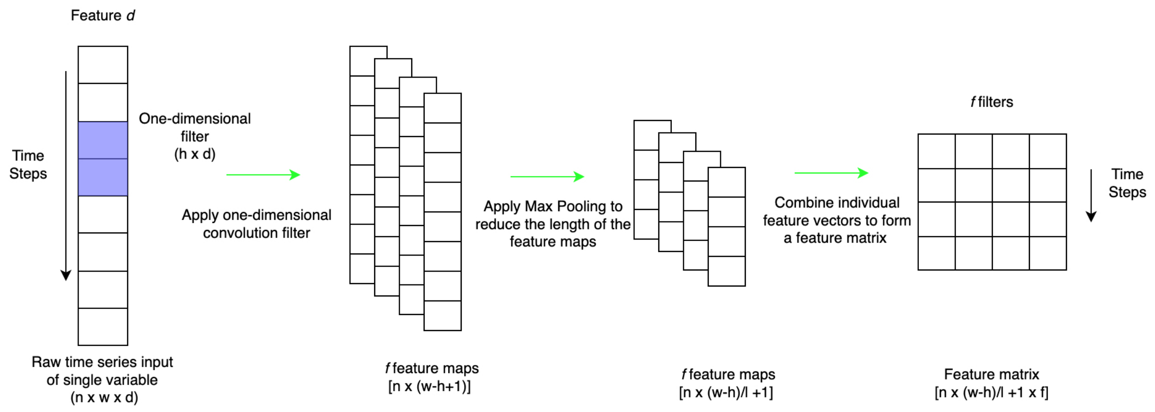
Feature d
Time Steps
One-dimensional filter (h x d)
Apply one-dimensional convolution filter
Raw time series input of single variable (n x w x d)
f feature maps [n x (w-h+1)]
Apply Max Pooling to reduce the length of the feature maps
f feature maps [n x (w-h)/l +1]
Combine individual feature vectors to form a feature matrix
f filters
Time Steps
Feature matrix [n x (w-h)/l +1 x f]

Figure 3.1: A one-dimensional convolution applied to a univariate raw time series input where n is the number of input samples, w is the number of lagged time steps in the input window, d is the dimension of the input (1 since it is univariate), h is width of the convolutional filter, f is the number of filters, l is the size of max pooling.

where the j^{th} index denotes the j^{th} convolutional filter. and w is the width of the input window of lagged values of $\mathbf{x}$. When the padding is set to *same* or *causal* $\mathbf{a}_j$ becomes

$$\mathbf{a}_j = [a_1, a_2, ..., a_w] \tag{3.13}$$

since the padding will equal $h-1$ to ensure that the output from the convolution and activation operations match the length of the input.

Max Pooling

The pooling layer reduces the length of the feature map. The max pool operation is applied $\lfloor \frac{w}{l} \rfloor$ times to each feature map $\mathbf{a}_j$ taking the maximum value of l continuous elements. The compressed feature will become

$$\mathbf{p} = [p_1, p_2, ..., p_{\frac{w-h}{l}+1}] \tag{3.14}$$

There are typically multiple filters f applied to the raw inputs, each with different initial weights to derive the output of the CNN layer.

The time series input shape to the CNN layer is $n \times w \times d$, where n is the number of samples. The size of the outputs will be $n \times (\frac{w-h}{l} + 1) \times f$. In this way, after convolutional and pooling operations, the raw input time dimension w, can be compressed from w to $(\frac{w-h}{l} + 1)$. The raw input feature dimension d, which is 1, is increased to the number of filters, f. Therefore, the CNN acts as a feature extractor

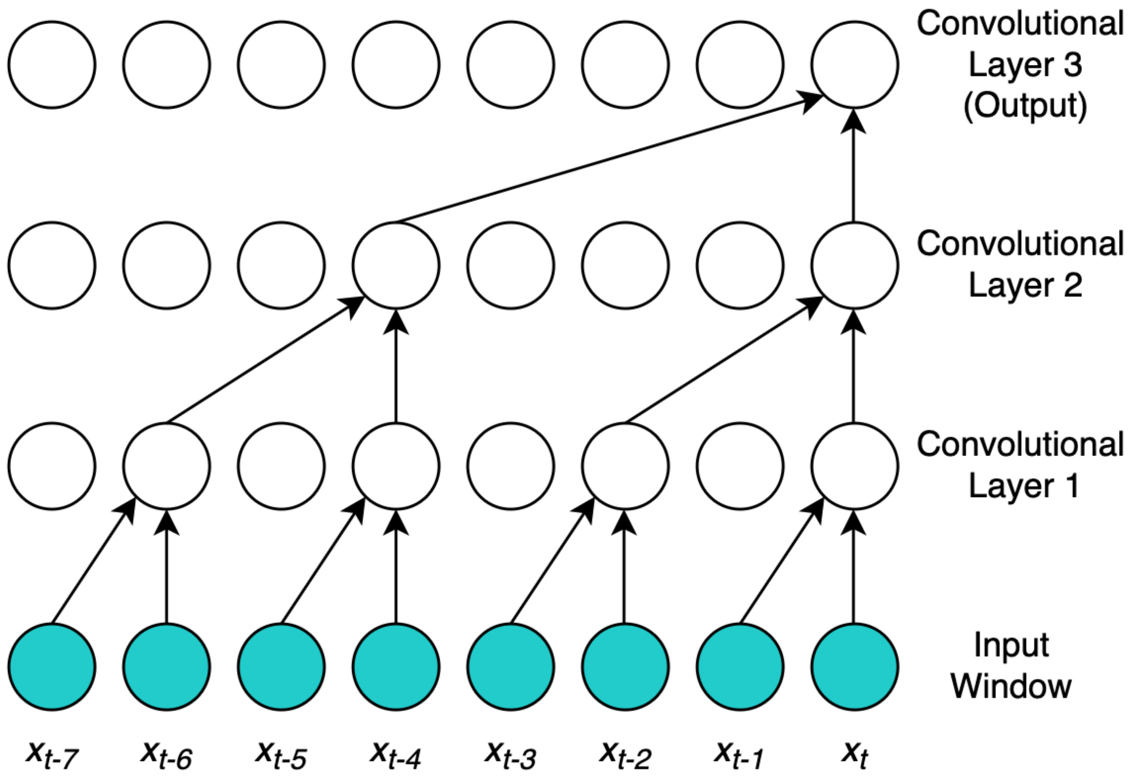

Figure 3.2: Stacked-dilated-causal convolutions for full history coverage while maintaining the causal structure of the time-series input. Note that convolutional layer 1 has a kernel with $2^0 = 1$ dilation, layer 2 has a $2^1 = 2$ dilated kernel, and layer 3 has a kernel that is dilated by $2^2 = 4$.

that feeds a more informative sequential representation to any subsequent layers, compared to the initial raw time series input. The complete transformation of the raw time series to the richer representation of each feature matrix is shown in Figure 3.1.

3.5.2 Stacked-dilated-causal Convolutions

There is *full history coverage* when every output from a convolution operation depends on all previous elements in the input window. This often requires stacking multiple convolutional layers. The total number of convolutional layers required for full history coverage is

$$\lceil \frac{(w-1)}{h-1} \rceil \tag{3.15}$$

so that a convolution with an input window, w, equal to 8, and a kernel length, h, set to 2 would result in seven convolutional layers for full history coverage.

To reduce the number of layers, a convolutional kernel can be *dilated*, which refers to the distance between the elements of the input that are convoluted with the kernel [38]. To increase the dilation at each layer, if c represents the number of convolutional layers before the layer being considered, then with a *dilation base*, b, the dilation of the kernel will be b^c at that layer. Then the number of convolutional layers required for full history coverage will be

$$\lceil \log_b \frac{(w-1)(b-1)}{h-1} + 1 \rceil \tag{3.16}$$

which would reduce the total number of layers for full history coverage from seven without a dilated kernel to three (see Figure 3.2). Evidently, without dilating the convolutional kernel, the ability to train the model would quickly become impractical as the input window increased in size.

3.5.3 Recurrent Neural Networks for Time Series

There are various forms of recurrent neural networks (RNNs). All RNNs use previous inputs rather than only the present input. The RNN that had the best forecasting performance was the bidirectional long short-term memory network, which is an extension of the long short-term memory network.

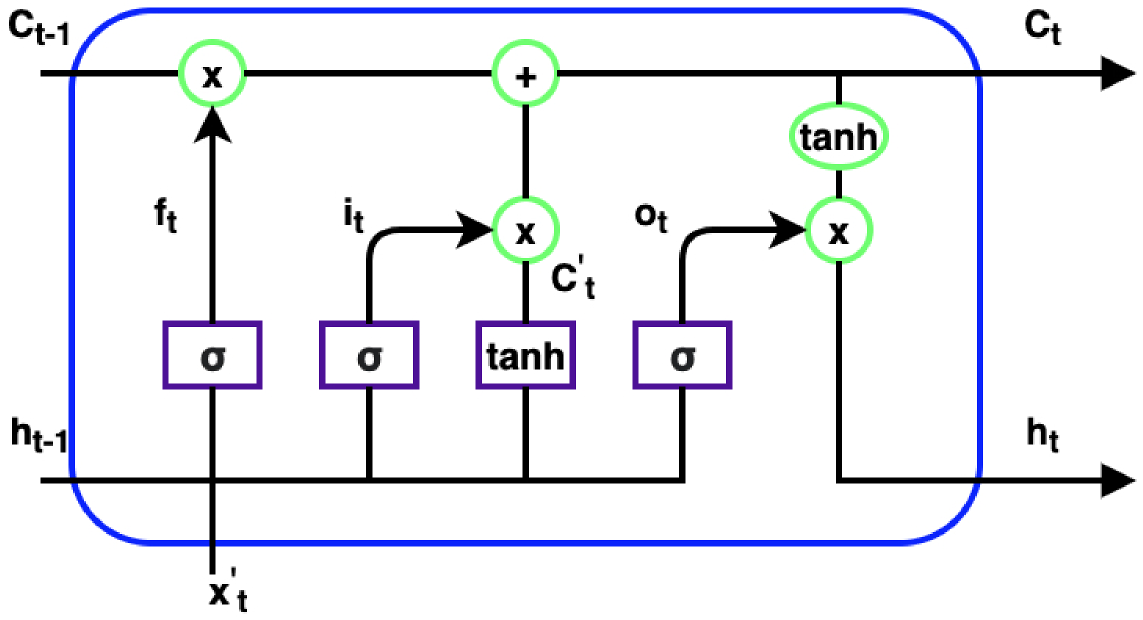

Figure 3.3: A basic LSTM for time series forecasting

The main idea behind long short-term memory networks (LSTMs) is that at each time step t gates are used to control the passage of information. The hidden state $\mathbf{h}_t$ at time t is updated by the current time step input $\mathbf{x}'_t$, the previous hidden state $\mathbf{h}_{t-1}$, the input gate $\mathbf{i}_t$, the forget gate $\mathbf{f}_t$, the output gate $\mathbf{o}_t$ and the memory cell state $\mathbf{C}_t$. The LSTM is shown in Figure3.3. The equations that describe a LSTM are

$$\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1} \oplus \mathbf{x}'_t] + \mathbf{b}_f) \tag{3.17}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1} \oplus \mathbf{x}'_t] + \mathbf{b}_i) \tag{3.18}$$

$$\mathbf{C}'_t = \tanh(\mathbf{W}_c[\mathbf{h}_{t-1} \oplus \mathbf{x}'_t] + \mathbf{b}_c) \tag{3.19}$$

$$\mathbf{C}_t = \mathbf{f}_t \otimes \mathbf{C}_{t-1} + \mathbf{i}_t \otimes \mathbf{C}'_t \tag{3.20}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1} \oplus \mathbf{x}'_t] + \mathbf{b}_o) \tag{3.21}$$

$$\mathbf{h}_t = \mathbf{o}_t \otimes \tanh(\mathbf{C}_t) \tag{3.22}$$

where $\mathbf{W} \in \mathbb{R}^{m\times(m+f)}$, $\mathbf{h_{t-1}} \in \mathbb{R}^{m\times 1}$, $\mathbf{x'_t} \in \mathbb{R}^{f\times 1}$, σ is the sigmoid activation function, tanh is the hyperbolic tangent activation function, $\otimes$ is the element-wise product, m is the number of nodes in the hidden layer, $\mathbf{b} \in \mathbb{R}^m$ is the bias vector. The LSTM processes the time series in temporal order. The output of the LSTM at the terminal time step is $\mathbf{h}_t$.

A single LSTM hidden layer will only process previous time steps forward in time (from past to future). However, Bi-directional LSTMs are able to process a time series in both the forward and backward temporal directions. Then the complete bi-directional LSTM output $\mathbf{h}_t$ is the concatenation of the forward and backward processes.

3.5.4 Complete Hybrid SCNN-BiLSTM Architecture

The proposed hybrid SCNN-BiLSTM combines Stacked-dilated-causal convolutions, a bidirectional LSTM, and fully connected layers. The stacked-dilated-causal convolutions guarantee that every output from the convolution operation depends on all previous elements in the time series input window, ensuring full history coverage while maintaining the causal structure.

The hidden state units from the LSTM are concatenated for each input variable. There are one or two more fully connected layers in succession connected to the layer of concatenated values from the LSTMs. The fully connected layers are defined by

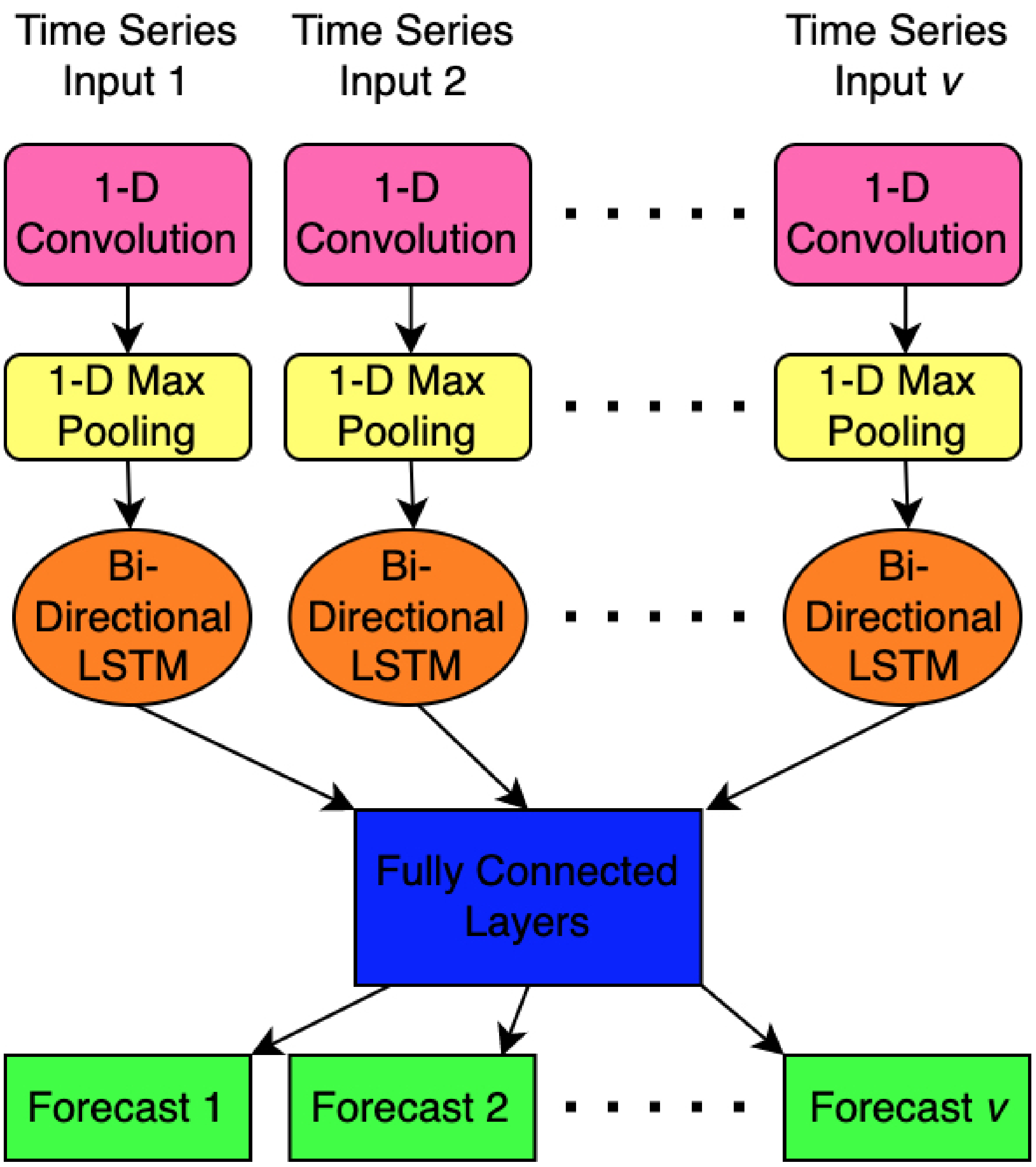

Figure 3.4: A high-level depiction of the proposed SCNN-BiLSTM. Note that the CNN component will involve multiple stacked layers of dilated causal convolutions.

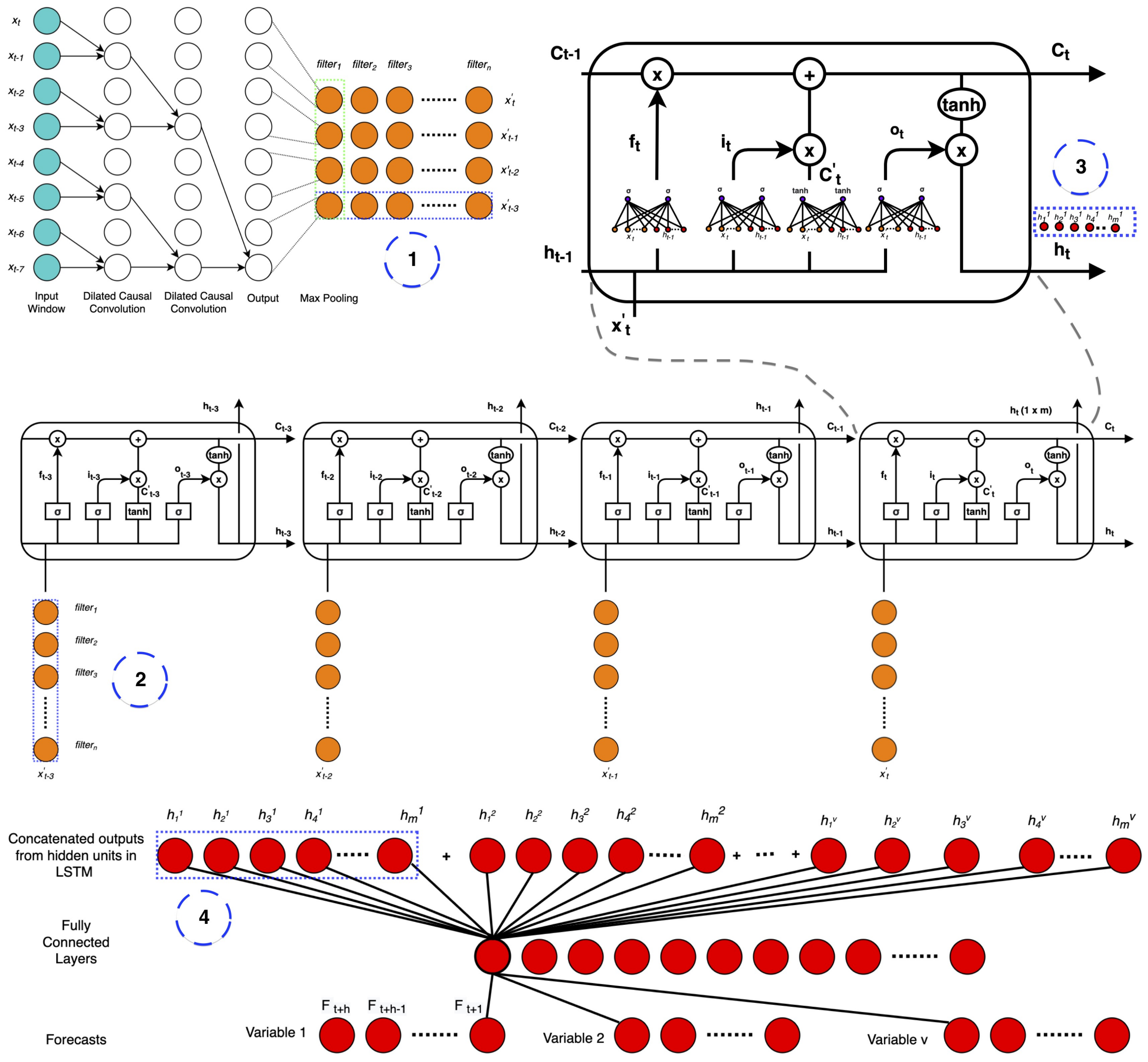
Input Window
Dilated Causal Convolution
Dilated Causal Convolution
Output
Max Pooling
filter1 filter2 filter3 filtern
1
2
3
4
Concatenated outputs from hidden units in LSTM
Fully Connected Layers
Forecasts
Variable 1
Variable 2
Variable v

Figure 3.5: The complete SCNN-BiLSTM hybrid neural network. The blue-dotted boxes highlight notable progressive stages. To indicate each stage, the blue boxes are labelled by a numbered blue circle. Blue-box 1 is part of the output from the convolutional component. The convolutional kernel with width h is applied to the raw time-series w which is $[x_t, x_{t-1}, ..., x_{t-7}]$. Then max-pooling with width l is applied to reduce the number of time steps to $[x_t^{'}, x_{t-1}^{'}, ..., x_{t-3}^{'}]$. The output from applying this process once is shown in a green-dotted box. The process is repeated for each filter n, resulting in n feature maps. The final output from the convolutional component is $\frac{w-h}{l} + 1$. Each variable v has separate convolutional and LSTM processing. This figure only shows the convolutional component being applied to one variable. The output from blue-box 1 is the first input to the LSTM (blue-box 2). Since the RNN component has bi-directional LSTMs, the output from the convolutional component is processed in both directions (the second LSTM in the backwards direction is not shown). There are m hidden units in the LSTM (blue-box 3). In the figure, the LSTM near blue-box 3 is a close up the final LSTM output. The m hidden units from the LSTM are concatenated with the LSTM hidden units of each variable v (blue-box 4). Next, there are fully connected. Finally, there are v outputs that are each h units long.

$$\mathbf{o}^i = r(\mathbf{W}_i\mathbf{h}^i + \mathbf{b}_i) \tag{3.23}$$

where $\mathbf{o}^i$ is the output and $\mathbf{h}^i$ denotes the input, $\mathbf{W}_i$ is the weight,$\mathbf{b}_i$ is bias term in of the i^{th} fully connected layer, r is the rectified linear unit (ReLU) activation function. There is an output for each input variable v that is the length of the forecast horizon.

The SCNN-BiLSTM is multi-headed, meaning each input variable has a separate CNN and RNN component before being concatenated and processed by fully connected layers. The hybrid neural network leverages the advantages of each separate neural network architecture to learn a complex representation of the raw time series input. The stacked-dilated-causal-convolutional component extracts temporal structural features from the raw time series. The temporal features are then processed forwards and backwards in time with an LSTM to capture the long-term time dependencies. Then the hidden LSTM units of each input variable are concatenated and processed by fully connected layers to determine the relationships between the variables.

The model is also able to forecast every time series variable at once, since it has an output for each variable. The hybrid model is shown in Figure 3.4 and Figure 3.5.

The multi-headed architecture provides recursive or direct forecasts. The outputs (forecast) are used as input in a recursive forecast until the desired forecasting horizon has been met. A direct forecast has the same number of outputs to match the forecasting horizon. If the desired forecasting horizon is longer than the number of outputs, the SCNN-BiLSTM will automatically produce the selected forecasting horizon with the recursive method.

Since the initial weights of a neural network are assigned randomly, the same configuration with that same training data will result in a different set of final trained weights. This will produce different forecasting performances. To address this, each hybrid configuration is trained and tested five times. Then the mean of all five forecasts becomes the final forecast. This also allows for a simple but robust method to estimate the uncertainty of the final forecast. The lower boundary of the uncertainty is defined as the mean of the forecasts subtracted by 1.95 times the standard deviation of the forecasts. Similarly, the upper boundary is the mean of the forecasts plus 1.95 times the standard deviation of the forecasts.

3.6 Summary

In this chapter we first covered the classical auto-regression methods, which are suitable for univariate time series forecasting. This was followed by presenting the basic building blocks of the proposed architecture. The building blocks consist of the stacked-dilated-causal convolutional neural network, a bidirectional long short-term memory network, and fully connected neural network layers.

Finally, the complete hybrid architecture consisting of Stacked-dilated-causal Convolutional Neural Network and Bidirectional Long Short-Term Memory (SCNN-BiLSTM) was presented. The main innovation of this hybrid technique is introducing stacked-causal convolutions for multivariate health forecasting, which efficiently provide full history coverage of the input window while maintaining the causal structure.

Chapter 4

Hospital Admission Forecasting

This chapter presents the the first health-forecasting use case to conduct experiments with the proposed SCNN-BiLSTM hybrid architecture. The aim of this use case is to predict number of hospital admission for mental health patients using multivariate time-series data.

This use case is framed as a series of experimental questions to answer, just like the infectious disease forecasting use case. In the Dataset section, the multivariate data used in the experiments are explained. Then in the Experiments and Results section, the first experiment investigates whether hybrid models are suitable for one-step forecasting. Next, long-range forecasting was examined to determine if multivariate models were better than univariate and if the stacked-dilated-causal Convolutional (SCNN) approach outperforms a standard CNN in a hybrid architecture. Finally, grid search and evolutionary neural architecture search were compared for the SCNN-BiLSTM to determine which approach would find the optimal configuration. The chapter will conclude with a discussion of the main experimental results.

4.1 Data

The data for the experiments consisted of the following variables:

1. Hospital admissions cases: The original dataset consisted of the date and time a patient at the Thunder Bay Regional Health Sciences Centre (TBRHSC) was admitted to adult mental health services from April 1, 2014, to February 15, 2021. In 2020 there was a shift in the time series distribution caused by the COVID-19 pandemic. Due to this shift in the time series distribution, the data

that followed 2019 were excluded from the test set. This time series dataset was re-sampled to forecast the total number of patients each week, with the beginning of each week defined as Monday. The weekly frequency was chosen because the daily frequency time series did not consist of significant regularities

2. Unemployment: The monthly unemployment in Thunder Bay. This variable has been found to influence mental health [39]. During an economic crisis, unemployment plays an important role in developing anxiety and depressive disorders and is closely related to suicide attempts.

3. Maximum temperature: The maximum weekly temperature in Thunder Bay. Meteorological variables such as maximum temperature have been linked with mental health [40].

4. Maximum precipitation: The maximum weekly precipitation in Thunder Bay. Precipitation is another meteorological variable along with temperature that is linked to mental health admissions in the literature. Since these variables have been shown to exhibit relationships with mental health admissions, multivariate forecasting models can exploit this information to produce better forecasts.

For the one-step forecasting, only hospital admissions cases were included. In the multi-step forecasting, the other three variables were included with the number of hospital admissions.

4.2 Experiments and Results

When conducting the experiments the following questions were considered:

1. For one-step forecasting, does a statistical, neural network, or hybrid neural network perform best?

2. Does the full coverage history provided by the SCNN-BiLSTM provide an advantage compared to a standard CNN for multi-step multivariate forecasting?

3. Will the evolutionary neural architecture search produce a configuration of the SCNN-BiLSTM that performs better than the grid search approach?

First, the CNN-BiLSTM hybrid architecture was compared with various classical statistical and machine learning techniques for one-step forecasting to examine if hybrid models can outperform statistical and neural network methods. The following methods were compared (each method is more fully explained in 3.2 and 3.3):

- The Auto-regressive Integrated Moving Average (ARIMA): The ARIMA model forecasts future values as a linear function of past observations and the random residuals.
- Seasonal ARIMA (SARIMA): The SARIMA model is an extension of ARIMA. It is useful for time series that exhibit seasonality. Seasonality is a repeating pattern that occurs over a fixed time interval.
- Fully Connected Neural Network (FCNN): The FCNN is a network of nodes organized in layers. The nodes in each layer are connected to every node from the previous layer.
- One-dimensional Convolutional Neural Network (CNN): The one-dimensional CNN uses filters to learn a more complex representation of the input time series.
- The Long Short-Term Memory Network (LSTM): The LSTM is a form of recurrent neural network that uses previous inputs rather than only the present input.

4.2.1 Experimental Setup

The neural network methods required an input that is a window of w lagged values of the each time series. This transformation of a time series is known as the *window* method (see 3.3.1). A time series of length N is reduced to

$$N - g - c + 1 \tag{4.1}$$

samples, where g is the window size, c is the output size.

The hyperparameters for the comparison models were all determined with grid searches over a range of possible values that will be described in the next section.

4.2.2 Performance Metric

The forecasting models required a performance metric to determine which model produces the best forecasts. In this study, the Mean Average Percentage Error (MAPE) was selected (see 3.1 for details).

4.2.3 Data Preprocessing

A neural network requires an input that is a window of w lagged values of each time series. This transformation of a time series is known as the *window* method. A time series of length N is reduced to $N - w - c + 1$ samples, where w is the window size, c is the output size.

After the dataset was transformed into windows of lagged windows of inputs and outputs, the values of each variable x were normalized with the following equation

$$\frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.2}$$

where x_{min} is the minimum value of that variable, x_{max} is the maximum value of the variable. The forecasts must be inverse-transformed so the values are consistent with the actual values.

Explicit denoising of the input time-series was not conducted since the dataset is not expected to contain any significant noise.

4.2.4 One-step Forecasting

The first part of this use case tested neural network, statistical, and hybrid neural network methods with a one-step hospital admissions forecast to answer experimental question 1.

The optimal hyperparameters of each method were determined with a grid search. The grid search trained each technique with the number of weekly hospital admissions from April 7, 2014, to December 31, 2018. It was tested over the entire year of 2019 with one-step walk-forward validation. The grid search pseudo-code is described in Algorithm 1.

The FCNN for this study consists of one hidden layer and an output layer. The number of nodes in the hidden layer, the number of epochs (number of iterations the model is trained with the complete data set), the batch size (number of observations to train the model prior to updating the weights), and the window size were

Algorithm 1: Grid Search for Hospital Admissions Forecasting

```
Input: Training Dataset $\mathbf{M}$, Walk-forward Validation Dataset $\mathbf{V}$, configurations $C$,
  forecast horizon $p$
Output: List of ranked model configurations $R$
  Initialisation: Empty list $R$
  for configuration $c$ in $C$ do
    initialize model $c$
    $c_{fit} = \mathbf{M}$ // fit model configuration with training set
    initialize $Results$ // List $Results$ to save forecast MAPE scores
    for step in $\mathbf{V}$ do
      $c_{forecast_p}$ // forecast one-step
      score $c_{forecast_p}$ with equation 3.1 // MAPE score of forecast
      $Results = c_{forecast_p}$ // append MAPE score of forecast to list
      $c_{update}$ // input with actual values for forecast horizon
    end for
    $R = Results$ // add MAPE results of model to $R$
  end for
  Sort $R$ // rank the configurations of model by MAPE score
  return $R$
```

all hyperparameters that needed to be determined. The grid search for this model involved the following permutations of hyperparameter values: 100, 200, 300, 400, and 500 nodes for the hidden layer; a batch size of 100, 200, and 300; 100, 200, and 300 epochs. The optimal hyperparameters determined by the grid search were 500 nodes, 300 epochs, a batch size of 100.

The CNN model has one hidden layer of fully connected nodes. It consists of the same hyperparameters as the FCNN, with the addition of filters, and kernels. The grid search for this model involved the following permutations of hyperparameter values: 40, and 50 nodes for the hidden layer; a batch size of 80, 90, and 100; 100, 200, and 300 epochs; kernel size of 3; 35, 40, 50 filters. The optimal hyperparameters were 40 nodes, 300 epochs, 35 filters, kernel size of 3, and a batch size of 80.

The LSTM model consists of a layer of LSTM nodes followed by a dense fully-connected layer and a single node as the output layer. The hyperparameters for this model are the number of nodes for the LSTM and fully-connected layer, the number of epochs, and the batch size. The number of nodes for the LSTM and fully-connected layer was set to be the same value. The grid search for this model included the following permutations of values: 100, 200 nodes; 100, 200, 300 epochs; 80, 90, 100 batch size. The optimal hyperparameters were 100 nodes, 100 epochs, and a batch size of 100.

The bidirectional LSTM (BiLSTM) consists of two layers of LSTM nodes. The first layer will consider the time series forward in time, while the second layer of the LSTM will consider the observations in reverse (backward) in time. In this way, the LSTM layers attempt to capture time-dependent regularities forward and backward in time. This approach will capture more time dependencies in the time series than a regular (forward) LSTM layer of nodes. The BiLSTM has the same hyperparameters as the LSTM. The optimal hyperparameters were 100 nodes, 200 epochs, and a batch size of 100.

The grid search for the CNN-LSTM involved the following permutations of hyperparameter values: 20 or 30 nodes for the hidden layer; a batch size of 40, 50, and 60; 100, 200, and 300 epochs; kernel size of 3; 10, 20, 30 filters. The optimal hyperparameters were 20 nodes, 300 epochs, 10 filters, kernel size of 3, and a batch size of 40.

The CNN-BiLSTM has the same hyperparameters as the CNN-LSTM except the model architecture that was implemented excludes a hidden layer of fully connected nodes. The optimal hyperparameters that were determined by a grid search were 300

Table 4.1: Results for Univariate One-Step Forecasting

Model	**Forecasting MAPE Score**
$SARIMA(0,1,1)(3,1,0)_7$	15.5
$ARIMA(5,1,0)$	16.8
FCNN	15.3
CNN	14.9
LSTM	15.2
BiLSTM	15.5
CNN-LSTM	14.8
CNN-BiLSTM	**13.8**

epochs, 10 filters, kernel size of 3, and a batch size of 30.

The parameters for the ARIMA model were determined with a grid search. The optimal p parameter was 5, the d parameter was 1, and the q parameter was 0, which resulted in an $ARIMA(5,1,0)$ model.

SARIMA adds four more parameters to the ARIMA model. These parameters are P for the seasonal auto-regressive order, D for the seasonal difference order, Q for the seasonal moving average order, and m for the number of time steps for a single seasonal period. The notation for a SARIMA is expressed as: $SARIMA(p,d,q)(P,D,Q)_m$. The model that was selected based on a grid search was the $SARIMA(0,1,1)(3,1,0)_7$.

The results of the one-step forecasts are shown in Table 4.1 with the optimal models from the grid search. The best performing technique was the CNN-BiLSTM, with a MAPE score of 13.8 percent. Notably, the only statistical method that performed as well as a deep learning method was the SARIMA, with a MAPE score of 15.5 percent.

The next best performing method was the CNN-LSTM model. The only difference between the CNN-LSTM and the CNN-BiLSTM is the bi-directional structure of the LSTM layer. The bi-directional structure enables the capture of past and future contexts, rather than exclusively past time series structures.

The two hybrid neural network approaches achieved the top performance metrics. This establishes the effectiveness of the hybrid neural network approach, which combines the strengths of various neural network architectures to improve time series forecasting. The results also answer question 1: The hybrid neural network approach outperforms the statistical and single neural network models for one-step hospital admissions forecasting.

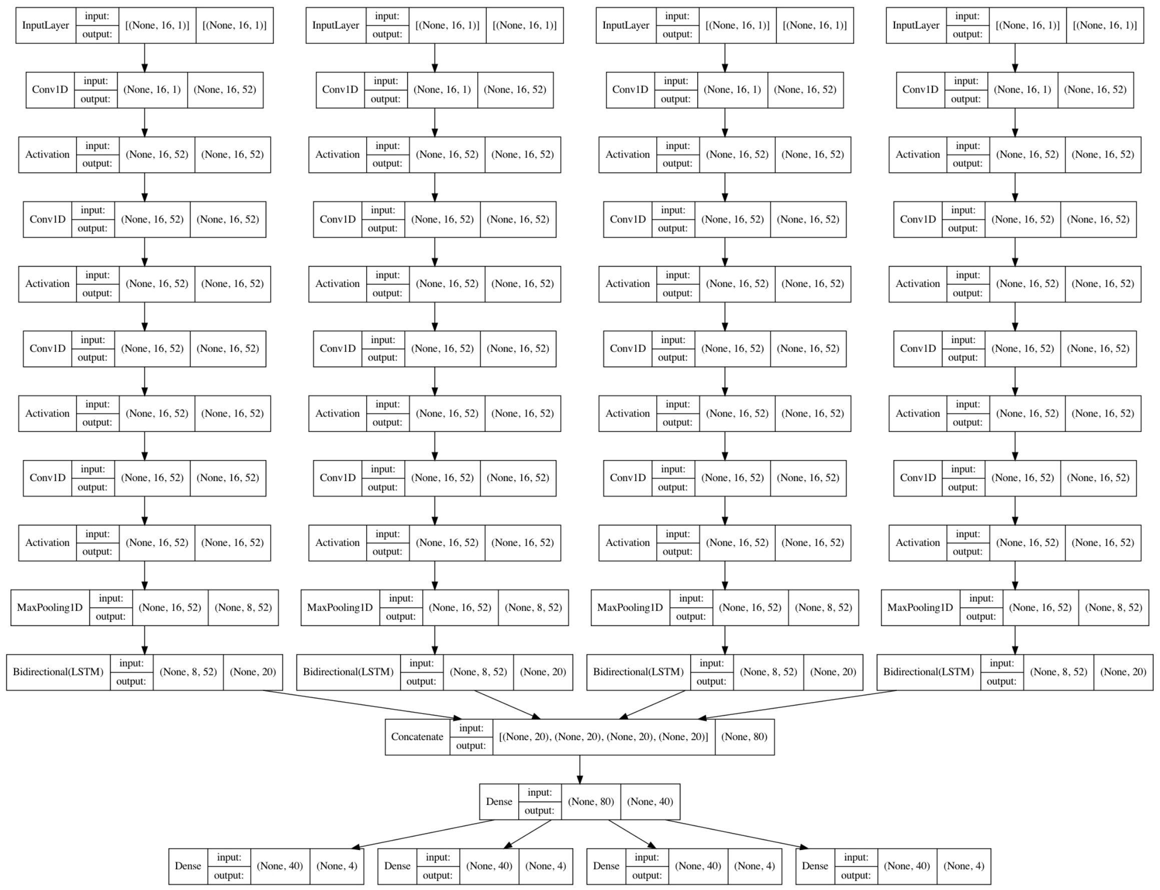
InputLayer input: output: [(None, 16, 1)] [(None, 16, 1)]
Conv1D input: output: (None, 16, 1) (None, 16, 52)
Activation input: output: (None, 16, 52) (None, 16, 52)
Conv1D input: output: (None, 16, 52) (None, 16, 52)
Activation input: output: (None, 16, 52) (None, 16, 52)
Conv1D input: output: (None, 16, 52) (None, 16, 52)
Activation input: output: (None, 16, 52) (None, 16, 52)
Conv1D input: output: (None, 16, 52) (None, 16, 52)
Activation input: output: (None, 16, 52) (None, 16, 52)
MaxPooling1D input: output: (None, 16, 52) (None, 8, 52)
Bidirectional(LSTM) input: output: (None, 8, 52) (None, 20)
Concatenate input: output: [(None, 20), (None, 20), (None, 20), (None, 20)] (None, 80)
Dense input: output: (None, 80) (None, 40)
Dense input: output: (None, 40) (None, 4)
Dense input: output: (None, 40) (None, 4)
Dense input: output: (None, 40) (None, 4)
Dense input: output: (None, 40) (None, 4)

Figure 4.1: SCNN-BiLSTM for the 4-step forecast. The multi-headed architecture accommodates the 4 time series variables.

4.2.5 Multi-step Forecasting

The next stage in this study was to extend the forecasting horizon from a one-step forecast to a multi-step forecast. A four-step forecasting horizon was selected, as a month (4 weeks) provides an adequate amount of time for health care management to prepare staff and resource demands.

Since the CNN-BiLSTM was the best performing model in the one-step forecasting experiment, variations of it were selected for the four-step forecasting. Notably, the proposed model, the SCNN-BiLSTM with full history coverage provided by stacked-dilated-causal convolutions, was included in determining if this architecture is better than a single layer of a standard CNN (CNN-BiLSTM).

Instead of considering a single variable (the number of hospital admissions), additional variables were included for the CNN-BiLSTM and SCNN-BiLSTM methods, creating multivariate models. The other variables included were the previously de-

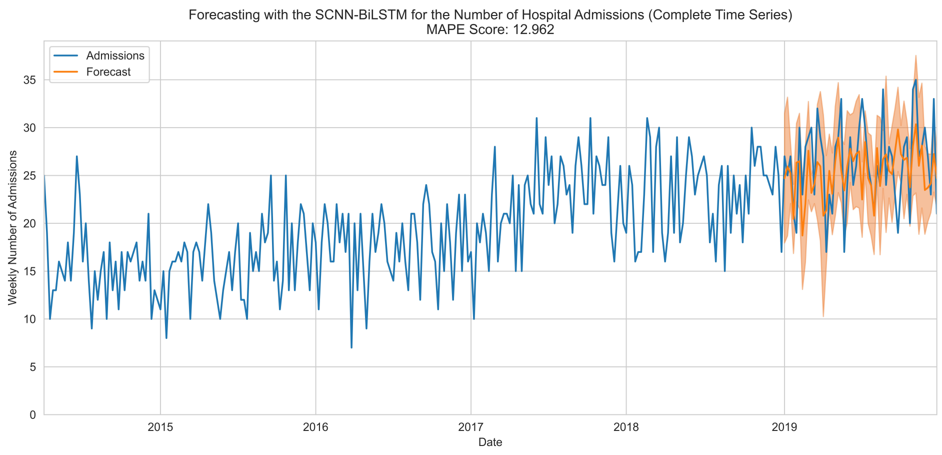

Figure 4.2: SCNN-BiLSTM 4-step forecast. The dark orange represents the forecast, the light orange is a 95 percent prediction interval. The entire data set is shown.

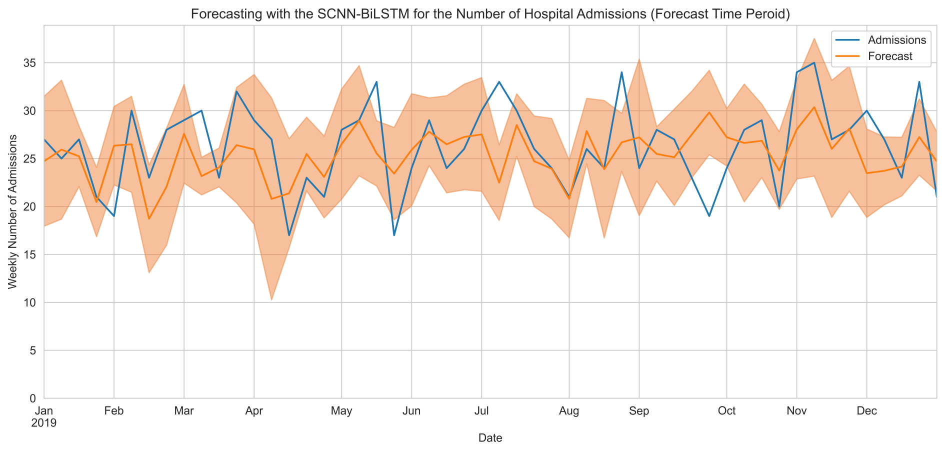

Figure 4.3: SCNN-BiLSTM 4-step forecast. The dark orange represents the forecast, the light orange is a 95 percent prediction interval. The figure is limited to the forecast range exclusively.

scribed unemployment, maximum temperature and total precipitation variables (see 5.1). To determine if the multivariate approach was superior to the univariate approach for long-range forecasting, a univariate CNN-BiLSTM was compared to the multivariate models.

Each of the four-step forecasting models was trained identically to the one-step forecasting models. This entailed training on the weekly time series data from April 7, 2014, to December 31, 2018. The year 2019 was the test range, which is also identical to the one-step forecasts. The walk-forward validation method was used to produce each 4-week forecast. This means that following each forecast, the actual values were provided as input for the next forecast for the entire test range.

A walk-forward validation MAPE score of various configurations of each model was conducted with a grid search. Each model considered 30 or 50 convolutional filters, an input window of 8,12, or 16, a fully connected component with 40 nodes or 80 and 40 nodes, and a maximum pool size of 1 or 2.

The best configuration for the univariate CNN-BiLSTM had 52 convolutional filters, an input window size of 8, fully connected layers with 80 and 40 nodes, and a max pool size of 1. The optimal configuration for the multivariate CNN-BiLSTM had 52 convolutional filters, an input window size of 8, fully connected layers with 80 and 40 nodes, and a max pool size of 2. Finally, the optimal architecture of the multivariate SCNN-BiLSTM had 52 convolutional filters, an input window size of 16, fully connected layer with 40 nodes, and a max pool size of 2. The architecture of the multivariate SCNN-BiLSTM is shown in Figure 4.1. The MAPE scores for the optimal configuration of each model are shown in Table 4.2.

Table 4.2: Performance Results for Four-Step Forecasting

Model	**Dataset**	**Forecasting MAPE Score**
CNN-BiLSTM	Univariate	14.78
CNN-BiLSTM	Multivariate	13.33
SCNN-BiLSTM	Multivariate	**12.96**

The results reveal the answer to experimental question 2: the stacked-dilated-causal convolutions of the SCNN-BiLSTM produced the best performing forecast for the four-step horizon. The forecast produced by the SCNN-BiLSTM is shown in Figure 4.2 and Figure 4.3. Visually, the pattern of the actual time series values (weekly hospital admissions data) is followed closely by the SCNN-BiLSTM forecast.

Once again the forecasts includes a prediction interval. The prediction interval assumes the errors are a Gaussian distribution. The prediction interval is 95 percent of the Gaussian distribution.

Evolutionary Neural Architecture Search

To answer the final experimental question, the evolutionary neural architecture search approach was compared to the grid search to determine if the best configuration of the SCNN-BiLSTM from evolutionary neural architecture search would outperform the best configuration of the same model from the grid search.

The following hyperparameters values were genes in each individual chromosome solution for the evolutionary neural architecture search: the window size had a range of 8 to 20 input lags; the output was from 1 to 12; the number of RNN nodes was from 5 to 100; the maximum pooling was from size 1 to 2; the first fully connected layer was from 5 to 200 nodes; the second fully connected layer was from -5 to 100 nodes (a negative value means this layer and all subsequent layers were not included); the third fully connected layer was from -5 to 100 nodes; the last fully connected layer was from -5 to 100 nodes; the number of convolutional filters was from 5 to 60.

Other values part of the evolutionary neural architecture search were: the crowding factor, which was set to 10; the number of individuals in the population, which was 20; the probability of crossover, set to 1.0; the probability of mutation, set to 1.0. The evolutionary neural architecture search was run for 5 generations. The results are shown in Figure 4.4

In the first generation, the best performing individual had a 14.2352 MAPE score. The average score of the first generation was 16.8665. In the last generation, or generation 5, the best performing individual got a 13.4669 MAPE score. The 5th generation of individuals had an average MAPE score of 14.863. Evolutionary neural architecture search improved the initial average score of the generation and the score of the best individual by the last generation.

The optimal solution from the evolutionary neural architecture search had a window size of 20, an output of 11, 81 RNN nodes, a maximum pooling size of 1, 42 convolutional filters, and fully connected layers with 50, 91, 67, 64 nodes. This configuration had a MAPE score of 13.4669, worse than the grid search MAPE score of 12.96.

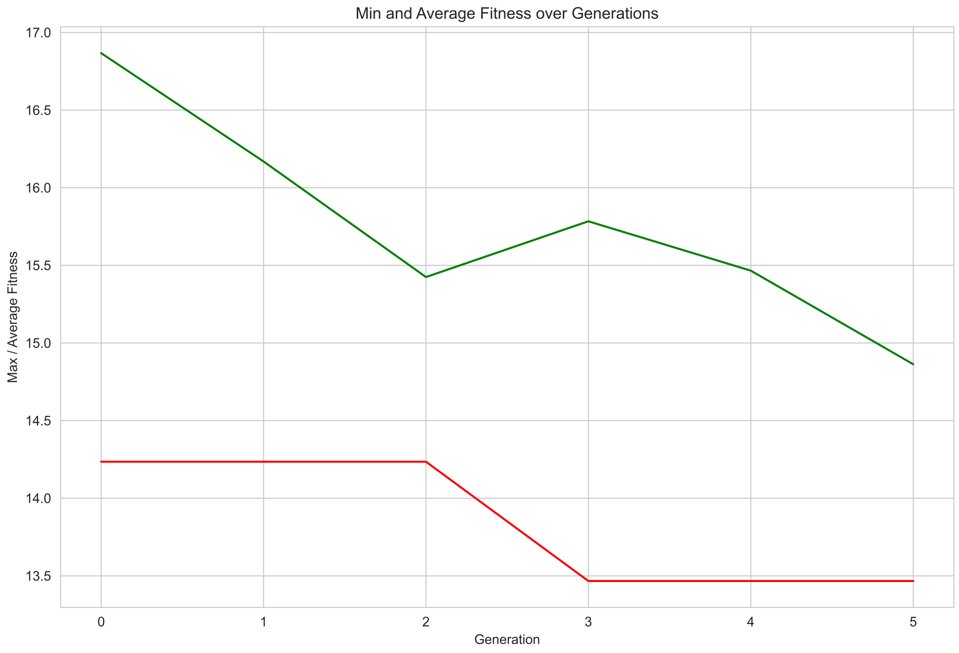

Figure 4.4: Results of the Evolutionary Neural Architecture Search. Each generation of solutions records the best performing individual (red) and the mean performance of the entire generation (green).

4.3 Discussion

This chapter presented the first use case for health forecasting with the SCNN-BiLSTM. It involved forecasting the weekly number of admissions to adult mental services at the Thunder Bay Regional Sciences Centre.

The first experiment of the use case established that hybrid models are suitable for one-step forecasting, as the CNN-BiLSTM outperformed statistical and neural network methods. Next, long-range forecasting was examined to determine if multivariate models were better than univariate and if the stacked-dilated-causal Convolutional (SCNN) approach outperforms a standard CNN in a hybrid model. The experiment revealed that the multivariate approach was superior to the univariate and that the SCNN outperformed the standard CNN.

Finally, the evolutionary neural architecture search was compared with the grid search to find the optimal configuration of the SCNN-BiLSTM. The configuration from the grid search had a better MAPE score than the evolutionary neural architecture search approach. Further experiments are required to establish if the evolutionary neural architecture search offers any benefits compared to the grid search.

Chapter 5

Infectious Disease Forecasting using Incomplete Multivariate Time-series

This chapter presents the second use case to test the proposed SCNN-BiLSTM architecture with a health forecasting problem. This use case considers infectious disease forecasting. Specifically, the spread of COVID-19 in Ontario, Canada, is forecast with various hybrid configurations of different types of Recurrent Neural Networks (RNN), and standard CNN or stacked-dilated-causal convolutions that provide full history coverage while maintaining the causal structure of the time series.

This chapter begins with the Data section, which describes the multivariate time-series data used for the experiments. The Experiments and Results section frames the experiments with four experimental questions. Then the experimental setup, the performance metric, the data prepossessing, and forecasting results are discussed. In the Forecasting Results subsection, there are answers to each of the four experimental questions. The last section of this chapter is the Discussion, which summarizes the experimental results from the use-case of infectious disease forecasting.

5.1 Data

The data for the experiments consisted of the following time series variables:

1. Number of confirmed COVID-19 cases: The number of confirmed COVID-19 cases was compiled daily from various public health units across Ontario [41].

2. Effective reproduction number: Ontario Health estimated the effective (or dynamic) reproduction number. It uses daily reported case counts and a 7-day rolling average for estimation, with a Markov Chain Monte Carlo sampling procedure. The mean serial interval was set to 4.5 days with a standard deviation of 2.5 days [42].

3. Total number of fully vaccinated individuals: Ontario Health collected the cumulative number of fully vaccinated individuals. To be *fully vaccinated*, an individual requires two doses of an approved COVID-19 vaccine.

4. Workplace mobility: Google collected the mobility measurements (workplace and residential) as part of their Community Mobility Reports [43]. The data shows how individuals in categorized places change compared to a baseline day. The baseline day is the median value for the five weeks from January 3, 2020, to February 6, 2020.

5. Residential mobility: Also collected by Google to compare the movement in residential areas to a baseline.

The collected multivariate time-series data represent epidemiological and mobility factors to capture the spread of COVID-19 infections across Ontario, Canada. Since COVID-19 spreads by close contact, the ability to quantify the movement of a population with the Google Mobility data is beneficial.

5.2 Experiments and Results

Since use case 1 established that hybrid neural networks are effective for health forecasting problems, this use case will not include statistical and neural network methods again. The experiments were conduced to answer the following key questions:

1. Is full history coverage using SCNN better than a standard CNN?

2. Is multivariate better than univariate forecasting with the CNN-RNN?

3. Which RNN (GRU, LSTM, or BiLSTM) performs best?

4. Will the evolutionary neural architecture search produce a configuration with a better MAPE score than grid search?

5.2.1 Experimental Setup

To answer the key experimental questions mentioned previously, there were two variations considered for the CNN-RNN: univariate and multivariate. These versions of the CNN-RNN were trained and tested with a grid search that exhaustively considered different architectures. The were five hyperparameters: the type of RNN, which was either a GRU, LSTM, or BiLSTM; the number of convolutional filters, which was either 10 or 50; the window size of either 8 or 16 lagged inputs; the fully connected layers, which could be a single layer of 50 nodes or two layers with 100 and 50 nodes; the number of convolutional layers, which could be 1, 2, or full history coverage. The total number of hyperparameters considered was $3 \times 2 \times 2 \times 2 \times 3 = 72$. The maximum pool size was set to 2, the convolutional padding was set to causal, the convolutional kernel size was set to 2, the dilation base for convolutional layers was set to 2, and the number of RNN hidden units was set to 10 for each configuration in the grid search. The grid search algorithm pseudo-code is shown in Algorithm 2.

The evolutionary neural architecture search (ENAS) was implemented as described in 3.4.2. The ENAS was only used for the SCNN-BiLSTM, as it proved to be the best architecture during the grid search for the first part of the experiments.

The univariate forecasting models were trained using the first $N-p$ observations, where N is the length of the daily number of confirmed COVID-19 cases in Ontario, and p is the forecast horizon. The forecast horizon was set to produce 28-step-ahead forecasts. The test range from April 1, 2021 to April 28, 2021 was chosen to be a difficult period to forecast accurately, as the number of confirmed cases initially increases, then peaks near the middle of the month, only to steadily decrease for the rest of the month. A 28-step forecast was also selected to produce a longer forecasting horizon than any previous CNN-RNN approach in the COVID-19 forecasting literature.

The length of this time series N was 412, it ranged from March 13, 2020 to March 31, 2021. The the trained models forecast the number of confirmed COVID-19 cases for the next 28 days from April 1, 2021 to April 28, 2021.

The multivariate forecasting models were fit with the daily number of confirmed COVID-19 cases, the effective reproduction number, the cumulative number of fully vaccinated individuals, baseline change in workplace mobility, and the baseline change in residential mobility in Ontario. To create a more realistic scenario, recent data was removed from some time series to simulate an incomplete multivariate dataset. Ontario Health published the effective reproduction number weekly. Likewise, other variables may not be entirely up to date throughout the pandemic.

Considering this scenario, in the experiment, the seven most recent days from the effective reproduction, the two most recent days of a cumulative number of fully vaccinated individuals, and the five most recent days of the baseline change in workplace mobility from the training set were removed. Since the dataset was incomplete, the multivariate CNN-RNN could not be trained with the first $N - p$ observations, like for the univariate version. Instead, the forecasting models were initially trained on the $N - p - q$ observations, q is computed as

$$\arg\max_{\mathbf{x}^j}(incomplete(\mathbf{x}^j) \quad \forall j \in [1, 2, ..., v] \tag{5.1}$$

where $incomplete(.)$ is the number of incomplete elements in each vector, $\mathbf{x}^j$ is the j^{th} variable in the multivariate dataset, and v is the number of variables considered. In the case of this study, q is 7, since the maximum time steps of incomplete data is 7 (from the effective reproduction number). After multivariate model was trained on the incomplete data it inferred the rest of the dataset. Then, with the completed dataset the model forecast the next 28 days of the number of confirmed cases.

Algorithm 2: Grid Search for Univariate and Multivariate Models

```
Input: Multivariate dataset $\mathbf{M} \in \mathbb{R}^{N \times d}$, configurations $C$, forecast horizon $p$
Output: List of ranked model configurations $R_{multivariate}$ and $R_{univariate}$
  Initialisation: Empty lists $R_{multivariate}$ and $R_{univariate}$
  for configuration $c$ in $C$ do
    initialize model $c$
    if ($c$ is multivariate model) then
      Determine $q$ by eq. (5.1)
      $c_{fit} = \mathbf{M} \in \mathbb{R}^{(N-p-q) \times d}$ // fit model with complete data
      $\mathbf{M_{incomplete}} = c_{forecast_q}$ // infer incomplete data
      $R_{multivariate} = c_{forecast_p}$ // forecast the next $p$ steps
      Sort $R_{multivariate}$ by score from eq. (3.1)
    end if
    if ($c$ is univariate model) then
      $c_{fit} = \mathbf{M} \in \mathbb{R}^{(N-p) \times 1}$
      $R_{univariate} = c_{forecast_p}$
      Sort $R_{univariate}$ by score from eq. (3.1)
    end if
  end for
  return $R_{multivariate}$ and $R_{univariate}$
```

5.2.2 Performance Metric

The forecasting models required a performance metric to determine which model produces the best forecasts. In this use-case experiment, the Mean Average Percentage Error (MAPE) was selected. The MAPE is a common metric to evaluate the performance of a forecast. See 3.1 for details on the MAPE forecasting metric.

5.2.3 Data Preprocessing

A neural network requires an input that is a window of w lagged values of each time series. This transformation of a time series is known as the *window* method. A time series of length N is reduced to $N - w - c + 1$ samples, where w is the window size, c is the output size.

After the dataset was transformed into windows of lagged windows of inputs and outputs, the values of each variable x were normalized with the following equation

$$\frac{x - x_{min}}{x_{max} - x_{min}} \tag{5.2}$$

where x_{min} is the minimum value of that variable, x_{max} is the maximum value of the variable. The forecasts must be inverse-transformed so the values are consistent with the actual values (number of confirmed cases).

Explicit denoising of the input time-series was not conducted since the dataset is not expected to contain any significant noise.

5.2.4 Forecasting Results

As previously mentioned, the univariate and multivariate CNN-RNN models were validated with a grid search of various architectures to determine the best performing univariate and multivariate configuration. The best configuration for the univariate model was found to be a bidirectional LSTM with 10 filters, 16 lagged values for the window size, a single layer of 50 fully connected nodes, and full history coverage for the number of stacked CNN layers. The best performing multivariate CNN-RNN had the identical configuration.

This answers key experimental question 1: the univariate and multivariate CNN-RNN models achieved the best MAPE scores with the stacked CNN with full history coverage of the input window values. Eight of the top ten results of the multivariate models from the grid search were full history coverage CNNs, and seven of the top

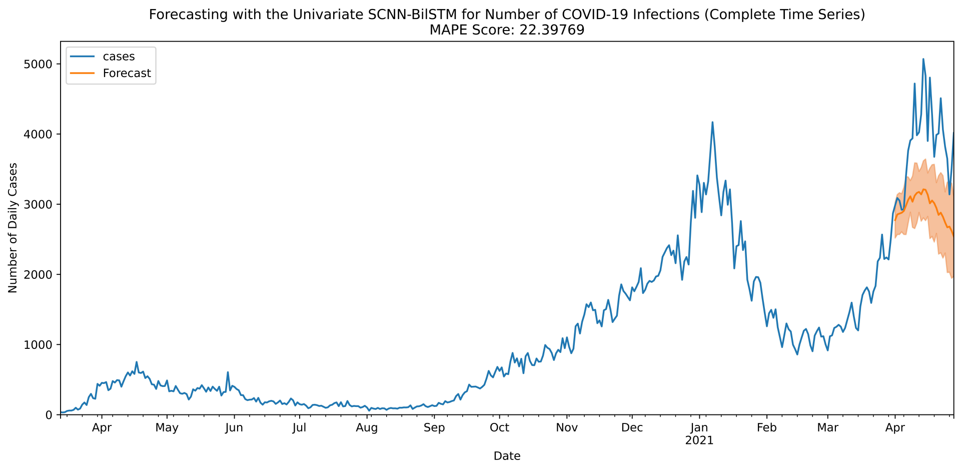

Figure 5.1: Univariate forecast for the number of confirmed COVID-19 cases in Ontario, Canada. The dark orange represents the forecast, the light orange is a 95 percent prediction interval. The entire data set is shown.

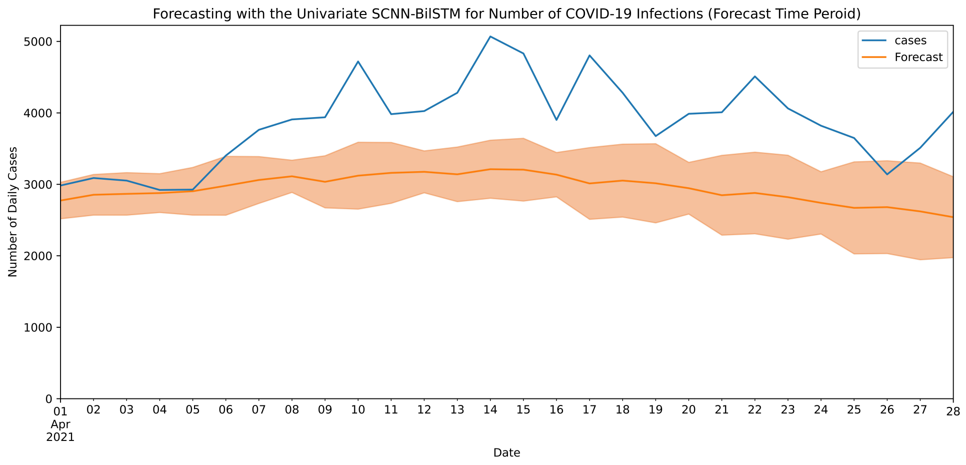

Figure 5.2: Univariate forecast for the number of confirmed COVID-19 cases in Ontario, Canada. The dark orange represents the forecast, the light orange is a 95 percent prediction interval. The figure is limited to the forecast range exclusively.

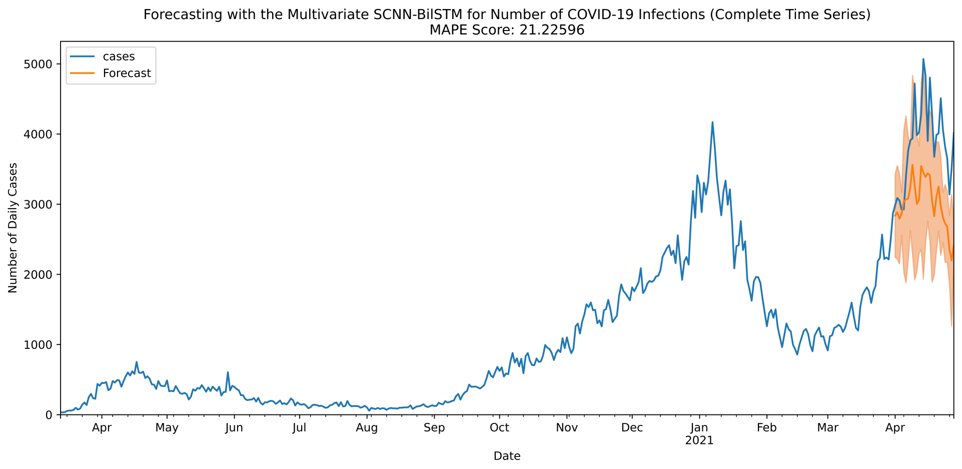

Figure 5.3: Multivariate forecast for the number of confirmed COVID-19 cases in Ontario, Canada. The dark orange represents the forecast, the light orange is a 95 percent prediction interval. The entire data set is shown.

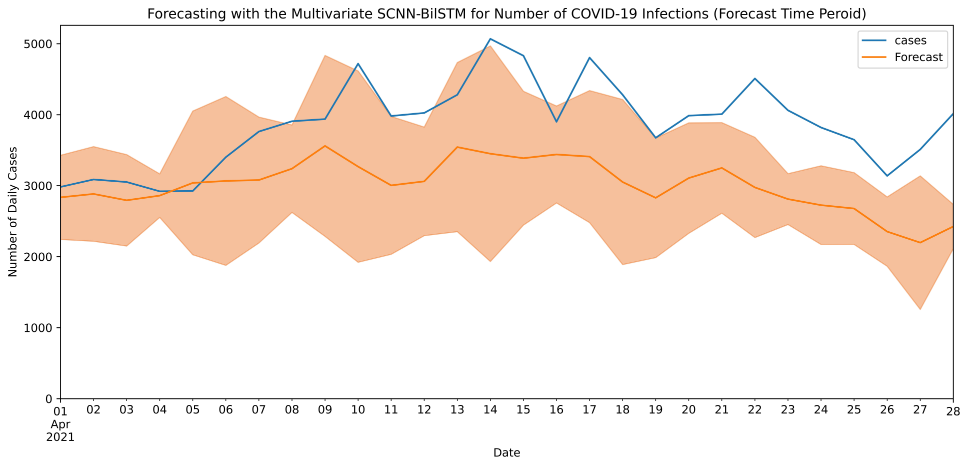

Figure 5.4: Multivariate forecast for the number of confirmed COVID-19 cases in Ontario, Canada. The dark orange represents the forecast, the light orange is a 95 percent prediction interval. The figure is limited to the forecast range exclusively.

ten results from the univariate models were full history coverage CNNs. These results suggest that the stacked causal dilated CNN approach is superior for this forecasting problem. The BiLSTM was the best RNN option for both the univariate and multivariate models, which answers key question 3.

To answer question 2, the 28-day-ahead forecast horizon from April 1, 2021, to April 28, 2021, produced by the univariate and multivariate SCNN-BiLSTM models, was grouped into one-step (1 day), medium-term (1-14 days), and long-term (15-28) forecasts. The results are displayed in Table 5.1. In the COVID-19 CNN-LSTM hybrid neural network forecasting literature, there is only one other study with a comparable result to this use case. It found that a CNN-LSTM had a MAPE score of 7.68 for a 14-day forecasting horizon. It should be noted that this forecast is for COVID-19 spread in India, not Ontario [25].

Given that the multivariate SCNN-BiLSTM had an incomplete dataset that first required completing with inferences from the model before the 28-step forecast, it is notable that multivariate SCNN-BiLSTM outperformed the univariate SCNN-BiLSTM for all forecasting horizons examined. These results support the claim that the multivariate hybrid forecasting approach is more effective than a univariate approach, at least for this study.

Table 5.1: Forecasting Performance (MAPE Score) of Best Univariate and Multivariate SCNN-BiLSTM models from Grid Search

Model	**One-step (1 day)**	**Medium Horizon (1-14 days)**	**Long Horizon (15-28 days)**
Univariate	7.043	16.878	27.917
Multivariate	4.928	14.924	27.528

The 28-step forecast produced by the multivariate SCNN-BiLSTM is displayed in Figure 5.3 and Figure 5.4. The same forecast horizon for the univariate SCNN-BiLSTM is shown in Figure 5.1 and Figure 5.2. The test range from April 1, 2021 to April 28, 2021 was chosen to be a difficult period to forecast accurately, as the number of confirmed cases initially increases, then peaks near the middle of the month, only to steadily decrease for the rest of the month. The multivariate SCNN-BiLSTM is able to skillfully forecast this challenging range of values. Visually, the univariate SCNN-BiLSTM appears to simply forecast the historical average of past cases.

The prediction interval provides an indication of the amount of uncertainty in the forecast. This is often not included in the machine learning literature [7], and was

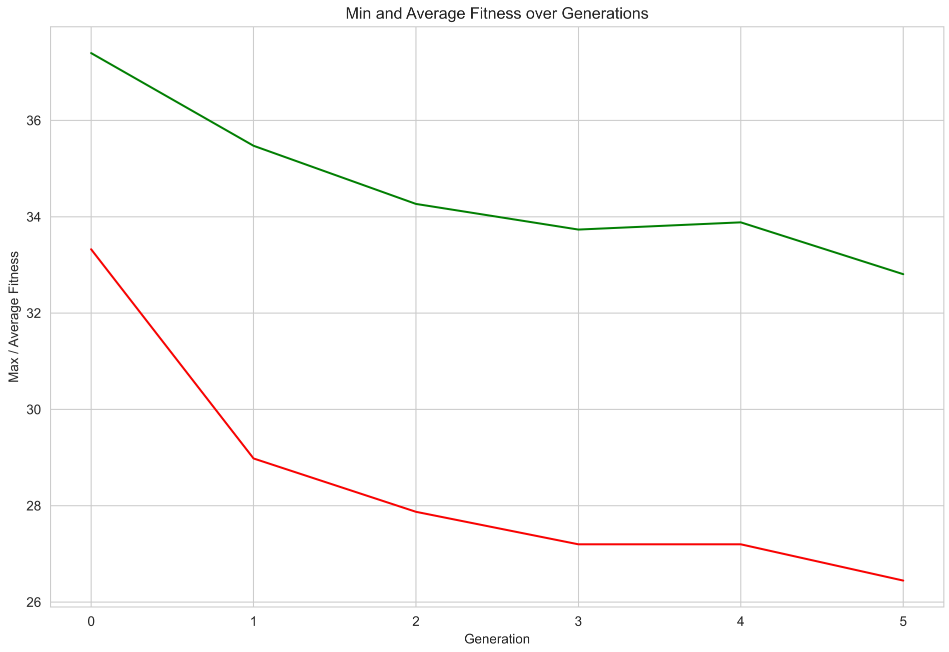

Figure 5.5: Results of the Evolutionary Neural Architecture Search. Each generation of solutions records the best performing individual (red) and the mean performance of the entire generation (green).

not part of past research on the CNN-RNN in COVID-19 forecasting. However, there is always uncertainty in every forecast, so it is essential to include. The prediction interval covers most of the actual cases of confirmed COVID-19 infections.

To answer question 4 the evolutionary neural architecture search was performed with the SCNN-BiLSTM to determine if it could produce a configuration that performed better than the grid search approach.

Evolutionary Neural Architecture Search

The evolutionary neural architecture search approach was compared to the grid search to determine if the best configuration of the SCNN-BiLSTM from evolutionary neural architecture search would outperform the best configuration of the same model from the grid search. The evolutionary neural architecture search was implemented with the evolutionary computation framework DEAP (Distributed Evolutionary Algorithms in Python) [44].

The following hyperparameter values were genes in each individual chromosome solution for the evolutionary neural architecture search: the window size had a range of 8 to 20 input lags; the output was from 1 to 12; the number of RNN nodes was from 5 to 100; the maximum pooling was from size 1 to 2; the first fully connected layer was from 5 to 200 nodes; the second fully connected layer was from -5 to 100 nodes (a negative value means this layer and all subsequent layers were not included); the third fully connected layer was from -5 to 100 nodes; the last fully connected layer was from -5 to 100 nodes; the number of convolutional filters was from 5 to 60.

Other values part of the evolutionary neural architecture search were: the crowding factor, which was set to 10; the number of individuals in the population, which was 20; the probability of crossover, set to 1.0; the probability of mutation, set to 1.0. The evolutionary neural architecture search was run for 5 generations. The results are shown in Figure 5.5.

In the first generation, the best performing individual had a 33.3244 MAPE score. The average score of the first generation was 37.3979. In the final generation, or generation 5, the best performing individual got a 26.4455 MAPE score. The complete generation of individuals had an average MAPE score of 32.8074. The initial average score of the generation and the score of the best individual improved from the first generation to the last.

The optimal solution had a window size of 21, an output of 7, 16 RNN nodes, a maximum pooling size of 1, and fully connected layers with 55, 78, 48, 33 nodes, and 23 convolutional filters. This configuration had a MAPE score of 26.4455 over the entire 28-day forecast, which was worse than the grid search MAPE score of 21.2260.

5.3 Discussion

The experimental results establish that the Stacked-dilated-causal Convolutional approach is better than a standard CNN in a hybrid model, the multivariate approach outperformed univariate models, and the BiLSTM was the best performing RNN compared to the LSTM and GRU (gated recurrent unit).

The multivariate dataset in the experiments is unique in infectious disease forecasting, as it has incomplete temporal data that must first be inferred before forecasting is possible. This scenario is more realistic since data for each variable is often not updated at the same rate.

Finally, to determine the optimal configuration of the SCNN-BiLSTM, the evolutionary neural architecture search was compared with the grid search. The grid search found the best-performing forecasting configuration of the SCNN-BiLSTM. Further experiments are required to determine if the intelligent search provided by the evolutionary neural architecture search offers any benefit to choosing the optimal design of the SCNN-BiLSTM for health forecasting problems.

Chapter 6

Conclusion

This thesis introduces the hybrid Stacked-dilated-causal Convolutional Neural Network and Bidirectional Long Short-Term Memory (SCNN-BiLSTM) for health forecasting using multivariate time-series data. The stacked-dilated-causal convolutions provide full history-coverage of the input window while maintaining the causal structure such that each output depends on all previous elements in a temporal sequence. The main contributions of this dissertation and future work are presented in this chapter.

6.1 Contributions

The proposed SCNN-BiLSTM architecture extends previous research on hybrid models for health forecasting in the following ways: (1) Stacked-dilated-causal CNN to provide full history coverage that maintains the causal structure of a temporal sequence; (2) automatic inference of incomplete temporal sequences; (3) multi-headed architecture to model multivariate time-series such that a separate combination of CNN and RNN processes each temporal sequences; (4) long-range forecast horizon. Two use-case scenarios were studied to establish the effectiveness of the proposed SCNN-BiLSTM architecture: hospital admissions forecasting and infectious disease forecasting.

In the use-case for hospital admissions forecasting, the number of admissions to adult mental health services at the Thunder Bay Regional Health Sciences Centre was predicted. The experiments were conducted to answer the following: (1) Is the hybrid neural network approach better than a single neural network or a statistical method when considering short-term forecasts? (2) for long-range forecasts, does the Stacked-dilated-causal CNN provide better forecasting than standard CNN? In the short-term (one-step) forecasts, it was established that the CNN-BiLSTM hybrid model outperformed various statistical and neural network techniques. In the four-step forecasting experiment, CNN-BiLSTM was compared with SCNN-BiLSTM. SCNN-BiLSTM was found to outperform CNN-BiLSTM for the long-term 4-step forecast.

The infectious disease experiment utilized COVID-19 data and mobility data in Ontario, Canada, to forecast the spread of new daily COVID-19 cases for the next 28 days. Various configurations of a CNN with a recurrent neural network (RNN) were tested to determine whether: (1) the full history coverage provided by Stacked-dilated-causal CNN performed better than standard CNN; (2) the multivariate or univariate approach has superior performance; (3) the LSTM, Bidirectional LSTM (BiLSTM) or the gated recurrent unit (GRU) was the optimal RNN for the hybrid model. The use-case experiments revealed that the full history coverage provided by the SCNN outperformed standard CNN, the multivariate approach was superior to the univariate even in the presence of an incomplete dataset, and the BiLSTM was the optimal RNN to be included in the hybrid model.

Finally, each use-case tested whether an intelligent search, such as the evolutionary neural architecture search, would find a better configuration of the SCNN-BiLSTM for the health forecasting problem than the grid search. However, the grid search approach produced better forecasting for both use cases. The results could be explained by the lack of available computing resources required for a detailed evolutionary neural architecture search.

Both use-cases demonstrate that the proposed SCNN-BiLSTM hybrid architecture is suitable for multivariate health forecasting. It was the best performing model in the infectious disease forecasting and long-range hospital admissions forecasting use-cases, suggesting that the model can generalize across various health forecasting problems.

6.2 Future Work

More experiments with evolutionary neural architecture search will be investigated in future. Various hyperparameters are part of the evolutionary neural architecture search. These hyperparameters could be explored more deeply. Notably, the number of generations and individuals in a generation could be varied to determine if either of these hyperparameters would improve the optimal configuration of the SCNN-

BiLSTM for health forecasting.

In addition to the grid search and evolutionary neural architecture search, a randomized search could be utilized to determine the optimal configuration of the SCNN-BiLSTM for health forecasting.

The architecture of the SCNN-BiLSTM could be altered in various ways. Larger kernels would reduce the number of layers required for full history coverage. The stacked-dilated-casual convolutions could be replaced with stacked-causal convolutions to determine which approach is superior in health forecasting. The multi-headed architecture could be changed so that one CNN and LSTM process all the inputs.

In this thesis, only multivariate time-series data was considered. However, there are other modalities of data that can be incorporated in forecasting models. One such modality would be text data. Text data collected from social media may play an important role in future to improve upon the forecasting results obtained in this research.

Another kind of feature I would include is *static* features. An example of a static feature would be the day of the week. There could be seasonal patterns in such static variables.

Bibliography

[1] D. Kahneman, “Article commentary: Judgment and decision making: A personal view,” *Psychological science*, vol. 2, no. 3, pp. 142–145, 1991.

[2] E. D. Craft, “Economic history of weather forecasting,” *EH. Net Encyclopedia, edited by Robert Whaples*, 2001.

[3] T. Gneiting and A. E. Raftery, “Weather forecasting with ensemble methods,” *Science*, vol. 310, no. 5746, pp. 248–249, 2005.

[4] L. Ferrara, C. Marsilli, and J.-P. Ortega, “Forecasting growth during the great recession: is financial volatility the missing ingredient?” *Economic Modelling*, vol. 36, pp. 44–50, 2014.

[5] J. Castle, D. F. Hendry, and M. P. Clements, *Forecasting.* Yale University Press, 2019.

[6] F. Lazzeri, *Machine learning for time series forecasting with Python.* John Wiley & Sons, 2020.

[7] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “Statistical and machine learning forecasting methods: Concerns and ways forward,” *PloS one*, vol. 13, no. 3, p. e0194889, 2018.

[8] I. N. Soyiri and D. D. Reidpath, “An overview of health forecasting,” *Environmental health and preventive medicine*, vol. 18, no. 1, pp. 1–9, 2013.

[9] E. Vynnycky and R. White, *An introduction to infectious disease modelling.* OUP oxford, 2010.

[10] C. S. Lutz, M. P. Huynh, M. Schroeder, S. Anyatonwu, F. S. Dahlgren, G. Danyluk, D. Fernandez, S. K. Greene, N. Kipshidze, L. Liu *et al.*, “Applying infectious

disease forecasting to public health: a path forward using influenza forecasting examples," *BMC Public Health*, vol. 19, no. 1, pp. 1–12, 2019.

[11] I. Cooper, A. Mondal, and C. G. Antonopoulos, "A sir model assumption for the spread of covid-19 in different communities," *Chaos, Solitons & Fractals*, vol. 139, p. 110057, 2020.

[12] Q. Chen, A. Allot, and Z. Lu, "Keep up with the latest coronavirus research." *Nature*, vol. 579, no. 7798, pp. 193–194, 2020.

[13] L. Kong, M. Duan, J. Shi, J. Hong, Z. Chang, and Z. Zhang, "Compartmental structures used in modeling covid-19: a scoping review," *Infectious diseases of poverty*, vol. 11, no. 1, pp. 1–9, 2022.

[14] J. Panovska-Griffiths, "Can mathematical modelling solve the current covid-19 crisis?" pp. 1–3, 2020.

[15] J. Whitworth, "Covid-19: a fast evolving pandemic," *Transactions of the Royal Society of Tropical Medicine and Hygiene*, vol. 114, no. 4, p. 241, 2020.

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

[17] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," *Advances in neural information processing systems*, vol. 2, 1989.

[18] I. E. Livieris, E. Pintelas, and P. Pintelas, "A cnn–lstm model for gold price time-series forecasting," *Neural computing and applications*, vol. 32, no. 23, pp. 17 351–17 360, 2020.

[19] J. Bi, X. Zhang, H. Yuan, J. Zhang, and M. Zhou, "A hybrid prediction method for realistic network traffic with temporal convolutional network and lstm," *IEEE Transactions on Automation Science and Engineering*, 2021.

[20] R. Zhao, R. Yan, J. Wang, and K. Mao, "Learning to monitor machine health with convolutional bi-directional lstm networks," *Sensors*, vol. 17, no. 2, p. 273, 2017.

[21] Z. M. Zain and N. M. Alturki, “Covid-19 pandemic forecasting using cnn-lstm: a hybrid approach,” *Journal of Control Science and Engineering*, vol. 2021, 2021.

[22] L. Xu, R. Magar, and A. B. Farimani, “Forecasting covid-19 new cases using deep learning methods,” *Computers in biology and medicine*, vol. 144, p. 105342, 2022.

[23] H. Widiputra, “Ga-optimized multivariate cnn-lstm model for predicting multi-channel mobility in the covid-19 pandemic,” *Emerging Science Journal*, vol. 5, no. 5, pp. 619–635, 2021.

[24] S. Ketu and P. K. Mishra, “India perspective: Cnn-lstm hybrid deep learning model-based covid-19 prediction and current status of medical resource availability,” *Soft Computing*, vol. 26, no. 2, pp. 645–664, 2022.

[25] H. Verma, S. Mandal, and A. Gupta, “Temporal deep learning architecture for prediction of covid-19 cases in india,” *Expert Systems with Applications*, vol. 195, p. 116611, 2022.

[26] C. A. Parker, N. Liu, S. X. Wu, Y. Shen, S. S. W. Lam, and M. E. H. Ong, “Predicting hospital admission at the emergency department triage: A novel prediction model,” *The American journal of emergency medicine*, vol. 37, no. 8, pp. 1498–1504, 2019.

[27] K. L. Khatri and L. S. Tamil, “Early detection of peak demand days of chronic respiratory diseases emergency department visits using artificial neural networks,” *IEEE journal of biomedical and health informatics*, vol. 22, no. 1, pp. 285–290, 2017.

[28] H. Bibi, A. Nutman, D. Shoseyov, M. Shalom, R. Peled, S. Kivity, and J. Nutman, “Prediction of emergency department visits for respiratory symptoms using an artificial neural network,” *Chest*, vol. 122, no. 5, pp. 1627–1632, 2002.

[29] K. P. Moustris, K. Douros, P. T. Nastos, I. K. Larissi, M. B. Anthracopoulos, A. G. Paliatsos, and K. N. Priftis, “Seven-days-ahead forecasting of childhood asthma admissions using artificial neural networks in athens, greece,” *International Journal of Environmental Health Research*, vol. 22, no. 2, pp. 93–104, 2012.

[30] L. Zhou, P. Zhao, D. Wu, C. Cheng, and H. Huang, "Time series model for forecasting the number of new admission inpatients," *BMC medical informatics and decision making*, vol. 18, no. 1, pp. 1–11, 2018.

[31] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE transactions on neural networks and learning systems*, 2021.

[32] Z.-H. Zhan, J.-Y. Li, and J. Zhang, "Evolutionary deep learning: A survey," *Neurocomputing*, vol. 483, pp. 42–58, 2022.

[33] M. Mitchell, "Genetic algorithms: An overview." in *Complex.*, vol. 1, no. 1. Citeseer, 1995, pp. 31–39.

[34] J. Yang and C. K. Soh, "Structural optimization by genetic algorithms with tournament selection," *Journal of computing in civil engineering*, vol. 11, no. 3, pp. 195–200, 1997.

[35] C. W. Ahn and R. S. Ramakrishna, "Elitism-based compact genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 367–385, 2003.

[36] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[37] A. Gulli and S. Pal, *Deep learning with Keras.* Packt Publishing Ltd, 2017.

[38] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[39] G. Konstantakopoulos, K. Pikouli, D. Ploumpidis, E. Bougonikolou, K. Kouyanou, M. Nystazaki, and M. Economou, "The impact of unemployment on mental health examined in a community mental health unit during the recent financial crisis in greece." *Psychiatrike= Psychiatriki*, vol. 30, no. 4, pp. 281–290, 2019.

[40] A. Aguglia, G. Serafini, A. Escelsior, M. Amore, and G. Maina, "What is the role of meteorological variables on involuntary admission in psychiatric ward?

an italian cross-sectional study," *Environmental Research*, vol. 180, p. 108800, 2020.

[41] Ontario Health. Confirmed positive cases of covid-19 in ontario. [Online]. Available: https://data.ontario.ca/dataset/confirmed-positive-cases-of-covid-19-in-ontario

[42] ——. Effective reproduction number (re) for covid-19 in ontario. [Online]. Available: https://data.ontario.ca/dataset/effective-reproduction-number-re-for-covid-19-in-ontario

[43] Google LLC. Google covid-19 community mobility reports. [Online]. Available: https://www.google.com/covid19/mobility

[44] F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné, "Deap: Evolutionary algorithms made easy," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2171–2175, 2012.