
UNIVERSITY OF CAPE TOWN

MASTERS THESIS

# Spatial-Temporal Graph Neural Networks for Weather Prediction in South Africa

*Author:*

Mikhail DAVIDSON

*Supervisor:*

A/Prof. Deshendran MOODLEY

*A thesis submitted in fulfilment of the requirements for the degree in Master of Science*

*in the*

Department of Computer Science

# Declaration of Authorship

I, Mikhail DAVIDSON, declare that this thesis titled, *"Deep Learning for Weather Prediction"* and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, that has been clearly stated.
- Where I have consulted the published work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Signed by candidate

Date:

# Declaration of Publications

1. M. Davidson, and D. Moodley 2022. ST-GNNs for Weather Prediction in South Africa. In *Artificial Intelligence Research*. Springer International Publishing. 1734(3):93–107

# Abstract

Spatial-temporal graph neural networks (ST-GNN) have been shown to be highly effective for flow prediction in dynamic systems but are under-explored for weather prediction applications. Additionally, current approaches for evaluating ST-GNN models do not take into account the robustness and stability of the trained models. This research compared and evaluated two ST-GNN models, i.e. Graph WaveNet (GWN) and the Low-Rank Weighted Graph Neural Network (WGN), for maximum surface temperature prediction in South Africa. The results of these two ST-GNN models are compared to two basic temporal deep neural network architectures, i.e. the LSTM and the TCN, for maximum surface temperature prediction across 21 weather stations in South Africa. A novel framework is presented in which to reliably evaluate model robustness and stability for maximum surface temperature. This framework was used to perform rigorous experiments to evaluate the stability and robustness of the ST-GNN models for maximum surface temperature prediction. The results show that the GWN model outperforms the other models across different prediction horizons with an average SMAPE score of 8.30%. Despite the GWN model outperforming the other models on average, the TCN model outperformed both ST-GNN models at particular weather stations. The results indicate that an ensemble approach consisting of ST-GNN models and basic temporal deep neural network architectures would be the most effective approach for maximum surface temperature prediction. Finally, the learnt adjacency matrices of the two ST-GNNs were analysed and compared to gain insights into the prominent spatial-temporal dependencies between weather stations.

# Acknowledgements

I want to thank the South African Weather Services for providing the data used in this thesis. I would also like to thank the Centre for Artificial Intelligence for its financial support. I am most grateful for my supervisor A/Prof. Deshen Moodley. Thank you for your making this opportunity possible. Your guidance, patience, and care helped me through this journey and has resulted in immense personal growth. I thank my friends, Carla Mathyse and Diana Swales, for their continuous support through this long journey. I thank my family, Jenny Paulse, Catherine, Krista, Yusha, and Dhanyal Davidson, for their unwavering support and encouragement. Finally, I thank my parents, Ishmet and Nazeema Davidson, to whom this thesis is dedicated. I am eternally grateful to you; thank you for showing me that the sky is not the limit.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **ABGNN** | Attention-Based Graph Neural Network |
| **ACF** | Autocorrelation Function |
| **ADF** | Augmented Dickey-Fuller |
| **AR** | Autoregressive |
| **ARIMA** | Autoregressive Integrated Moving Average |
| **ANN** | Artificial Neural Network |
| **CNN** | Convolutional Neural Network |
| **DGNN** | Dynamic Graph Convolutional Neural Network |
| **DNN** | Deep Neural Network |
| **FCN** | Fully-Connected Network |
| **GCN** | Graph Convolutional Network |
| **GGRU** | Graph Gated Recurrent Unit Network |
| **GPU** | Graphics Processing Unit |
| **GWN** | Graph WaveNet |
| **HPO** | Hyper-Parameter Optimisation |
| **IDW** | Inverse Distance Weighting |
| **LSTM** | Long Short-Term Memory Network |
| **MA** | Moving Average |
| **MSE** | Mean Squared Error |
| **MLPNN** | Multi-Layer Perceptron Neural Network |
| **MAPE** | Mean Absolute Percentage Error |
| **MIMO** | Multi-Input Multi-Output |
| **MISO** | Multi-Input Single-Output |
| **MAE** | Mean Absolute Error |
| **NWP** | Numerical Weather Prediction |

| | |
|---|---|
| **ReLU** | Rectified Linear Unit |
| **RMSE** | Root Mean Squared Error |
| **RNN** | Recurrent Neural Network |
| **SARIMA** | Seasonal Autoregressive Integrated Moving Average |
| **SMAPE** | Symmetrical Mean Absolute Percentage Error |
| **ST** | Spatial-Temporal |
| **ST-GNN** | Spatial-Temporal Graph Neural Network |
| **SVM** | Support Vector Machine |
| **TCN** | Temporal Convolutional Network |
| **WGN** | Low-Rank Weighted Graph Convolutional Network |

# Chapter 1

# Introduction

## 1.1 Background, Context and Problems

Deep learning is an under-explored tool for predicting the weather in South Africa, having only been used to predict lightning occurrences [1] and thunderstorm severity [2]. Current weather prediction approaches are computationally expensive and have several shortcomings that lower accuracy and require a substantial amount of prior knowledge. The lack of deep learning research for weather prediction in South Africa hinders the potential use and further exploration of deep learning as an alternative for weather prediction.

The weather system can be characterised as a highly dynamic and chaotic system that exhibits complex and often latent spatial and temporal dependencies. While Deep Neural Network (DNN) approaches like LSTMs and TCNs [3] can capture both long-term and short-term temporal patterns, they do not cater for spatial patterns. A novel group of emerging deep neural network architectures, Spatial-Temporal Graph Neural Networks (ST-GNNs), have emerged to capture temporal and spatial patterns that are highly effective for flow prediction in dynamic systems [4] [5].

Each variable is typically represented as a node in a graph that captures the local dynamics at that location, whilst the system level or global dynamics are captured by weighted edges that reflect the strength of the dependencies between locations [5]. Thus a variable (node) has strong connections (links) to variables that are affected by changes in its values and weaker links to variables that are not affected by changes in its values. In this way, a node captures local dynamics (single location), while the overall graph structure captures the system's global (inter-location) dynamics. The prevalent application predicts traffic flow at different points in a city traffic network [4] [5]. The inter-variable dependencies for this case are spatial relations between traffic flow at different points in the network. These approaches are referred to as spatial-temporal GNNs (ST-GNNs) [5].

One of the challenges with ST-GNNs is that the spatial dependencies captured in the graph structure can be dynamic. Early ST-GNN approaches [4] used a static graph structure provided as prior knowledge as a fixed adjacency matrix before training. However, spatial dependencies in dynamic systems can change and evolve. Graph WaveNet (GWN) [5] was amongst the first to include a dynamic adjacency matrix that learnt and adapted to evolving spatial dependencies. While these

techniques have been applied to other domains like stock market prediction [6], they have not been applied to weather prediction. Despite the emergence of newer approaches like StemGNN [7] and MTGNN [8], which have outperformed Graph WaveNet on traffic flow prediction, GraphWaveNet still provides better performance on some applications like stock market prediction [6].

As mentioned above, the extensive research of ST-GNNs for spatial-temporal prediction in the traffic flow domain has resulted in the development of state-of-the-art (SOTA) spatial and temporal dependency capturing techniques. The traffic flow and weather domain largely face similar spatial-temporal prediction challenges. The SOTA ST-GNN models developed in the traffic flow domain have also been shown to be generalisable to other domains like stock market price prediction. These SOTA ST-GNN models like GWN, have not been implemented and tested for their potential for weather prediction.

The use of ST-GNNs for weather prediction is under-explored. We found only one ST-GNN architecture proposed specifically for weather prediction, i.e. the Low-Rank Weighted Graph Neural Network (WGN) [9]. While the WGN showed good accuracy compared to LSTMs and graph convolutional neural networks, the performance of the WGN was not compared to recent ST-GNN approaches, like GWN emerging from the traffic domain. Furthermore, it was only evaluated on weather prediction in a single region, i.e. the continental United States. Since the weather system is highly dynamic and erratic, the WGN model may perform differently when implemented over a different geographical area with different weather systems.

In dynamic systems like weather, DNN models become outdated and must be frequently updated as new data becomes available to cater for new spatial and temporal patterns that may emerge. Hold-out validation is traditionally used across most studies in this area [3, 9, 5]. The use of a single hold-out set does not provide a sufficiently robust performance measure for datasets that are erratic and non-stationary. Furthermore, DNNs, as a result of random initialisation and possibly other random parameter settings, could yield substantially different results when re-run with the same hyper-parameter values on the same data set. It is also crucial that optimal DNN configurations should be stable [10], i.e. have a minimal deviation from the mean test loss across multiple runs. Model robustness tests for model performance changes when trained on new data, and model stability indicate the variation in a model's performance over repeated experiments. However, many studies do not perform adequate model robustness and stability tests for the configured model [3, 9, 5].

Finally, it is difficult to gain deep insight into weather systems, both locally and globally, due to weather being a large, dynamic and interconnected system. Dynamic adjacency matrices offer a unique alternative to gain insights into spatial-temporal dependencies underlying weather systems. These insights can be used to verify existing weather knowledge or present new information. Since ST-GNNs have not been explored for weather prediction in South Africa, there is currently no research into the insights that analysing learnt spatial-temporal dependencies could provide on the relationship between weather stations. Furthermore, current literature has not compared the learnt spatial-temporal dependencies between different ST-GNNs trained on the same data.

## 1.2 Aim and Objectives

The overall aim of this work is to reliably evaluate and compare ST-GNN models i.e. WGN and GWN with basic temporal deep neural networks, for weather prediction, specifically maximum surface temperature prediction, in South Africa. The specific research objectives are to:

1. Compare and evaluate two basic temporal deep neural networks architectures, i.e. the LSTM and the TCN for temperature prediction across 21 weather stations in South Africa.
2. Compare the results to Graph WaveNet (GWN) and the Low-Rank Weighted Graph Neural Network (WGN).
3. Perform appropriate experiments to evaluate the stability and robustness of temporal and ST-GNN models i.e. LSTM, TCN, WGN, and GWN.
4. Analyse and compare the learnt adjacency matrices of the two ST-GNNs to gain insights into the prominent spatial-temporal dependencies between weather stations.

## 1.3 Methodology

This study is conducted on weather observations recorded at 22 weather stations found in the Western Cape and Northern Province in South Africa. These data sets each contain hourly recordings of six weather features from January 2012 through to December 2019. The data sets were obtained from the South African Weather Services.

Five models namely, SARIMA, LSTM, TCN, WGN, and GWN are implemented and compared for weather maximum surface temperature prediction in the Western Cape and Northern Province across five prediction horizons i.e. 3, 6, 9, 12, and 24 hours. The SARIMA method is a classical statistical approach, whereas the LSTM and TCN are temporal DNNs. These three models comprise the models against which the ST-GNNs will be evaluated. The WGN model is an ST-GNN model developed in the weather domain, whereas the GWN model is a model developed in the traffic flow domain. We build the ST-GNN models using code from the original implementation of the WGN and GWN models.

A SARIMA, LSTM, and TCN model were developed for each individual weather station for each of the five prediction horizons. The experimental design in which these models were developed is as follows. The data for the models underwent the pre-processing steps of missing data imputation, standardisation, and input-output processing. This processed data were then split into successive train, validation, and test sets through walk-forward validation. The temporal DNN models then underwent manual and random search hyper-parameter optimisation over the 24-hour prediction horizon. The optimal hyper-parameters identified on the 24-hour prediction horizon were then used to train the evaluate the models across the remaining four prediction horizons.

In contrast to the temporal DNN experimental design, the ST-GNN models are trained on data from all weather stations simultaneously. These ST-GNNs then also output predictions across all weather stations simultaneously. Therefore, ST-GNN models were only developed for each of the

five prediction horizons. The ST-GNN experimental design is similar to the temporal DNN design in that it underwent the same pre-processing steps, followed by walk-forward validation. The hyper-parameter optimisation process was the same, with the optimal parameters found on the 24-hour prediction horizon used to train and test the ST-GNN models across the remaining prediction horizons. Finally, the self-adaptive adjacency matrices for each ST-GNN model were extracted, analysed, and compared.

The models are evaluated using the walk-forward validation technique enabling the robustness of the temporal and ST-GNN models to be tested. To evaluate the stability of the models, each experiment is run 10 times with random seeds. The variance in the performance metric across the 10 runs determines the model's stability. The MAE, MSE, and SMAPE metrics were used to evaluate the performance of the models. In general, the performance of the models is reported as the mean and the standard deviation of the SMAPE values for each prediction horizon.

## 1.4 Importance and Contribution

To our knowledge, this is the first work that explores both temporal DNNs and ST-GNNs for weather prediction in South Africa. The study extends the evaluation of the WGN ST-GNN model developed in the weather domain to other geographical locations since the original study was confined to the continental United States. The study also provides a comparison of ST-GNN models developed in the weather domain to ST-GNN models developed in the traffic flow domain.

This study contributes additional evidence to the claim that ST-GNN models like GWN are generalisable to other spatial-temporal domains. In addition to this, this study further extends the exploration of GWN to other spatial-temporal domains i.e. the weather domain.

This study provides the first effective evaluation, via walk-forward validation, of temporal DNNs and ST-GNNs that measure the robustness and performance of these models when developed for dynamic and complex systems like the weather. Finally, the study provides a measure of the stability of these deep learning models through the repetition of experiments.

## 1.5 Chapter Outline

The remainder of this dissertation is structured as follows:

- Chapter 2 - **Literature Review**: This chapter provides a review of the current approaches for weather prediction, temporal deep learning models, and finally ST-GNN models for spatial-temporal prediction.
- Chapter 3 - **Experimental Design**: This chapter describes the data set, design of the experiments, and implementation of the temporal DNNs and ST-GNN models.
- Chapter 4 - **Results and Analysis**: This chapter presents the results produced after following the experimental design mentioned above and an analysis of the results.

- Chapter 5 - **Discussion**: This chapter discusses the aim and objectives, a comparison to related literature and the contributions made by this thesis.
- Chapter 6 - **Conclusion**: This section highlights the impact of the work, followed by limitations, and finally discusses the options for future work.

# Chapter 2

# Literature Review

The aim of the literature review is four-fold. The first is to provide the contextual background (weather) in which the deep learning models will be implemented for maximum temperature prediction, i.e. predicting the highest temperature at a place in a given time period. The second is to introduce various deep neural networks that have been used to predict temperature. We then present and explore spatial-temporal graph neural networks (ST-GNN) for temperature prediction and traffic flow prediction. Finally, we provide a summary of the key findings of the literature review.

## 2.1 Weather Prediction

Weather refers to the specific condition of the atmosphere at a particular place and time. It is measured in terms of features including wind speed and direction, air temperature, humidity, atmospheric pressure, cloudiness, and precipitation. Weather can change from hour to hour, day to day, and season to season [11]. The weather is a chaotic system that is affected by many other features e.g. vegetation, geographical contour, and human factors [12]. Weather features are also non-linear, where a change in one feature does not necessarily indicate a direct, proportional change in another variable [3].

### 2.1.1 Numerical Weather Prediction

Numerical weather prediction (NWP) is one of the most popular methods of weather prediction [3]. It utilises computer algorithms to provide a prediction based on current weather conditions by solving a large system of non-linear mathematical equations. These models define a coordinate system, which divides the earth into a 3-dimensional grid. The parameters measured within each grid and their interactions are used to predict atmospheric properties in the future.[3] These NWP methods can predict minutes to weeks ahead for regions with meter-to-kilometre resolutions.

The NWP approach is getting more complex resulting in a growing demand for computation power and also electricity consumption [12]. Further shortcomings of NWP include parameterization. This is a method in NWP that replaces complex or small-scale atmospheric processes with simplified processes. Parameterizations in NWP are imperfect and therefore reduce the accuracy of

predictions. Additional weaknesses of NWP include an insufficient resolution to resolve fine-scale weather and very high computational cost for operational prediction [13]. Deep learning offers an alternative approach to weather prediction which does not succumb to these weaknesses.

### 2.1.2 Classical Statistical Approach To Temperature Prediction

The SARIMA model is a classical statistical method that has been used to predict temperature with varying degrees of success [14, 15, 16]. SARIMA is a popular baseline model against which to compare novel deep learning approaches in both weather and traffic flow prediction.

Early components of the SARIMA model are the Autoregressive (AR) and the Moving Average (MA) models. Combining these two models resulted in the formulation of the ARIMA model. SARIMA is a generalized model of the original ARIMA model, used to deal with seasonality within time-series data [17] that the ARIMA models were not capable of modelling. SARIMA stands for Seasonal Autoregressive Integrated Moving Average.

In an AR($p$) model, the future value of a variable is assumed to be a linear combination of $p$ past observations and a random error together with a constant term. In the same fashion that an AR($p$) model regresses against past values of a series, an MA($q$) model uses past errors as the explanatory variables. The moving average model is a linear regression of the current observation of the time series against the random errors of one or more prior observations. These models were combined to create ARMA models. These models can only be used for stationary time series data. The ARIMA models were introduced as a generalization of the ARMA models to account for non-stationarity in time-series data. The ARIMA model was then modified into the SARIMA model to handle non-seasonal non-stationary data [17]. The mathematical formulation of a SARIMA$(p,d,q)$x$(P,D,Q)^s$ model is shown below in equation 2.1 below:

$$\phi_p(L^s)\varphi_p(L)(1-L)^d(1-L^s)^D y_t = \Theta_Q(L^s)\theta_q(L)\varepsilon_t \tag{2.1}$$

The $p, d$ and $q$ are integers greater than or equal to zero and refer to the order of the autoregressive, integrated, and moving average parts of the model respectively. The $P, D$ and $Q$ are the seasonal counterparts of the aforementioned variables. The integer $d$ controls the level of differencing, with the integer $D$ controlling the level of seasonal differencing [17].

Chen et al. [15] predict the monthly mean temperatures in Nanjing using the SARIMA model. They split the data into a train and test set, training on the train set and achieving a reasonably low MSE of 0.89 on the test set. Ayitey et al. [14] also predict the monthly temperature in Ghana following the same train, test split as Chen. Their model had an RMSE of 495.74 on the test set which they claimed to be an acceptable value. Dimri et al [16] predict monthly mean and maximum temperatures for the Bhagirathi river basin at two different weather stations. Again, the train-test split methodology was used. Dimri et al. achieved an acceptable mean RMSE of 1.43 and 1.41 at each weather station. However, they found that over-predictions were found in extreme tempera-

ture results.

The aforementioned papers all used the Box-Jenkins methodology for optimal model selection which is a time-inefficient method. This methodology however is not necessary as modern computational power allows us to search through a large number of randomly initialised SARIMA models using hyper-optimisation to find the optimal model.

## 2.2 Deep Neural Networks For Temperature Prediction

This section begins by providing a high-level overview of early deep neural networks used for temperature prediction. We then introduce two temporal deep learning models that have been successfully used for temperature prediction i.e. LSTM and TCN. We provide an in-depth mathematical formulation of the model along with the performance of these models for temperature prediction.

### 2.2.1 Early Deep Neural Networks For Temperature Prediction

An initial deep neural network (DNN) based method to temperature prediction developed by Hippert et al. [18] used a hybrid prediction system, consisting of a multi-layer perceptron neural network (MLPNN) and an autoregressive model. The hybrid model was trained on weather station data from Brazil, achieving a mean absolute percentage error (MAPE) of 2.82. This model was compared to linear models on the same data and the hybrid MLPNN model outperformed the linear models. The MLPNNs improved results are a result of the neural network's ability to model non-linear data. Cifuentes et al. [19] published a paper reviewing various machine learning techniques for air temperature prediction noting that the latest advances in deep learning have considerably improved the accuracy rates in temperature prediction. A comparative section on temperature prediction using hourly data reveals that certain Artificial Neural Networks (ANNs) [20, 21, 22] and Support Vector Machines (SVM) offer similar predictive accuracy. The ANNs mentioned in the study include Radial Basis Function Neural Networks [23, 22], Multi-Layer Perceptron Neural Networks [20, 22] and ensemble methods. The SVM approach is preferred over the aforementioned ANN approaches due to ease of implementation and similar prediction performance with a mean average error (MAE) ranging between 0.513°C and 2.303°C across different prediction horizons. The aforementioned ANNs are not capable of effectively capturing temporal and spatial dependencies [24, 9, 4]. Hewage et al. [3] produced the best prediction results over a 12-hour prediction range using an LSTM, capable of capturing temporal dependencies, with an MSE of 0.00017°K, outperforming the NWP model mentioned in section 2.3.

The advancements in deep learning strategies have led to effective deep learning architectures, capable of learning temporal and spatial dependencies. The following subsections delve deeper into the LSTM and TCN architectures.

### 2.2.2 Long Short-Term Memory Networks

Long short-term memory (LSTM) networks, a temporal model, are a specialised form of Recurrent Neural Networks (RNN) with the ability to capture both short and long-term temporal dependen-

cies [3]. An LSTM cell, as shown in figure 2.1, works by utilising three types of gating mechanisms in combination with information from the current and previous time steps. These gating mechanisms control the flow and capture of sequential data through the internal memory found in each cell. These networks are capable of learning temporal dependencies and do not suffer from the vanishing gradient problem that early temporal models such as Recurrent Neural Networks could not overcome [9].

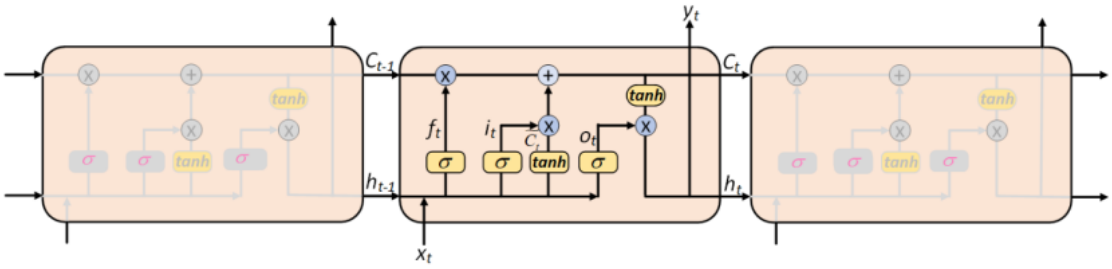


Figure 2.1: LSTM Cell (copied from [25])

The gates, represented by rectangles containing $\sigma$ are the forget, input, and output gates. The first gate is the forget gate which determines the extent to which the previous input, $h_{(t-1)}$, and current input, $x_t$, affects the output of the cell.

$$F_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.2}$$

$\sigma$ is the sigmoid activation function, $W_f$ is the weight of the forget gate, $h_{t-1}$ is the hidden state output at time $t-1$, $x_t$ is the current input at time $t$, and $b_f$ is the bias of the forget gate. The output of this gate is between 0 and 1, if the $f_t$ value is close to 0 most of the information from the previous cell is removed, whereas if the output is close to 1, the information is passed through the gate [25].

The next gate is the input gate. This gate, in combination with the intermediate cell state, determines what information from the current input should be added to the cell state (long-term memory).

$$I_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.3}$$

The input gate also uses a sigmoid activation function, with $W_i$ being the weight of the input gate, and $b_i$ the bias of the input gate. The value of the input gate is then combined through the multiplication operation with the intermediate cell state ($\tilde{C}_t$) to obtain the new cell state [25].

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.4}$$

Equation 2.4, $\tilde{C}_t$, is the intermediate state of the cell with a *tanh* activation function. The previous cell state ($C_{t-1}$) is multiplied by the forget gate. The result of this multiplication is then added to the product of the intermediate cell state and input gate output as mentioned above. These operations yield the new cell state as shown in equation 2.5 [25]:

$$C_t = (i_t \tilde{C}_t + f_t \times C_{t-1}) \tag{2.5}$$

The last gate is the output gate which determines how much information of that cell is output.

$$O_t = \sigma(W_o \bullet [h_{t-1}, x_t] + b_o) \tag{2.6}$$

The cell state with *tanh* $c_t$, calculated using the forget, input gate and intermediate cell state is then multiplied by the value of the output gate which becomes the hidden state output, $h_t$. This is shown in equation 2.7 [25]:

$$h_t = o_t \times tanh(c_t) \tag{2.7}$$

Finally, the input, output, and forget gates have independent weights and bias allowing the network to learn how much of the last output must remain stored, how much of the current input must be stored, and how much of the cell state must be sent to the output [25].

Bidirectional LSTMs are a variation of LSTMs which includes two parallel LSTM tracks defined by forward and backward loops. These forwards and backwards loops extract patterns from the past and the future, in order to better model time dependency [26].

Zahroh et al. [25] used an LSTM to predict the daily minimum and maximum temperature in Bandung. The data was split into an 80:20 ratio between training and test sets. They achieved the lowest MAPE scores of 0.0225 and 0.015 for minimum and maximum temperature prediction respectively. The LSTM model that achieved these results consisted of a single hidden layer with 4 neurons, trained over 50 epochs.

Zaytar et al. [27] developed two LSTM models to predict hourly temperature data 24 and 72 hours into the future across nine different cities in Morocco. The data was split into train, validation and test sets. The models consisted of 2 LSTM layers and a fully connected hidden layer with 100 neurons. The models were trained and validated on the train and validation sets, achieving an average MSE of 0.0068 and 0.009 over the 24 and 72-hour prediction horizon respectively.

Hewage et al. [3] developed two separate LSTM models. The first model was a multi-input multi-output(MIMO) model, and the second was a multi-input single-output model(MISO). Hewage et al. split the data into a train, validation and test set. They performed hyper-parameter optimisation on the MIMO-LSTM and MISO-LSTM models, varying the number of LSTM layers and nodes within each layer. The final proposed models consisted of 3 LSTM layers consisting of 128, 512, and 256 nodes respectively.

The results between the MISO-LSTM and MIMO-LSTM were found to be similar, and because the MIMO-LSTM predicts all weather parameters simultaneously, whereas the MISO-LSTM requires an LSTM for each parameter, the MIMO-LSTM was said to be more ideal for weather prediction. Hewage et al. trained MIMO-LSTM models across four different prediction horizons: 3, 9, 24, and 48 hours. Using the MIMO-LSTM models, three different strategies were used for prediction across the prediction horizons:

1. Train the MIMO-LSTM on the 3-hour prediction horizon, then load these weights onto other models and fine-tune them to predict the remaining prediction horizons.

2. Train models for each prediction horizon without loading model weights.

3. Train a Bidirectional-LSTM for each prediction horizon.

The bidirectional LSTM achieved the best results over the 9, 24 and 48-hour horizon with an overall MSE of 0.014, 0.034 and 0.035 respectively. The standard MIMO-LSTM trained using the 2nd strategy in the list above achieved the lowest performance on the 3-hour prediction horizon with an overall MSE of 0.006399. Overall, the results between the Bidirectional-LSTM and MIMO-LSTM trained using strategy two were similar. The time taken to train, test, and predict the Bidirectional-LSTM is significantly greater than that of the MIMO-LSTM trained using the 2nd strategy. The MIMO-LSTM was therefore identified as the effective and efficient method to train the LSTM models.

### 2.2.3 Temporal Convolutional Network

Although LSTMs have presented a better alternative to RNNs, Bai et al. [28] noted that temporal convolutional neural networks have outperformed LSTMs across a diverse range of tasks. Temporal Convolutional Networks (TCN) [3] is another temporal model that takes in a sequence of inputs of length $L$ and predicts the output, whose length $l$ is equal to that of the input sequence. The properties of the TCN are twofold, the first is that the output of the network should have the same length as its input. The second property is that the network can only use the information from past time steps. This network uses a 1D fully-convolutional network (FCN) architecture with causal convolution to fulfil the aforementioned properties and to capture temporal trends [28]. The 1D-FCN uses zero padding to keep subsequent layers the same length as previous ones, adhering to the first property. The second key feature of TCNs is causal convolutions. Causal convolutions uphold the second property of TCNs by ensuring that convolutions at time $\mathbf{t}$ are convolved only with elements from time $t$ and earlier in the previous layer. [28]

The TCN architecture described above requires a deep network or large filters to have a large receptive field. Dilated convolutions and residual connections allow for large receptive fields and a very deep network. [28] Dilated convolutions, as shown in figure 2.2, is a convolution where the filter is applied over a region larger than its size by skipping input values with a given step [29]. Stacking dilated causal convolution layers with dilation factors in increasing order allows for an exponentially growing receptive field. The receptive field is the region of input that a neuron or unit within the network is exposed to. A larger receptive field allows the network to take in and capture longer temporal sequences and dependencies with fewer layers, saving computation resources. [3] [5] Residual connections utilise skip connections that speeds up the training process and avoids vanishing gradient problem in deep models. [29]

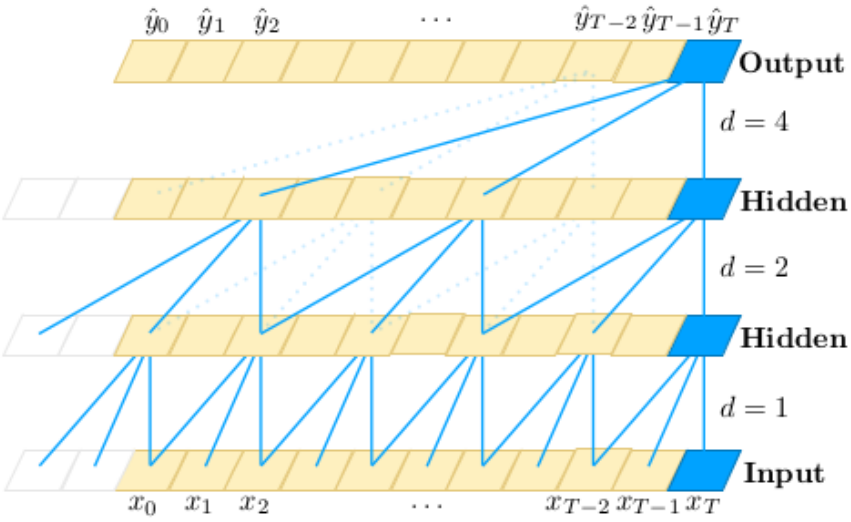


Figure 2.2: Dilated Convolution (copied from [3])

Mathematically, given a 1D sequence input $\mathbf{x} \epsilon \mathbf{R}^T$ and a filter $\mathbf{f} \epsilon \mathbf{R}^k$, the dilated causal convolution operation of $\mathbf{x}$ with $\mathbf{f}$ at step $t$ is represented below in equation 2.8 as:

$$\mathbf{x} \star \mathbf{f}(t) = \sum_{s=0}^{K-1} \mathbf{f}(s)\mathbf{x}(t - d \times s) \tag{2.8}$$

Hewage et al. [3], implemented a TCN in addition to the LSTM models discussed in section 2.2.2. Similarly to the LSTMs, MISO-TCN and MIMO-TCN models were developed. The MIMO-TCN and MISO-TCN were implemented with different configurations, varying the filter sizes, number of TCN layers and activation functions to find the optimal hyper-parameters. On the MISO models, surprisingly, the MISO-LSTM outperformed the MIMO-TCN on 6 of the 10 weather parameters. The MIMO-LSTM also outperformed the MISO-TCN on 6 of the 10 weather parameters.

## 2.3 Spatial-Temporal Graph Neural Networks

This section begins by introducing spatial-temporal modelling and the challenges it poses. This is followed by an explanation of graph convolution and adjacency matrices. We then explore the spatial-temporal graph neural networks in the weather and traffic flow domain that will be used in this project to predict temperature.

### 2.3.1 Spatial-Temporal Modelling

Space and time are ubiquitous aspects of observations within the weather domain. Observations and predictions are recorded and predicted at a specific time and location. This type of data is referred to as spatial-temporal (ST) data. ST prediction poses myriad problems, most of which

relate to effectively capturing the spatial and temporal features and dependencies found within the data.

Spatial dependency is defined as the property of random variables taking values, at pairs of locations a certain distance apart, that are more similar (positive autocorrelation) or less similar (negative autocorrelation) than expected for randomly associated pairs of observations [30]. The dependency within the context of this research is the spatial dependency between weather stations. The spatial dependencies between weather stations are affected by spatial features, which are features found in the spatial realm. These include, but are not limited to, distance, topology, jet streams and prevailing winds. [9]

Temporal dependency can be thought of in a similar fashion to spatial dependency, which is the property of random variables taking values, at pairs of temporal values at certain intervals apart, that are more similar or less similar than expected for randomly associated pairs of observations [30]. Within the context of weather, temporal dependencies can be understood as the impact of weather features recorded at earlier time steps on weather features further into the future. When predicting the weather, it might be assumed that recent weather data is more useful when predicting within the next 24 hours, but the prediction is also dependent on information that is significantly older than the recent window. This can be understood as temporal dependency and these dependencies are both long and short-term [27]. Temporal features relate to factors affecting temporal dependencies such as the seasons and climate change.

Capturing these dependencies presents unique modelling challenges. The training data needs to be represented in a way that explicitly expresses spatial dependencies so that the model can learn these dependencies. Spatial and temporal dependencies are not static, for example, the relationship between the stations changes over time and this means that models need a dynamic method of representing and capturing dependencies [24]. Models need to be robust and computationally cheap in order to capture both the long and short-term temporal dependencies [5]. These models also need to have some degree of explainability which will assist in the exploration and understanding of weather phenomena. Finally, models need a structure which simultaneously captures spatial and temporal dependencies, allowing them to capture spatial-temporal dependencies. Successfully tackling the aforementioned challenges is key to accurate, reliable predictions.

### 2.3.2 Graph Convolution and Adjacency Matrices

A Convolutional Neural Network (CNN) is a deep learning approach for modelling spatial relationships in data. CNNs are capable of learning representations of data by taking linear combinations of each data point with its neighbours. CNNs have shown to be extremely effective at learning spatial relationships within gridded data sets [9]. Convolutional neural networks consist of several building blocks of which the most important is the convolutional layer.

The convolutional layer performs feature extraction through convolutional operations and activation functions. Convolution, as shown in figure 2.3, is a specialized type of linear operation used for feature extraction, where a small array of numbers, called a kernel, is applied across the input, which is an array of numbers, called a tensor. An element-wise product between each element of

the kernel and the input tensor is calculated at each location of the tensor and summed to obtain the output value in the corresponding position of the output tensor, called a feature map. This procedure is repeated, applying multiple kernels to form an arbitrary number of feature maps, which represent different characteristics of the input tensors; different kernels can thus be considered as different feature extractors. [31]

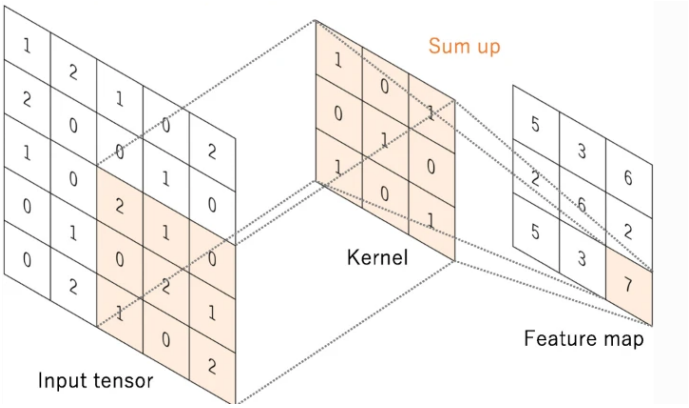


Figure 2.3: Convolution Operation

The standard convolution operation which works on gridded data is not applicable to the weather station data used in this project. As shown in figure 2.4 below, the blue pins represent the weather stations where observations are recorded, if the observations were represented in a grid format and the standard convolution operation applied, only a few cells of the grid would have values, and the rest would be empty. This would result in the neural network convolving over meaningless, empty cells, effectively capturing no spatial dependencies.

Graph convolution operations offer a generalised approach to capturing spatial dependencies in weather data represented in the form of graphs. The weather station network can be modelled as a graph, with the nodes of the graph, $V$, representing weather stations and $E$ being the set of edges between the nodes. The goal of graph neural networks is to generate a node's representation by aggregating its own features and its neighbours' features through convolution. The graph of weather stations can be represented in a matrix containing each weather station's feature vector at each time step $t$. The neighbourhood of a weather station, $v$, is the set of weather stations $N$ that are connected to $v$. An adjacency matrix is used to indicate if weather stations have connected edges [32] and the strength of the relationship between these stations. The adjacency matrix is key to accurate predictions as it determines the size of the neighbourhood, or receptive field, that the graph convolution operates over [24]. Linking back to figure 2.3, the adjacency matrix would indicate which values from the input tensor to include in the convolution operation. In our context, the adjacency matrix indicates which weather station's data should be used for graph convolution when making predictions. An accurate adjacency matrix is also crucial to graph convolution because if it indicates that no spatial dependency exists between two weather stations, the prediction accuracy will be further reduced. A further challenge of adjacency matrices is that the spatial dependencies between stations are not static, this requires a dynamic approach to determining the adjacency

matrix.

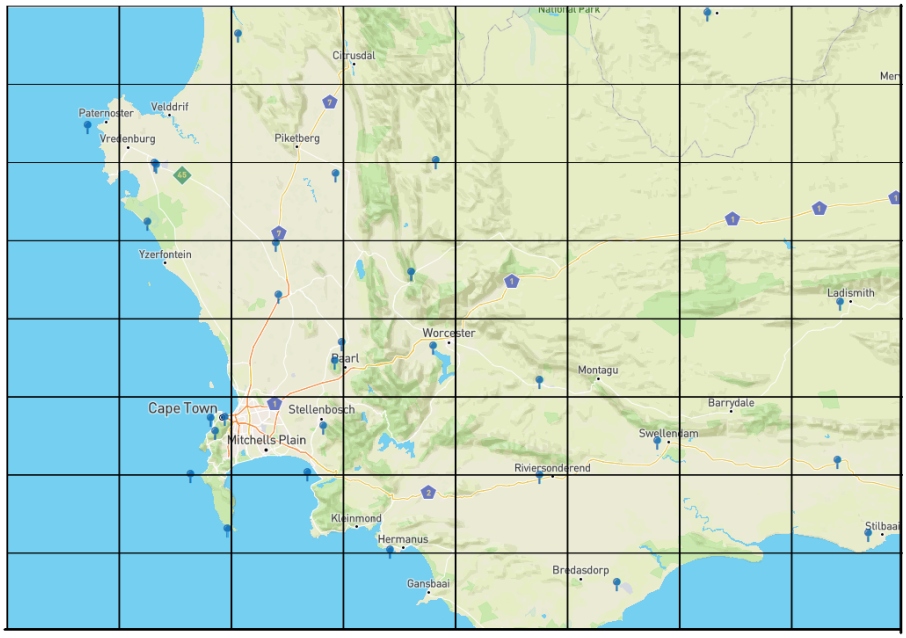


Figure 2.4: Gridded Weather Map

The LSTM and TCN models described above, in addition to graph convolution, form the building blocks for the Spatial-Temporal Graph Neural Networks (ST-GNNs). These ST-GNN models are capable of capturing both spatial and temporal dependencies. The remaining sections provide an overview of the ST-GNN models developed in the weather and traffic flow domain.

### 2.3.3 Low-Rank Weighted Graph Convolutional Network (WGN)

Our literature review revealed that only one ST-GNN model has been explored for weather prediction. The following section describes the aforementioned ST-GNN model implemented by Wilson et al. [9] for weather prediction in the continental United States.

Wilson et al. [9] proposed a Low-Rank Weighted Graph Convolutional (WGN) ST-GNN approach to weather prediction by combining LSTMs and graph convolution. Initially training an LSTM on historical records, they found the model predicted temperature values that are similar to the temperature in the previous time step. The model was unable to anticipate rapid changes in the temperature time series. Upon further analysis, it was found that the rapid changes in temperature at one location tended to be proceeded by large temperature changes at nearby locations in the previous time step. This information reveals that capturing spatial dependencies is crucial to accurate predictions and that previous temporal models were unable to incorporate and capture

spatial information.

Wilson et al. built a spatial-temporal weather prediction model that is able to capture both spatial and temporal dependencies. The temporal building block of this GNN was the LSTM network. Wilson et al [9] noted that spatial dependencies can be influenced by spatial features besides geographical distance such as topography, prevailing winds and jet streams. Wilson et al. therefore recognised the need for a dynamic adjacency matrix. The WGN model treated the adjacency matrix as a model parameter that can be learned from the data. This allows the model to capture dynamic spatial dependencies. The general network structure, as seen in figure 2.5, shows fully connected layers within each LSTM cell replaced with graph convolutional (GCN) layers. The LSTM provided the network structure into which the spatial model of a GCN was fused, enabling the network to simultaneously capture temporal and spatial dependencies.

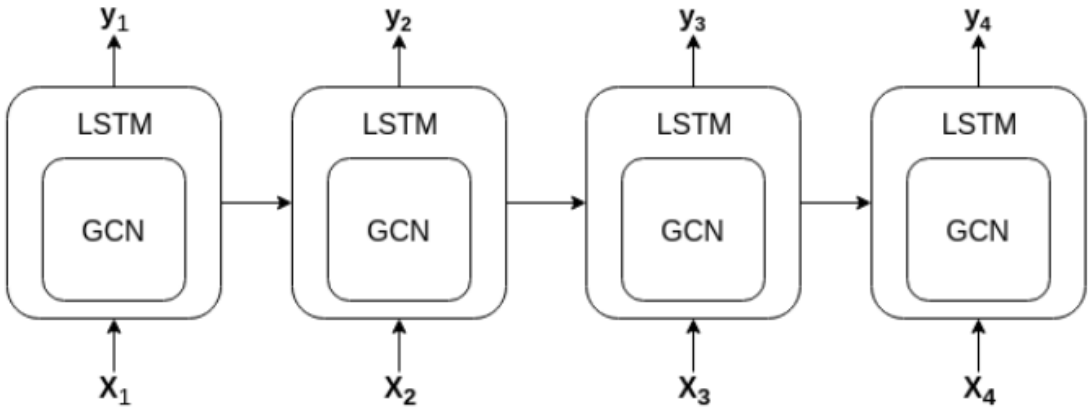


Figure 2.5: WGN Network Structure (copied from [9])

The WGN model takes a sequence of $T$ weather observations at $\mathbf{V}$ locations i.e $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$ where each $\mathbf{X}_t \in \mathbb{R}^{|V| \times d}$. The $d$ features are the weather variables such as pressure, humidity, and wind speed. The goal of this paper was, given an input sequence of historical observations, to predict the target sequence $y_1$, $y_2$, ..., $y_T$ where $\mathbf{y}_t \in \mathbb{R}^{|V|}$ is a vector of the targeted weather variable to be predicted at time $t$ for all $|V|$ locations [9].

Wilson et al. utilised the following form of graph convolution. For a graph convolution neural network with $L$ layers, the output of each layer, $H^{(l)}$, is a $|V|$ x $d^l$ matrix with each row representing one of the vertices (weather stations). The number of vertices in each matrix stays the same, only the dimension of the vertex representation $d^l$ changes [9]. The graph convolution of a set of vertices, $\mathbf{H}^{l-1}$ with a matrix of weights, $\mathbf{W}^l$ is defined as:

$$\mathbf{H}^{l-1} * \mathbf{W}^l = \mathbf{A}\mathbf{H}^{l-1}\mathbf{W}^l \tag{2.9}$$

where $\mathbf{A}$ is the adjacency matrix associated with the graph and * represents graph convolution.
. The product $\mathbf{A}\mathbf{H}_t^{l-1}$ sums up features in the neighbourhood around each weather station, while

right multiplying this product with $\mathbb{W}^l$ allows the network to consider linear combinations of features [9].

The graph convolution can then be utilized in each layer of the neural network, the operations performed in each graph convolutional layer can be described as:

$$\begin{aligned}\mathbf{H}^l &= f^l(\mathbf{H}^{l-1}) \\ &= \sigma(\mathbf{H}^{l-1} * \mathbf{W}^l) \\ &= \sigma(\mathbf{A}\mathbf{H}^{l-1}\mathbf{W}^l)\end{aligned} \tag{2.10}$$

Wilson et al. combined the LSTM cell and graph convolutions by inserting the graph convolution operation described in equation 2.10 into the LSTM cell. Since graph convolutions are capable of capturing spatial dependencies, and the LSTM cell is capable of capturing temporal dependencies, the WGN architecture is able to simultaneously capture spatial and temporal dependencies. The modified LSTM equations from section 1.3.1 are:

$$\begin{aligned}\mathbf{C}_t^{(l)} =& \mathbf{I}_t \circ \tilde{\mathbf{C}}_t^{(l)} + \mathbf{F}_t^{(l)} \circ \mathbf{C}_{t-1}^{(l)} \\ \mathbf{F}_t^{(l)} =& \sigma\left(\mathbf{H}_t^{(l-1)} * \mathbf{W}_f^{(l)} + \mathbf{H}_{t-1}^{(l)} * \mathbf{U}_f^{(l)} + \mathbf{b}_f^{(l)}\right) \\ \mathbf{I}_t^{(l)} =& \sigma\left(\mathbf{H}_t^{(l-1)} * \mathbf{W}_i^{(l)} + \mathbf{H}_{t-1}^{(l)} * \mathbf{U}_i^{(l)} + \mathbf{b}_i^{(l)}\right) \\ \tilde{\mathbf{C}}_t^{(l)} =& \tanh\left(\mathbf{H}_t^{(l-1)} * \mathbf{W}_c^{(l)} + \mathbf{H}_{t-1}^{(l)} * \mathbf{U}_c^{(l)} + \mathbf{b}_c^{(l)}\right) \\ \mathbf{H}_t^{(l)} =& \mathbf{O}_t^{(l)} \circ \tanh\left(\mathbf{C}_t^{(l)}\right) \\ \mathbf{O}_t^{(l)} =& \sigma\left(\mathbf{H}_t^{(l-1)} * \mathbf{W}_o^{(l)} + \mathbf{H}_{t-1}^{(l)} * \mathbf{U}_o^{(l)}\right. \\ & \left. + \mathbf{C}_t^{(l)} * \mathbf{V}_o^{(l)} + \mathbf{b}_o^{(l)}\right)\end{aligned} \tag{2.11}$$

This paper used two-real world data sets with measurements taken from locations within the United States. Weather stations are only included in the data set if only less than 10% of their data is missing. If data is missing, the data is linearly interpolated. This paper aimed to predict either temperature or wind speed twelve hours into the future based on historical weather measurements. The model was trained on the first 10,000 time steps, with the next 2,300 used as the validation set, and the remaining 2,300 time steps as the test set.

The WGN model was compared to four baselines: Gaussian Process, GCN, Local LSTM and Global LSTM. On predicting temperature, this model outperformed the baselines at 99.7%, 97.6%, 90.4%, and 99.4% of weather stations respectively. Outperforming the baselines in wind speed prediction at 100.0%, 97.3%, 93.1% and 96.4% of weather stations. The results of this graph neural network show that capturing both spatial and temporal dependencies leads to increased prediction accuracy over solely temporal or spatial models.

The WGN achieved near-optimal performance with only 2 layers on the first data set whereas the second data set's performance increased as more layers were added. The addition of more layers into the network allows the model to incorporate weather information from larger areas.

### 2.3.4 Spatial Dependency Visualisation

Visualising the spatial dependencies between weather stations enables us to compare current knowledge of weather systems with those found by graph neural networks. This visualisation can also provide insight into previously unknown weather phenomena.

Wilson et al. visualized the spatial dependency between weather stations, as shown in figure 2.6, by plotting the most important weights within the learnt adjacency matrix. This was done by taking the absolute value of the learned adjacency matrix and finding the largest edge weight in each row of the adjacency matrix. The edge corresponding to each row's largest weight is then plotted along with its direction. In cases where the pairs of weather stations are spatially dependent on each other, the spatial dependency between these stations is represented using a purple line. [9]

Figure 2.6: Spatial Dependency Visualisation (copied from [9])

### 2.3.5 ST-GNNs For Traffic Flow Prediction

Spatial-temporal prediction is a task that is currently being tackled in the traffic flow domain. Deep neural networks [33, 5, 4] that have been implemented in this domain face similar spatial-temporal prediction challenges, with some models being generalisable across different ST domains [4]. There are numerous similarities between traffic flow and the weather domain. The first is that the observations in both domains are captured at fixed locations in fixed intervals in time. This leads to the spatial-temporal prediction challenge of needing to represent the data in a way that captures spatial features other than geographical distance. The spatial dependencies in the traffic flow domain are

also dynamic and vary over different temporal levels. Traffic flow prediction also requires temporal dependencies to be captured, also facing the challenge of how to capture long-term temporal dependencies and take in long input sequences without high computation costs and running into the vanishing gradient problem.

As seen above, the challenges faced in both the traffic flow and weather domains are similar, with minor domain-specific challenges. As a result of the similarities between the domains, ST-GNNs in the traffic flow domain have been analysed to determine their potential application within the weather domain. Spatial-temporal prediction in the traffic flow domain using ST-GNNs is significantly more advanced than those found in the weather domain. Since spatial-temporal prediction in both domains tackles similar challenges, an ST-GNN developed for traffic flow prediction was chosen to tackle the weather prediction problem in addition to the WGN model.

The Spatial-Temporal Graph Convolution Neural Network (ST-GNN) [4] model was the first graph neural network used in the traffic flow domain, using the adjacency matrix to represent spatial dependencies. This model was limited to static spatial dependencies and short-range dependencies. The ST-GNN model showed state-of-the-art performance over previous temporal methods in the urban flow domain [4]. This model however was not able to capture dynamic spatial dependencies [24]. To overcome ST-GNN's limitation of capturing only static dependencies, the Dynamic Graph Convolutional Neural Network (DGNN) [24] model was developed which used a dynamic adjacency matrix, allowing the model to successfully capture changing spatial dependencies. The limitation of this model included an inability to capture long-range temporal dependencies and discover hidden spatial dependencies [24]. An attention-based ST-GNN (ABGNN) [33] was then proposed which also addressed the issue of a static adjacency matrix in the ST-GNN model and overcame the DGNN's inability to capture long-range dependencies, this model, however, was not able to tackle the issue of discovering hidden spatial links [33]. Table 2.1 below provides a comparison between the aforementioned ST-GNN models from the traffic flow domain.

| Traffic Flow Graph Neural Networks | | | | |
|---|---|---|---|---|
| Graph Neural Network Properties | **ST-GNN** | **DGNN** | **ABGNN** | **GWN** |
| *Spatial Dependency Capturing Ability* | Low | Medium | High | High |
| *Spatial Dependency Capturing Component* | GCN | GCN | GCN | GCN |
| *Dynamic Spatial Dependencies* | No | Yes | Yes | Yes |
| *Capture Missing Spatial Dependencies* | No | No | No | Yes |
| *Temporal Dependency Capturing Ability* | Low | Medium | High | High |
| *Temporal Dependency Capturing* Component | Gated CNN | Gated CNN | TCN | TCN |
| *Short/Long Range Temporal Dependencies* | Short | Short | Both | Both |
| *Computational Complexity* | Low | Medium | High | Medium |
| *Potential For Application in Weather Domain* | Low | Medium | High | High |

Table 2.1: Table comparing graph neural networks from the traffic flow domain

As seen in table 2.1, the two standout GNNs are the Attention Based GNN and Graph WaveNet. The ABGNN and Graph WaveNet each have a strong ability to capture spatial dependencies through the use of attention mechanisms and adaptive adjacency matrices respectively. Both models are able to capture dynamic spatial dependencies through a dynamic adjacency matrix. Each

model makes use of TCNs to capture temporal dependencies. TCNs are computationally faster and highly parallel which makes them more appealing than traditional LSTMs. Furthermore, TCNs have been shown to outperform LSTMs in a wide variety of domains. TCNs are the superior method of capturing temporal dependencies when compared to gated CNNs and LSTMs. The inclusion of TCNs in the ABGNN and Graph WaveNet architecture enables the models to capture both short and long-term temporal dependencies. Both models have a high potential for application in the weather domain. A significant difference between the ABGNN and Graph WaveNet model aside from the use of attention mechanisms in capturing spatial dependencies is the computational complexity. Graph WaveNet is less computationally complex than the ABGNN. Graph WaveNet has also been shown to outperform SOTA ST-GNNs developed after GWN i.e. MTGNN [7] and StemGNN [8] in other domains like stock price prediction [6]. Graph WaveNet was therefore chosen as one of the GNNs that will be implemented.

### 2.3.6 Graph WaveNet

Graph WaveNet (GWN) [5] is an ST-GNN used in traffic flow prediction. This GNN differs from WGN in its method to capture spatial and temporal dependencies and network structure. The GWN model can capture long-range temporal sequences and find missing spatial dependencies between nodes in the graph. Spatial dependencies are captured using a GCN with a self-adaptive adjacency matrix while temporal dependencies are captured using a TCN, instead of an LSTM as used in the WGN model.

Wu et al. [5] represented a graph by $G = (V, E)$ where $V$ is the set of nodes and $E$ is the set of edges. The adjacency matrix representing the spatial dependencies of the graph is denoted by $\mathbf{A} \in \mathbb{R}^{N \times N}$. Given a graph $G$, Wu et al. [5] used its historical $S$ step graph signals to learn a function $f$ that is capable of predicting its next $T$ step graph signals.

This paper proposed a self-adaptive adjacency matrix, built on modified versions of Kipf et al's. [34] spectral graph convolution, and Li et al's [35] diffusion convolution layer. Kipf et al. [34] proposed a first approximation of Chebyshev spectral filtering. This method smoothed a node's signal by aggregating and transforming its neighbourhood information. The advantages of this method are that is a compositional layer with a filter localized in space, and it supports multi-dimensional inputs [5]. Let $\tilde{\mathbf{A}} \in \mathbb{R}^{N \times N}$ represent the normalized adjacency matrix with self loops, $\mathbf{X} \in \mathbb{R}^{N \times D}$ represent the the input signals, $\mathbf{Z} \in \mathbb{R}^{N \times M}$ represent the output, and $\mathbf{W} \in \mathbb{R}^{D \times M}$ denote the model parameter matrix, Kipf et al [34] defined the graph convolutional layer as:

$$\mathbf{Z} = \tilde{\mathbf{A}}\mathbf{X}\mathbf{W} \tag{2.12}$$

Li et al. [35] then proposed a diffusion convolution layer which proved to be effective in spatial-temporal modelling. Graph diffusion convolution smooths out the neighbourhood over the graph, this process can be thought of as a denoising filter. This assists with graph learning because both features and edges in real graphs are often noisy [36]. The diffusion process of graph signals was modelled with $K$ finite steps. Wu et al. [5] generalized the diffusion convolution layer into the following form:

$$\mathbf{Z} = \sum_{\mathbf{k=0}}^{\mathbf{K}} \mathbf{P^k X W_k} \tag{2.13}$$

where $\mathbf{P^k}$ represents the power series of the transition matrix. In the case of a directed graph, the diffusion process has two directions, forwards and backwards, where the forward transition matrix $\mathbf{P}_f = \mathbf{A}/rowsum(\mathbf{A})$ and the backwards transition matrix $\mathbf{P}_b = \mathbf{A^T}/rowsum(\mathbf{A^T})$, resulting in the diffusion graph convolution layer [5]:

$$\mathbf{Z} = \sum_{\mathbf{k=0}}^{\mathbf{K}} \mathbf{P_f^k X W_{k1}} + \mathbf{P_b^k X W_{k2}} \tag{2.14}$$

In the Graph WaveNet model, Wu et al. [5] proposed a self-adaptive adjacency matrix $\tilde{\mathbf{A}}_{adp}$ that does not require any prior knowledge and is learned end-to-end though stochastic gradient descent. This allows the model to discover hidden spatial dependencies by itself. This is achieved by randomly initializing two node embedding dictionaries with learnable parameters $E_1, E_2 \in \mathbb{R}^{Nxc}$. Equation 2.15 describes the self-adaptive adjacency matrix as [5]:

$$\tilde{\mathbf{A}}_{adp} = SoftMax(ReLU(\mathbf{E}_1\mathbf{E}_2^T)) \tag{2.15}$$

where $\mathbf{E}_1$ is the source node embedding and $\mathbf{E}_2$ is the target node embedding. Multiplying $\mathbf{E}_1$ and $\mathbf{E}_2$ derives the spatial dependency weights between the source nodes and target nodes. The ReLU activation function is used to eliminate weak connections and the SoftMax function is applied to normalize the self-adaptive adjacency matrix [5]. Combining the pre-defined spatial dependencies and self-learned hidden graph dependencies, Wu et al. [5] proposed the following graph convolution layer shown in equation 2.16 below:

$$\mathbf{Z} = \sum_{\mathbf{k=0}}^{\mathbf{K}} \mathbf{P_f^k X W_{k1}} + \mathbf{P_b^k X W_{k2}} + \tilde{\mathbf{A}}_{adp}^k \mathbf{X W}_{k3} \tag{2.16}$$

When the graph structure is unavailable, Wu et al. [5] uses the self-adaptive adjacency matrix alone to capture hidden spatial dependencies, which is equation 2.17 [5] shown below:

$$\mathbf{Z} = \sum_{\mathbf{k=0}}^{\mathbf{K}} \tilde{\mathbf{A}}_{adp}^k \mathbf{X W}_k \tag{2.17}$$

A spatial-temporal layer is made up of the TCN and graph convolutional layer described above. These layers are stacked according to different temporal levels. For example, the bottom layer will receive short-term temporal information which would then result in capturing short-term spatial dependencies. Through the stacked architecture, as shown in figure 2.7, the network is able to capture different spatial dependencies at different temporal levels.

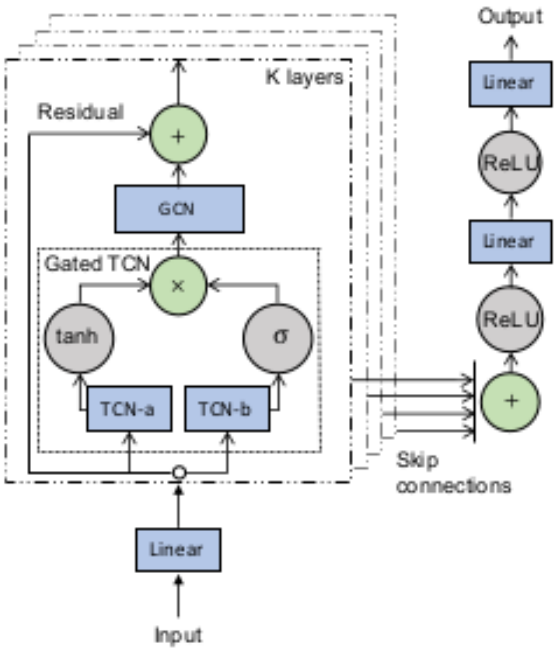


Figure 2.7: Graph WaveNet Structure (copied from [5])

The data sets used in this paper consisted of two public traffic network data sets. The intervals between measurements are 5 minutes. The data set is split in chronological order with 70% for training, 10% for validation and 20% for testing. The validation method applied to the WGN and Graph WaveNet models is not optimal for building robust, reliable models. This approach does not indicate if the model would have similar accuracy in other periods which may have different dynamics to the final time period used for testing.

Three metrics were used to evaluate the performance of the models: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE). Graph WaveNet was compared to six different baselines: ARIMA, LSTM, WaveNet, Diffusion Convolution Recurrent Neural Network (DCRNN), Graph Gated Recurrent Unit Network (GGRU) and STGCN. The models were compared over three different future prediction windows, 15, 30 and 60 minutes. The Graph WaveNet model outperformed the temporal and graph neural network baseline models with RMSE values of 5.15, 6.22, and 7.37 in each future prediction window respectively. Graph WaveNet was able to successfully learn hidden spatial dependencies automatically from the data, achieving state-of-the-art results in urban traffic flow prediction.

The adjacency matrix was plotting a correlation heat map of the final learned adaptive adjacency matrix. This visualisation is significantly harder to understand than the visualisation approach taken by Wilson et al. [9]. Furthermore, Wilson et al. overlayed the adjacency matrix over the

geographical map where the nodes in the graph were found. This allowed for easy visual extrapolation of spatial relationships when viewed geographically.

There was no evidence that either Graph WaveNet or the WGN model were evaluated for model stability or robustness. Random initialisation and other random parameters could result in different results when re-run with the same hyper-parameter space [37]. The models implemented in this project will be tested for model stability and robustness to ensure stable GNN configurations.

## 2.4 Summary

The SARIMA approach provides an adequate baseline against which the temporal deep neural networks can be compared to. We have chosen to implement the LSTM and TCN as our temporal deep neural networks due its state-of-the-art performance for temperature prediction. The WGN model has been identified as the only ST-GNN model that has been developed and implemented in the weather domain. We thus choose to implement the WGN as the first of our two ST-GNNs. Finally, the GWN model was identified as the best model from the traffic flow domain to predict the temperature in the weather domain. The following chapter outlines the experimental design in which all of the aforementioned models will be implemented.

# Chapter 3

# Experimental Design

This chapter describes the temperature prediction process and the experiments performed to carry out the objectives set out in section 1.3. It begins by providing an overview of the experimental process in section 3.1. The data sets are then described in section 3.2. Then, the pre-processing steps are described in section 3.3, followed by the partitioning of the data in section 3.4. This is then followed by describing how the models are evaluated in section 3.4.2. The process of hyper-parameter optimisation is then explained in section 3.5. The design of the temporal deep neural networks and ST-GNNs are described in section 3.6 along with the metrics with which the models are evaluated. Finally, the extraction and visualisation methods applied to the spatial-temporal dependencies are described in section 3.8.

## 3.1 Overview

Figure 3.1 provides a broad overview of the experimental design followed to predict temperature. The time series data collected from the weather stations are analysed and then go through a series of pre-processing techniques. Following this, the data are then partitioned using the sliding-window technique for the appropriate models. The partitioned data is then used to perform hyper-parameter optimisation using a combination of methods consisting of manual tuning and random search. Following hyper-parameter optimisation, the best hyper-parameters are selected and then used to train models to predict the temperature over the prediction horizons.

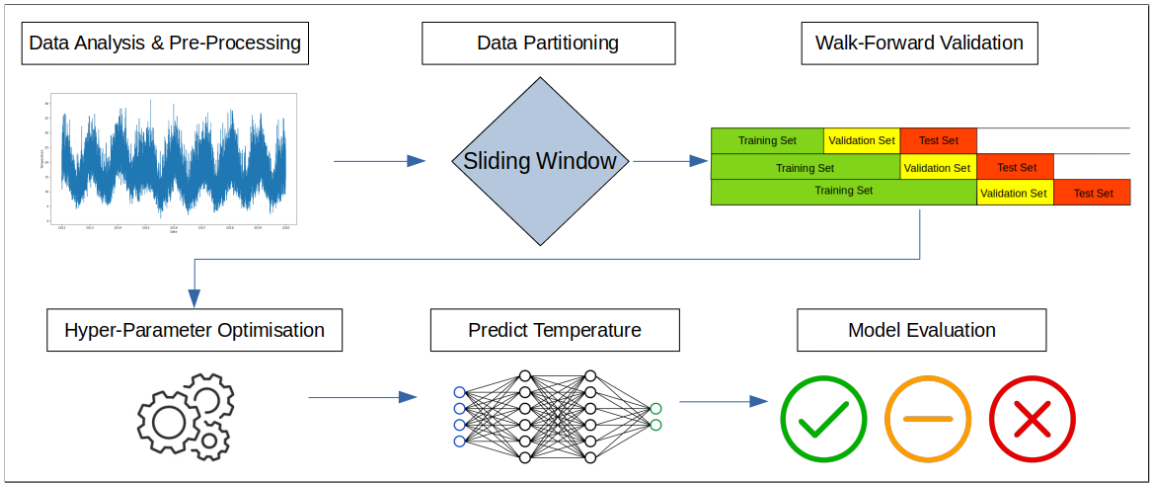


Figure 3.1: Overview of the experimental design

## 3.2 Data Set

The data set used to carry out the experiments performed was provided by the South African Weather Services. It consists of surface weather observations, which are observations made near the earth's surface by automatic weather stations [38]. The original data set comprises surface weather observations across 22 weather stations within the Western and Northern Provinces of South Africa as shown in figure 3.2.

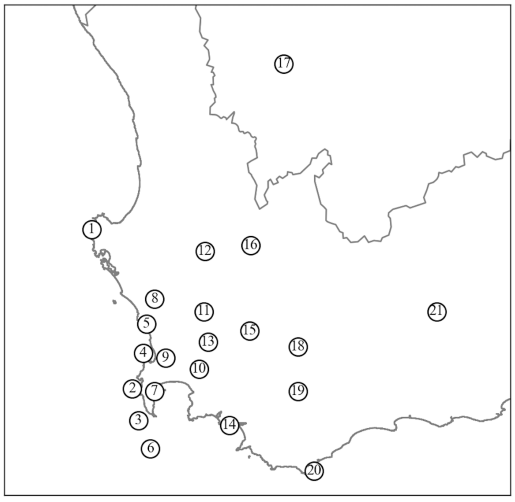


Figure 3.2: Map of Weather Stations in South Africa

The observations, shown in table 3.1 are recorded at hourly intervals over a period of 8 years from January 2012 up to December 2019 yielding 70128 time steps. At each timestamp, all 6 aforementioned surface weather observations are recorded at each weather station.

| | Data Set |
|---|---|
| No. of stations | 22 |
| Features | Temperature(Celsius) |
| | Wind Speed(km) |
| | Wind Direction |
| | Pressure |
| | Rain(mm) |
| | Humidity |
| Time Period | 01/01/2012 - 12/12/2019 |
| Recording Interval | Hourly |
| No. of time steps | 70128 |

Table 3.1: Table describing data set

The next section will describe the missing data, the criteria under which a station was chosen to be included in the experiments, and the missing data for each station was imputed.

### 3.2.1 Missing Data Imputation

The missing data of each station is shown in table 3.2. Following Wilson et al. [9], a station is only included in the experiments if less than 10% of its data is missing. As shown in table 3.2, Swellendam had more than 10% of its data missing and was therefore not included in the experiments.

| | Total Missing Data | % Missing Data |
|---|---|---|
| Atlantis | 935 | 0.22% |
| Calvinia | 1,800 | 0.42% |
| Cape Columbine | 1,905 | 0.45% |
| Cape Point | 11,929 | 2.83% |
| Cape Town Royal Yacht Club | 11,949 | 2.83% |
| Cape Town Slangkop | 4,673 | 1.11% |
| Excelsior | 1,946 | 0.46% |
| Hermanus | 5,861 | 1.39% |
| Jonkershoek | 1,078 | 0.25% |
| Kirstenbosch | 31 | 0.01% |
| Ladismith | 264 | 0.06% |
| Molteno Resevoir | 49 | 0.01% |
| Paarl | 7,259 | 1.72% |
| Porterville | 2,305 | 0.54% |
| Robben Island | 8,778 | 2.08% |
| Robertson | 793 | 0.18% |
| SA Astronomical Observatory | 4,098 | 0.97% |
| Struisbaai | 3,585 | 0.85% |
| Swellendam | 71,733 | 17.04% |
| Tygerhoek | 2,539 | 0.60% |
| Wellington | 861 | 0.20% |
| Worcester | 1,879 | 0.44% |

Table 3.2: Table describing missing data at each station

The inverse distance weighting (IDW) method was used to impute missing data in all 21 stations. This method is computationally easy to implement and showed the strongest performance when imputing meteorological data [39]. The IDW method is shown in equation 3.1 below:

$$z(x_j) = \frac{\sum_{i=1}^{n} z(x_i) d_{ij}^{-r}}{\sum_{i=1}^{n} d_{ij}^{-r}} \tag{3.1}$$

where $n$ is the number of stations, $d$ is the distance between two stations, $z(x_j)$ is the expected predicted value at a station, $z(x_i)$ is the observed value at a station, and $r$ is a real positive number usually set to 2. This method uses the values at neighbouring stations to impute the missing value, where stations that are closer have a greater influence on the imputed value.

The next section will detail the pre-processing techniques that were implemented on the aforementioned data set after the missing data were imputed.

## 3.3 Pre-Processing

Data preprocessing is a technique that is used to transform raw data into some meaningful and understandable format [40]. This section covers the preprocessing methods of normalization and processing the data into input-output instances using the sliding window technique.

### 3.3.1 Normalization

The multivariate models used in these experiments require that the scales of measurement across all the variables are the same. Following the pre-processing techniques used by Hewage et al. [3], all the variables underwent min-max normalization so that the range of values of a feature lies within the range [0, 1]. The formula for min-max normalization is shown in equation 3.2 below.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.2}$$

### 3.3.2 Input-output Processing

The sliding window technique, shown in figure 3.3, is used to form input-output samples from a given time series $X$. Let $x_0$ be the current time step, where $w$ is the length of the input sequence window and $H$ is the length of the prediction horizon. The input-output pair of the sliding window technique would be $[x_0 - w, ..., x_0]$ and $[x_1, ..., x_1 + H]$ respectively.. As can be seen in figure 3.3, the window is shifted by a single timestamp until the end of the series is reached.

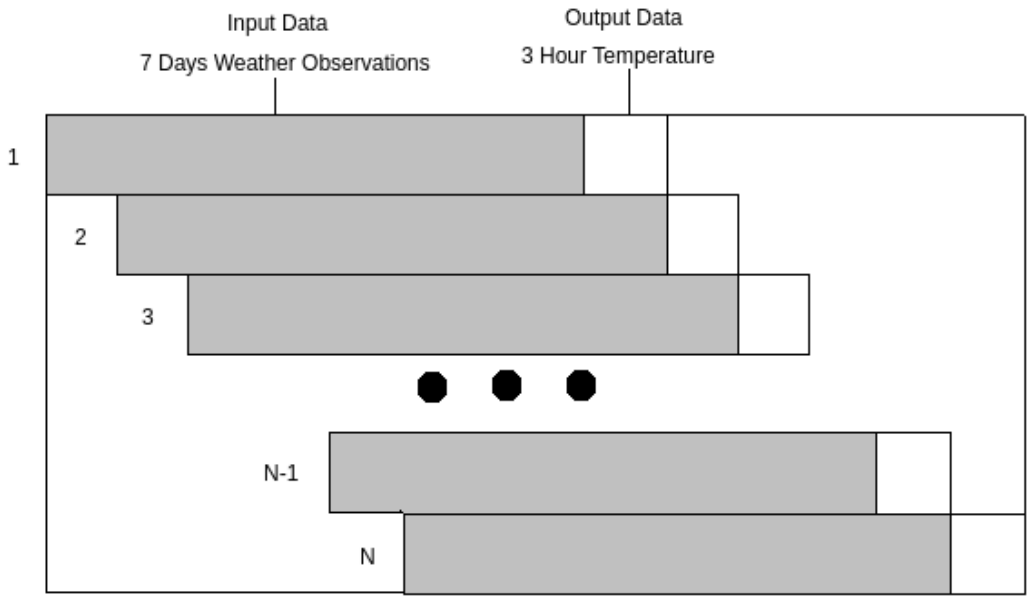


Figure 3.3: Sliding Window Technique

## 3.4 Walk-Forward Validation

Walk-forward validation is a technique that has largely been used in validating models for stock price prediction. This technique is appropriate for time series data as it accounts for the temporal nature of the data [41]. In walk-forward validation, the data set is split into the successive train, validation and test sets shown in figure 3.4. The purpose of the sets is as follows [42]:

- Training Set - set of samples used for learning, to fit the parameters of the model.
- Validation Set - set of samples used the fine tune the model or measure the performance of the model on unseen data as a way of selecting hyper-paremeters.
- Test Set - set of samples used to measure the performance of the final fine-tuned model.

This division is ideal because the model should be evaluated on samples that were not used to build or fine-tune the model so that they provide an unbiased sense of model effectiveness [43]. The performance of the model on the test set is averaged over the walks to reflect a robust metric of performance [44].

The expanding walk-forward validation technique is used as shown in figure 3.4. This version of the walk-forward technique expands the size of the training set with each split (walk), while the size of the validation and test sets remain the same.

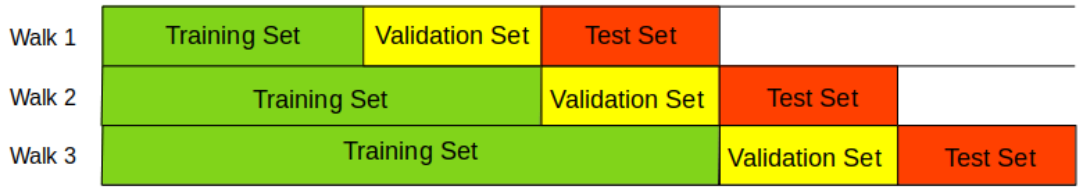


Figure 3.4: Expanding Window Walk-Forward Validation

### 3.4.1 Partitioning of Weather Data

The weather data consists of 8 years worth of surface observations. The first split consisted of the 1st year's observations as the training set, the following 3 months as the validation set, and the 3 months after as the test set. The size of the validation and test set are kept the same, whereas the size of the training set expanded with each successive split. Using the expanding walk-forward validation technique, this method resulted in a total of 27 splits over the data set as shown in table 3.3. Using walk-forward validation, we were able to use 84.38% of the data set for testing.

| | **Surface Weather Observations** |
|---|---|
| *Number of data instances* | 70128 |
| *Chosen test set size* | 3 Months |
| *Chosen total tests percentage* | 84.38% |
| *Number of Splits* | 27 |
| *Validation set size* | 3 Months |
| *Training set size* | 1 year + 3n months<br>where n=[0, ..., 26] |

Table 3.3: Table showing the walk-forward validation splits

### 3.4.2 Model Evaluation Within Walk-Forward Validation

As mentioned in section 3.4, the models are evaluated using the expanding walk-forward validation method. Traditionally, the metrics are averaged across the walks to provide a final metric of the model's performance. However, Kouamme et al. [44] demonstrated that this method may lead to an incorrect estimation of the model's performance, where averaging the metrics over all splits equates to an incorrect estimation of the final metrics. In order to avoid this, the predictions for all the test sets are concatenated into a single series and then used to calculate the final metrics.

## 3.5 Hyper-parameter Optimisation

Hyper-parameter optimisation (HPO) is the process of finding a configuration of parameters that results in the best performing model. Hyper-parameters refer to parameters that cannot be updated during the training of machine learning models [45]. Grid search and random search are the two most popular methods of HPO. Bergstra and Bengio [46] showed empirically and theoretically that random search HPO is the more efficient method. Furthermore, Bergstra and Bengio also found that manual tuning can provide useful insight into the parameters to be selected for random search HPO.

### 3.5.1 Manual Tuning

Manual tuning is the process of selecting a particular hyper-parameter, leaving all others constant, and then evaluating the performance of the algorithm as this hyper-parameter is adjusted. For parameters that are continuous, an arbitrary starting point is chosen and then the value is either increased or decreased while observing the validation loss. This process is done until the incremental performance improvement is negligible, or there is no further improvement in the validation loss. For categorical parameters, an exhaustive search is performed to find the categorical value that results in the lowest validation loss [44].

### 3.5.2 Random Search

In grid search, a search space is defined for each hyper-parameter and tests all possible configurations of those values to find the configuration with the lowest validation loss [44]. Random search is similar to grid search but, instead of testing all values in the search space, random search randomly selects a pre-defined number of configurations between the upper and lower bounds as candidate hyper-parameter values, and then trains these candidates until the defined budget is exhausted [44]. If the configuration space of random search is large enough, then the global optimums for the hyper-parameters or their approximations can be found.

## 3.6 Experimental Pipeline

This section describes the experimental pipeline in which each of the models is developed. Additionally, it also describes the hyper-parameters that were explored for each model.

### 3.6.1 SARIMA

In designing the SARIMA experiments, the methodology followed a variation of the experimental design implemented by Tadesse et al. [47]. The methodology is shown in figure 3.5. This methodology was implemented for each of the 21 weather stations. Following figure 3.5, a stationarity test, followed by an analysis of the Autocorrelation (ACF) plot was performed on the temperature series of each station. The data was then split into train, validation and test sets using walk-forward validation. The best model for each weather station was identified using the validation test, this model was then evaluated on the test set for each of the 27 splits of walk-forward validation.

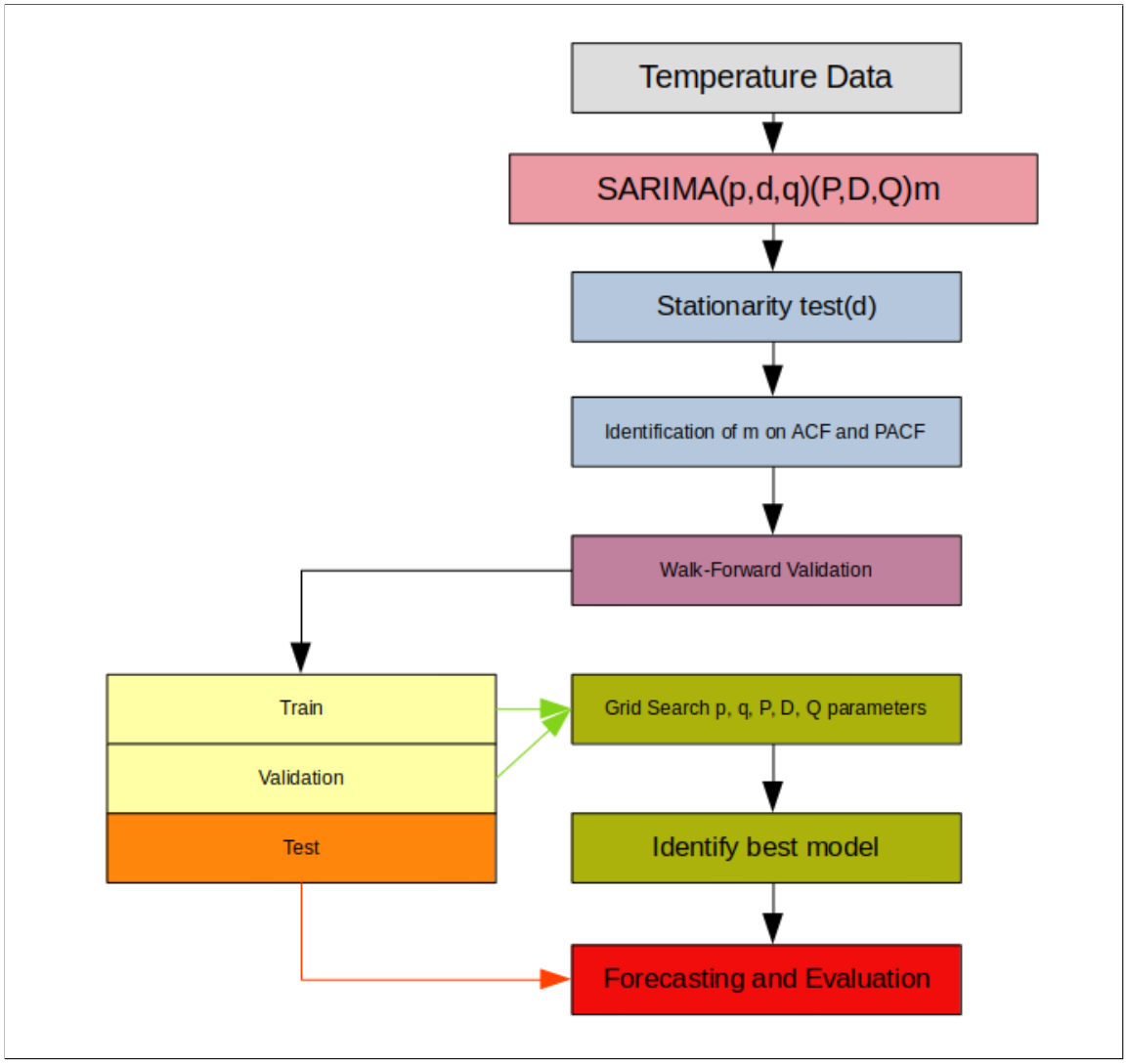


Figure 3.5: Experimental Design of SARIMA Model

### Pre-Processing

The SARIMA model is a univariate statistical method. It uses only the temperature feature in training to predict future temperature values. The only pre-processing technique that was applied to data in this experiment pipeline was imputing missing data using the IDW method described in section 3.3.

### Model Identification

The Augmented Dickey-Fuller (ADF) test for stationarity was done on each of the weather stations to determine the value of the $d$ parameter in the SARIMA model. Following this, the ACF plots were used to determine the value of the $m$ parameter. This was done by identifying the lag which showed a pattern of periodic significant spikes in the ACF plots.

Having identified the $d$ and $m$ parameters, the SARIMA model underwent a random search to find the optimal remaining values for the $p, q, P, D, Q$ for each weather station. According to Brockwell and Davis [48], the $P$ and $Q$ parameters are usually less than 3 and $D$ is seldom greater than one. The $p$ and $q$ parameters were set to a maximum of 3, to ensure that the final models are parsimonious.

| | Search Space |
|---|---|
| d | 0 |
| m | 24 |
| D | [0,1] |
| p | [0,1,2,3] |
| q | [0,1,2,3] |
| P | [0,1,2,3] |
| Q | [0,1,2,3] |

Table 3.4: Search space of SARIMA parameters

During the random search process, a configuration of parameters was selected from table 3.4. This model would then be trained over each of the first two splits created through walk-forward validation. Only the first two splits were used because doing the random search validation over all 27 splits would take too long. For each of the two splits, the predictions on the validation set are appended to a list. Once the model produced predictions for each of the validation sets in the two splits, the MSE was calculated over the entire list of predictions. The MSE over the validation sets were used to measure the performance of the models. This process was done for each weather station, and the model with the lowest MSE on the validation set was then chosen as the best model. The best performing model was then tested on each of the 27 test sets. The predictions are appended to a list and the metrics over the entire list are then calculated and reported as the performance of the model for each station.

### Software and Hardware Environment

The SARIMA models were implemented in python 3.8 on Ubuntu. The SARIMA method in the *statsmodels* library was used to run the experiments. The experiments were run on a Ryzen 5600X CPU.

### 3.6.2 LSTM and TCN

The LSTM and TCN models followed the experiment pipeline shown in figure 3.6 that follows closely with the method used by Hewage et al. [3] to predict temperature. LSTM and TCN models were designed for each weather station individually following this experimental design. Broadly, the data from a single station goes through a series of pre-processing methods. This data is then used to perform manual hyper-parameter optimisation to determine the range of the hyper-parameters that will be used in the random search HPO. The best model for the 24-hour prediction horizon is then found through random search HPO using the validation set. The identified optimal hyper-parameters are then used as the hyper-parameters on which the models are re-tuned to predict the temperature across the remaining prediction horizons (3, 6, 9, 12h).

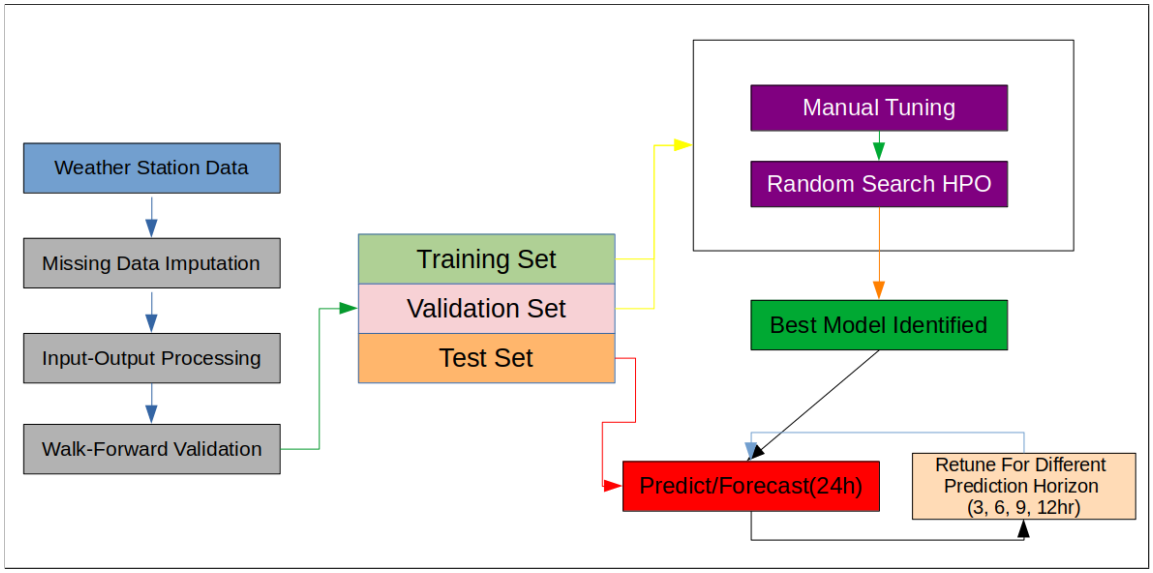


Figure 3.6: LSTM and TCN Experiment Pipeline

#### Pre-Processing

The process starts with imputing the missing data using the IDW method. Once the missing data have been imputed, the complete data were then processed into input-output pairs where the input window size is treated as a hyper-parameter. The range of values for the input window size is shown in table 3.5. Once the data has been processed into input-output pairs, the data is then split into multiple train, validation, and test splits using walk-forward validation.

#### Manual Tuning

The range of continuous and values of categorical parameters are identified using the manual tuning method described in section 3.5.1 which is then used as the search space for the random search

method. The manual tuning was only conducted on two of the twenty-seven splits as this would take a significant amount of time if the tuning was conducted over each of the 27 splits over the 24-hour prediction horizon. The parameters that were manually tuned for both the LSTM and TCN are shown in table 3.5. Additionally, the kernel size and optimiser values shown in table 3.5 were set and underwent no hyper-parameter optimisation.

| **LSTM/TCN** | **Model** | **Value Type** | **Type** | **Values** |
|---|---|---|---|---|
| *Activation Function* | LSTM/TCN | Categorical | Algorithm | $[tanh, linear, ReLu]$ |
| *# of Epochs* | LSTM/TCN | Discrete | Training | [20, 30, 40] |
| *Batch Size* | LSTM/TCN | Discrete | Training | [32, 64, 125, 250] |
| *Learning Rate* | LSTM/TCN | Continuous | Training | (0.001-0.01) |
| *Input Window Size (hours)* | LSTM/TCN | Discrete | Structure | [12, 24, 48, 72, 168] |
| *# of Layers* | LSTM/TCN | Discrete | Structure | [1, 2, 3, 4] |
| *Optimiser* | LSTM/TCN | Categorical | Structure | Adam |
| *Dropout Rate* | LSTM/TCN | Discrete | Regularisation | [0.1, 0.2, 0.3, 0.4] |
| *# of units* | LSTM | Discrete | Structure | [20, 30, 40, 50] |
| *L1 and L2 Regularisers* | LSTM | Continuous | Regularisation | (0.0001-0.01) |
| *Layer Normalisation* | TCN | Training | Regularisation | True/False |
| *# of Filters* | TCN | Discrete | Structure | [32, 64, 125] |
| *Kernel Size* | TCN | Discrete | Structure | 2 |
| *Dilations* | TCN | Discrete | Structure | [1, 2, 4, 8, 16, 32, 64] |

Table 3.5: Hyper-parameter values for LSTM and TCN models

The *activation function* function determines how inputs in a neuron are converted to outputs. The *number of epochs* indicates the number of times the model is trained on the entire data set. The *batch size* determines the number of sub-samples of the data set that are found in a single batch used to train the model. The *learning rate* affects the rate at which the weights of the network are updated. The *input window size* is the number of timestamps to use as input. The *number of layers* indicates the number of TCN or LSTM layers in the model. The *optimiser* is the method by which the loss function is minimised. The *dropout rate* determines the probability of training a given node in the network. The *number of units* in the LSTM model determines the dimensionality of the output space. The *L1 and L2 regularisers* in the LSTM model are used to prevent overfitting on the training data set.

*Layer normalisations* are used in the TCN models to speed up and stabilise training. The *number of filters* in the TCN model indicates the number of filters to use in the convolutional layer used to capture features in the data. The *kernel size* represents the size of the aforementioned filters. Finally, the *dilations* in combination with the kernel size determines the size of the receptive field. The *receptive field* is the region of input that a neuron or unit within the network is exposed to [28]. Hewage et al. [3] state that changing the dilations and kernel size does not heavily impact the model's performance [3]. The dilations and kernel size are fixed at the values shown in table 3.5. Finally, as shown in table 3.5, we have chosen to vary eleven of the thirteen parameters while keeping two fixed.

### Random Search HPO

The random search HPO process is done after the aforementioned manual tuning step above has been completed. The parameters identified in the manual tuning step are used as the search space for the random search HPO step. The models underwent random search HPO over 2 of the 27 splits over the 24-hour prediction horizon using the training and validation data sets. The predictions from each of the 2 splits are appended to a list, and the hyper-parameter configurations that result in the lowest MSE over the 2 splits are selected as the final configurations to train and test the models with.

### Evaluation

After having found the optimal hyper-parameter configurations for each model, the models are then trained and tested over all 27 splits of the walk-forward validation technique. For each split, the model is trained using the train data set and then tested on the test data set. The predictions from all 27 splits are appended to a list. The complete list of predictions is then used to calculate the metrics which measure the model's performance.

### Remaining Prediction Horizons

For each station, once the hyper-parameters of the best model have been identified for the 24-hour prediction horizon, the model is then re-tuned to predict the temperature across the 3, 6, 9, and 12-hour prediction horizons using the identified hyper-parameters. These models go through the same pre-processing methods, with the only difference being that the length of the output sequence is adjusted when creating the input-output sequences in accordance with the corresponding prediction horizon. These models are then evaluated in the same manner as mentioned above.

### Results

The above experimental design yields the results shown in table 3.6. For each station, the optimal parameters for the best performing model are identified for the 24-hour prediction horizon. The SMAPE, MAE, and MSE performance metrics are reported across all 5 prediction horizons.

| **Results** | **Prediction Horizon (hours)** |
|---|---|
| *Optimal Hyper-Parameters* | 24 |
| *MSE, SMAPE, MAE* | 3, 6, 9, 12, 24 |

Table 3.6: Table showing results that will be produced for LSTM and TCN models

### Software and Hardware Environment

The LSTM and TCN models were implemented in python 3.8 on Ubuntu. The *Keras* [49] framework was used to implement both of the models. The experiments were run on an NVIDIA RTX3070 GPU.

### 3.6.3 Spatial-Temporal Graph Neural Networks

The experimental design of the graph neural networks is shown in figure 3.7. The design is similar to that of the LSTM and TCN, without the additional step of having to apply the model on each station individually as the GNNs are trained on, and predict the temperature at all stations simultaneously. Broadly, the data goes through a series of pre-processing steps. The models then go through hyper-parameter optimisation to find the best set of parameters when predicting over the 24-hour prediction horizon. Once the parameters have been identified and the predictions are made on the 24-hour prediction horizon, the ST-GNN is then re-tuned using the same hyper-parameters to predict over the 3, 6, 9, and 12-hour prediction horizons.

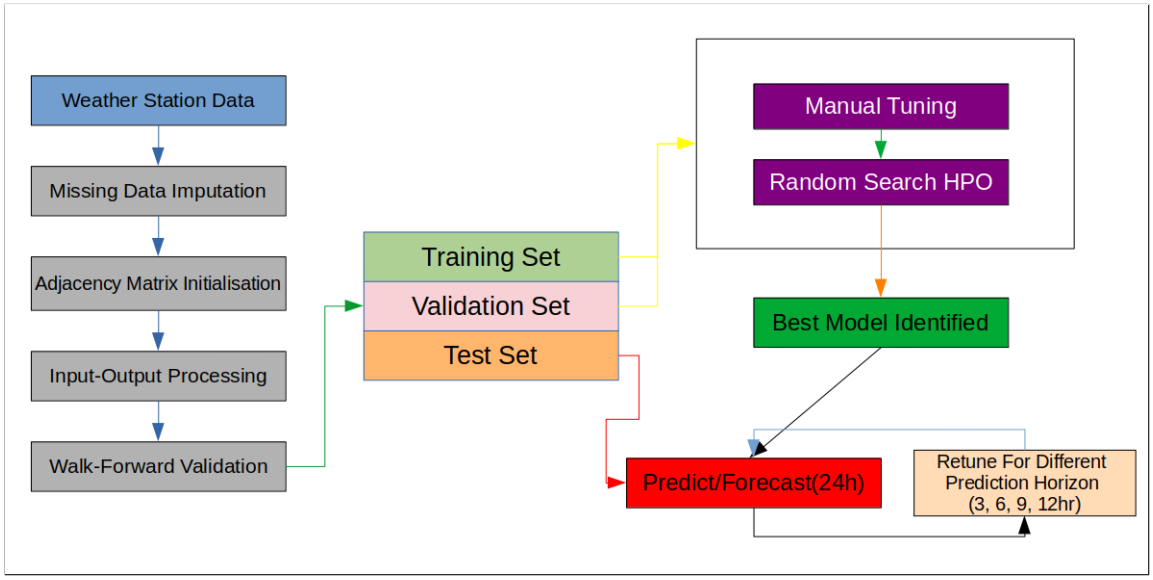


Figure 3.7: Graph Neural Network Experiment Pipeline

#### Pre-processing

The pre-processing methods applied to the data in the ST-GNN experimental design were the same as that in the LSTM and TCN experimental design. Additionally, the data were originally separated between stations. The ST-GNNs required that all the stations' data are collated into a single data set consisting of all stations' data.

#### Manual Tuning

Prior to the random search HPO, manual tuning was conducted on both ST-GNNs to identify the range of continuous and values of categorical parameters that will be used in creating the search space for the random search HPO. The parameters that were manually tuned can be found in table 3.7 below:

| WGN/GWN | Model(s) | Value Type | Type | Values |
|---|---|---|---|---|
| *# of Epochs* | GWN/WGN | Discrete | Training | [20, 30, 40, 50, 70, 100] |
| *Batch Size* | GWN/WGN | Discrete | Training | [32, 64, 125, 250] |
| *Learning Rate* | GWN/WGN | Continuous | Training | (0.001-0.01) |
| *Input Window Size (hours)* | GWN/WGN | Discrete | Training | [12, 24, 48, 72, 168] |
| *# of hidden neurons* | GWN/WGN | Discrete | Structure | [10, 20, 30, 40, 50] |
| *Dropout Rate* | GWN/WGN | Continuous | Regularisation | (0.001-0.01) |
| *Mask Length* | WGN | Discrete | Training | [0, 12, 24, 48, 72] |
| *# of Graph Convolution Layers* | WGN | Discrete | Structure | [1, 2, 3, 4] |
| *# of Spatial-Temporal Layers* | GWN | Discrete | Structure | [1, 2, 3, 4] |
| *Dilations* | GWN | Discrete | Structure | [1, 2, 1, 2, 1, 2, 1, 2] |

Table 3.7: ST-GNN Hyper-Parameters

The *mask length* in the WGN model is the number of time steps in the input sequence that will be used as context for the WGN model before making predictions. The number of *graph convolutional layers* in the WGN model indicates how many graph convolution layers are used to capture the temporal and spatial dependencies. The number of *spatial-temporal layers* in the GWN model indicates the number of spatial-temporal blocks consisting of TCNs and graph convolutions. The *spatial-temporal blocks* can be stacked to capture temporal dependencies of different lengths. Finally, the *dilations* in the GWN model determines the receptive field size of the model. The receptive field is the region of input that a neuron or unit within the network is exposed to [28]. The dilations are fixed at the same values used by Wu et al. [5] in their original implementation of GWN as Hewage et al. [3] found that varying the dilations parameter does not significantly impact model performance. Finally, as shown in table 3.7, we have chosen to vary nine of the ten parameters while keeping the dilations fixed.

In addition to the hyper-parameters listed in table 3.7, the ST-GNNs had additional parameters related to the adjacency matrix. The adjacency matrix hyper-parameters are shown in table 3.8 below:

| GNN | Adjacency Matrix | Value Type | Type | Values |
|---|---|---|---|---|
| *WGN/GWN* | Initialisation | Categorical | Algorithm | Random, Gaussian Kernel Distance |
| *WGN* | Adjacency Matrix Rank (r) | Discrete | Structure | [1, 1.5, 2] |
| *GWN* | Adjacency Type | Categorical | Algorithm | Adaptive Only |
| | | | | Forward-Backward |
| | | | | Forwards-Backwards Adaptive |

Table 3.8: Table showing hyper-parameters related to adjacency matrix

The adjacency matrices could either be randomly initialised or initialised by the Euclidean distance between weather stations with a thresholded Gaussian kernel. The WGN ST-GNN had an additional hyper-parameter of rank. A higher rank reduces the number of connections in the adjacency matrix thereby reducing computation complexity at the cost of accuracy. The rank, $r$, parameter shown in table 3.8 reduces the size of the adjacency matrix by reducing it to contain only $s/r$ connections where $s$ is the number of stations. The Graph WaveNet model could utilise just a learnt self-adaptive adjacency matrix that requires no prior knowledge for graph convolution. Alternatively, prior knowledge is given in the form of the thresholded Gaussian kernel values. The prior knowledge is used to derive the forwards and backwards transition matrix that is used in combination with the self-adaptive adjacency matrix in the graph convolution layers. Finally, all the parameters shown in table 3.8 were varied as hyper-parameters.

### Random Search HPO

The hyper-parameters identified in the manual tuning step mentioned above are then used as the search space for the random search HPO step. The graph neural networks underwent random search HPO over 2 of the 27 splits using the training and validation data sets. The predictions for each of the two splits are appended to a list and the hyper-parameter configurations that result in the lowest MSE over the 2 splits and then selected as the final configurations to train and test the models with.

### Evaluation

Using the optimal hyper-parameter configurations for the GNN model, the model is then trained and tested over the 27 splits and of the walk-forward validation technique. For each split, the model is trained and tested using the train and test sets respectively. The predictions from each of the 27 splits are appended to a list, and the complete list of predictions is then used to calculate the metrics which measure the model's performance.

### Remaining Prediction Horizons

The hyper-parameters identified in the random search HPO step are then used to train new GNN models to predict temperature across the remaining prediction horizons (3, 6, 9, and 12 hours). These models follow the same method of evaluation mentioned above.

### Results

The above experimental design yields the results shown in table 3.9. For each GNN, the optimal parameters for the best performing model are identified for the 24-hour prediction horizon. Performance metrics are reported across all 5 prediction windows.

| **Results** | **Prediction Horizon (hours)** |
|---|---|
| *Optimal Parameters* | 24 |
| *Adjacency Matrix* | 24 |
| *MSE/MAE/SMAPE* | 3, 6, 9, 12, 24 |

Table 3.9: Table showing results that will be produced for WGN and GWN models

### Software and Hardware Environment

The GNNs were implemented in python 3.8 on Ubuntu. The *Tensorflow* [50] framework was used to implement the low-rank GNN whereas the *PyTorch* [51] framework was used to implement the Graph WaveNet model. The experiments were run on an NVIDIA RTX3070 GPU. The original implementations of the GWN[1] and WGN[2] models were used.

[1] https://github.com/nnzhan/Graph-WaveNet
[2] https://github.com/TylerPWilson/wgc-lstm

## 3.7 Evaluation Metrics

This section will cover the metrics used as the loss function for the models and the metric that is used to evaluate and compare the models.

The experimental design above is inspired by Hewage et al. [3]. Hewage et al. used LSTMs and TCNs to predict temperature. Wilson et al. [9] developed the WGN model that was used to predict temperature. Lastly, Wu et al. [5] developed the GWN model which was used to predict traffic flow. Since these are the models implemented in this paper, we examine the metrics used as the loss functions for the models, and the metrics used to evaluate and compare the final models.

Hewage et al. [3] used the Mean Squared Error (MSE) metric as the loss function and to compare models. Wilson et al. [9] also used the MSE metric as the loss function and to compare models. The models built by Hewage et al. and Wilson et al. yielded accurate predictions by using the MSE metric as the loss function. Wu et al. [5] used the MAE metric as the loss function and achieved state-of-the-art results. Having said this, the MSE metric has been used as the loss function in weather prediction models with high accuracy. Despite Wu et al.[5] using the MAE metric as the loss function, we have selected the MSE metric as the loss function because highly accurate models have been developed in the weather domain using the MSE as the loss function. The mathematical formulation for the MSE metric is shown below in equation 3.3:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_r - Y_p)^2 \tag{3.3}$$

where $Y_r$ is the real value expected or target value, $Y_p$ is the model's predicted value, and $n$ is the number of samples.

The MSE metric used as the loss function is scale-dependent. This becomes an issue when comparing models whose data have different scales. We therefore introduce the Symmetrical Mean Absolute Error (SMAPE) and Mean Absolute Error (MAE) metrics. The MAE and SMAPE metrics are independent of scale and can therefore be used to compare different models. The MAE metric is defined below in equation 3.4:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_p - Y_r| \tag{3.4}$$

where $Y_r$ is the real value expected or target value, $Y_p$ is the model's predicted value, and n is the number of samples. The SMAPE metric is defined below in equation 3.5:

$$SMAPE = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|Y_p - Y_r|}{(|Y_p| + |Y_r|)/2} \tag{3.5}$$

where $Y_r$ is the real value expected or target value, $Y_p$ is the model's predicted value, and n is the number of samples. The MAE does however have a drawback, the prediction values can range between zero and infinity, and a single value does not say much about the performance of the regression with respect to the distribution of the ground truth elements [52]. The performance and comparison between models will therefore focus on the SMAPE metric. We will therefore report the MSE, MAE, and SMAPE metrics with the focus being on the SMAPE metric to evaluate and compare models.

## 3.8 Spatial-Temporal Dependency Visualisation

The WGN adjacency matrix only consists of the self-adaptive adjacency matrix and is therefore directly extracted from the model. The GWN adjacency matrix can consist of both the forwards-backwards transition matrices and the self-adaptive adjacency matrix. The spatial-temporal dependencies in the GWN model are encoded in the self-adaptive adjacency matrix. To extract this matrix, we matrix multiply the models' two node embeddings, then apply Relu to eliminate weak connections, and finally apply SoftMax.

In visualising the spatial-temporal dependencies we follow the method used by Wilson et al. [9] mentioned in the literature review. Wilson et al. visualized the spatial-temporal dependencies between weather stations by plotting the most important weights within the learnt adjacency matrix. This was done by taking the absolute value of the learned adjacency matrix and finding the largest edge weight in each row of the adjacency matrix. Once these weights are determined, we apply normalisation on the weights to scale their range to [0, 1]. The edge corresponding to each row's largest weight is then plotted along with its direction. [9]

In order to analyse the influence that a weather station exerts over other stations, we measure and present the out-degree of each station alongside the visualisations. The out-degree is the number of edges directed out of a station to other stations.

## 3.9 Summary

The data will go through a series of pre-processing steps. The processed data will then be used to train the SARIMA, LSTM, TCN, WGN, and GWN models. The models will undergo manual and random search hyper-parameter optimisation. The ideal hyper-parameters will then be used to train models over the 3, 6, 9, 12, and 24-hour prediction horizons. The ideal hyper-parameters and the performance of the models at each station will be presented in the following chapter. The spatial-temporal dependencies learnt by the WGN and GWN models will then be visualised.

# Chapter 4

# Results and Analysis

This chapter presents the results of the experiments described in chapter 3. The results are accompanied by an analysis thereof which will later be discussed in chapter 5. This section begins by presenting the results of the SARIMA model. This is followed by presenting the results of the hyper-parameter optimisation results of the LSTM, TCN, WGN, and GWN models. The station results of the LSTM and TCN models are then presented. This is followed by the station results of the WGN and GWN model. A summary of the aforementioned results is then provided. We then perform an investigation into the data set and model predictions which elucidate the results. Finally, this chapter concludes by presenting a visualisation, analysis, and comparison of the learnt spatial-temporal dependencies of the WGN and GWN model.

## 4.1 SARIMA

This section presents the results of exploring SARIMA for temperature prediction. We present the results obtained by following the experimental design laid out in section 3.6.1. First, we present the results of the hyper-parameter optimisation of the models, this is then followed by the performance of the models for each weather station.

### 4.1.1 Hyper-Parameter Optimisation

The $m$ and $d$ parameters were determined to be 0 and 24 as shown in appendix A.1. The search space was configured to the range of values shown in table 4.1. A random search hyper-parameter optimisation (HPO) process was then implemented to determine the optimal parameters for each station's SARIMA model. The random search was conducted using the training and validation sets of the first two splits of the twenty-seven walk-forward validation splits. This was done because carrying out the random search HPO process on all the splits for each station would have been time inefficient. From the search space shown in table 4.1, 30 random configurations were selected by drawing random parameters from the aforementioned search space.

| | **Search Space** |
|---|---|
| $d$ | 0 |
| $m$ | 24 |
| $D$ | [0,1] |
| $p$ | [0,1,2,3] |
| $q$ | [0,1,2,3] |
| $P$ | [0,1,2,3] |
| $Q$ | [0,1,2,3] |

Table 4.1: Search space of SARIMA parameters

The hyper-parameters found through the random search HPO process are shown in table 4.2 below. From table 4.2, we can see that 14 of the 21 stations included an AR component in their model. This means that the majority of SARIMA models are using lagged observations (inputs from previous time steps) to predict the value at the next step. The $d$ parameter is 0 for all models because of the stationarity tests mentioned above. The moving average (MA) component for all models except one was found to be 1. The inclusion of the MA component in the SARIMA models indicates that the mean of past observations is highly correlated with future values.

| | **Parameters** | | | | | | |
|---|---|---|---|---|---|---|---|
| **Stations** | $p$ | $d$ | $q$ | $P$ | $D$ | $Q$ | $m$ |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 24 |
| 2 | 1 | 0 | 1 | 0 | 1 | 0 | 24 |
| 3 | 0 | 0 | 1 | 0 | 1 | 0 | 24 |
| 4 | 0 | 0 | 1 | 0 | 1 | 1 | 24 |
| 5 | 0 | 0 | 1 | 0 | 1 | 1 | 24 |
| 6 | 1 | 0 | 1 | 0 | 0 | 1 | 24 |
| 7 | 1 | 0 | 1 | 0 | 1 | 1 | 24 |
| 8 | 0 | 0 | 2 | 0 | 1 | 2 | 24 |
| 9 | 1 | 0 | 1 | 1 | 0 | 1 | 24 |
| 10 | 1 | 0 | 1 | 0 | 0 | 1 | 24 |
| 11 | 1 | 0 | 1 | 0 | 0 | 1 | 24 |
| 12 | 1 | 0 | 1 | 0 | 1 | 1 | 24 |
| 13 | 1 | 0 | 1 | 0 | 1 | 2 | 24 |
| 14 | 1 | 0 | 1 | 0 | 1 | 1 | 24 |
| 15 | 0 | 0 | 1 | 1 | 1 | 2 | 24 |
| 16 | 1 | 0 | 1 | 0 | 1 | 0 | 24 |
| 17 | 1 | 0 | 1 | 0 | 1 | 1 | 24 |
| 18 | 1 | 0 | 1 | 1 | 1 | 1 | 24 |
| 19 | 1 | 0 | 1 | 0 | 1 | 2 | 24 |
| 20 | 0 | 0 | 1 | 0 | 1 | 0 | 24 |
| 21 | 1 | 0 | 1 | 0 | 1 | 0 | 24 |

Table 4.2: Optimal hyper-parameters for each station

For the seasonal number of autoregressive terms included in the model ($P$), only 3 of the models included a seasonal autoregressive component. The majority of the models did not find seasonally

lagged observations to be useful when predicting future temperature values. The $D$ parameter indicates if a station's data must be seasonally differenced to remove any seasonal trends. A value greater than 0 indicates the data set was seasonally differenced. From table 4.2, we can see that 17 of the 21 stations included a seasonally differenced component to remove trends. This reveals that some stations have strong seasonal trends that need to be removed whereas other stations have weak or no seasonal trends. Finally, the seasonal moving average ($Q$) parameter ranges from 0 to 2. Again we see the models relying heavily on the mean of past observations to predict future values with sixteen of the twenty-one stations including a seasonal moving average component.

### 4.1.2 Model Performance

The performance metrics of the SARIMA models for each station are shown in table 4.3 below. The data set was split via walk-forward validation into 27 splits. For each station, a SARIMA model was trained on the train set and then evaluated on the test set. This was done for all 27 splits. The results below show the average performance of SARIMA models at each station across all 27 splits. It is important to note that the SARIMA data was not processed into input-output pairs as done for the deep learning models. The SARIMA models were also not tested across the five prediction horizons (3, 6, 9, 12, 24) because this would take exceptionally long to compute.

| | **Metrics** | | |
|---|---|---|---|
| **Station** | *MSE* | *SMAPE* | *MAE* |
| 1 | 56.54 | 40.61 | 6.61 |
| 2 | 111.93 | 57.10 | 8.86 |
| 3 | 18.40 | 23.11 | 3.54 |
| 4 | 25.91 | 28.61 | 4.51 |
| 5 | 39.78 | 30.52 | 5.49 |
| 6 | 270.53 | 199.99 | 16.16 |
| 7 | 96.30 | 66.98 | 8.69 |
| 8 | 25.99 | 26.73 | 4.51 |
| 9 | 80.62 | 50.73 | 7.28 |
| 10 | 266.67 | 200.00 | 15.37 |
| 11 | 356.51 | 200.00 | 17.53 |
| 12 | 42.30 | 31.87 | 5.64 |
| 13 | 85.93 | 47.55 | 8.35 |
| 14 | 97.40 | 52.96 | 8.85 |
| 15 | 26.71 | 26.90 | 4.45 |
| 16 | 75.90 | 44.22 | 7.46 |
| 17 | 36.58 | 30.55 | 5.36 |
| 18 | 38.55 | 34.44 | 5.61 |
| 19 | 68.48 | 47.78 | 7.40 |
| 20 | 98.99 | 49.08 | 8.48 |
| 21 | 94.33 | 49.07 | 8.28 |

Table 4.3: Performance Metrics of SARIMA models for each station

The MSE of the SARIMA models ranges from 18.40 to 200. The MSE metric is sensitive to outliers

and noise. The MSE values and the range of those values are high which suggests that the SARIMA models were not able to predict and model outliers and noise present in the temperature data sets. The high RMSE of the models, ranging from 3.13 to 7.22, reveals that the models are not able to fit the temperature data sets well. The high MSE and RMSE metrics of the models show that the SARIMA models have high biases. The magnitude and range of the MAE results shown in table 4.3 are low in comparison to the MSE and RMSE metrics because the MAE metric is less sensitive to large error values. The large range of values in the MSE metric recorded for the SARIMA models indicate that the models are not capable of learning temporal relationships that exist at each individual station. Finally, the range of SMAPE values lies between a poor predictive performance range of 26.9% all the way up to an exceptionally poor SMAPE value of 200% at 3 stations.

Linking the results from table 4.3 to 4.2, we see some interesting findings. The models trained on stations 6, 10, and 11 have the worst performance. These models are three of the four models that do not include seasonal differences ($D$). The worst performing models also all have a $p$ value of 1 indicating that they use a linear combination of past values to predict temperature. Comparatively, the top four best performing models at stations 3, 4, 8, and 15 all include a seasonally differenced component ($D$). Three of these four models also have $p$ values of 0. These findings suggest that including a seasonally differenced component improves accuracy, whereas including the $p$ component significantly reduces model performance.

Figure 4.1 below shows the SMAPE of the SARIMA models at each station.

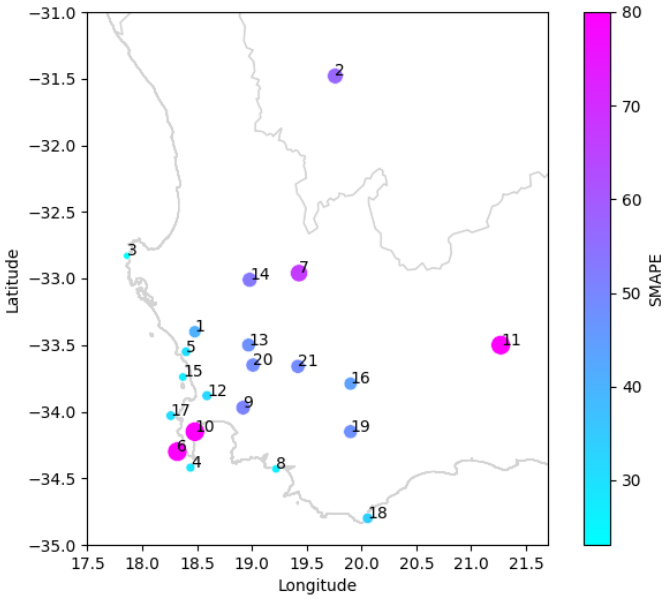


Figure 4.1: Plot of SARIMA models' SMAPE at each weather station

The colour of the dot representing a station reflects the values of SMAPE scores for each station reported in table 4.3. The size of the dot indicates the size of the SMAPE metric with larger dots representing poorer performance i.e. higher SMAPE values. In order to ensure that the colour range is small enough to see value differences across all stations, the three stations (6, 10, 11) that achieved the highest SMAPE of 200% were reduced to 80%. From this plot, we can see that the SARIMA model performed better when predicting temperatures along the coast with the exception being the two stations that recorded the highest SMAPEs of 199.99% and 200%. The performance of the models further inland is significantly worse than those along the coast. As mentioned earlier, the high SMAPE values are an indication of the SARIMA model's inability to predict outliers, deal with noise, and capture temporal dependencies. This suggests that the temperature data sets recorded further inland could have a larger number of outliers and the patterns underlying the temperature series are more complex.

## 4.2 Hyper-Parameter Optimisation

This section presents and discusses the results of the hyper-parameter optimisation process performed on the LSTM, TCN, WGN, and GWN models. For each model, the manual tuning results are presented. This is then followed by a table showing the optimal hyper-parameters found during the random search HPO step and an analysis thereof.

### 4.2.1 LSTM

#### Manual Tuning Results

The results of the manual tuning are shown below in table 4.4 below.

| Parameter | Value Type | Type | Search Space | Optimal |
|---|---|---|---|---|
| *Activation Function* | Categorical | Algorithm | *tanh*, *linear*, *sigmoid* | *tanh* |
| *# of Epochs* | Discrete | Training | [20, 30, 40, 50] | [40] |
| *Batch Size* | Discrete | Training | [32, 64, 125] | [32, 64, 125] |
| *Learning Rate* | Continuous | Training | (0.0001, 0.01) | (0.0001, 0.005) |
| *Input Window Size (hours)* | Discrete | Structure | [12, 24, 48, 72, 168] | [48, 72, 168] |
| *# of Layers* | Discrete | Structure | [1, 2] | [1] |
| *Optimiser* | Categorical | Structure | Adam | Adam |
| *# of units* | Discrete | Structure | [20, 30, 40, 50] | [30, 40, 60] |
| *Dropout Rate* | Continuous | Regularisation | [0.1 0.2,...,0.5] | [0.1, 0.3] |
| *L1 and L2* | Discrete | Regularisation | [1e-2, 1e-3, 1e-4] | [1e-2, 1e-3] |

Table 4.4: Optimal hyper-parameters found through manual tuning

Of the 3 activation functions, $tanh, linear, ReLu$, it was found that the $tanh$ function results in the best performance. The epoch size of 40 was found to be an ideal epoch size in the range of 20-40 epochs. The LSTM models would typically reach their lowest training and validation losses after 30-40 epochs. The ideal batch size values for the LSTM were found to range between 32 and 125, where the difference in performance across the different batch sizes was not huge. The ideal

learning rate ranged from 0.005 to 0.0001, the values in this range were found to effectively control how much to change the model's response to the calculated error. The optimal input window sizes fed into the LSTM models range from 48 hours (2 days) to 168 hours (7 days) hours of historical weather data. The manual tuning revealed that only a singly LSTM layer is necessary to obtain the best performing model. The optimal single layer indicates that the temporal dependencies at the weather stations are not complex, as more complex dependencies would require additional layers.

The optimal range for the number of units for the LSTM HPO process was found to be between 30 and 60. Similar to the range of batch sizes, the performance difference within the range of the number of hidden neurons was not large. From the manual tuning process, the LSTM models performed much better on the training set than the validation set indicating that the models were overfitting the training data. Dropout and $L1$ and $L2$ regularisers were found to be effective in reducing how much the LSTM models were overfitting the data and modelling noise in the train sets. The range of the effective dropout and $L1$ and $L2$ parameters were found to be between 0.1 and 0.3, and $1e-2$ and $1e-3$ respectively. Of the aforementioned hyper-parameters, the learning rate, input window size, number of epochs, and regularisation parameters were found to have the largest impact on the performance of the models.

The values in the optimal column in table 4.4 were then used as the search space for the random search tuning. The batch size, learning rate, input window size, number of units, dropout rate and regularisation were used as the hyper-parameters in the random search tuning. The total possible number of configurations to search through was 4860. The random search tuning randomly selected 30 configurations from this total search space. The number of configurations to search from was kept low because this process needed to be repeated for each weather station.

#### Random Search Results

The range of continuous and values of categorical parameters found in section 4.2.1 were used as the search space for the random search HPO process. The results of the random search HPO process; the optimal parameters for the LSTM model for each station are shown in table 4.5 below.

| **Station** | *# of Units* | *Input Window Size* | *Batch Size* | *Learning Rate* | *Dropout* | *L1 and L2* |
|---|---|---|---|---|---|---|
| 1 | 40 | 168 | 32 | 0.001 | 0.1 | 1e-2, 1e-2 |
| 2 | 40 | 168 | 32 | 0.001 | 0.1 | 1e-3, 1e-3 |
| 3 | 40 | 168 | 64 | 0.005 | 0.3 | 1e-2, 1e-2 |
| 4 | 60 | 72 | 32 | 0.001 | 0.3 | 1e-3, 1e-3 |
| 5 | 30 | 168 | 64 | 0.001 | 0.3 | 1e-2, 1e-2 |
| 6 | 30 | 72 | 64 | 0.003 | 0.2 | 1e-3, 1e-3 |
| 7 | 60 | 168 | 64 | 0.001 | 0.1 | 1e-2, 1e-2 |
| 8 | 30 | 48 | 64 | 0.0007 | 0.3 | 1e-3, 1e-3 |
| 9 | 40 | 168 | 64 | 0.0007 | 0.3 | 1e-2, 1e-2 |
| 10 | 30 | 48 | 64 | 0.001 | 0.3 | 1e-3, 1e-3 |
| 11 | 30 | 168 | 64 | 0.001 | 0.3 | 1e-2, 1e-2 |
| 12 | 50 | 72 | 64 | 0.0005 | 0.3 | 1e-3, 1e-3 |
| 13 | 30 | 168 | 64 | 0.0005 | 0.1 | 1e-3, 1e-3 |
| 14 | 50 | 168 | 64 | 0.001 | 0.1 | 1e-3, 1e-3 |
| 15 | 60 | 168 | 125 | 0.001 | 0.1 | 1e-2, 1e-2 |
| 16 | 40 | 48 | 32 | 0.001 | 0.3 | 1e-2, 1e-2 |
| 17 | 30 | 72 | 64 | 0.0003 | 0.2 | 1e-2, 1e-2 |
| 18 | 50 | 168 | 64 | 0.0003 | 0.3 | 1e-3, 1e-3 |
| 19 | 40 | 72 | 64 | 0.0005 | 0.3 | 1e-2, 1e-2 |
| 20 | 40 | 72 | 64 | 0.001 | 0.1 | 1e-2, 1e-2 |
| 21 | 30 | 72 | 125 | 0.001 | 0.1 | 1e-3, 1e-3 |

Table 4.5: Table showing optimal parameters for each LSTM Model

The percentage of hyper-parameter values used in the best configurations shown in table 4.5 are visualised in the pie chart in figure 4.2. As found during the manual tuning step in section 4.2.1, the activation function, number of layers, optimiser and number of epochs were set to *tanh*, 1, *adam*, and 40 respectively. Fourteen of the twenty-one stations found that a batch size of 64 would lead to the best performance, four stations found a batch size of 32 to be effective, and the remaining three stations found a batch size of 128 to lead to the best performance. Although a batch size of 128 would result in faster computation through efficient use of GPU computing, the higher batch sizes resulted in poor generalization indicated by a higher training and validation loss and overfitting. The batch size of 64 offers slightly faster computation than the lowest batch size of 32 and results in better generalization in the models, as evidenced by similar low training and validation losses.

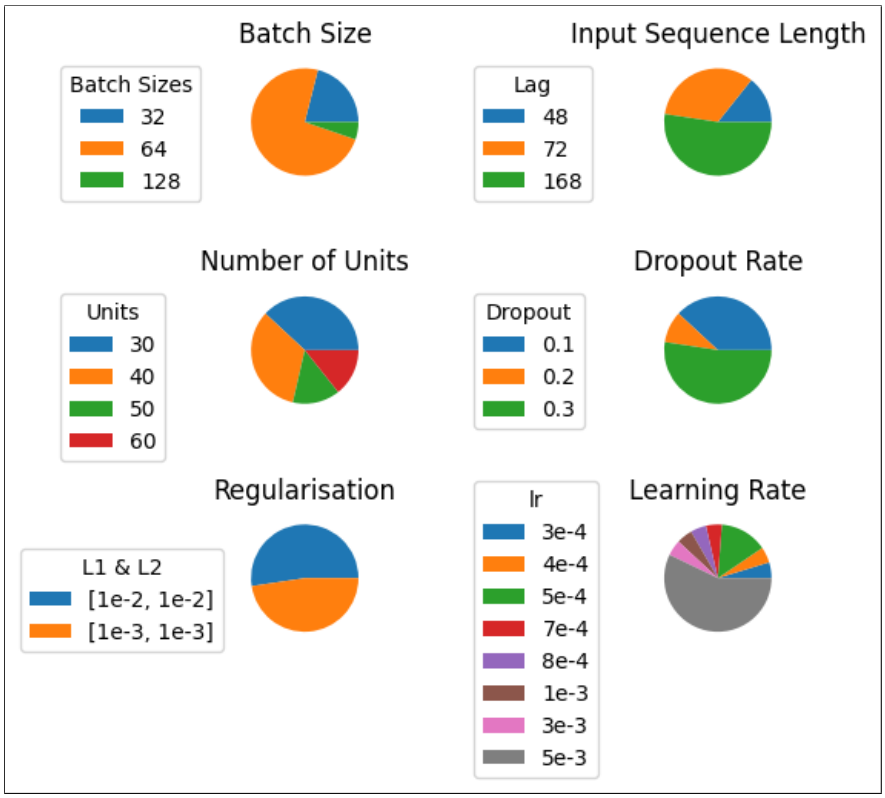


Figure 4.2: Pie Charts of optimal hyper-parameters for LSTM models

Eleven of the twenty-one stations found that an input window size of 168 hours resulted in the best performance. Seven stations found the ideal length to be 72 hours, and the remaining three stations used an input sequence length of 48 hours to achieve optimal performance. With the range of input sequence length from 12 hours to 168 hours, it is clear that the LSTM models achieved better performance when using longer input sequences.

The ideal learning rate for the LSTM models ranged between 0.005 and 0.0001. Twelve of the stations used a learning rate of 0.001 with the remaining stations using a learning rate spread between 0.0003 to 0.0008. As mentioned in section 4.2.1, the learning rate values in the aforementioned range did not result in significantly different performance. The range of the ideal number of units was between 30 and 60, with the different values in the range not leading to significant performance differences. Furthermore, the number of hidden neurons did not significantly reduce or increase the model's ability to generalise.

All of the LSTM models trained initially had significantly lower training losses than validation losses, indicating poor generalisation and overfitting. Adjusting the number of units did not assist in preventing overfitting. As a result, $L1$ and $L2$ regularisation were used in all the models with values ranging from $1e-2$ to $1e-3$. Dropout was used in preventing overfitting with values ranging

from 0.1 to 0.3. Regularisation in combination with dropout proved effective in producing models capable of generalisation that performed well on unseen data.

### 4.2.2 Temporal Convolutional Network

#### Manual Tuning Results

The results of the manual tuning performed for the TCNs are shown below in table 4.6. Of the 3 activation functions, *sigmoid*, *linear*, *relu*, it was found that *relu* resulted in the best performance. *ReLu* resulted in both faster training times and lower training and validation losses. Similar to the epoch size for the LSTMs, the TCNs would converge within 30-40 epochs. The ideal epoch size was therefore set to 40. The ideal batch size for the TCN models was found to be 64, having said this the performance difference between the different batch sizes was not significant. The ideal batch size for TCNs is similar to that of the LSTMs where the majority of LSTM models used 64 as the optimal batch size. Both models therefore benefit from faster computation as a result of larger batch sizes and better generalisation of the models.

The ideal learning rate for all models was 0.01. When compared to the LSTM models, this is higher than the ideal learning rate used in all the LSTM models. The larger learning rate for the TCNs allowed the model to reach optimal performance faster than the LSTM models. The TCN models found that within the range of 12-168 hours, 24 and 48 hours were the ideal input window sizes. The TCN models therefore preferred short input sequences. This preference for shorter input sequence lengths is due to the TCN having a long memory. The manual tuning process found that either 1 or 2 layers are the ideal number of layers. The ideal number of filters ranged from 32 to 128 for each of the TCN models. Finally, all the TCN models suffered from overfitting and poor generalisation with low training losses and comparatively higher validation losses. Dropout was therefore included in all the models, with dropout ranging from 0 to 0.5.

| **Parameter** | **Value Type** | **Type** | **Search Space** | **Optimal** |
|---|---|---|---|---|
| *Activation Function* | Categorical | Algorithm | *tanh*, *linear*, *relu* | *relu* |
| *# of Epochs* | Discrete | Training | [20, 30, 40, 50] | 40 |
| *Batch Size* | Discrete | Training | [32, 64, 125] | 64 |
| *Learning Rate* | Continuous | Training | (0.0001, 0.01) | 0.01 |
| *Layer Normalisation* | Categorical | Training | True/False | True |
| *Optimiser* | Categorical | Structure | Adam | Adam |
| *Input Window Size (hours)* | Discrete | Structure | [12, 24, 48, 72, 168] | [24, 48] |
| *# of Layers* | Discrete | Structure | [1, 2] | [1, 2] |
| *# of Filters* | Discrete | Structure | [32, 64, 128] | [32, 64, 128] |
| *Dilations* | Discrete | Structure | [1, 2, 4, 8, 16, 32, 64] | [1, 2, 4, 8, 16, 32, 64] |
| *Dropout Rate* | Continuous | Regularisation | [0, 0.1,...,0.5] | [0, 0.1,..., 0.5] |

Table 4.6: Optimal hyper-parameters found through manual tuning

#### Random Search Results

The range of continuous and values of categorical parameters found in section 4.2.2 were used as the search space for the random search HPO process. The results of the HPO process; the optimal

parameters for the TCN model for each station are shown in table 4.7 below.

As found during the manual tuning step, the activation function, number of epochs, layer normalisation, dilations and optimiser were set to the values shown in table 4.6 above. From table 4.7, we can see that nineteen of the twenty-one stations only needed a single layer for the best results, whereas the remaining two stations required two layers. The addition of the extra layer did not result in a significant change in performance. Twenty stations found the ideal input window size to be 48 hours with one station using 24 hours. When compared to the input window size preferred by the LSTM models shown in table 4.7, it is clear that TCNs need a shorter input size than LSTMs as a result of having a longer memory. The filters used for each of the models range from 32 to 128. Since the filters capture the temporal dependencies, the range in filters suggests varying complexities of temporal dependencies across stations. We can see that the majority of the TCN models use 64 filters to capture temporal dependencies, whereas the remaining models use 32, and 128 filters. What is interesting to note is that for stations 7 and 8, the models required 2 layers, and the maximum number of filters to capture the dependencies found within the data. This suggests that the data at stations 7 and 8 have more complex temporal dependencies than the other stations.

| **Station** | *# of Layers* | *Input Window Size* | *Dropout* | *Filters* |
|---|---|---|---|---|
| 1 | 1 | 24 | 0.4 | 64 |
| 2 | 1 | 48 | 0.3 | 64 |
| 3 | 1 | 48 | 0.3 | 64 |
| 4 | 1 | 48 | 0.4 | 64 |
| 5 | 1 | 48 | 0.3 | 64 |
| 6 | 1 | 48 | 0.2 | 64 |
| 7 | 2 | 48 | 0.2 | 128 |
| 8 | 2 | 48 | 0.3 | 128 |
| 9 | 1 | 48 | 0.3 | 32 |
| 10 | 1 | 48 | 0.2 | 64 |
| 11 | 1 | 48 | 0 | 64 |
| 12 | 1 | 48 | 0.5 | 64 |
| 13 | 1 | 48 | 0.1 | 128 |
| 14 | 1 | 48 | 0.2 | 64 |
| 15 | 1 | 48 | 0.3 | 32 |
| 16 | 1 | 48 | 0.3 | 32 |
| 17 | 1 | 48 | 0.3 | 64 |
| 18 | 1 | 48 | 0.2 | 64 |
| 19 | 1 | 48 | 0.1 | 64 |
| 20 | 1 | 48 | 0.2 | 128 |
| 21 | 1 | 48 | 0.3 | 64 |

Table 4.7: Table showing optimal parameters for each TCN Model

### 4.2.3 WGN

**Manual Tuning**

The results of the manual tuning performed on the WGN model are shown below in table 4.8. The ideal number of batches to train the WGN model was 15 000. The ideal batch size was found to be 32 which is a smaller batch size than the batch size of 64 that the majority of the LSTM models used. The low batch size of 32 resulted in better generalisation in comparison to the higher batch sizes. The learning rate of the WGN model, 0.01, is significantly greater than that of the LSTM models. This resulted in the WGN model converging faster than the LSTM models.

Similarly to the GWN model, the WGN model uses a shorter input window size (48 hours) than the size used in its corresponding LSTM model (72 and 168 hours). This provides further evidence that capturing spatial dependencies in addition to temporal dependencies results in an ST-GNN model requiring less temporal information to make accurate weather predictions. The mask length in the WGN model indicates the number of historical observations to use as context when making predictions. This feature was found to have little to no impact on the model's performance. It was found that 2 graph convolutional layers resulted in the best performance. Finally, the ideal number of hidden neurons in the WGN model was found to be either 10 or 20 during the manual tuning process.

| Parameter | Value Type | Type | Search Space | Optimal Values |
|---|---|---|---|---|
| *# of Batches* | Discrete | Training | [10000, 12500, 15000] | 15000 |
| *Batch Size* | Discrete | Training | [32, 64, 125, 250] | 32 |
| *Learning Rate* | Continuous | Training | (0.001-0.01) | 0.01 |
| *Input Window Size (hours)* | Discrete | Training | [12, 24, 48, 72, 168] | 48 |
| *Mask Length* | Discrete | Training | [0, 12, 24, 48, 72] | 0 |
| *# of Graph Convolution Layers* | Discrete | Structure | [1, 2, 3] | 2 |
| *# of hidden neurons* | Discrete | Structure | [10, 20, 30, 40, 50] | [10, 20] |

Table 4.8: Table showing hyper-parameters of WGN model

The next set of parameters that were tuned related to the adjacency matrix. The parameters and their ideal values are shown in table 1.6.1. The adjacency matrix could either be randomly initialised or initialised using the Gaussian Kernel distance between nodes. It was found that the WGN model performed better when the adjacency matrix was initialised using the distance between the nodes. These results indicate that the WGN model requires predefined knowledge of the graph structure which is a limitation of the model. Finally, we found that the optimal value for $r$, the parameter controlling the number of connections in the adjacency matrix, was 1. This value indicates that the full dense adjacency matrix is used with connections between all stations.

| Adjacency Matrix | Value Type | Type | Search Space | Optimal Values |
|---|---|---|---|---|
| Initialisation | Categorical | Algorithm | [Random, Distance-Based] | Distance-Based |
| Adjacency Matrix Rank (r) | Discrete | Algorithm | [1, 1.5, 2] | 1 |

Table 4.9: Adjacency Matrix Hyper-Parameters for WGN model

**Random Search Results**

The range of continuous and values of categorical parameters found in section 4.2.3 were used as the search space for the random search HPO process. Of the 7 possible parameters, 5 were tuned during manual tuning. From the WGN search space shown in table 4.10, all 4 possible configurations were validated over two of the total 27 splits. The results of the HPO process, the optimal parameters for the WGN model, are shown in table 4.10.

| Parameter | Value Type | Type | Search Space | Optimal |
|---|---|---|---|---|
| *Input Window Size (hours)* | Discrete | Training | [24, 48] | 24 |
| *# of hidden neurons* | Discrete | Structure | [10, 20] | 10 |

Table 4.10: Optimal hyper-parameters found through manual tuning

The optimal input window size is 48 hours' worth of historical weather observations. Since the WGN model follows the architecture of an LSTM cell, we would think that the ideal input window size would be similar to that of the LSTM models. However, the WGN model uses a shorter input window size than the LSTM models which generally used input window sizes of 72 and 168 hours. What is important to note alongside the aforementioned finding is that the GWN model also used a smaller input window size when compared to the TCN model which captures its temporal dependencies. This provides further evidence that when a model is able to capture spatial and temporal dependencies, it requires less temporal information from the input sequences as the models now have spatial dependencies which further improves their performance.

### 4.2.4 GWN

**Manual Tuning Results**

The results of the manual tuning performed for Graph WaveNet are shown below in table 4.11. The *ReLu* activation function was identified as the best performing activation function. Since Graph WaveNet uses temporal convolutional layers to capture temporal dependencies, it follows that the *ReLu* activation function was the ideal function for both the TCN models and Graph WaveNet. Through the manual tuning process, it was found that a learning rate of 0.001 is ideal. The ideal number of epochs was found to be 30, this is a comparatively shorter number of epochs when compared to the LSTM and TCN models. The ideal batch size through manual tuning was 64. Again, this is the same batch size used by the majority of the TCN models.

Of interest in table 4.11, the ideal input window size is 12 hours. This is the shortest length within the range of 12 - 168 hours. As mentioned earlier, Graph WaveNet consists of temporal convolutional layers which have a long memory, enabling it to take in shorter input sequences as the temporal dependencies are stored within the memory. Furthermore, since GWN also makes use of spatial dependencies through the adjacency matrix, it is possible that the GWN model needs less temporal information as it makes use of the spatial dependencies in addition to the temporal information. The number of hidden units did not significantly affect the model's performance and the ideal values of these parameters were found to be 32. Finally, the GWN model also suffered from overfitting on the training data with low training losses and comparatively higher validation losses. Dropout was included to mitigate overfitting, with an ideal dropout range found to be [0.1, 0.2, 0.3].

| **Parameter** | **Value Type** | **Type** | **Search Space** | **Optimal Values** |
|---|---|---|---|---|
| *Activation Function* | Categorical | Algorithm | *relu* | *relu* |
| *# of Epochs* | Discrete | Training | [20, 40, 50, 60, 70] | 30 |
| *Batch Size* | Discrete | Training | [32, 64, 128, 250] | 64 |
| *Learning Rate* | Continuous | Training | (0.001-0.01) | 0.001 |
| *Weight Decay* | Continuous | Training | 0.0001 | 0.0001 |
| *Input Window Size (hours)* | Discrete | Structure | [12, 24, 48, 72, 168] | 12 |
| *# of Hidden Units* | Discrete | Structure | [22, 32, 42] | [22, 32, 42] |
| *Optimiser* | Categorical | Structure | Adam | Adam |
| *Dropout Rate* | Continuous | Regularisation | [0.1, 0.2, 0.3, 0.4] | [0.1, 0.2, 0.3] |

Table 4.11: Optimal hyper-parameters found through manual tuning

The next set of parameters that were tuned was related to the adjacency matrix. The parameters and their ideal values are shown in table 4.12. The adjacency matrix could either be initialised using random values or a Gaussian Kernel using the geographic distance as the values. The difference in performance was not significant, therefore the randomly initialised adjacency matrix was used because it does not require the additional computation to find the Gaussian Kernel distance between nodes. Lastly, the adaptive-only adjacency matrix was found to result in similar performance as the forwards-backwards, and forward-backwards adaptive adjacency matrices. The adaptive-only adjacency matrix was therefore chosen as the adjacency type because it lowers the computational cost and training time of the model. These results indicate that using diffusion convolution in addition to the learnt-self adaptive adjacency matrix does not improve performance. The GWN model therefore did not need predefined knowledge of the graph structure and can learn spatial dependencies through stochastic gradient descent highlighting the strength of GWN architecture for capturing hidden spatial dependencies with a low computational cost.

| Adjacency Matrix | Search Space | Optimal Value |
|---|---|---|
| Initialisation | Gaussian Kernel Distance, Random | Random |
| Adjacency Type | Adaptive Only | Adaptive |
| | Forwards-Backwards | |
| | Adaptive Forwards-Backwards | |

Table 4.12: Optimal adjacency matrix hyper-parameters found through manual tuning

### Random Search Results

The range of continuous and values of categorical parameters found in section 4.2.4 were used as the search space for the random search HPO process. From the Graph WaveNet search space, all 6 possible random configurations were validated over two of the total 27 splits. Of the 6 possible hyper-parameters, 4 were tuned during the manual tuning process resulting in a small search space consisting of 2 parameters. The results of the HPO process, the optimal parameters for the Graph WaveNet model, are shown in table 4.13 below.

| Parameter | Value Type | Type | Search Space | Optimal |
|---|---|---|---|---|
| *Hidden Units* | Discrete | Structure | [22, 32] | 32 |
| *Dropout* | Discrete | Regularisation | [0.1, 0.2, 0.3] | 0.3 |

Table 4.13: Optimal hyper-parameters found through manual tuning

The notable finding from the random search process is the high dropout. Similar to all prior models, we found dropout was necessary to reduce the model from overfitting on the training data.

## 4.3 LSTM and TCN Model Performance

The SMAPE values recorded for the LSTM and TCN models at each station across all prediction horizons are shown below in figure 4.14. All performance metrics for the LSTM and TCN models can be found in appendix tables A.2 and A.3. We do not include a comparison to the SARIMA models since the performance of the SARIMA models indicated that they are not capable of producing useful predictions and thus provide no benefit to the section below.

| Stations | **SMAPE(LSTM \| TCN)** | | | | |
|---|---|---|---|---|---|
| | 3 | 6 | 9 | 12 | 24 |
| 1 | 16.48 \| **8.94** | 16.97 \| **10.18** | 17.31 \| **10.92** | 17.55 \| **11.49** | 16.48 \| **12.91** |
| 2 | 10.06 \| **6.87** | 11.19 \| **8.23** | 11.93 \| **9.18** | 12.45 \| **9.91** | 10.06 \| **11.71** |
| 3 | 19.78 \| **9.87** | 20.17 \| **11.15** | 20.45 \| **11.98** | 20.69 \| **12.62** | 19.78 \| **14.16** |
| 4 | 15.76 \| **8.64** | 16.31 \| **10.08** | 16.65 \| **10.98** | 16.86 \| **11.62** | 15.76 \| **13.13** |
| 5 | 15.50 \| **8.92** | 16.02 \| **10.25** | 16.40 \| **11.19** | 16.69 \| **11.88** | 15.50 \| **13.43** |
| 6 | 15.11 \| **10.01** | 15.88 \| **11.41** | 16.37 \| **12.30** | 16.67 \| **12.97** | 15.11 \| **14.44** |
| 7 | 13.49 \| **8.95** | 14.38 \| **10.07** | 15.00 \| **10.95** | 15.42 \| **11.61** | 13.49 \| **13.16** |
| 8 | 15.57 \| **8.90** | 16.13 \| **9.95** | 16.43 \| **10.68** | 16.56 \| **11.22** | 15.57 \| **12.48** |
| 9 | 16.34 \| **10.07** | 17.16 \| **11.75** | 17.69 \| **12.89** | 18.03 \| **13.72** | 16.34 \| **15.49** |
| 10 | 18.84 \| **10.53** | 19.39 \| **12.11** | 19.73 \| **13.11** | 19.91 \| **13.83** | 18.84 \| **15.32** |
| 11 | 14.25 \| **8.00** | 14.93 \| **9.69** | 15.46 \| **10.75** | 15.89 \| **11.58** | 14.25 \| **13.37** |
| 12 | 15.00 \| **8.45** | 15.81 \| **9.95** | 16.43 \| **10.95** | 16.90 \| **11.78** | 15.00 \| **13.81** |
| 13 | 13.40 \| **6.58** | 14.08 \| **8.05** | 14.54 \| **9.09** | 14.84 \| **9.86** | 13.40 \| **11.56** |
| 14 | 14.10 \| **8.40** | 14.84 \| **9.79** | 15.27 \| **10.73** | 15.53 \| **11.42** | 14.10 \| **12.90** |
| 15 | 20.88 \| **8.13** | 20.98 \| **9.04** | 21.02 \| **9.60** | 21.04 \| **10.00** | 20.88 \| **10.88** |
| 16 | 14.40 \| **8.52** | 15.14 \| **9.93** | 15.61 \| **10.87** | 15.96 \| **11.56** | 14.40 \| **13.11** |
| 17 | 16.47 \| **7.19** | 16.92 \| **8.33** | 17.21 \| **9.08** | 17.39 \| **9.63** | 16.47 \| **10.96** |
| 18 | 12.10 \| **6.73** | 12.43 \| **7.88** | 12.63 \| **8.65** | 12.74 \| **9.22** | 12.10 \| **10.51** |
| 19 | 16.74 \| **8.28** | 17.24 \| **10.00** | 17.55 \| **11.10** | 17.74 \| **11.89** | 16.74 \| **13.56** |
| 20 | 12.96 \| **8.13** | 13.90 \| **9.48** | 14.56 \| **10.40** | 15.03 \| **11.15** | 12.96 \| **12.92** |
| 21 | 12.92 \| **6.57** | 13.57 \| **7.88** | 13.98 \| **8.79** | 14.20 \| **9.48** | 12.92 \| **11.03** |
| **Average** | 15.25 \| **8.41** | 15.87 \| **9.77** | 16.30 \| **10.69** | 16.58 \| **11.35** | 17.27 \| **12.90** |
| **Difference** | *6.84%* | *6.10%* | *5.61%* | *5.23%* | *4.37%* |

Table 4.14: Table comparing LSTM and TCN models performance for each weather station across all prediction horizons

We can see that the LSTM models produced mediocre results, with SMAPE scores ranging from 10.06% to 21.27% across all prediction horizons. The LSTM achieved a 15.25%, 15.87%, 16.30%, 16.58%, and 17.27% average SMAPE score across the 3, 6, 9, 12, and 24-hour prediction horizons respectively. There is a minor difference of 2.02% between the SMAPE recorded across the shortest 3-hour horizon versus the 24-hour prediction horizon. The difference is not significant and the LSTM provides reasonable predictive power across all prediction horizons.

The TCN models achieved SMAPE scores of 8.41%, 9.77%, 10.69%, 11.35%, and 12.90% for the 3,

6, 9, 12, and 24-hour prediction horizons respectively. The low SMAPE scores across the stations suggest that the TCN models are capable of modelling the temporal dependencies within the data and can accurately and reliably predict the temperature at these prediction horizons. However, we see a 4.49% increase in the average SMAPE value from the 3 to 24-hour prediction horizon which is larger than the 2.02% difference recorded by the LSTM models.

The TCN model outperforms the LSTM model at all stations across all prediction horizons. The TCN model significantly outperforms the LSTM on average across all prediction horizons with performance improvements ranging between 6.84% to 4.37%. This confirms that the TCNs use of causal dilated convolution, allowing for a larger receptive field, is better able to capture temporal dependencies. Both the LSTM and TCN model's performance reduces as the prediction horizon grows suggesting that temporal dependencies become more complex for longer prediction horizons.

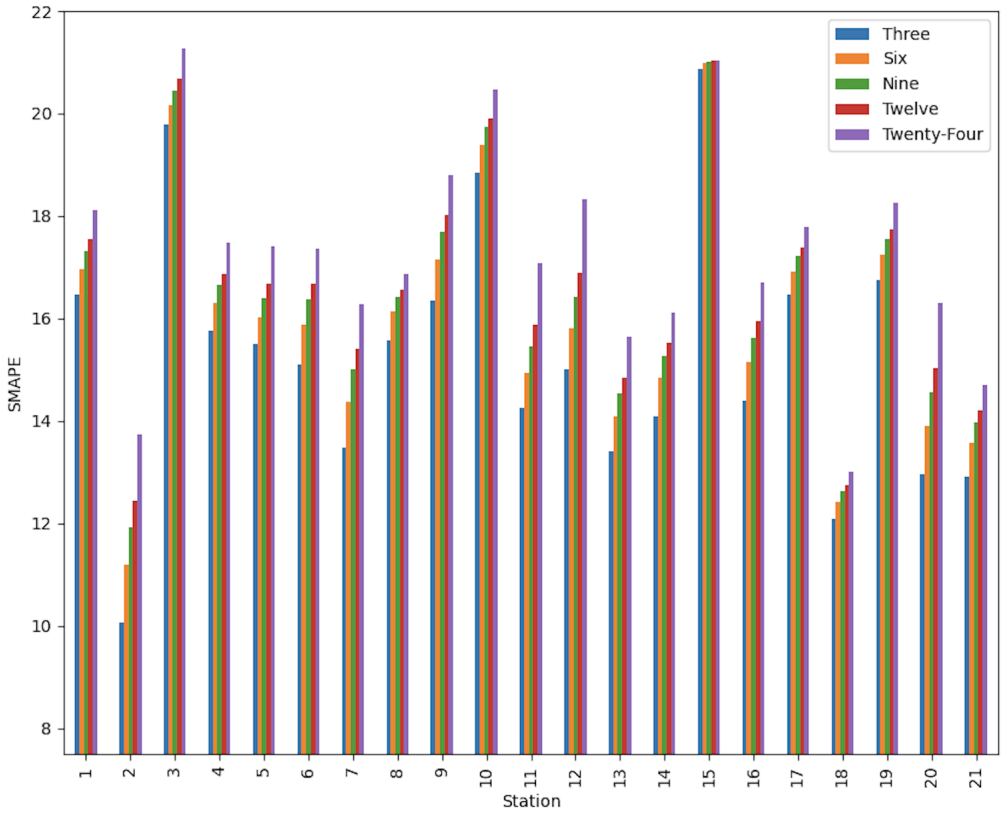


Figure 4.3: Bar graph of LSTM SMAPE values recorded across the prediction horizons

Figure 4.3 shows a bar graph of the SMAPE values recorded at each station across all prediction horizons. We see a general trend of SMAPE scores increasing as the horizon gets longer. The lowest SMAPE scores are achieved on the 3-hour prediction horizon whereas the highest SMAPE

scores are recorded over the 24-hour prediction window. We can see that the LSTM had the worst performance at stations 3, 10, and 15, with high SMAPE scores recorded across all prediction horizons. This suggests that these stations either have complex temporal dependencies or the observed temperatures at these stations are heavily influenced by spatial dependencies. The LSTMs developed for stations 2 and 18 perform the best across all the prediction horizons indicating that these stations have less complex temporal dependencies that are easily captured by the LSTM model. The difference in performance suggests that the complexity of spatial and temporal dependencies varies between stations.

Figure 4.3 below shows the SMAPE values recorded across the prediction horizons at each station.

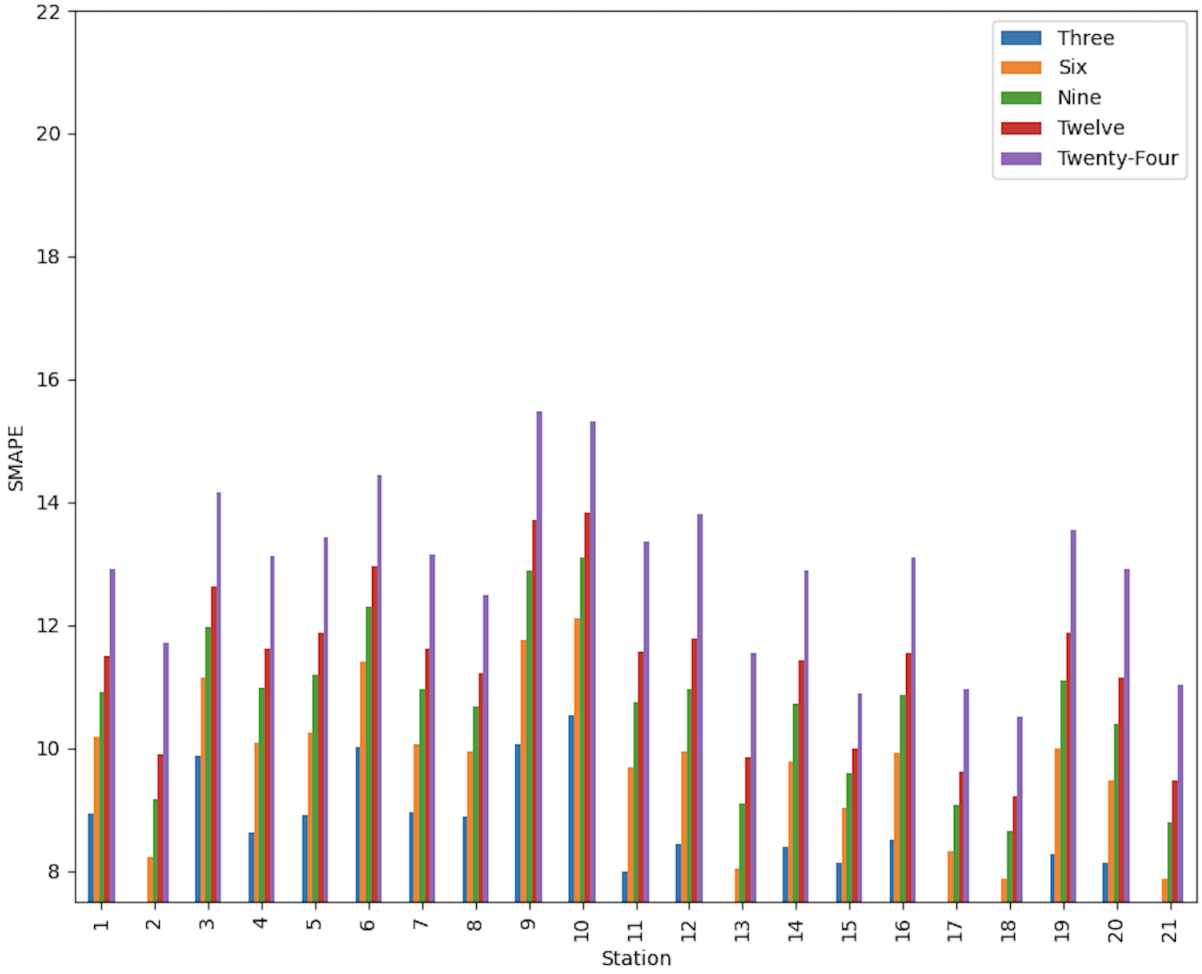


Figure 4.4: Bar graph of TCN SMAPE values recorded across the prediction horizons

The first thing we identify is that there is a larger increase in the SMAPE scores as the prediction

horizon grows longer when compared to the LSTM models. We can see that the SMAPE scores are all below 16%, with some stations predicting temperature with a SMAPE score of less than 8%. Overall, the performance of the TCN models across all the stations appears to be consistently low indicating that the TCN models are capable of reliably and accurately predicting temperature across all the prediction horizons.

TCN models developed for stations 2, 13, 15, 17, 18, and 21 have the lowest SMAPE values. The LSTM models for stations 2, 18, and 21 are also the best performing models suggesting that these stations have less complex temporal dependencies easily captured by both temporal models. These stations also provide a good point of comparison between the LSTM and TCN, where we can clearly see the TCN outperforming the LSTM at the stations where the LSTM has the best performance.

The stations with the worst performing TCN models are stations 3, 6, 9, and 10. These stations also provide a good point of comparison between the LSTM and TCN models. The LSTM models also show poor performance at stations 3 and 10 suggesting that capturing temporal dependencies are not sufficient for accurate predictions at these stations. Stations 9 and 10 also provide a good point of comparison between the LSTM and TCN models, where we see the TCN model outperform the LSTM model at stations where the temporal dependencies are more complex and most likely are heavily influenced by spatial dependencies.

Figure 4.5 shows the box plots of the SMAPE values of the TCN and LSTM models across all five prediction horizons.

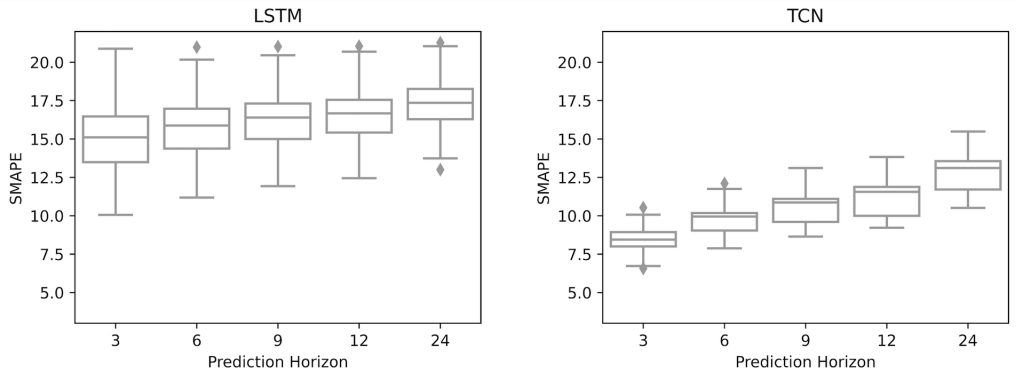


Figure 4.5: Box plot of LSTM and TCN SMAPE values recorded across the prediction horizons

The distribution of the LSTM model's performance at the 3-hour prediction horizon is spread over a longer range of values as shown by the long box plot. This indicates that the LSTM is not able to produce consistently accurate predictions on this horizon. We can also see that the range of SMAPE scores becomes shorter as the prediction horizon grows. The 3 and 24-hour SMAPE values are distributed normally. The 6, 9, and 12 SMAPE values show a slight negative skew which suggests that these horizons consist of a higher frequency of low SMAPE scores. The LSTM has a high number of outliers. We see that the number of outliers increases as the prediction horizon

grows suggesting that LSTM becomes more unstable when predicting further into the future.

The distribution of TCN scores across all prediction horizons is significantly lower than the LSTM. However, an unusual finding is that the TCN model only has outliers on the short prediction horizons i.e. 3 and 6. The spread of scores of the TCN model on the shorter horizons is also significantly smaller indicating that the TCN model produces more reliable, accurate predictions than the LSTM on these short horizons. It is clear from the figure that the TCN performance worsens significantly as the prediction horizon goes on, more so than the LSTM.

Figure 4.6 shows a scatter plot of the SMAPE values on the 24-hour prediction horizon at each of the twenty-one stations. The colour and size represent the SMAPE value recorded at the weather station with a larger dot representing poorer performance i.e. higher SMAPE value.

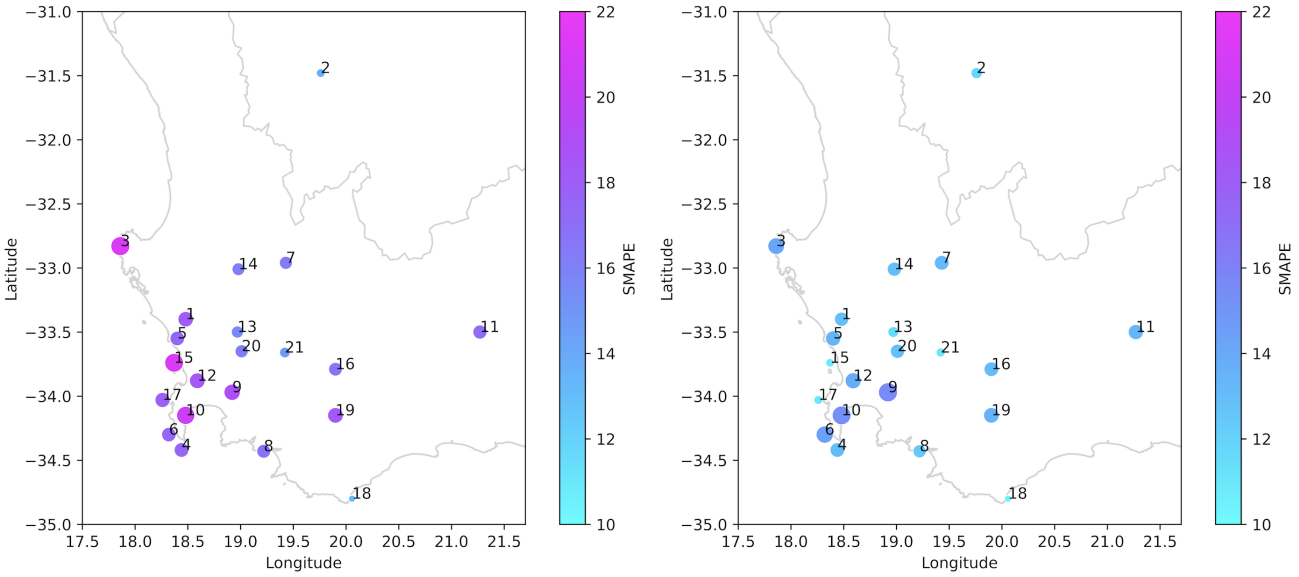


Figure 4.6: Scatter plot of LSTM (left) and TCN (right) SMAPE values recorded across the 24-hour prediction horizon

It is immediately clear that the TCN outperforms the LSTM at all stations indicated by lighter colours and smaller dots. Generally, the stations where the LSTM has poor performance are also where the TCN has larger dots indicating higher SMAPE values. Similarly, stations where the LSTM performs well, are where the TCN also performs well. This provides strong evidence that the TCN is more capable of capturing temporal dependencies than the LSTM. However, stations 15 and 17 do not follow this trend, and we see the TCN model significantly outperform the LSTM models at these stations.

The worst performing models for both stations are generally found along the west coast at stations 3, 6, 4, 9, and 10. Comparatively, the performance of both models is better at stations found further inland. This finding suggests that the temporal dependencies at stations along the west coast are

more complex and most likely heavily influenced by spatial dependencies and weather phenomena from the ocean. The best performing stations for both models are stations 2 and 18 despite these models being geographically distant.

## 4.4 ST-GNN Model Performance

This section describes the results of following the experimental design specified in section 3.6.3. The performance metrics of the ST-GNN models are presented alongside the best performing temporal model i.e. TCN.

### 4.4.1 WGN and GWN Model Performance

The TCN, WGN, and GWN SMAPE values for each station across all prediction horizons are shown in table 4.15 below. The difference row indicates the percentage difference between the TCN and WGN model when compared to the GWN model. All performance metrics for the WGN and GWN models can be found in appendix tables A.4 and A.5.

| Stations | **SMAPE (TCN \| WGN \| GWN)** | | | | |
|---|---|---|---|---|---|
| | 3 | 6 | 9 | 12 | 24 |
| 1 | 8.94 \| 7.10 \| **6.46** | 10.18 \| 9.23 \| **7.99** | 10.92 \| 9.57 \| **8.78** | 11.49 \| 9.51 \| **9.10** | 12.91 \| 9.37 \| **9.48** |
| 2 | **6.87** \| 10.03 \| 9.41 | **8.23** \| 14.33 \| 11.57 | **9.18** \| 15.82 \| 12.84 | **9.91** \| 15.57 \| 13.51 | **11.71** \| 15.41 \| 14.53 |
| 3 | 9.87 \| 5.54 \| **5.28** | 11.15 \| 7.01 \| **6.02** | 11.98 \| 7.55 \| **6.48** | 12.62 \| 7.62 \| **6.75** | 14.16 \| 7.50 \| **7.20** |
| 4 | 8.64 \| 4.88 \| **4.81** | 10.08 \| 6.56 \| **5.77** | 10.98 \| 7.31 \| **6.42** | 11.62 \| 7.33 \| **6.78** | 13.13 \| 7.26 \| **7.16** |
| 5 | 8.92 \| 5.45 \| **4.91** | 10.25 \| 7.22 \| **5.89** | 11.19 \| 7.91 \| **6.55** | 11.88 \| 7.84 \| **6.97** | 13.43 \| 8.16 \| **7.72** |
| 6 | 10.01 \| 5.18 \| **4.96** | 11.41 \| 6.65 \| **5.77** | 12.30 \| 7.17 \| **6.32** | 12.97 \| 7.09 \| **6.66** | 14.44 \| 7.19 \| **7.11** |
| 7 | **8.95** \| 10.14 \| 9.76 | **10.07** \| 13.60 \| 11.77 | **10.95** \| 14.88 \| 12.79 | **11.61** \| 14.71 \| 13.24 | **13.16** \| 13.88 \| 13.94 |
| 8 | 8.90 \| 4.86 \| **4.56** | 9.95 \| 6.65 \| **5.38** | 10.68 \| 7.17 \| **5.89** | 11.22 \| 7.10 \| **6.19** | 12.48 \| 6.77 \| **6.66** |
| 9 | 10.07 \| 8.72 \| **7.84** | 11.75 \| 11.53 \| **9.82** | 12.89 \| 12.23 \| **10.88** | 13.72 \| 12.39 \| **11.36** | 15.49 \| 12.46 \| **12.12** |
| 10 | 10.53 \| 8.34 \| **7.34** | 12.11 \| 11.06 \| **8.93** | 13.11 \| 11.61 \| **9.70** | 13.83 \| 11.48 \| **10.04** | 15.32 \| 10.63 \| **10.40** |
| 11 | 8.00 \| 7.28 \| **7.11** | 9.69 \| 10.26 \| **8.53** | 10.75 \| 11.34 \| **9.39** | 11.58 \| 11.32 \| **9.85** | 13.37 \| 11.02 \| **10.88** |
| 12 | 8.45 \| 4.85 \| **5.06** | 9.95 \| 6.77 \| **6.21** | 10.95 \| 7.54 \| **7.00** | 11.78 \| 7.82 \| **7.49** | 13.81 \| 8.50 \| **8.16** |
| 13 | 6.58 \| 5.96 \| **5.76** | 8.05 \| 8.69 \| **7.39** | 9.09 \| 9.68 \| **8.34** | 9.86 \| 9.57 \| **8.80** | 11.56 \| 9.67 \| **9.45** |
| 14 | 8.40 \| 7.94 \| **8.02** | **9.79** \| 10.95 \| 10.08 | **10.73** \| 11.67 \| 11.22 | **11.42** \| 11.43 \| 11.65 | 12.90 \| 10.85 \| **11.79** |
| 15 | 8.13 \| 5.79 \| **5.23** | 9.04 \| 7.64 \| **6.25** | 9.60 \| 8.00 \| **6.78** | 10.00 \| 7.89 \| **6.99** | 10.88 \| 7.41 \| **7.29** |
| 16 | 8.52 \| 7.25 \| **7.27** | 9.93 \| 10.37 \| **9.04** | 10.87 \| 11.36 \| **10.01** | 11.56 \| 11.11 \| **10.41** | 13.11 \| 10.64 \| **10.94** |
| 17 | 7.19 \| 5.20 \| **4.66** | 8.33 \| 7.14 \| **5.79** | 9.08 \| 7.59 \| **6.43** | 9.63 \| 7.60 \| **6.75** | 10.96 \| 7.55 \| **7.27** |
| 18 | 6.73 \| 5.91 \| **5.01** | 7.88 \| 7.91 \| **6.08** | 8.65 \| 8.37 \| **6.74** | 9.22 \| 8.18 \| **7.08** | 10.51 \| 8.02 \| **7.54** |
| 19 | 8.28 \| 8.03 \| **7.73** | 10.00 \| **11.39** \| 9.78 | 11.10 \| 12.22 \| **10.89** | 11.89 \| 11.95 \| **11.33** | 13.56 \| 11.37 \| **11.73** |
| 20 | 8.13 \| 6.53 \| **6.07** | 9.48 \| 9.11 \| **7.71** | 10.40 \| 9.92 \| **8.61** | 11.15 \| 9.78 \| **9.03** | 12.92 \| 9.93 \| **9.67** |
| 21 | 6.57 \| 6.55 \| **6.76** | **7.88** \| 9.71 \| 8.53 | **8.79** \| 10.96 \| 9.56 | **9.48** \| 10.99 \| 10.02 | 11.03 \| 10.30 \| **10.52** |
| **Average** | 8.41 \| 6.74 \| **6.39** | 9.77 \| 8.75 \| **7.82** | 10.69 \| 9.99 \| **8.65** | 11.35 \| 9.92 \| **9.05** | 12.90 \| 9.71 \| **9.60** |
| % Diff. | 2.02 \| 0.35 \| 0.00 | 1.95 \| 0.93 \| 0.00 | 2.04 \| 1.34 \| 0.00 | 3.30 \| 0.87 \| 0.00 | 3.19 \| 0.11 \| 0.00 |

Table 4.15: Table comparing LSTM, TCN, and WGN results across prediction horizons

From table 4.15 we can see that the GWN model on average outperforms both the WGN and TCN models across all prediction horizons with SMAPE values of 6.39%, 7.82%, 8.65%, 9.05% and

9.60%. This is expected because the GWN uses the superior temporal model i.e. TCN to capture temporal dependencies in addition to being able to capture hidden spatial dependencies. The WGN model on average outperforms the TCN model across all prediction horizons despite using the weaker temporal LSTM to capture temporal dependencies. This is due to the WGN including spatial information through the learnt self-adaptive adjacency matrix. These results provide clear evidence that capturing spatial dependencies in addition to temporal dependencies significantly improves performance.

On average, the difference between the performance of the WGN and GWN model is small despite the WGN model using the weaker LSTM temporal model to capture temporal dependencies. This provides additional evidence that capturing spatial dependencies significantly improves performance even when a weaker temporal model is used. The difference in performance between the TCN and GWN models however is significant. We see that the performance difference grows larger as the prediction horizon grows. This suggests that capturing temporal dependencies alone produces significantly weaker results for longer prediction horizons and highlights the importance of capturing spatial dependencies for longer prediction horizons.

The most interesting findings from table 4.15 relate to the TCN. The TCN on average outperforms both ST-GNN models at stations 2, 7, 14, and 21. The TCN outperforms the WGN and GWN model across the 6, 9, and 12 prediction horizons. However, the TCN only outperforms the GWN and WGN at two stations on the 24-hour prediction horizon. These results align with our finding that spatial dependencies are significantly more influential on performance at longer prediction horizons. Since the GWN model utilised the TCN architecture and the TCN outperformed both ST-GNNs on average at stations 2, 7, 14 and 21, it is likely that including spatial dependencies at these stations actually reduces the performance of the models.

From table 4.15 we can see an unusual trend across the 9, 12, and 24 hour prediction horizons. In all the previous models, the model's performance worsens as the prediction horizon grows, however, the WGN results show that the model performs on average, better on the 24 hour prediction horizon than when compared to the 9 and 12-hour horizon. Following on from our finding that spatial dependencies significantly improve performance on longer prediction horizons, the WGN captures temporal dependencies using the weaker LSTM architecture. It is likely that the WGN shows stronger performance on the longest horizons by utilising spatial information and weaker performance on the shorter horizons as spatial information doesn't significantly improve performance and the LSTM component of the WGN reduces performance.

The WGN SMAPE values recorded across the prediction horizons at each weather station are shown below in figure 4.7.

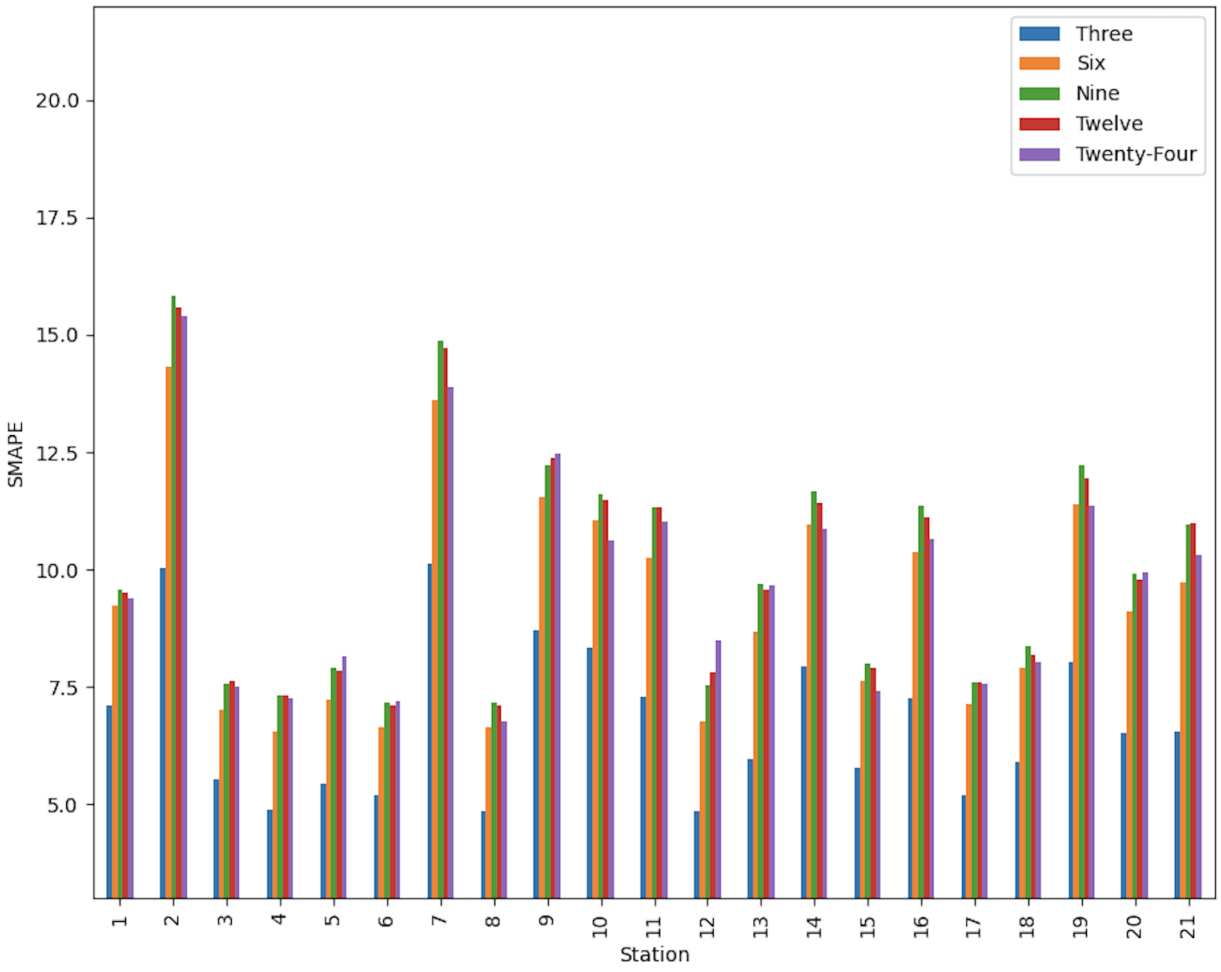


Figure 4.7: Bar chart of WGN SMAPE values recorded at each station across the prediction horizons

We can see that there is a large variation in the performance of the WGN models between stations. We see the strongest performance at stations 3, 4, 5, 6, 8, 15, 17, and 18 suggesting that capturing temporal and spatial dependencies results in accurate predictions at these stations. The WGN model shows average performance at stations 10, 11, 14, 16, 19, and 20 suggesting that the temporal and spatial dependencies at these stations are slightly more complex than the aforementioned stations. The weakest performance of the WGN model is recorded at stations 2 and 7 where we found that including spatial information reduces the accuracy of predictions.

The GWN SMAPE values recorded across the prediction horizons at each weather station are shown below in figure 4.7.

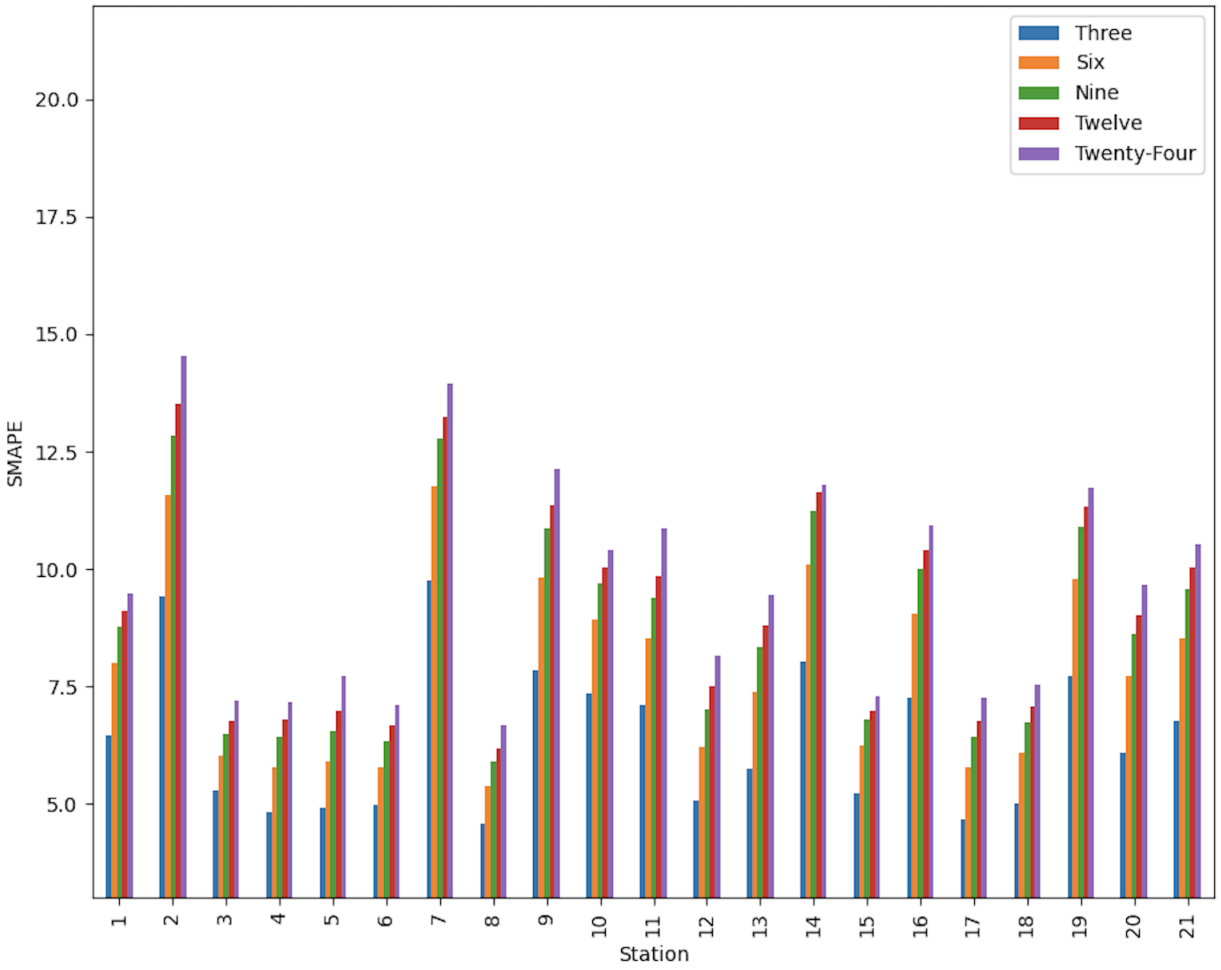


Figure 4.8: Bar chart of GWN SMAPE values at each station recorded across the prediction horizons

From the figure, we can see that the GWN model performs best at stations 3, 4, 5, 6, 8, 15, 17, and 18. The performance of the WGN model is similar at these stations providing further evidence that both ST-GNN models are able to easily capture spatial and temporal dependencies at these stations. The stations with the weakest performance are stations 2, 7, 10, 14, and 19. These results are also similar to the WGN results suggesting that the temporal and spatial dependencies are complex. From figures 4.8 and 4.7, we can see that the performance of the GWN and WGN models are similar indicating that both ST-GNN models have similar capabilities of capturing temporal and spatial dependencies at all stations.

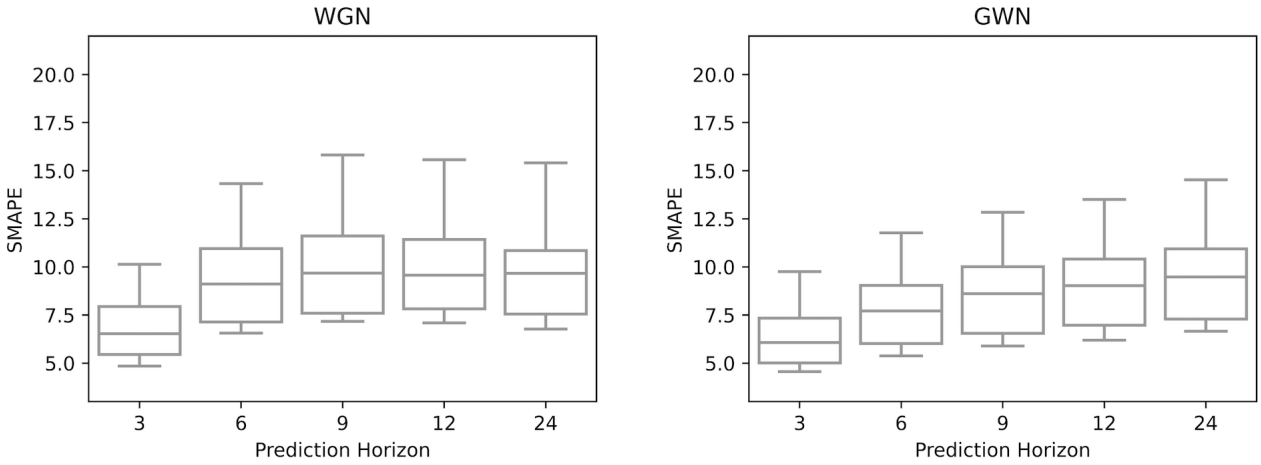


Figure 4.9: Box plots of WGN and GWN models' SMAPE across the prediction horizons

The box plot of the WGN average performance across all five prediction horizons is shown in 4.9. This plot is the most unique of the previous ones considering that we do not see a linear increase in the distribution of SMAPE metrics recorded against the prediction horizons, the reason for which was elucidated earlier. The box plot also shows that the distribution of SMAPE scores across all stations at each prediction horizon is positively skewed with most of the SMAPE metrics for each weather station falling closer to the minimum of the SMAPE value for that particular prediction horizon.

The GWN model has the best and most similar performance on the 3-hour prediction horizon. We see that the range of the distribution of performance increases as the prediction horizon which is a similar finding to the TCN model. This makes sense considering that the GWN uses the TCN to capture temporal dependencies. In contrast to the LSTM and TCN box plots, we see that the WGN and GWN models do not have any outliers suggesting that their performance across each prediction horizon is more consistent.

The maps shown in figure 4.10 below show the SMAPE scores for the WGN and GWN model over the 24 hour prediction horizon over a map of South Africa. Colour and size are used to represent SMAPE values with larger dots representing poorer performance i.e. higher SMAPE values. The difference in performance between the stations is barely noticeable. However, is interesting to see that the performance of the WGN and GWN models are similar at each station despite using different architectures to capture temporal dependencies.

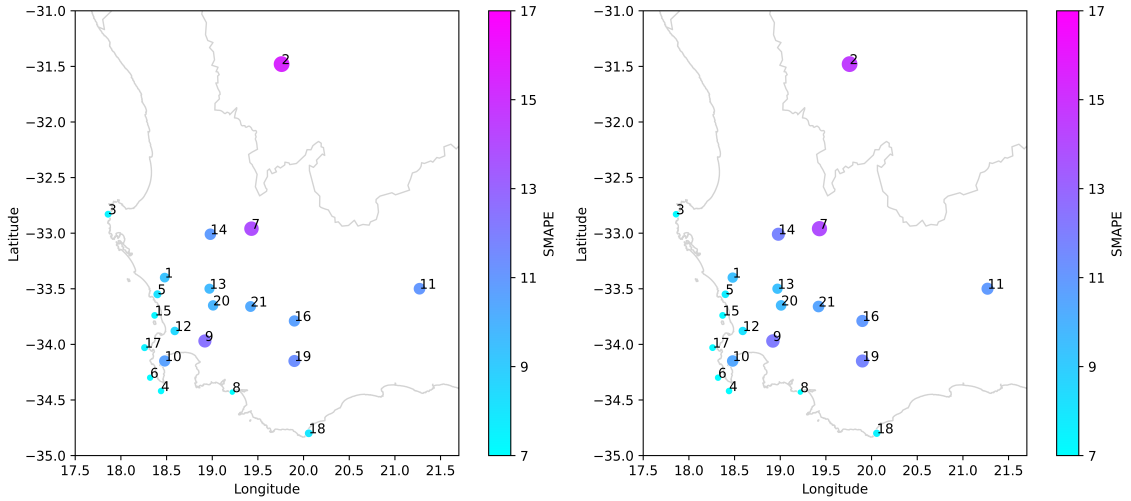


Figure 4.10: Scatter plot of WGN (left) and GWN (right) SMAPE scores at each weather station over the 24 hour prediction horizon

The TCN and LSTM models produced weaker results at stations found along the west coast than at stations further inland. From figure 4.10, we can see that the ST-GNN models have more accurate predictions along the west coast than stations further inland. The stations along the coast are geographically closer together than those found inland. Since the ST-GNN models capture spatial dependencies, it stands to reason that the ST-GNN models perform better where there are more stations located closer together which enables them to utilise spatial dependencies to further improve their performance. Furthermore, the ST-GNNs had the worst performance stations 2 and 7. These models do not have many nearby stations from which to use spatial dependencies to improve performance. The inclusion of spatial information at isolated stations therefore reduces the accuracy of predictions.

## 4.5 Results Summary

This section presents the summarised final results of the four deep learning models implemented across all prediction horizons.

Table 4.16 shows the average SMAPE and normalized MSE results for each deep learning model and the standard deviation of the results over 10 independent runs. The results are presented across all five prediction horizons. The LSTM model records the worst performance across all prediction horizons with an average accuracy of 16.25% over all prediction horizons. The LSTM model is the least stable model with high standard deviations between SMAPE metrics ranging between 2.52 to 4.12. The TCN model shows a significant accuracy improvement over the LSTM model with an average prediction accuracy of 10.62%. The TCN model performs well, with only a small percentage improvement in accuracy between itself and the GWN and WGN models. The standard deviations across the prediction horizons suggest that the model is also more stable than the LSTM model.

| Model | Prediction Horizon | | | | | Average |
|---|---|---|---|---|---|---|
| | 3 | 6 | 9 | 12 | 24 | |
| LSTM $\frac{SMAPE}{MSE}$ | 15.25 ±2.54 / 0.005 | 15.87 ±2.38 / 0.006 | 16.30 ±2.28 / 0.006 | 16.58 ±2.22 / 0.007 | 17.27 ±2.11 / 0.008 | 16.25% / 0.006 |
| TCN $\frac{SMAPE}{MSE}$ | 8.41 ±1.15 / 0.002 | 9.77 ±1.22 / 0.002 | 10.69 ±1.26 / 0.003 | 11.35 ±1.30 / 0.003 | 12.90 ±1.39 / 0.003 | 10.62% / 0.003 |
| WGN $\frac{SMAPE}{MSE}$ | 6.74 ±1.63 / 0.002 | 8.75 ±2.32 / 0.003 | 9.99 ±2.54 / 0.004 | 9.92 ±2.48 / 0.004 | 9.71 ±2.32 / 0.005 | 9.02% / 0.004 |
| GWN $\frac{SMAPE}{MSE}$ | **6.39 ±1.56** / 0.002 | **7.82 ±2.00** / 0.002 | **8.65 ±2.20** / 0.003 | **9.05 ±2.27** / 0.003 | **9.60 ±2.35** / 0.004 | **8.30%** / 0.003 |

Table 4.16: Table comparing deep learning models' SMAPE and MSE scores across the prediction horizons

Figure 4.11 below shows the distribution of deep learning models' performance across all prediction horizons.

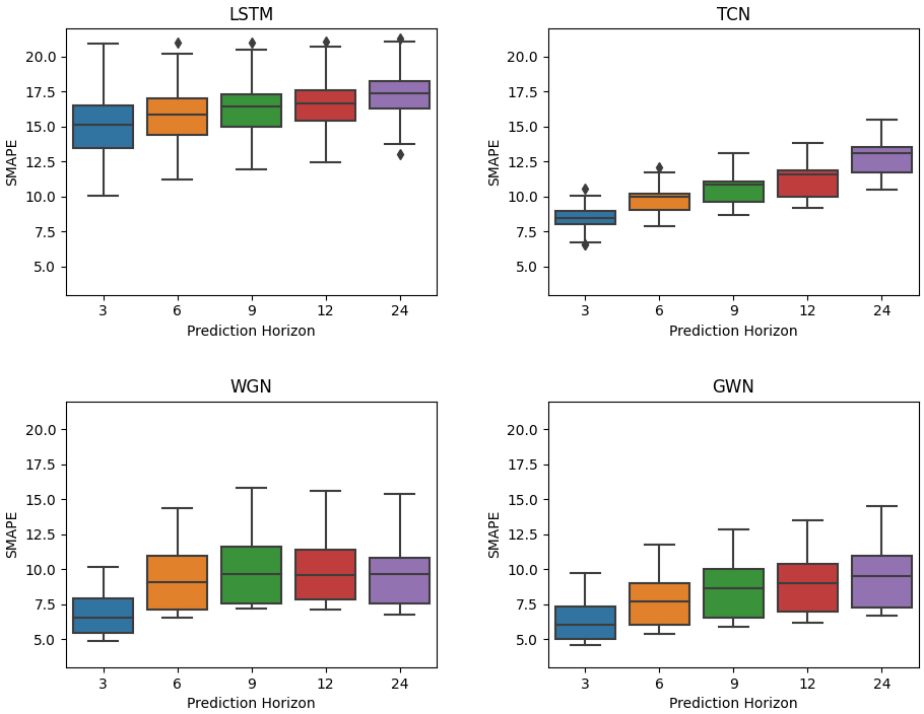


Figure 4.11: Box plots showing SMAPE scores of deep learning models over all prediction horizons

It is clear that the LSTM has the worst performance across the deep learning models, additionally, the LSTM plot has outliers across 4 of the 5 prediction horizons indicating that is it unstable. We can see that the TCN has the smallest distribution of scores of all the deep learning models in

addition to low SMAPE scores. The TCN has the best stability across all prediction horizons. The GWN and WGN distributions are similar. Both ST-GNN models show the best performance with the lowest SMAPE scores across the prediction horizons. However, it is clear that the TCN model is more robust and stable when predicting temperature across all the prediction horizons. This suggests that the inclusion of spatial information reduces model stability.

Finally, figure 4.12 below shows the best performing model for each individual station averaged across all prediction horizons, averaged over 10 runs. Figure 4.12 provides further insight into the results presented in table 4.11. The first important thing to note is that the WGN model was not the best performing model on average at any of the weather stations. Having said this, it did outperform the other models at certain stations at particular prediction horizons, just not on average. The GWN model was on average the best performing model at 17 of the 21 weather stations. We can see that the GWN model outperformed the TCN and WGN models at all stations along the coast. The TCN model was on average the best performing model at the remaining 4 weather stations. We can see that the TCN generally performs better at stations further north. Finally, it is interesting to note that the most accurate predictions across the weather stations and predictions horizons are a mixture of both ST-GNN and temporal deep learning architectures.

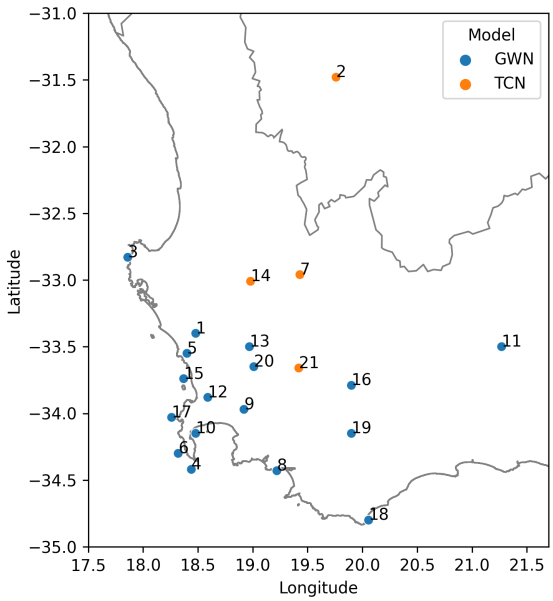


Figure 4.12: Plot showing best performing deep learning model at each weather station

## 4.6 Results Analysis

Having presented and summarised the results, we now provide an explanation for the performance of each model at every individual weather station. We do this by analysing the data sets of each weather station in combination with the predictions of the model. We are particularly interested in why the TCN model outperformed both ST-GNN models at stations 2, 7, 14, and 21.

Table 4.17 below shows the descriptive statistics of the maximum temperature observations at each station. It must be noted that the $Min.Temp$ column indicates the minimum of the recorded maximum temperature. The table also shows the SMAPE metric of each deep learning model averaged over all five prediction horizons. The values in bold for the models indicate the lowest SMAPE value among the models i.e. the best performing model at each station.

| **Station** | Max. Temp | Min. Temp | Mean | Std. | LSTM | TCN | GWN | WGN |
|---|---|---|---|---|---|---|---|---|
| 1 | 43.0 | 1.1 | 16.92 | 5.97 | 17.28 | 10.88 | **8.36** | 8.95 |
| 2 | **40.8** | **-6.0** | 16.79 | **8.46** | 11.87 | **9.18** | 12.37 | 14.23 |
| 3 | 34.8 | 5.6 | 15.13 | 3.21 | 20.47 | 11.95 | **6.34** | 7.04 |
| 4 | 38.1 | 4.2 | 15.89 | 3.59 | 16.61 | 10.89 | **6.18** | 6.66 |
| 5 | 40.5 | 4.5 | 18.28 | 4.72 | 16.40 | 11.13 | **6.40** | 7.31 |
| 6 | 36.1 | 6.3 | 16.21 | 3.14 | 16.27 | 12.22 | **6.16** | 6.65 |
| 7 | **37.5** | **-4.2** | 14.33 | **8.71** | 14.91 | **10.94** | 12.29 | 13.44 |
| 8 | 36.2 | 4.6 | 17.13 | 3.69 | 16.31 | 10.64 | **5.73** | 6.51 |
| 9 | 41.8 | -0.2 | 16.69 | 5.87 | 17.60 | 12.78 | **10.40** | 11.46 |
| 10 | 41.3 | 0.9 | 15.63 | 5.54 | 19.67 | 12.98 | **9.28** | 10.62 |
| 11 | 42.2 | -0.8 | 17.54 | 5.09 | 15.52 | 10.67 | **9.15** | 10.24 |
| 12 | 38.7 | 5.1 | 17.77 | 4.89 | 16.49 | 10.98 | **6.78** | 7.09 |
| 13 | 42.4 | 0.5 | 18.38 | 5.86 | 14.50 | 9.02 | **7.94** | 8.71 |
| 14 | **43.2** | **-1.0** | 17.68 | **7.69** | 15.16 | **10.44** | 10.55 | 10.56 |
| 15 | 39.5 | 2.2 | 16.61 | 4.12 | 20.99 | 9.53 | **6.50** | 7.34 |
| 16 | 43.2 | -0.8 | 17.75 | 5.82 | 15.56 | 10.79 | **9.53** | 10.14 |
| 17 | 41.3 | 3.0 | 17.85 | 4.66 | 17.15 | 9.03 | **6.17** | 7.01 |
| 18 | 32.4 | 1.8 | 17.01 | 4.25 | 12.58 | 8.59 | **6.48** | 7.67 |
| 19 | 44.4 | -1.5 | 16.80 | 5.29 | 17.50 | 10.96 | **10.29** | 10.99 |
| 20 | 43.3 | 1.7 | 18.14 | 5.04 | 14.55 | 10.41 | **8.21** | 9.05 |
| 21 | **43.3** | **-1.6** | 17.92 | **8.02** | 13.87 | **8.75** | 9.07 | 9.70 |

Table 4.17: Table showing descriptive statistics on maximum temperature for each station along with model performance

The maximum temperatures observed at each station range between 32.4 and 44.4. The minimum temperatures range between -6 and 6.3. We can see that the mean of the temperatures observed at each station is generally similar. However, we see a high variation in the standard deviation at each

station ranging from 3.14 to 8.71. We are particularly interested in stations 2, 7, 14, and 21 where the TCN model outperformed the LSTM, WGN, and GWN models. The maximum and minimum temperatures for these stations are highlighted in bold. We can see that these stations have the highest standard deviations and all have negative minimum temperatures. These findings suggest that the ST-GNN models have poorer performance at stations where the temperature observations vary significantly from the mean temperature.

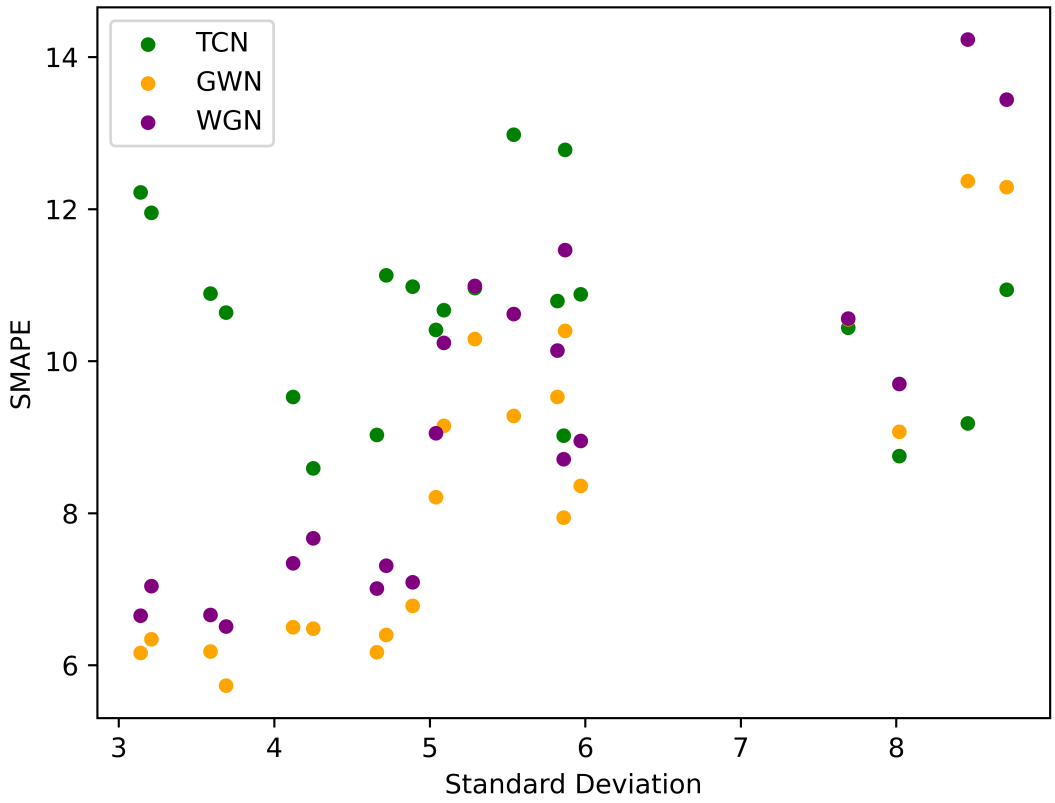


Figure 4.13: Scatter plot of the standard deviation of TCN, GWN, and WGN models against SMAPE

We plot the standard deviation and model performance from table 4.17 in figure 4.13. It is clear that at lower standard deviations the WGN and GWN models significantly outperform the TCN model. We see the aforementioned trend of the performance of ST-GNN models weakening as the standard deviation increases. Finally, we see the TCN model has the best performance at the highest standard deviations.

In order to further investigate the findings above, we provide the mean and standard deviations of the prediction performance of each model at each individual station in table 4.18 below.

| **Station** | Mean | Std. | LSTM | TCN | GWN | WGN |
|---|---|---|---|---|---|---|
| 1 | 16.92 | 5.97 | 17.20 ±6.47 | 16.90 ±5.98 | 16.95 ±5.77 | 16.91 ±5.93 |
| 2 | 16.79 | 8.46 | 17.02 ±7.98 | 16.56 ±8.50 | 16.77 ±5.23 | 16.76 ±5.21 |
| 3 | 15.13 | 3.21 | 16.60 ±3.24 | 15.03 ±2.87 | 15.14 ±2.18 | 15.11 ±2.22 |
| 4 | 15.89 | 3.59 | 15.12 ±4.58 | 15.68 ±3.21 | 15.92 ±3.18 | 15.88 ±3.22 |
| 5 | 18.28 | 4.72 | 17.21 ±5.92 | 17.95 ±4.33 | 18.26 ±4.11 | 18.19 ±4.31 |
| 6 | 16.21 | 3.14 | 17.90 ±5.42 | 16.21 ±2.76 | 16.10 ±3.21 | 16.43 ±3.43 |
| 7 | 14.33 | 8.71 | 15.11 ±6.77 | 14.07 ±7.75 | 14.22 ±5.38 | 14.65 ±4.37 |
| 8 | 17.13 | 3.69 | 16.11 ±4.22 | 17.03 ±3.42 | 17.09 ±3.08 | 17.11 ±3.17 |
| 9 | 16.69 | 5.87 | 17.25 ±7.76 | 16.73 ±6.58 | 16.68 ±5.92 | 16.64 ±5.80 |
| 10 | 15.63 | 5.54 | 16.90 ±5.98 | 15.29 ±5.14 | 15.55 ±4.98 | 16.02 ±5.11 |
| 11 | 17.54 | 5.09 | 19.21 ±7.88 | 17.55 ±6.92 | 17.53 ±5.06 | 17.45 ±5.98 |
| 12 | 17.77 | 4.89 | 18.90 ±6.58 | 17.54 ±4.69 | 17.79 ±4.22 | 17.12 ±4.11 |
| 13 | 18.38 | 5.86 | 16.23 ±7.98 | 18.19 ±6.94 | 18.32 ±5.62 | 18.67 ±5.23 |
| 14 | 17.68 | 7.69 | 18.21 ±7.14 | 17.34 ±7.47 | 17.69 ±5.23 | 17.99 ±5.11 |
| 15 | 16.61 | 4.12 | 17.93 ±6.01 | 16.57 ±3.95 | 16.66 ±4.05 | 16.33 ±4.13 |
| 16 | 17.75 | 5.82 | 18.11 ±6.22 | 17.67 ±6.80 | 17.66 ±5.48 | 17.63 ±5.22 |
| 17 | 17.85 | 4.66 | 15.90 ±4.38 | 17.76 ±4.44 | 17.88 ±4.46 | 17.41 ±4.33 |
| 18 | 17.01 | 4.25 | 17.21 ±5.11 | 16.75 ±4.18 | 16.97 ±4.33 | 16.89 ±4.54 |
| 19 | 16.80 | 5.29 | 17.24 ±7.22 | 16.60 ±6.22 | 16.90 ±5.76 | 16.77 ±5.89 |
| 20 | 18.14 | 5.04 | 16.31 ±6.98 | 18.06 ±7.02 | 18.22 ±5.12 | 18.31 ±5.44 |
| 21 | 17.92 | 8.02 | 15.90 ±9.92 | 17.75 ±7.08 | 18.01 ±5.65 | 18.92 ±5.55 |

Table 4.18: Mean and standard deviations of each models predictions

The mean of the predictions of the TCN, WGN, and GWN models tend to lie closer to the mean of the observed temperature values. We see that the standard deviations of both ST-GNN models tend to be lower than the temporal models at each station. We are particularly interested in the results at stations 2, 7, 14, and 21 where the TCN model outperformed the GWN and WGN models. We can see that although the ST-GNN models' predictions tend to lie closer to the mean, they have a significantly lower standard deviation than the TCN model at the aforementioned stations. Our analysis of the weather data in combination with each model's predictions suggests that the ST-GNN models produce less accurate predictions at stations where the temperature values vary greatly from the mean. It is likely that as a result of developing the temporal models on each station individually, that these models are more capable of learning the greater spread of temperature values at stations with high standard deviations. In contrast, the ST-GNN models tend to perform better where the spread of temperature values around the mean is lower, as these models tend to predict temperature values closer to the mean.

To further investigate the poor performance of the ST-GNN models at stations 2, 7, 14, and 21, we plot the predicted values versus the targets in figures 4.14, 4.15, 4.16, and 4.17 below. The predictions and target values are from the final split of the walk-forward validation test sets.

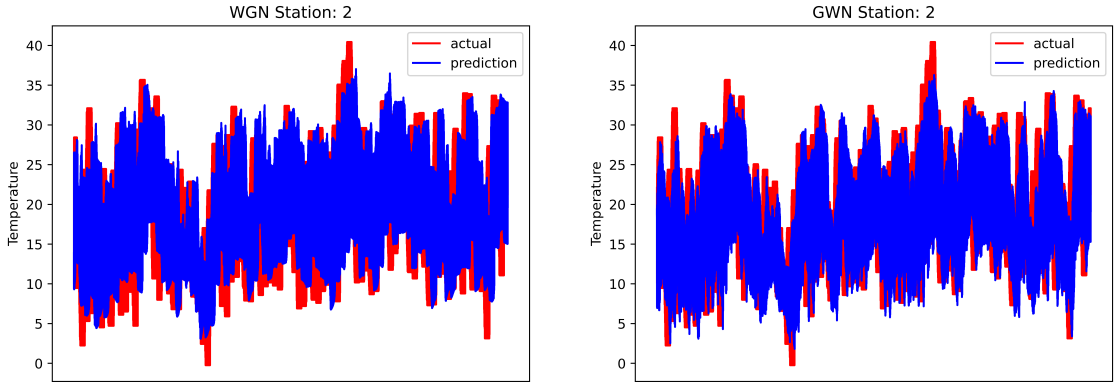


Figure 4.14: ST-GNN plots of predicted vs actual temperature values at station 2

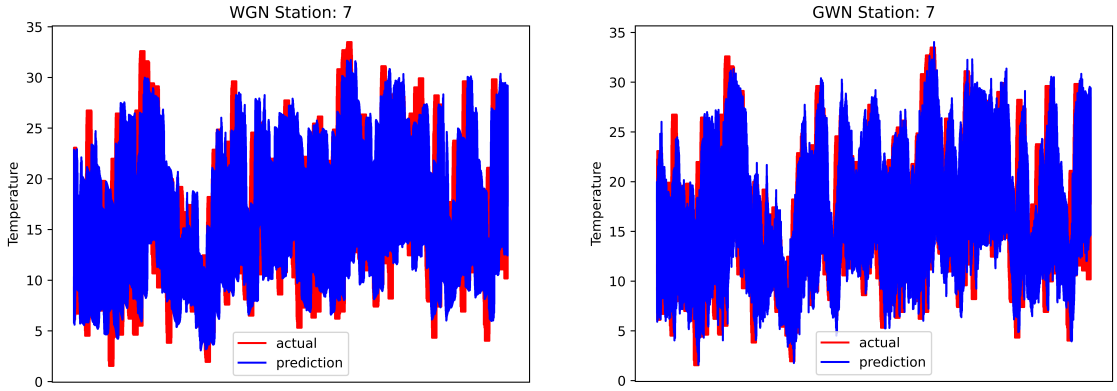


Figure 4.15: ST-GNN plots of predicted vs actual temperature values at station 7

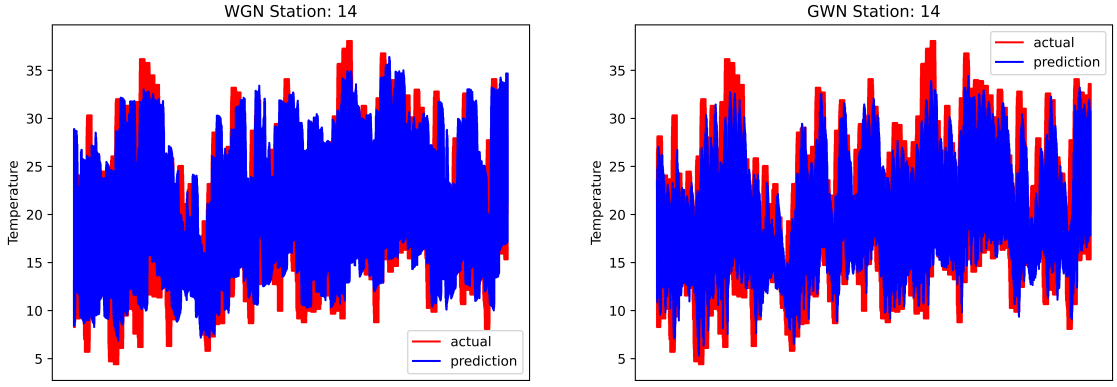


Figure 4.16: ST-GNN plots of predicted vs actual temperature values at station 14

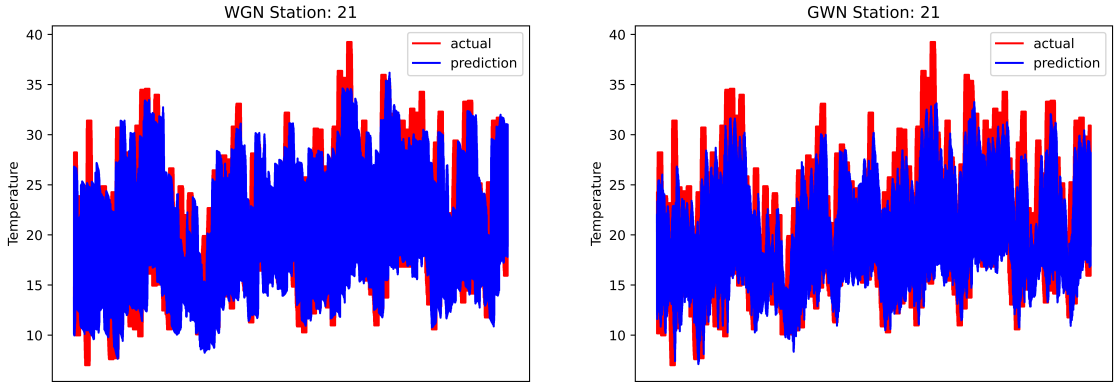


Figure 4.17: ST-GNN plots of predicted vs actual temperature values at station 21

In all of the visualised plots above, we see that both the GWN and WGN models struggle to predict extreme values far from the mean. This trend is true for both high and low extreme temperature values. Our analysis of the mean and standard deviations of each model's predictions in table 4.18 revealed how ST-GNN models tend to predict values closer to the mean. The plots above visualising the target temperature values show high standard deviations from the mean and many extreme values. It therefore makes sense that the ST-GNN model's performance was weaker at these stations since the target temperature observations have high standard deviations from the target mean and a high number of extreme values. From the figures above we see that the GWN model is better able to predict extreme low temperatures when compared to the WGN model. However, we can also see that both models struggle when predicting extreme high temperatures. Finally, we see a larger temporal lag between the WGN model's predictions and the target temperature values than the GWN model.

## 4.7 Spatial-Temporal Dependency Analysis

This section analyses and compares the learnt self-adaptive adjacency matrices of the WGN and GWN models through different visualisation layouts and extraction techniques. The extraction and visualisation process follows the process described in section 3.8

### 4.7.1 Spatial-Temporal Dependency Geographical Visualisation

Following the extraction technique laid out in section 3.8, we extracted and visualised the spatial-temporal dependencies for the GWN and WGN models below.

Figure 4.18 below shows the spatial-temporal dependencies learnt by the WGN model. The out-degree indicates the number of stations that a station influences. These dependencies are overlaid with a geographical map of South Africa. This overlay allows us to see the spatial-temporal dependencies between stations based on their spatial position within South Africa. The visualisation also indicates the out-degree of each station, enabling us to see the influence that each weather station

has. Figure 4.18 shows a directed edge from stations 2 to 11. This indicates that station 2 exerts influence over the temperature at station 11.

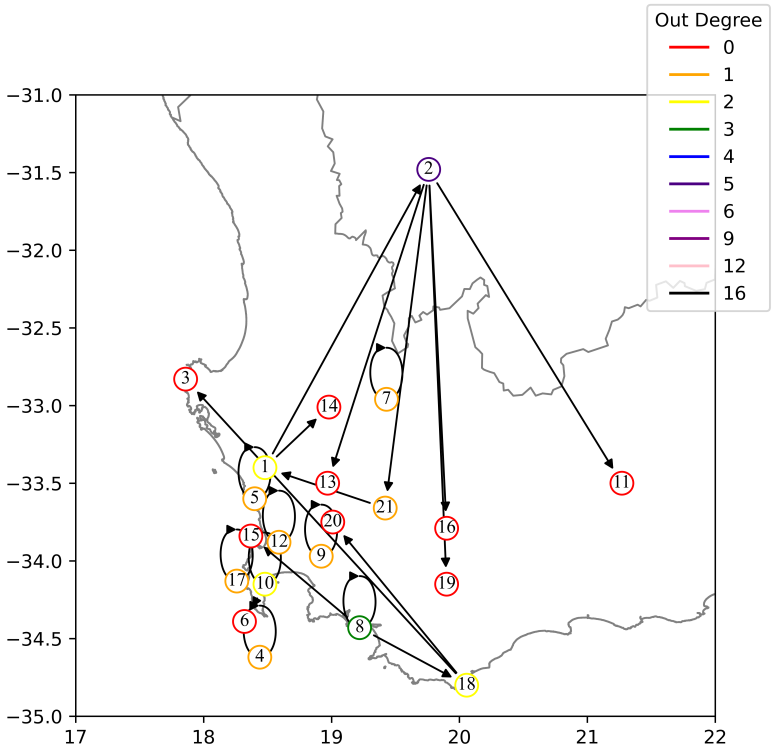


Figure 4.18: WGN spatial-temporal dependencies

From figure 4.18, we can see that station 2 has the highest out-degree of all weather stations, influencing 5 geographically distant weather stations. It is interesting to note that station 2 has the highest out-degree despite being the furthest distance away from the other weather stations. We also see a number of stations with self-loops and out-degrees varying between 0 and 5.

Similarly to the WGN spatial-temporal dependency visualisation above, we visualise the spatial-temporal dependencies found by the GWN model in figure 4.19. The GWN visualisation displays a striking set of spatial-temporal dependencies. We see that station 2 has an exceptionally high out-degree of 16 nodes, exerting influence on the majority of weather stations. A similar trend was found in the WGN visualisation, but not as extreme. From figure 4.19 we can see that the only other station with an out-degree is station 7, with an out-degree of 5. The remaining stations all have an out-degree of 0.

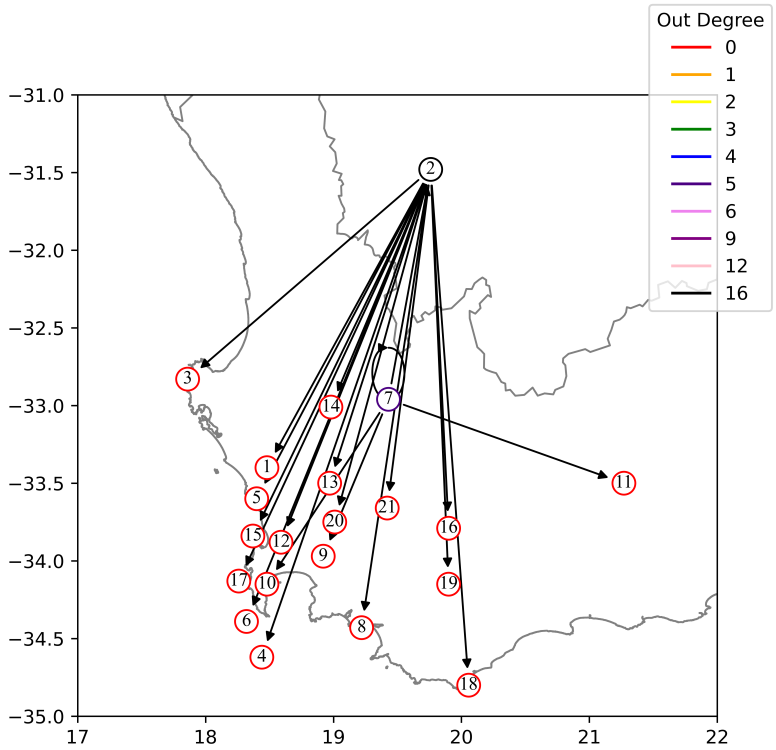


Figure 4.19: GWN spatial-temporal dependencies

### 4.7.2 Spatial-Temporal Path Visualisation

In the previous section, we looked at the spatial-temporal dependencies where only the largest edge weights were visualised. We then found that as a result of the high number of self-loops in the WGN visualisation, and station 2 having the highest out-degree of 16 in the GWN visualisation, we were not able to visualise the paths within the network.

In order to overcome this limitation, we chose to include the second-largest edge weight, in addition to the largest edge weight. This ensures that self-loops do not close off the weather station from the network and allows us to see the influence of other weather stations besides station 2 in the context of the GWN model. This enables us to visualise how information propagates between stations.

We visualise the spatial-temporal dependencies overlaid with the geographical map of South Africa. We explored different layout techniques for the network but none resulted in a visualisation where you could easily see the paths in the network. We therefore chose to use the geographical overlay method. The WGN and GWN spatial-temporal dependencies are shown below in figures 4.20 and

4.21 respectively.

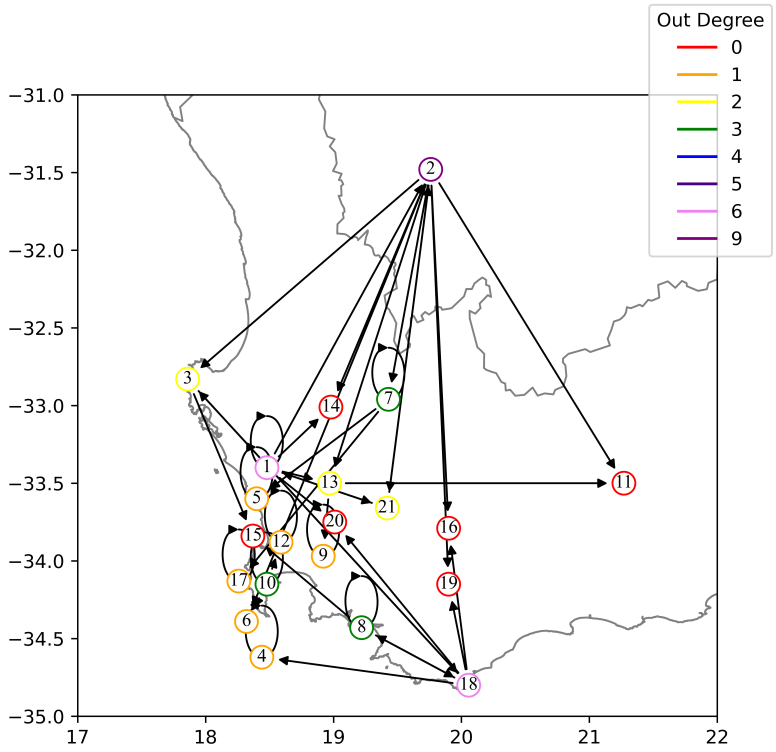


Figure 4.20: WGN spatial-temporal dependencies with additional edges

From the figure above we can see that the spatial-temporal dependency network for the WGN model has far more connections as a result of plotting additional edges aside from the self-loop edges. Stations 1 and 18 now have an out-degree of 6, whereas in the original visualisation they only had an out-degree of 2. Station 2 remains the node with the highest out-degree, with an increase in out-degree from 5 to 9. In addition to nodes having a higher number of edges, there are a significant number of paths within the network that show how information propagates between stations. From the visualisation, we can see that stations 1, 17, and 12 propagate information through the network to the remaining nodes. These remaining nodes then have smaller paths with lower out-degrees.

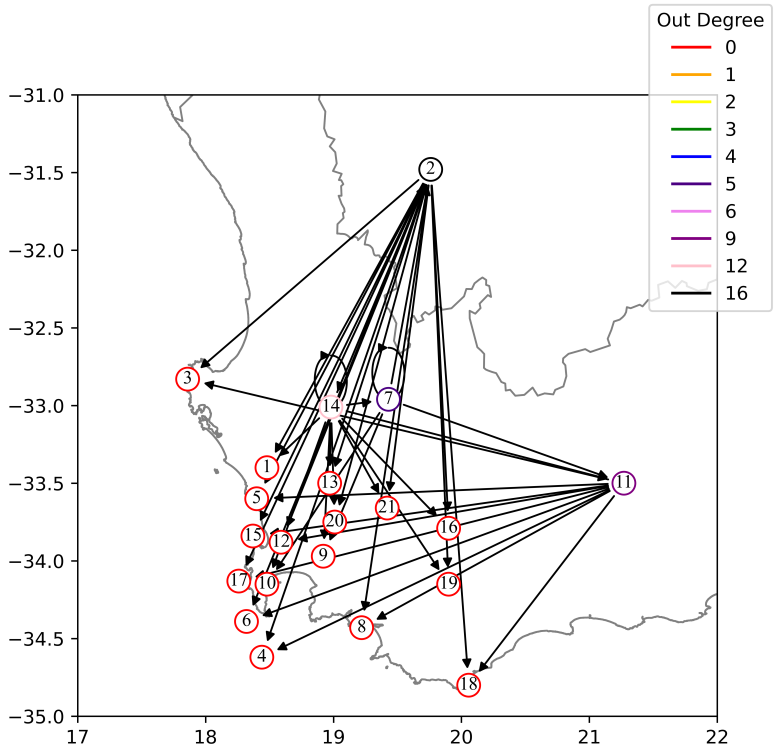


Figure 4.21: GWN spatial-temporal dependencies with additional edges

The GWN visualisation above again presents a striking set of spatial-temporal dependencies. Station 2 remains to be the station with the largest out-degree of 16, having the most influence on the remaining weather stations. In the original visualisation of only the strongest edge weights, both stations 11 and 14 had an out-degree of 0, influencing no other stations on the map. After adding an additional edge, we see stations 7, 11, and 14 now exerting a high degree of influence over the other stations with out-degrees of 9, 9, and 12 respectively.

Comparing the learnt spatial-temporal dependencies of the WGN reveals that that is only one similarity between the two ST-GNN models. Both ST-GNN models indicate that station 2 influences the most number of stations. This is particularly interesting because we found that the ST-GNN models had the weakest performance on station 2. The out-degree of stations in the WGN model varies between 0 and 9 with most stations having an out-degree greater than 1. In contrast, most stations in the GWN model have an out-degree of 0 with only 4 stations having high out-degrees ranging from 5 to 16. The WGN model also has a significantly higher number of self-loops at weather stations than the GWN model. Finally, there are far more paths in the spatial-temporal dependency visualisation of the WGN model than in the GWN model.

# Chapter 5

# Discussion

The overarching aim of this study was to reliably evaluate and compare ST-GNN models i.e. WGN and GWN with basic temporal deep neural networks (DNNs) i.e. TCN and LSTM, for weather prediction in South Africa. Within this overarching goal, we set four specific objectives. The first objective was to compare and evaluate two temporal deep neural networks i.e. TCN and LSTM. The second was objective was to then evaluate and compare the basic temporal DNNs to the Graph WaveNet and Low-Rank Weighted spatial-temporal graph neural networks (ST-GNN). The third objective was to evaluate the stability and robustness of temporal and ST-GNN models. The final objective was to analyse and compare the learnt adjacency matrix of the two ST-GNN models to gain insight into the prominent spatial-temporal dependencies between weather stations. This discussion chapter reflects on our findings related and the extent to which we met the aforementioned aim and objectives. We then compare our findings to related literature and finally discuss our contribution.

## 5.1 Objectives

In this section, we sequentially discuss each objective.

### 5.1.1 Temporal Deep Neural Network Evaluation and Comparison

Our first objective was to compare and evaluate two basic temporal deep neural network architectures, i.e. the LSTM and the TCN for maximum temperature prediction across 21 weather stations in South Africa.

We successfully built LSTM and TCN models for each of the 21 weather stations across the five prediction horizons. The models first went through two hyper-parameter optimisation (HPO) steps i.e. manual tuning and random search. Hyper-parameter optimisation for the LSTM and TCN models was lengthy. This is due to the LSTM and TCN having the most hyper-parameters (10) of all the deep learning models. The high number of hyper-parameters resulted in a large search space which greatly increased the length of the hyper-parameter optimisation (HPO) process. The lengthy hyper-parameter optimisation process that the LSTM and TCN models went through is a

weakness of both models. This weakness is only compounded by having to develop a model for each of the 21 stations. We found that the TCN generally required a smaller input window size than the LSTM which is due to the TCN having a larger memory than the LSTM. Requiring a smaller input window is a strength of the TCN since it significantly reduces computational complexity. The LSTM and TCN models both suffered from overfitting on the training data across the 27 splits. Dropout and regularisation were necessary to ensure that these models do not overfit.

The LSTM results presented in chapter 4 revealed that the LSTM's performance was at best mediocre. The LSTM outperformed the SARIMA baseline model which was to be expected given its ability to capture temporal dependencies and non-linear patterns. However, the LSTM was significantly outperformed by the TCN model at all 21 stations across all five prediction horizons. The TCN outperforming the LSTM was to be expected. The TCN has a larger receptive field size due to its ability to stack convolutional layers and keep the computational complexity of these layers low through the use of dilations. This large receptive field of the TCN models also allows the TCN models to capture longer dependencies than the LSTM model. The TCN models were also significantly more stable and robust than the LSTM models reflected by a lower standard deviation of performance across 10 runs and a lower SMAPE score across the 27 splits of walk-forward validation. When we analysed the performance of the TCN and LSTM models spatially, we found that the performance of models at stations along the west coast was significantly poorer than at stations found further inland.

The LSTM is not a good model to use to predict temperature across the 3, 6, 9, 12 and 24-hour prediction horizons. The hyper-parameter optimisation process was lengthy and any further performance improvements from the HPO process would require the process to be lengthened. When we consider the HPO process that had to be conducted for each station, in addition to producing models with mediocre results, it is clear that the LSTM model is not suitable for maximum temperature prediction in the Western Cape and Northern Province in South Africa. Comparatively, the TCN model is clearly the better temporal deep neural network for maximum temperature prediction, despite it also being affected by a lengthy HPO process. Finally, the TCN and LSTM performance could potentially be improved by conducting HPO over more splits.

### 5.1.2 ST-GNN Evaluation and Comparison

The second objective was to evaluate the ST-GNN models i.e. GWN and WGN and then compare the results to the temporal models.

The hyper-parameter optimisation of the GWN model took a significantly shorter amount of time than the temporal models. Six parameters were manually tuned and 4 of the parameters were set after the manual tuning step. This resulted in a small search space for the random search hyper-parameter optimisation step resulting in a short HPO computation time. Furthermore, the hyper-parameter optimisation process only had to be done once, and not for each weather station independently which saved a significant amount of time when compared to the LSTM and TCN models. Similarly to the GWN model, the hyper-parameter optimisation process of the WGN model took a shorter amount of time than temporal models. Seven hyper-parameters were tuned through hyper-parameter optimisation. Five of these parameters were tuned during the manual tuning step

resulting in only two parameters needing to be tuned during the random search process. This resulted in a small search space and ultimately a quick hyper-parameter optimisation process. Again, the optimisation process only had to be done once, whereas the temporal models required it do be done for each of the twenty-one stations.

Tuning the hyper-parameters around the adjacency matrix presented interesting findings. The GWN model can utilise a predefined graph structure in addition to a learnt self-adaptive adjacency matrix, or just the self-adaptive adjacency matrix alone for graph convolution. We found that using predefined knowledge in the form of a Gaussian Kernel threshold on the geographic distance between the weather stations did not improve accuracy. The GWN model was able to learn the hidden spatial dependencies without prior knowledge by randomly initializing two node embeddings and learning these embeddings via stochastic gradient descent. This is a promising finding and strength of the GWN model as it removes the need for any information around the predefined graph structure. The WGN model required predefined knowledge of the graph structure to initialise the nadjacency matrix to achieve optimal performance whereas the GWN did not. This is a drawback of the WGN model since it requires some form of predefined knowledge which in some cases may not be available.

The results presented in the previous chapter show that on average, Graph WaveNet is the model that produces the most accurate predictions across all prediction horizons. When comparing the graph neural networks, GWN outperforms the WGN on average across all prediction horizons by 0.72%. This small performance improvement is unexpected. The WGN and GWN model use LSTMs and TCNs respectively as part of their architecture to capture temporal dependencies. The TCN provided on average a significant improvement of 5.63% across all prediction horizons against the LSTM model. We therefore expected the GWN model to outperform the WGN model to a similar degree. However, the results revealed that the GWN model outperformed the WGN model by a meagre 0.72%. We can therefore infer that capturing spatial dependencies significantly improved the performance of both ST-GNN models to similar accuracy scores, with the GWN model outperforming the WGN on average due to the TCN component. These findings highlight the importance of capturing spatial dependencies in addition to temporal dependencies for temperature prediction.

Graph WaveNet results in an average performance improvement across all prediction horizons of 7.95% on the LSTM models. This result was expected because GWN captures spatial dependencies and has a superior architecture for capturing temporal dependencies. The WGN model outperforms the LSTM model at all stations across all prediction horizons. WGN uses the LSTM cell to capture temporal dependencies while using graph convolution to capture spatial dependencies. These results provide clear evidence that capturing spatial dependencies in addition to temporal dependencies improves the accuracy of predictions.

As mentioned earlier, the TCN outperformed the LSTM and SARIMA models at all weather stations across all prediction horizons. However, the performance comparison between the TCN and ST-GNN models is particularly interesting. When comparing the TCN to the ST-GNNs, WGN and GWN outperformed the TCN model on average across all prediction horizons. However, the TCN outperformed the WGN and GWN models on average at 4 stations. Upon further investigation into why the TCN model outperformed the ST-GNNs at these stations, we found interesting weaknesses in both the GWN and WGN models. We found that the TCN model outperforms the

ST-GNN models at stations with a high standard deviation in the observed temperature values. We also found that these stations with high standard deviations have recorded negative minimum temperatures. The analysis in section 4.6 indicated that the ST-GNN models' predictions tend to lie closer to the mean even at stations with a high standard deviation. Furthermore, both the WGN and GWN models struggled to accurately predict extreme high temperatures. The WGN model also struggled to predict extreme low temperatures and had a larger temporal lag than the GWN model. We therefore found two weaknesses in the ST-GNN models i.e. when predicting extreme high and low temperatures and at stations with a high standard deviation.

It is clear that the GWN model is the best model for maximum temperature prediction in the Western Cape and Northern Province of South Africa. The GWN model has a short hyper-parameter optimisation process, requires no predefined knowledge of the structure of the graph, captures hidden spatial dependencies, predicts temperature at all stations simultaneously, and makes accurate predictions across all five prediction horizons. However, this model has some limitations as described above. We therefore suggest an ensemble approach for maximum temperature prediction that consists of the GWN and TCN model. Although the TCN has a lengthy HPO process, this model only needs to be implemented at stations with a high standard deviation in the observed temperature values. The GWN model can then be used at the remaining stations.

### 5.1.3 Stability and Robustness

The third objective was to perform appropriate experiments to evaluate the stability and robustness of the models implemented i.e. LSTM, TCN, WGN, and GWN.

We repeated the experiments for each model 10 times to evaluate the stability. The LSTM models had the highest average standard deviation across the five prediction horizons along with the worst performance. The high standard deviation along with the weak prediction performance of the LSTM model makes it the weakest model of all DNNs implemented. Comparatively, the TCN model had the lowest standard deviation of all models implemented, along with accurate predictions of temperature. This finding indicates that the TCN model was the most stable temporal model. The WGN model was significantly more stable than its temporal counterpart, the LSTM. This is interesting and suggests that including spatial dependencies improves stability when using the LSTM as the mechanism with which to capture temporal dependencies. The WGN model is less stable than the TCN model which suggests that the TCN produces stabler, more accurate results when compared to the WGN. However, when we compare the GWN model's stability to its temporal counterpart, the TCN, we see that the GWN is less stable than the TCN model. This finding suggests that when using the TCN to capture temporal dependencies, including spatial dependencies can negatively impact the stability of the model. We can then infer that ST-GNN models are slightly less stable than temporal models like the TCN which are highly capable of capturing temporal dependencies.

We used walk-forward validation to evaluate the models for robustness. The LSTM again was the least robust model, producing weak results across the splits of walk-forward validation. The TCN was found to be a robust model producing accurate results across all the splits. The WGN and GWN were found to both be highly robust models producing accurate results across the splits. Our

findings reveal a strength of the TCN, WGN, and GWN models in that they are all highly robust models.

### 5.1.4 Spatial-Temporal Dependencies

The final objective was to analyse and compare the learnt adjacency matrices of the two ST-GNNs to gain insights into the prominent spatial-temporal dependencies between weather stations.

The spatial-temporal dependency visualisations presented the most peculiar results. As previously mentioned, the GWN only marginally outperformed the WGN model across all prediction horizons. Since the accuracy of the two models is fairly similar, we would expect that the spatial-temporal dependencies found by each station would then be similar. However, our visualisations in section 4.7 found almost no similarities between the learn spatial-temporal dependencies of the two models. The only common finding between the visualisations was that the northern most station exerted the most influence of all the weather stations. This is a particularly interesting finding considering that this station is one of the most geographically distant station from the other stations. We found that this northern most station is one of two stations found along the interior plateau, an area of land that is significantly higher above sea level than the other stations. It is possible that the unique positioning makes the information from this station important to predicting temperature at stations along the coast at lower sea levels.

In visualising the spatial-temporal dependencies we followed the method used by Wilson et al. [9]. Wilson et al. only implemented a single ST-GNN, whereas we implemented two and then compared the learnt spatial-temporal dependencies. Our findings suggest that the current visualisation and analysis techniques of spatial-temporal dependencies are limited and not sufficiently robust to provide highly insightful findings.

## 5.2 Comparison to Related Literature

In the introduction, we set out an aim and objectives to address certain research gaps identified in the literature review. In this section, we reflect on the gaps we found, the extent to which these were addressed, and how our findings compare to those in related literature.

For the LSTM and TCN, we compare our work against Hewage et al. [3] whose work inspired our experimental design and also implemented an LSTM and TCN for temperature prediction. The first significant difference between this project and that of Hewage et al. is in the experimental design. Hewage et al. did a single train, validation, and test split of the data whereas we validated and tested our models through walk-forward validation. Our results therefore provide a more reliable estimate of the model's robustness through our use of walk-forward validation. We also measured the stability of the models indicated by the standard deviations across the 10 runs of the experiments. The other notable difference is the search space of the hyper-parameters. Hewage et al. searched models with significantly more trainable parameters and layers than those searched through in this paper. They found that a 3-layer LSTM architecture performed the best whereas our research found a single-layer LSTM sufficient to capture temporal dependencies. Their study covered a significantly higher number of stations and area of land, this would explain the difference in the number of layers as our model needed to learn fewer temporal dependencies as it covers fewer

stations and a smaller geographical area.

Our findings revealed that the TCN model was the best model temporal model for predicting temperature. This finding is in line with Hewage et al. However, our analysis of the stations where the TCN performs well, and even outperforms ST-GNN models, indicated that these models are particularly strong at stations whose temperature values have a high standard deviation. This is a new finding that reflects the depth of our analysis of the predictions of each model. According to Hewage et al. the kernel size, dilations, batch size, dropout rate and learning rate does not significantly impact the final performance of the TCN models. We found that these findings are true with the exception that the dropout rate and learning rate do significantly impact the performance of the final models. This is a significant finding since these tuning these parameters drastically affected performance.

We now compare the findings of our research to that of the original Graph WaveNet implementation by Wu et al. [5]. Our data were normalised using a MinMax scaler whereas Wu et al. [5] used Z-score normalisation. The original implementation of the GWN model found that including the predefined graph structure in addition to the self-adaptive adjacency matrix yields the most accurate predictions, where the adaptive adjacency matrix was initialised using a Gaussian kernel of the distances between nodes. Our findings, however, revealed that we could just use a randomly initialised adaptive adjacency matrix and still achieve optimal performance. This is a positive finding that adds additional evidence to the claim that GWN can effectively capture hidden spatial dependencies without prior knowledge in myriad domains. We did however find weaknesses in the GWN model which Wu et al. did not identify. These weaknesses related to the weak performance of the GWN model where the target variable has a high standard deviation and when predicting extreme maximum temperatures at certain stations.

We compare the findings of our research to that of the original Low-Rank Weighted GNN implementation by Wilson et al. [9]. Wilson et al. also used daily measurements of minimum, maximum and average temperature and wind speed whereas we used hourly recordings of humidity, pressure, temperature, rain, wind speed and direction to predict temperature. We found that these features provided sufficient information to accurately predict temperature across all the prediction horizons. The original implementation of the WGN model used a sparse adjacency matrix that uses less spatial information from distant weather stations to predict the temperature in an attempt to reduce computational complexity and training time without losing predictive performance. Through our manual tuning process, we found that using a sparse adjacency matrix resulted in worse performance. The reason for this is that our weather stations were located in close proximity to one another, whereas the original WGN implementation used data from weather stations across the United States and therefore a greater distance. Their sparse adjacency matrix therefore needn't consider spatial relationships between stations that are extremely distant from one another. Finally, similarly to the original WGN model, the adjacency matrix had to be initialised using a Gaussian kernel between the distance between the weather stations to achieve optimal performance. Similarly to the authors of the GWN paper, Wilson et al. did not identify the weaknesses of the model when predicting extreme maximum and minimum temperatures, and at stations whose temperature values have a high standard deviation. Additionally, we found that the WGN model suffers from a large temporal lag that lowers model performance which Wilson et al. did not mention.

Neither Hewage et al., Wu et al., nor Wilson et al. evaluated their models for stability and robustness. We were able to effectively evaluate model stability and robustness of all the models through repetition of experiments and walk-forward validation respectively. These led to new findings that suggest temporal models like the TCN are more stable than ST-GNNs and that capturing spatial dependencies in addition to temporal dependencies can reduce stability.

## 5.3 Contribution

Our first contribution is the use of walk-forward validation as an experimental framework in which models that are trained on temporal data can be evaluated for performance, stability and robustness. We showed that walk-forward validation provides a better evaluation of the model's performance and robustness rather than a single train-validation-test split. We also showed that walk-forward validation can be effectively used in the weather domain. This is an important contribution as weather models need to constantly be updated with new data to learn the dynamic temporal and spatial dependencies.

Our next contribution was that our research revealed that the input window size of the models is the most important hyper-parameter across all deep learning models we implemented. The related literature in weather and traffic flow prediction made no mention of exploring the size of the input window as a hyper-parameter. Our research indicated that the input window size should be included as a hyper-parameter when tuning models.

Our work contributed by deepening our understanding of the relationship between weather stations by analysing the spatial-temporal dependencies. We found similarities in the spatial-temporal dependencies learnt by the WGN and GWN model. The stations along the elevated interior of the land exert a lot of influence on stations found at lower sea levels.

The next contribution made by our work provides insight into the strengths and weaknesses of the WGN and GWN models. We found that the ST-GNN models produce less accurate predictions of temperature at stations with a high standard deviation in their recorded temperature. Additionally, we found that the ST-GNN models struggled to accurately predict extreme high and low temperatures at certain stations.

Our research contributed to the field of using deep learning for weather prediction in the Western Cape and Northern Cape provinces in South Africa. Our work showed that deep learning shows promising, accurate results when predicting temperature. This work suggests that deep learning can be used as an alternative to current popular weather prediction techniques and should be further explored.

Our results and unique insights lead us to contribute by suggesting an ensemble approach for maximum temperature prediction. The TCN models should be implemented at stations where the temperature observations have a high standard deviation. The GWN model can then be used to predict the temperature at the remaining stations where the standard deviation in the temperature observations is comparatively lower.

Our final contribution is that we are the first to implement ST-GNN models i.e. GWN from the traffic flow domain for weather prediction, showing that the traffic flow ST-GNNs are capable of learning spatial and temporal dependencies in the weather domain.

# Chapter 6

# Conclusions and Future Work

This final chapter covers the conclusions of this paper, and finally, we briefly discuss possible future work in deep learning for weather prediction.

The research of this thesis was guided by a set of research questions aimed at exploring the use of deep learning techniques in weather prediction laid out at the start of this thesis. The overall aim of this work is to reliably evaluate and compare ST-GNN models i.e. WGN and GWN, with temporal deep neural networks i.e. LSTM and TCN, for weather prediction in South Africa. In this research, we implemented and evaluated five different approaches for weather prediction i.e. SARIMA, TCN, LSTM, GWN, and WGN. We implemented these models using the walk-forward validation technique and repeated our experiments 10 times which enabled us to reliably evaluate models for robustness and stability. The models were used to predict temperature across twenty-one weather stations in two provinces in South Africa across five prediction horizons i.e. 3, 6, 9, 12 and 24 hours.

Our investigation into the SARIMA model showed that it was not capable of producing accurate results at any weather station. We found that the LSTM model was neither stable nor robust, producing mediocre predictions across all weather stations and prediction horizons. We also showed that the LSTM model is significantly outperformed by the TCN temporal model. The TCN model produced unexpected results, outperforming the ST-GNN models at particular weather stations across different prediction horizons. Our results from the TCN model revealed that a temporal model, without utilising any spatial information, can outperform models that incorporate spatial information. However, the WGN and GWN model expectedly outperformed the temporal models on average across all prediction horizons highlighting the importance of capturing spatial dependencies. Additionally, the TCN model when evaluated using walk-forward validation was found to be a robust and stable model, capable of producing accurate predictions.

We implemented the WGN model developed in the weather domain on South African weather data. We were able to determine the accuracy of state-of-the-art GNN models developed in the weather domain when applied to South African data. Our research found that the WGN model has a high prediction accuracy and largely outperforms temporal models. This model, when compared to its temporal counterpart, the LSTM, showed that capturing spatial dependencies in addition to temporal dependencies generally improves the accuracy of predictions.

We also implemented the GWN model which was originally developed and implemented in the traffic flow domain. We found that the GWN model developed in the traffic flow domain produced the most accurate predictions. Additionally, we found that the GWN model is a robust and stable model for weather prediction. The performance of the GWN model in the weather domain provides evidence for the claim that ST-GNNs developed in the traffic flow domain can be used to model spatial-temporal data regardless of domain.

Our research followed an experimental design which tested model stability and robustness when predicting in the weather domain. Through walk-forward validation, we were able to show that the TCN, WGN, and GWN models were all sufficiently robust to produce medium to highly accurate predictions of temperature. Furthermore, by repeating the experiments ten times, we were able to present reliable results that show that the TCN, WGN, and GWN models were stable when predicting temperature.

Our research presented spatial-temporal dependency visualisations for WGN and GWN models. Our visualisations highlighted the strong influence of particular weather stations when predicting temperature at other stations. We found two stations, common to both ST-GNNs, to have a strong influence over predicting temperature at the other stations. Having said that, the comparison between the spatial-temporal dependencies revealed that the learnt spatial-temporal dependencies were dissimilar while producing similar accurate predictions. Finally, through our research, we found that an ensemble approach, combining the temporal TCN model and GWN model would provide the most accurate weather predictions.

## 6.1 Future Work and Limitations

Finally, we share possible extensions or future work that can be explored, highlighting the limitations of our work.

- Our research treated the input window size as a hyper-parameter. We found that this parameter has a significant impact on the performance of models. Future work should scrutinize and investigate the input window size from the perspective of the model and domain.
- Our research chose to evaluate deep learning models on a three-month rolling-window walk-forward validation technique. Our choice of three-month increments stemmed from the number of seasons in a year and keeping the maximum number of splits low to mitigate the lengths of experiments. Future work should explore the choice of increment size for rolling-window walk-forward validation in relation to model performance.
- Hyper-parameter optimisation (HPO) was only implemented on the first 3 splits of the full data set. Future work could run HPO over additional splits to potentially produce more accurate models.
- Our research only compared deep learning models and classical statistical approaches to weather prediction. Our deep learning models produced highly accurate predictions, but there was no comparison made between the models we developed and those used by the South African Weather Services(SAWS) to predict the weather. Future work should therefore

consider including the methods used by the SAWS as a baseline against which to compare deep learning models.

- Our analysis of the learnt spatial-temporal dependencies of the WGN and GWN model did not determine the truthfulness of the learnt dependencies. Future work should explore techniques to determine this truthfulness.

# Appendices

# Appendix A

# Results

## A.1 SARIMA Model Identification Results

The Augmented Dickey-Fuller (ADF) test was carried out on each of the 21 weather stations. This test determines if the temperature series was stationary. The null and alternative hypotheses for the ADF test are as follows:

**Null Hypothesis($H_0$):** *A unit root exists*
**Alternative Hypothesis($H_1$):** *The series is stationary*

The p-values for all of the 21 weather stations were found to be less than the significance level at the 0.01 level. We were thus able to confidently reject the null hypothesis and conclude that the time series of temperatures at each station is stationary. We therefore set the $d$ parameter to 0 for the SARIMA models.

The next step in the model identification process is determining the value of the $m$ parameter which indicates the seasonal period of the data. The autocorrelation function (ACF) plot for a single station is shown in figure A.1. The ACF plots for all stations followed the autocorrelation pattern shown in figure A.1. The ACF plot indicated that there are significant seasonal spikes every 24 lags. The value of the $m$ parameter was therefore set to 24.

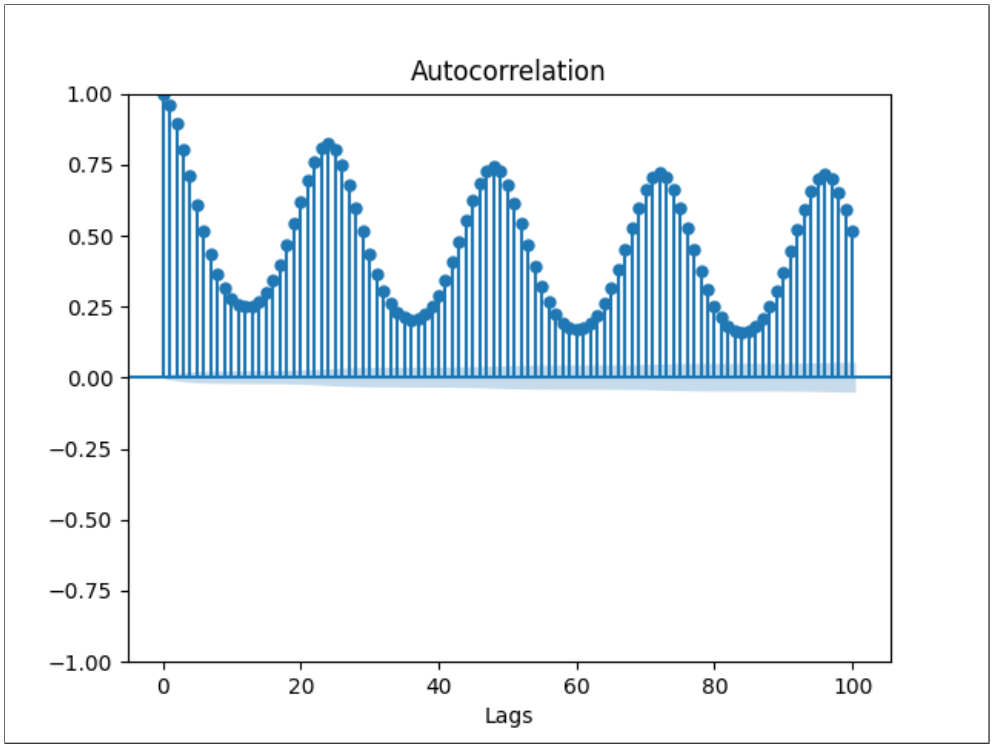


Figure A.1: ACF Plot for a single station

## A.2 LSTM Results

| | MSE | | | | | SMAPE | | | | | MAE | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **3** | **6** | **9** | **12** | **24** | **3** | **6** | **9** | **12** | **24** | **3** | **6** | **9** | **12** | **24** |
| 1 | 0.006 | 0.007 | 0.007 | 0.007 | 0.008 | 16.48 | 16.97 | 17.31 | 17.55 | 18.11 | 0.057 | 0.058 | 0.060 | 0.061 | 0.063 |
| 2 | 0.003 | 0.003 | 0.004 | 0.004 | 0.006 | 10.06 | 11.19 | 11.93 | 12.45 | 13.74 | 0.041 | 0.046 | 0.049 | 0.051 | 0.058 |
| 3 | 0.008 | 0.008 | 0.009 | 0.009 | 0.009 | 19.78 | 20.17 | 20.45 | 20.69 | 21.27 | 0.064 | 0.065 | 0.066 | 0.067 | 0.069 |
| 4 | 0.004 | 0.005 | 0.005 | 0.005 | 0.006 | 15.76 | 16.31 | 16.65 | 16.86 | 17.47 | 0.049 | 0.051 | 0.052 | 0.052 | 0.054 |
| 5 | 0.006 | 0.006 | 0.006 | 0.007 | 0.007 | 15.50 | 16.02 | 16.40 | 16.69 | 17.40 | 0.056 | 0.058 | 0.059 | 0.060 | 0.063 |
| 6 | 0.005 | 0.005 | 0.006 | 0.006 | 0.006 | 15.11 | 15.88 | 16.37 | 16.67 | 17.36 | 0.051 | 0.054 | 0.056 | 0.057 | 0.059 |
| 7 | 0.004 | 0.004 | 0.005 | 0.005 | 0.006 | 13.49 | 14.38 | 15.00 | 15.42 | 16.29 | 0.048 | 0.052 | 0.055 | 0.056 | 0.060 |
| 8 | 0.005 | 0.005 | 0.006 | 0.006 | 0.006 | 15.57 | 16.13 | 16.43 | 16.56 | 16.87 | 0.054 | 0.056 | 0.057 | 0.058 | 0.059 |
| 9 | 0.005 | 0.006 | 0.006 | 0.007 | 0.007 | 16.34 | 17.16 | 17.69 | 18.03 | 18.81 | 0.058 | 0.061 | 0.063 | 0.065 | 0.068 |
| 10 | 0.006 | 0.006 | 0.007 | 0.007 | 0.007 | 18.84 | 19.39 | 19.73 | 19.91 | 20.48 | 0.059 | 0.061 | 0.062 | 0.063 | 0.065 |
| 11 | 0.005 | 0.006 | 0.006 | 0.007 | 0.008 | 14.25 | 14.93 | 15.46 | 15.89 | 17.08 | 0.055 | 0.058 | 0.061 | 0.062 | 0.067 |
| 12 | 0.005 | 0.005 | 0.006 | 0.006 | 0.007 | 15.00 | 15.81 | 16.43 | 16.90 | 18.32 | 0.052 | 0.056 | 0.058 | 0.060 | 0.066 |
| 13 | 0.004 | 0.004 | 0.005 | 0.005 | 0.005 | 13.40 | 14.08 | 14.54 | 14.84 | 15.65 | 0.048 | 0.051 | 0.053 | 0.054 | 0.058 |
| 14 | 0.004 | 0.004 | 0.005 | 0.005 | 0.005 | 14.10 | 14.84 | 15.27 | 15.53 | 16.11 | 0.048 | 0.051 | 0.053 | 0.054 | 0.056 |
| 15 | 0.012 | 0.012 | 0.012 | 0.012 | 0.012 | 20.88 | 20.98 | 21.02 | 21.04 | 21.04 | 0.082 | 0.082 | 0.082 | 0.082 | 0.082 |
| 16 | 0.005 | 0.005 | 0.005 | 0.005 | 0.006 | 14.40 | 15.14 | 15.61 | 15.96 | 16.71 | 0.052 | 0.055 | 0.056 | 0.058 | 0.061 |
| 17 | 0.007 | 0.008 | 0.008 | 0.008 | 0.008 | 16.47 | 16.92 | 17.21 | 17.39 | 17.79 | 0.061 | 0.063 | 0.064 | 0.065 | 0.066 |
| 18 | 0.005 | 0.005 | 0.005 | 0.005 | 0.006 | 12.10 | 12.43 | 12.63 | 12.74 | 13.00 | 0.053 | 0.054 | 0.055 | 0.056 | 0.057 |
| 19 | 0.007 | 0.007 | 0.007 | 0.007 | 0.008 | 16.74 | 17.24 | 17.55 | 17.74 | 18.26 | 0.060 | 0.062 | 0.063 | 0.063 | 0.065 |
| 20 | 0.003 | 0.004 | 0.004 | 0.005 | 0.005 | 12.96 | 13.90 | 14.56 | 15.03 | 16.31 | 0.044 | 0.048 | 0.050 | 0.052 | 0.057 |
| 21 | 0.004 | 0.004 | 0.004 | 0.004 | 0.005 | 12.92 | 13.57 | 13.98 | 14.20 | 14.70 | 0.047 | 0.049 | 0.051 | 0.052 | 0.054 |
| **Ave.** | 0.005 | 0.006 | 0.006 | 0.007 | 0.008 | 15.25 | 15.87 | 16.30 | 16.58 | 17.27 | 0.053 | 0.055 | 0.057 | 0.058 | 0.063 |

Table A.1: Metrics recorded for LSTM models for each of the 21 stations across the prediction horizons

## A.3 TCN Results

| | **MSE** | | | | | **SMAPE** | | | | | **MAE** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **3** | **6** | **9** | **12** | **24** | **3** | **6** | **9** | **12** | **24** | **3** | **6** | **9** | **12** | **24** |
| 1 | 0.002 | 0.002 | 0.002 | 0.003 | 0.003 | 8.94 | 10.18 | 10.92 | 11.49 | 12.91 | 0.029 | 0.034 | 0.036 | 0.038 | 0.043 |
| 2 | 0.001 | 0.002 | 0.003 | 0.003 | 0.004 | 6.87 | 8.23 | 9.18 | 9.91 | 11.71 | 0.028 | 0.034 | 0.038 | 0.041 | 0.049 |
| 3 | 0.002 | 0.003 | 0.003 | 0.003 | 0.004 | 9.87 | 11.15 | 11.98 | 12.62 | 14.16 | 0.031 | 0.035 | 0.038 | 0.040 | 0.045 |
| 4 | 0.001 | 0.002 | 0.002 | 0.003 | 0.003 | 8.64 | 10.08 | 10.98 | 11.62 | 13.13 | 0.026 | 0.031 | 0.034 | 0.036 | 0.041 |
| 5 | 0.002 | 0.002 | 0.003 | 0.003 | 0.004 | 8.92 | 10.25 | 11.19 | 11.88 | 13.43 | 0.031 | 0.036 | 0.039 | 0.042 | 0.047 |
| 6 | 0.002 | 0.003 | 0.004 | 0.004 | 0.005 | 10.01 | 11.41 | 12.30 | 12.97 | 14.44 | 0.034 | 0.039 | 0.042 | 0.044 | 0.049 |
| 7 | 0.002 | 0.002 | 0.003 | 0.003 | 0.004 | 8.95 | 10.07 | 10.95 | 11.61 | 13.16 | 0.032 | 0.036 | 0.039 | 0.041 | 0.047 |
| 8 | 0.002 | 0.002 | 0.003 | 0.003 | 0.003 | 8.90 | 9.95 | 10.68 | 11.22 | 12.48 | 0.030 | 0.034 | 0.037 | 0.039 | 0.043 |
| 9 | 0.002 | 0.003 | 0.004 | 0.004 | 0.005 | 10.07 | 11.75 | 12.89 | 13.72 | 15.49 | 0.035 | 0.041 | 0.045 | 0.048 | 0.055 |
| 10 | 0.002 | 0.003 | 0.003 | 0.003 | 0.004 | 10.53 | 12.11 | 13.11 | 13.83 | 15.32 | 0.032 | 0.037 | 0.040 | 0.042 | 0.047 |
| 11 | 0.002 | 0.002 | 0.003 | 0.004 | 0.005 | 8.00 | 9.69 | 10.75 | 11.58 | 13.37 | 0.029 | 0.036 | 0.040 | 0.044 | 0.051 |
| 12 | 0.002 | 0.002 | 0.003 | 0.003 | 0.004 | 8.45 | 9.95 | 10.95 | 11.78 | 13.81 | 0.029 | 0.034 | 0.038 | 0.041 | 0.049 |
| 13 | 0.001 | 0.002 | 0.002 | 0.002 | 0.003 | 6.58 | 8.05 | 9.09 | 9.86 | 11.56 | 0.024 | 0.029 | 0.033 | 0.036 | 0.042 |
| 14 | 0.001 | 0.002 | 0.002 | 0.003 | 0.004 | 8.40 | 9.79 | 10.73 | 11.42 | 12.90 | 0.028 | 0.033 | 0.037 | 0.040 | 0.045 |
| 15 | 0.002 | 0.002 | 0.002 | 0.003 | 0.003 | 8.13 | 9.04 | 9.60 | 10.00 | 10.88 | 0.029 | 0.033 | 0.035 | 0.037 | 0.040 |
| 16 | 0.002 | 0.002 | 0.003 | 0.003 | 0.004 | 8.52 | 9.93 | 10.87 | 11.56 | 13.11 | 0.029 | 0.035 | 0.038 | 0.041 | 0.047 |
| 17 | 0.001 | 0.002 | 0.002 | 0.002 | 0.003 | 7.19 | 8.33 | 9.08 | 9.63 | 10.96 | 0.026 | 0.030 | 0.033 | 0.035 | 0.040 |
| 18 | 0.002 | 0.002 | 0.003 | 0.003 | 0.004 | 6.73 | 7.88 | 8.65 | 9.22 | 10.51 | 0.029 | 0.034 | 0.037 | 0.040 | 0.045 |
| 19 | 0.001 | 0.002 | 0.003 | 0.003 | 0.004 | 8.28 | 10.00 | 11.10 | 11.89 | 13.56 | 0.028 | 0.035 | 0.038 | 0.041 | 0.047 |
| 20 | 0.001 | 0.002 | 0.002 | 0.003 | 0.003 | 8.13 | 9.48 | 10.40 | 11.15 | 12.92 | 0.027 | 0.032 | 0.035 | 0.038 | 0.044 |
| 21 | 0.001 | 0.001 | 0.002 | 0.002 | 0.003 | 6.57 | 7.88 | 8.79 | 9.48 | 11.03 | 0.023 | 0.029 | 0.032 | 0.035 | 0.040 |
| **Ave.** | 0.002 | 0.002 | 0.003 | 0.003 | 0.004 | 8.41 | 9.77 | 10.69 | 11.35 | 12.90 | 0.028 | 0.033 | 0.038 | 0.040 | 0.044 |

Table A.2: Metrics recorded for TCN models for each of the 21 stations across the prediction horizons

## A.4 WGN Results

| | **MSE** | | | | | **SMAPE** | | | | | **MAE** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **3** | **6** | **9** | **12** | **24** | **3** | **6** | **9** | **12** | **24** | **3** | **6** | **9** | **12** | **24** |
| 1 | 0.002 | 0.003 | 0.003 | 0.003 | 0.003 | 7.10 | 9.23 | 9.57 | 9.51 | 9.37 | 0.032 | 0.042 | 0.043 | 0.043 | 0.042 |
| 2 | 0.003 | 0.006 | 0.007 | 0.006 | 0.006 | 10.03 | 14.33 | 15.82 | 15.57 | 15.41 | 0.039 | 0.058 | 0.064 | 0.063 | 0.061 |
| 3 | 0.001 | 0.002 | 0.002 | 0.002 | 0.002 | 5.54 | 7.01 | 7.55 | 7.62 | 7.50 | 0.024 | 0.030 | 0.033 | 0.033 | 0.032 |
| 4 | 0.001 | 0.002 | 0.002 | 0.002 | 0.002 | 4.88 | 6.56 | 7.31 | 7.33 | 7.26 | 0.022 | 0.029 | 0.032 | 0.033 | 0.032 |
| 5 | 0.001 | 0.002 | 0.003 | 0.003 | 0.003 | 5.45 | 7.22 | 7.91 | 7.84 | 8.16 | 0.026 | 0.035 | 0.039 | 0.038 | 0.040 |
| 6 | 0.001 | 0.002 | 0.002 | 0.002 | 0.002 | 5.18 | 6.65 | 7.17 | 7.09 | 7.19 | 0.024 | 0.030 | 0.032 | 0.032 | 0.032 |
| 7 | 0.002 | 0.005 | 0.005 | 0.005 | 0.004 | 10.14 | 13.60 | 14.88 | 14.71 | 13.88 | 0.036 | 0.050 | 0.055 | 0.053 | 0.049 |
| 8 | 0.001 | 0.002 | 0.002 | 0.002 | 0.002 | 4.86 | 6.65 | 7.17 | 7.10 | 6.77 | 0.022 | 0.031 | 0.033 | 0.033 | 0.031 |
| 9 | 0.003 | 0.004 | 0.005 | 0.005 | 0.005 | 8.72 | 11.53 | 12.23 | 12.39 | 12.46 | 0.037 | 0.050 | 0.053 | 0.053 | 0.053 |
| 10 | 0.002 | 0.004 | 0.004 | 0.004 | 0.003 | 8.34 | 11.06 | 11.61 | 11.48 | 10.63 | 0.035 | 0.047 | 0.049 | 0.049 | 0.045 |
| 11 | 0.002 | 0.004 | 0.005 | 0.005 | 0.004 | 7.28 | 10.26 | 11.34 | 11.32 | 11.02 | 0.032 | 0.047 | 0.052 | 0.052 | 0.050 |
| 12 | 0.001 | 0.002 | 0.002 | 0.003 | 0.003 | 4.85 | 6.77 | 7.54 | 7.83 | 8.50 | 0.023 | 0.033 | 0.036 | 0.038 | 0.041 |
| 13 | 0.001 | 0.003 | 0.004 | 0.003 | 0.003 | 5.96 | 8.69 | 9.69 | 9.57 | 9.67 | 0.027 | 0.041 | 0.045 | 0.045 | 0.045 |
| 14 | 0.002 | 0.004 | 0.005 | 0.005 | 0.004 | 7.94 | 10.95 | 11.67 | 11.43 | 10.85 | 0.034 | 0.049 | 0.053 | 0.051 | 0.048 |
| 15 | 0.001 | 0.002 | 0.002 | 0.002 | 0.002 | 5.79 | 7.64 | 8.00 | 7.89 | 7.41 | 0.026 | 0.034 | 0.036 | 0.035 | 0.033 |
| 16 | 0.002 | 0.004 | 0.005 | 0.004 | 0.004 | 7.25 | 10.37 | 11.36 | 11.11 | 10.64 | 0.032 | 0.047 | 0.052 | 0.050 | 0.047 |
| 17 | 0.001 | 0.002 | 0.002 | 0.002 | 0.002 | 5.20 | 7.14 | 7.59 | 7.59 | 7.55 | 0.025 | 0.034 | 0.036 | 0.036 | 0.036 |
| 18 | 0.001 | 0.002 | 0.002 | 0.002 | 0.002 | 5.91 | 7.91 | 8.37 | 8.18 | 8.02 | 0.026 | 0.034 | 0.036 | 0.035 | 0.034 |
| 19 | 0.002 | 0.004 | 0.005 | 0.004 | 0.004 | 8.03 | 11.39 | 12.22 | 11.95 | 11.38 | 0.034 | 0.049 | 0.052 | 0.051 | 0.048 |
| 20 | 0.002 | 0.003 | 0.004 | 0.004 | 0.004 | 6.53 | 9.11 | 9.92 | 9.78 | 9.93 | 0.030 | 0.043 | 0.047 | 0.046 | 0.047 |
| 21 | 0.002 | 0.003 | 0.004 | 0.004 | 0.003 | 6.55 | 9.71 | 10.96 | 10.99 | 10.30 | 0.029 | 0.043 | 0.049 | 0.049 | 0.045 |
| Ave. | 0.002 | 0.003 | 0.004 | 0.004 | 0.003 | 6.74 | 8.75 | 9.99 | 9.92 | 9.71 | 0.029 | 0.043 | 0.049 | 0.049 | 0.045 |

Table A.3: Table showing performance metrics for WGN model at each station across all prediction horizons

## A.5 Graph WaveNet Results

| | MSE | | | | | SMAPE | | | | | MAE | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **3** | **6** | **9** | **12** | **24** | **3** | **6** | **9** | **12** | **24** | **3** | **6** | **9** | **12** | **24** |
| 1 | 0.002 | 0.002 | 0.003 | 0.003 | 0.003 | 6.46 | 7.99 | 8.78 | 9.10 | 9.48 | 0.029 | 0.036 | 0.040 | 0.041 | 0.043 |
| 2 | 0.002 | 0.004 | 0.005 | 0.005 | 0.005 | 9.41 | 11.57 | 12.84 | 13.51 | 14.53 | 0.036 | 0.046 | 0.051 | 0.054 | 0.058 |
| 3 | 0.001 | 0.001 | 0.002 | 0.002 | 0.002 | 5.28 | 6.02 | 6.47 | 6.75 | 7.20 | 0.023 | 0.026 | 0.028 | 0.030 | 0.032 |
| 4 | 0.001 | 0.001 | 0.002 | 0.002 | 0.002 | 4.81 | 5.77 | 6.42 | 6.78 | 7.16 | 0.021 | 0.026 | 0.028 | 0.030 | 0.032 |
| 5 | 0.001 | 0.002 | 0.002 | 0.002 | 0.003 | 4.91 | 5.89 | 6.55 | 6.97 | 7.72 | 0.024 | 0.029 | 0.032 | 0.034 | 0.038 |
| 6 | 0.001 | 0.001 | 0.002 | 0.002 | 0.002 | 4.96 | 5.77 | 6.32 | 6.66 | 7.11 | 0.023 | 0.026 | 0.029 | 0.031 | 0.033 |
| 7 | 0.002 | 0.003 | 0.004 | 0.004 | 0.004 | 9.76 | 11.77 | 12.79 | 13.24 | 13.94 | 0.034 | 0.042 | 0.046 | 0.048 | 0.050 |
| 8 | 0.001 | 0.001 | 0.001 | 0.001 | 0.002 | 4.55 | 5.38 | 5.89 | 6.20 | 6.66 | 0.021 | 0.025 | 0.027 | 0.029 | 0.031 |
| 9 | 0.002 | 0.003 | 0.004 | 0.004 | 0.004 | 7.84 | 9.82 | 10.88 | 11.36 | 12.12 | 0.034 | 0.043 | 0.047 | 0.049 | 0.052 |
| 10 | 0.002 | 0.003 | 0.003 | 0.003 | 0.003 | 7.34 | 8.93 | 9.70 | 10.04 | 10.40 | 0.031 | 0.038 | 0.042 | 0.043 | 0.044 |
| 11 | 0.002 | 0.003 | 0.003 | 0.004 | 0.004 | 7.11 | 8.53 | 9.39 | 9.85 | 10.88 | 0.032 | 0.038 | 0.043 | 0.045 | 0.049 |
| 12 | 0.001 | 0.002 | 0.002 | 0.002 | 0.003 | 5.06 | 6.21 | 7.00 | 7.49 | 8.16 | 0.024 | 0.030 | 0.033 | 0.036 | 0.039 |
| 13 | 0.001 | 0.002 | 0.003 | 0.003 | 0.003 | 5.76 | 7.39 | 8.34 | 8.80 | 9.45 | 0.027 | 0.034 | 0.039 | 0.041 | 0.044 |
| 14 | 0.002 | 0.004 | 0.004 | 0.005 | 0.005 | 8.02 | 10.08 | 11.22 | 11.65 | 11.79 | 0.035 | 0.045 | 0.050 | 0.052 | 0.052 |
| 15 | 0.001 | 0.001 | 0.002 | 0.002 | 0.002 | 5.23 | 6.25 | 6.78 | 6.99 | 7.29 | 0.023 | 0.028 | 0.030 | 0.031 | 0.033 |
| 16 | 0.002 | 0.003 | 0.004 | 0.004 | 0.004 | 7.27 | 9.04 | 10.01 | 10.41 | 10.94 | 0.032 | 0.041 | 0.045 | 0.047 | 0.049 |
| 17 | 0.001 | 0.001 | 0.002 | 0.002 | 0.002 | 4.66 | 5.79 | 6.43 | 6.75 | 7.27 | 0.022 | 0.027 | 0.031 | 0.032 | 0.035 |
| 18 | 0.001 | 0.001 | 0.002 | 0.002 | 0.002 | 5.01 | 6.08 | 6.74 | 7.08 | 7.54 | 0.022 | 0.026 | 0.029 | 0.030 | 0.032 |
| 19 | 0.002 | 0.003 | 0.004 | 0.004 | 0.004 | 7.73 | 9.78 | 10.89 | 11.33 | 11.73 | 0.032 | 0.042 | 0.046 | 0.048 | 0.049 |
| 20 | 0.001 | 0.002 | 0.003 | 0.003 | 0.004 | 6.07 | 7.71 | 8.62 | 9.03 | 9.67 | 0.028 | 0.036 | 0.041 | 0.043 | 0.046 |
| 21 | 0.002 | 0.003 | 0.003 | 0.004 | 0.004 | 6.75 | 8.54 | 9.56 | 10.02 | 10.52 | 0.030 | 0.039 | 0.043 | 0.045 | 0.047 |
| **Ave.** | 0.002 | 0.002 | 0.003 | 0.003 | 0.004 | 6.39 | 7.82 | 8.65 | 9.05 | 9.60 | 0.028 | 0.033 | 0.038 | 0.040 | 0.044 |

Table A.4: Table showing the GWN results for each station across all prediction horizons

# Bibliography


[1] Sizwe Mzila. "Using Neural Networks to Predict Lightning Occurrences". PhD thesis. 2021.

[2] Yaseen Essa. "Thunderstorm Severity Prediction for South Africa between 2015 to 2016 using LSTM Model Variants and Remote Sensing Instruments". PhD thesis. 2022.

[3] P. Hewage, M. Trovati, E. Pereira, et al. *Deep learning-based effective fine-grained weather forecasting model*. 2020.

[4] Bing Yu, Haoteng Yin, and Zhanxing Zhu. "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence* (July 2018). DOI: 10.24963/ijcai.2018/505. URL: http://dx.doi.org/10.24963/ijcai.2018/505.

[5] Zonghan Wu et al. "Graph WaveNet for Deep Spatial-Temporal Graph Modeling". In: Aug. 2019, pp. 1907–1913. DOI: 10.24963/ijcai.2019/264.

[6] Kialan Pillay and Deshendran Moodley. "Exploring Graph Neural Networks for Stock Market Prediction on the JSE". In: *Artificial Intelligence Research*. Cham: Springer International Publishing, 2022, pp. 95–110. ISBN: 978-3-030-95070-5.

[7] Defu Cao et al. "Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting". In: (2021). eprint: arXiv:2103.07719.

[8] Zonghan Wu et al. "Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 753–763.

[9] T. Wilson, P. Tan, and L. Luo. "A Low Rank Weighted Graph Convolutional Approach to Weather Prediction". In: *2018 IEEE International Conference on Data Mining (ICDM)*. 2018, pp. 627–636. DOI: 10.1109/ICDM.2018.00078.

[10] Kouame Hermann Kouassi and Deshendran Moodley. "An analysis of deep neural networks for predicting trends in time series data". In: *Southern African Conference for Artificial Intelligence Research*. Springer. 2021, pp. 119–140.

[11] R.D. Moore et al. "Weather and Climate". In: *Draft* (Jan. 2008), pp. 3–55.

[12] Selim Furkan Tekin et al. "Spatio-temporal Weather Forecasting and Attention Mechanism on Convolutional LSTMs". In: *CoRR* abs/2102.00696 (2021). arXiv: 2102.00696. URL: https://arxiv.org/abs/2102.00696.

[13] Jonathan Weyn, Dale Durran, and Rich Caruana. "Can Machines Learn to Predict Weather? Using Deep Learning to Predict Gridded 500-hPa Geopotential Height From Historical Weather Data". In: *Journal of Advances in Modeling Earth Systems* 11 (July 2019). DOI: 10.1029/2019MS001705.

[14] Emmanuel Ayitey, Justice Kangah, and Frank Twenefour. "Sarima Modeling of Monthly Temperature in the Northern part of Ghana". In: 12 (Apr. 2021), pp. 37–45. DOI: 10.9734/AJPAS/2021/v12i330287.

[15] Peng Chen et al. "Time Series Forecasting of Temperatures using SARIMA: An Example from Nanjing". In: *IOP Conference Series: Materials Science and Engineering* 394 (Aug. 2018), p. 052024. DOI: 10.1088/1757-899X/394/5/052024.

[16] T. Dimri, Shamshad Ahmad, and Mohammad Sharif. "Time series analysis of climate variables using seasonal ARIMA approach". In: *Journal of Earth System Science* 129 (Dec. 2020). DOI: 10.1007/s12040-020-01408-x.

[17] Ratnadip Adhikari and R. Agrawal. *An Introductory Study on Time series Modeling and Forecasting*. Jan. 2013. ISBN: 978-3-659-33508-2. DOI: 10.13140/2.1.2771.8084.

[18] H. S. Hippert, C. E. Pedreira, and R. C. Souza. "Combining neural networks and ARIMA models for hourly temperature forecast". In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. Vol. 4. 2000, 414–419 vol.4. DOI: 10.1109/IJCNN.2000.860807.

[19] Jenny Cifuentes Quintero et al. "Air Temperature Forecasting Using Machine Learning Techniques: A Review". In: *Energies* 13 (Aug. 2020), p. 4215. DOI: 10.3390/en13164215.

[20] R.E. Abdel-Aal. "Hourly temperature forecasting using abductive networks". In: *Engineering Applications of Artificial Intelligence* 17 (Aug. 2004), pp. 543–556. DOI: 10.1016/j.engappai.2004.04.002.

[21] M. A. Jallal et al. "Air temperature forecasting using artificial neural networks with delayed exogenous input". In: *2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*. 2019, pp. 1–6. DOI: 10.1109/WITS.2019.8723699.

[22] Imran Tasadduq and Khaled Bubshait. "Application of neural networks for the prediction of hourly mean surface temperatures in Saudi Arabia". In: *Renewable Energy* 25 (Apr. 2002), pp. 545–554. DOI: 10.1016/S0960-1481(01)00082-9.

[23] Pedro Lanza and Jesús Zamarreño. "A Short-Term Temperature Forecaster Based on a Novel Radial Basis Functions Neural Network." In: *Int. J. Neural Syst.* 11 (Feb. 2001), pp. 71–77. DOI: 10.1016/S0129-0657(01)00050-3.

[24] Zulong Diao et al. "Dynamic Spatial-Temporal Graph Convolutional Neural Networks for Traffic Forecasting". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (July 2019), pp. 890–897. DOI: 10.1609/aaai.v33i01.3301890.

[25] Solichatus Zahroh et al. "Modeling and Forecasting Daily Temperature in Bandung". In: Nov. 2019.

[26] Daniele Liciotti et al. "A Sequential Deep Learning Application for Recognising Human Activities in Smart Homes". In: *Neurocomputing* 396 (Apr. 2019). DOI: 10.1016/j.neucom.2018.10.104.

[27] Mohamed Akram Zaytar and Chaker El Amrani. "Sequence to Sequence Weather Forecasting with Long Short-Term Memory Recurrent Neural Networks". In: *International Journal of Computer Applications* 143 (June 2016), pp. 7–11. DOI: 10.5120/ijca2016910497.

[28] Shaojie Bai, J. Kolter, and Vladlen Koltun. "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". In: (Mar. 2018).

[29] Yangdong He and Jiabao Zhao. "Temporal Convolutional Networks for Anomaly Detection in Time Series". In: *Journal of Physics: Conference Series* 1213 (June 2019), p. 042050. DOI: 10.1088/1742-6596/1213/4/042050. URL: https://doi.org/10.1088/1742-6596/1213/4/042050.

[30] Jennifer Miller, Janet Franklin, and Richard Aspinall. "Incorporating spatial dependence in predictive vegetation models". In: *Ecological Modelling* 202.3 (2007), pp. 225–242. ISSN: 0304-3800. DOI: https://doi.org/10.1016/j.ecolmodel.2006.12.012. URL: https://www.sciencedirect.com/science/article/pii/S0304380006006247.

[31] Rikiya Yamashita et al. "Convolutional neural networks: an overview and application in radiology". In: *Insights into Imaging* 9 (June 2018). DOI: 10.1007/s13244-018-0639-9.

[32] Zonghan Wu et al. "A Comprehensive Survey on Graph Neural Networks". In: *CoRR* abs/1901.00596 (2019). arXiv: 1901.00596. URL: http://arxiv.org/abs/1901.00596.

[33] S. Guo et al. "Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting". In: *AAAI*. 2019.

[34] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *CoRR* abs/1609.02907 (2016). arXiv: 1609.02907. URL: http://arxiv.org/abs/1609.02907.

[35] Yaguang Li et al. "Graph Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting". In: *CoRR* abs/1707.01926 (2017). arXiv: 1707.01926. URL: http://arxiv.org/abs/1707.01926.

[36] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. "Diffusion Improves Graph Learning". In: *CoRR* abs/1911.05485 (2019). arXiv: 1911.05485. URL: http://arxiv.org/abs/1911.05485.

[37] Kouame Hermann Kouassi and Deshendran Moodley. *An analysis of deep neural networks for predicting trends in time series data*. 2020. arXiv: 2009.07943 [cs.LG].

[38] J. Iseh.A. and Y. Woma.T. "Weather Forecasting Models, Methods and Applications". In: *International journal of engineering research and technology* 2 (2013).

[39] Feng-Wen Chen and Chen-Wuing Liu. "Estimation of the spatial rainfall distribution using inverse distance weighting (IDW) in the middle of Taiwan". In: *Paddy and Water Environment* 10 (Sept. 2012). DOI: 10.1007/s10333-012-0319-1.

[40] Huma Khan et al. "Data Preprocessing: A preliminary step for web data mining". In: (May 2019), pp. 206–221. DOI: 10.17993/3ctecno.2019.specialissue2.206-221.

[41] Davide Falessi et al. *On the Need of Preserving Order of Data When Validating Within-Project Defect Classifiers*. 2018. DOI: 10.48550/ARXIV.1809.01510. URL: https://arxiv.org/abs/1809.01510.

[42] Brian Ripley. *Pattern Recognition And Neural Networks*. Vol. 11. Jan. 2008. ISBN: 978-0521717700. DOI: 10.1017/CBO9780511812651.

[43] Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. Jan. 2013. ISBN: 978-1-4614-6848-6. DOI: 10.1007/978-1-4614-6849-3.

[44] Kouame Kouassi and Deshendran Moodley. *Automatic deep learning for trend prediction in time series data*. Sept. 2020.

[45] Li Yang and Abdallah Shami. "On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice". In: *Neurocomputing* 415 (July 2020). DOI: 10.1016/j.neucom.2020.07.061.

[46] James Bergstra and Y. Bengio. "Random Search for Hyper-Parameter Optimization". In: *The Journal of Machine Learning Research* 13 (Mar. 2012), pp. 281–305.

[47] Kassahun Tadesse and Megersa Dinka. "Application of SARIMA model to forecasting monthly flows in Waterval River, South Africa". In: *Journal of Water and Land Development* 35 (Dec. 2017). DOI: 10.1515/jwld-2017-0088.

[48] Peter Brockwell and Richard Davis. *An Introduction to Time Series and Forecasting*. Vol. 39. Jan. 2002. ISBN: 978-1-4757-2528-5. DOI: 10.1007/978-1-4757-2526-1.

[49] Francois Chollet et al. *Keras*. 2015. URL: https://github.com/fchollet/keras.

[50] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[51] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[52] Davide Chicco, Matthijs Warrens, and Giuseppe Jurman. "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation." In: *PeerJ Comput. Sci.* 7 (2021), 7:e623. DOI: 10.7717/peerj-cs.15. URL: https://doi.org/10.7717/peerj-cs.15.