Multivariate Forecasting for Spatio-Temporal Systems

A Study of Multi-Output Prediction and Accuracy Trade-Offs

Candidate Number: PGLR0

MSc Computational Statistics and Machine Learning[1]

Vasileios Lampos

Submission date: 22 September 2025

Abstract

This dissertation investigates how to extend Sonnet, a state-of-the-art deep learning model for time-series forecasting, from single-output to multi-output prediction in the context of spatio-temporal weather data. I propose *MultiSonnet*, which jointly forecasts temperature at 850 hPa across a 3×3 spatial grid, and evaluate it on five climatically diverse regions worldwide. Results reveal a clear trade-off. *MultiSonnet* delivers accurate forecasts for the peripheral grid points and consistently outperforms a seasonal persistence baseline. However, its accuracy at the central grid point is reduced compared to the original single-output Sonnet. To mitigate this issue, I introduce two weighted-loss variants, *Distance Weights*, which assigns fixed importance based off spatial proximity to the central grid, and *Learnable Weights*, which considers the weights trainable model parameters during optimisation. Both reduce central point degradation, but *Learnable Weights* achieves the best results, improving the MAE by an average of 2.6%, and enhancing peripheral accuracy at longer horizons. Regional analysis shows that spatial heterogeneity, as observed in Singapore's dataset, strengthens the benefits of multi-output learning, whereas more homogeneous regions limit the gains. Overall, this dissertation demonstrates when and how multi-output extensions of Sonnet can balance central and peripheral accuracy, and thus provide insight into the conditions under which spatial forecasting yields the greatest benefit.

Contents

Chapter 1

Introduction

1.1 Motivation

Weather is a complex spatio-temporal system, where conditions in one area are largely influenced by those in surrounding regions [45]. Effective forecasting therefore depends on models that learn both temporal and spatial dependencies across locations. For the last century, Numerical Weather Prediction (NWP) has dominated this field, using physics-based mathematical models to represent the atmosphere [30]. However, recent advances in machine learning, along with growing computational resources, have demonstrated that data-driven deep learning models can achieve competitive performance and complement traditional NWP approaches [25].

These developments open the door for data-driven forecasting approaches, where the focus shifts from explicitly modelling physical processes to learning patterns directly from historical data. In the context of machine learning, this naturally reduces to a multivariate forecasting problem, where multiple variables are jointly predicted over a given length in the future [44]. This setup allows for models to learn relationships among variables and across time and has generally been shown to improve predictive accuracy compared to modelling each variable independently [17, 23, 42]. In the context of weather forecasting, multivariate models naturally involve either predicting different meteorological features or the same feature across various locations.

Because of this, many state-of-the-art deep learning time series models are tested on weather forecasting benchmarks. One such model, Sonnet [44], introduced a novel architecture centred around a frequency-domain attention mechanism. It achieved superior performance in forecasting the temperature at 850 hPa (T850), as well as on other tasks

like influenza-like-illness rate and oil temperature prediction. However, Sonnet's weather experiments were limited to predicting a single meteorological feature, T850, at a single location. Features measured at surrounding locations were treated as exogenous inputs rather than additional forecasting targets.

This raises a natural question: instead of using neighbouring locations solely as exogenous inputs, can forecasts be improved by predicting them jointly? Framing the task this way turns it into a multivariate forecasting problem, where the same variable, T850, is predicted across multiple locations simultaneously. This can naturally be viewed as a multi-task learning (MTL) problem, with each location treated as a separate, but related forecasting task. Prior research in MTL shows that joint training can improve generalisation when tasks are strongly related, but may harm performance when relationships are weak [1].

In this dissertation, I address this gap by developing *MultiSonnet*, a multi-output extension of Sonnet designed to forecast T850 across multiple locations jointly. I evaluate its performance on five regions worldwide, with T850 forecasts made simultaneously at nine sub-locations within each region. The primary objective is to improve forecasts at the central point of each region, which serves as the benchmark for comparison to the original single-output Sonnet model. Additionally, to further investigate how spatial structure influences performance, I introduce and compare two loss-weighting strategies: *Distance Weights* and *Learnable Weights*, which bias training towards the centre of each region.

The results reveal that multi-output forecasting introduces a trade-off: forecast at surrounding points improve, but accuracy at the central location declines unless weighted loss strategies are used. Of the two approaches tested, *Distance Weights* offers limited short-horizon gains, while *Learnable Weights* consistently reduces central error and strengthen peripheral forecasts. Performance also varies by region, with heterogeneous areas such as Singapore benefiting the most from joint prediction.

Building on this motivation, this dissertation is guided by three key research questions, designed to systematically evaluate the impact of multi-output forecasting and the role of spatial structure:

- **RQ1:** How does extending Sonnet from single-target to multi-target forecasting affect performance, particularly at the central location?
- **RQ2:** How does joint forecasting with a location-weighted loss influence predictive accuracy at the central location?

- **RQ3:** How do spatial and temporal dependencies shape the effectiveness of multi-target forecasting?

1.2 Report Overview

The subsequent chapters of this dissertation are organised as follows:

The next chapter, entitled **Background**, highlights the relevant work that motivates this dissertation. It presents background on time series forecasting and how machine learning model approaches differ from widely-used statistical forecasting models. It then discusses Attention, an integral component of deep learning architectures in recent years and the backbone of Sonnet, which is then discussed in detail. Finally, it examines how the multi-output model fits between two seemingly contradictory paradigms throughout machine learning literature: Global Forecasting Models and Multi-Task Learning, which motivate different weighted loss strategies.

The following chapter, entitled **Methodology**, is broken up into two subsections: *Multi-Region Weather Datasets* and *Experiments*. The *Multi-Region Weather Datasets* section describes and analyses each dataset for the five tested regions as well as the training, validation, and testing splits used for all experiments. The *Experiments* section then lays out the methodology and setup of three multi-output variants of Shu and Lampos's Sonnet model. It covers architecture, training, hyperparameter tuning, and evaluation, as well as the motivation for each step in the process.

Next, the chapter entitled **Results and Discussion** compares the results of each model variant relative to the single-output model and to each other. It systematically addresses each of the three research questions from Section 1.1 and provides the main discussion on how model performance varies by region, horizon, and the underlying spatial structure of the targets.

The final chapter, entitled **Conclusion**, summarises the main results and contributions of this dissertation as well as a discussion of the limitations and future work.

Chapter 2

Background

This section lays out the prior work that motivated the design and experiments tested in this dissertation. Specifically, it starts with an overview of time series forecasting models, moving from traditional statistical models to more complex deep learning architectures. Next, it introduces Attention, a key component of many deep learning models, including Shu and Lampos's Sonnet [44]. Then, it gives an overview of Sonnet's architecture. Finally, it outlines a key contradiction between global forecasting models and multi-task learning research, and how this contradiction shapes my hypotheses.

2.1 Time Series Forecasting

Time series forecasting involves predicting future values of a sequence given past observations. Traditional statistical models like autoregressive integrated moving average (ARIMA) and vector autoregression (VAR) have long been standard approaches due to their simplicity. ARIMA models capture dependencies within a single time series by exploiting relationships with its past values and errors, while VAR extends this framework to model linear interdependencies among multiple time series [32]. While interpretable and practical under assumptions of linearity and stationarity, these approaches often struggle with the complex, non-linear dynamics observed in real-world meteorological data [41].

With advances in computation, many works have shown that neural-network-based forecasting models outperform traditional statistical models on the most complex forecasting tasks. For example, De Saa et al. [19] used hourly temperature data in Hungary and compared an ARIMA model with a deep learning model that used convolutional layers to extract spatial dependencies and LSTM layers to capture temporal dynamics. Their results

showed that the deep learning model reduced the mean absolute error by 21% compared to the ARIMA model. Similar improvements have also been reported outside of meteorological forecasting; for instance, Namini et al. [11] found that a simple long-short-term memory (LSTM) architecture reduced the forecast error by $84\% - 87\%$ compared to an ARIMA model across multiple horizons of financial data. More broadly, Cerqueira et al. [14] found that deep learning models tend to outperform classical time series models as the sample size of the data increases.

Neural-network-based methods, such as those presented by Namini et al. and De Saa et al., approach time series forecasting by framing it as a supervised learning problem. Formally, let $\mathbf{x}_t = [x_t^{(1)}, x_t^{(2)}, \dots, x_t^{(n)}]^\top$ denote a vector of observations at time t. Given a lookback window of L past observations, the model is trained to learn a function f_θ that maps the sequence of past observations to the next H future values:

$$f_\theta : (\mathbf{x}_{t-L}, \dots, \mathbf{x}_t) \longmapsto (\mathbf{x}_{t+1}, \dots, \mathbf{x}_{t+H}), \tag{2.1}$$

where θ are the model parameters and H the forecasting horizon, which can range from short, one-step ahead forecasts to longer, multi-step predictions. The model is then trained by minimising a loss function $\mathcal{L}(\theta)$, typically the mean squared error (MSE), over a dataset of N training examples. These examples are created by extracting all possible contiguous windows of length $L + H$ from the time series, using the first L points as input and the subsequent H points as the target:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left\| f_\theta(\mathbf{x}_{(i-L):i}) - \mathbf{y}_i \right\|^2, \tag{2.2}$$

where $\mathbf{y}_i = (\mathbf{x}_{i+1}, \dots, \mathbf{x}_{i+H})$ denotes the vector of target future values for the ith training example. Gradient-based optimisation algorithms, such as Stochastic Gradient Descent or Adam [4], are then used to iteratively update the model parameters θ to minimise this loss.

Framing forecasting as a supervised learning problem is more flexible than traditional models. Now, the inputs and outputs can both be represented as matrices or tensors, which allows for exogenous variables or multiple targets to be seamlessly incorporated into the same framework without the heavy statistical assumptions present in models like VAR or ARIMAX. As a result, these approaches can model the non-linear, complex interdependencies within the data that models like ARIMA and VAR struggle to capture.

Throughout deep-learning-based time series literature, there exist two distinct forecast-

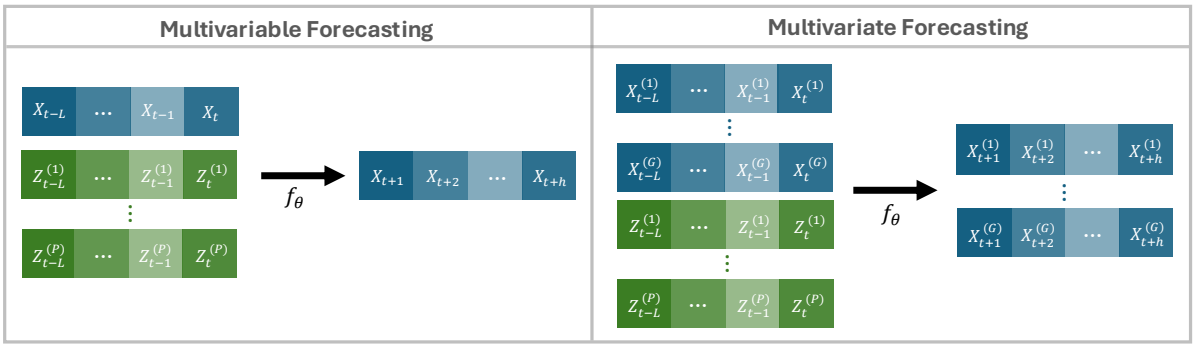


Figure 2.1: The difference between multivariable and multivariate forecasting setups. The left-hand side illustrates a multivariable forecasting setup with P exogenous features and one target. The right-hand side illustrates a multivariate forecasting setup with P exogenous features and G targets. Each X represents endogenous variables, while Z represents exogenous variables.

ing setups that will be referenced throughout this dissertation: Multivariate and Multivariable forecasting. Multivariate forecasting models use information from multiple features to generate joint predictions for several targets (endogenous features) simultaneously, capturing dependencies amongst all target series [44]. As shown on the right-hand side of Figure 2.1, for G targets and P exogenous variables, the model f_θ produces a forecast for all G targets over all steps in the forecasting horizon H. In contrast, multivariable forecasting models also use multiple input series, but focus on predicting a single target variable, treating the other series as exogenous predictors [44]. Similarly, as shown on the left-hand side of Figure 2.1, the model f_θ maps P exogenous features to the single target over the horizon H.

Because time series data is not independently and identically distributed (IID), specialised architectures such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and attention-based models have been developed to capture temporal dependencies explicitly. Recurrent and convolutional networks have demonstrated strong performance [19, 11], but often struggle with long-range dependencies due to the vanishing gradient problem [33]. Attention-based architectures address these challenges by allowing the model to focus on the most relevant past observations and variables when making predictions, making them particularly well-suited for multivariable and multivariate forecasting tasks [26, 44].

2.2 Attention

Attention has become a fundamental building block in machine learning and is the foundation of the many state-of-the-art time series forecasting models [44, 26, 39, 38]. By allowing models to dynamically focus on the most relevant parts of the input time series when making predictions, Attention overcomes the limitations of recurrent and convolutional networks in capturing long-range dependencies over long forecasting horizons [34]. First widely used in natural language processing through the Transformer architecture [8], Attention has since been commonly used in many forecasting tasks, including weather prediction.

Attention mechanisms learn a weighted combination of an input time series by mapping it into three learned linear transformations: queries (Q), keys (K), and values (V). A dot product is taken between each query and key to output a measure of similarity, which is then scaled by the dimensionality of the keys d_k. Finally, a softmax is used to normalise these and generate a set of attention weights. The output is a vector that is a weighted sum of the value vectors V using these weights [8]:

$$\text{Attention}\left(Q, K, V\right) = \texttt{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V\,. \tag{2.3}$$

The resulting attention vector gives a representation of the input time series that emphasises the most relevant information.

In the context of multivariable and multivariate forecasting, Attention serves as a powerful building block. It can be adapted to capture short and long-range temporal dependencies as well as complex interactions across endogenous and exogenous features. Consequently, many models use specialised attention variants that selectively weight information from different time steps, features, or both. As a result, models are able to learn which past observations and variables are most informative for prediction.

Spatial Feature Attention Long Short Term Memory (SFA-LSTM) [29] is one such model. SFA-LSTM extends a standard LSTM by integrating a form of spatial feature attention in the decoder to capture spatial and temporal dependencies in Canadian weather data. They found that their spatial attention variant reduces the mean squared error by 76.2% on average compared to other LSTM variants.

Deformtime [38] introduces two deformable attention blocks that capture variable and temporal dependencies separately. Each block dynamically focuses on the most relevant past observations, rather than all inputs equally. Shu and Lampos found that it outper-

formed other competitive forecasting models by 7.2% on average across different tasks, including oil temperature forecasting and influenza-like-illness (ILI) rate forecasting across various regions.

Building on these developments, Sonnet [44] introduces a variant of Attention that operates in the frequency domain. Rather than comparing values at each time step directly, working in the frequency domain allows the Attention to focus on how the signal fluctuates over different frequencies, which is particularly useful for highly seasonal data like weather patterns. The following section provides a detailed overview of Sonnet, including its architecture and the specific Multivariable Coherence Attention mechanism (MVCA) that underpins its performance.

2.3 Spectral Operator Neural Network (Sonnet)

Spectral **O**perator **N**eural **Net**work (Sonnet), developed by Shu and Lampos [44], is a specific attention-based architecture designed for multivariable forecasting tasks. Shu and Lampos found that Sonnet outperforms other state-of-the-art models from literature on 72% of their tested tasks. They also highlighted the value of Sonnet's attention variant, denoted Multivariable Coherence Attention (MVCA). When MVCA is used in place of the standard Attention shown in Equation 2.3, it reduces the mean absolute error by 10.7% on average across various benchmark tasks.

Sonnet consists of five main components: Embedding, Adaptive Wavelet, Multivariable Coherence Attention, Koopman Operator, and Decoder.

Embedding

The Sonnet architecture was experimented with on multivariable forecasting tasks. The input consists of a single target variable $\mathbf{y} \in \mathbb{R}^{L}$ and the remaining $N-1$ features $\mathbf{X} \in \mathbb{R}^{L\times(N-1)}$ treated as exogenous, where L denotes the lookback window. In order to ensure that all N variables are on the same scale and to create a richer representation, the data must be embedded into a latent dimension of dimensionality d. By transforming raw time series data into this latent space, embeddings can uncover relevant features and temporal dependencies that are not immediately obvious in the original data [43], such as patterns in weather data. This is done by separately passing $\mathbf{X}$ and $\mathbf{y}$ through a single feed-forward layer, which projects the data into a latent space to form $E_x \in \mathbb{R}^{L\times\alpha d}$ and $E_y \in \mathbb{R}^{L\times(1-\alpha)d}$ respectively, where $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$ is used to control the relative

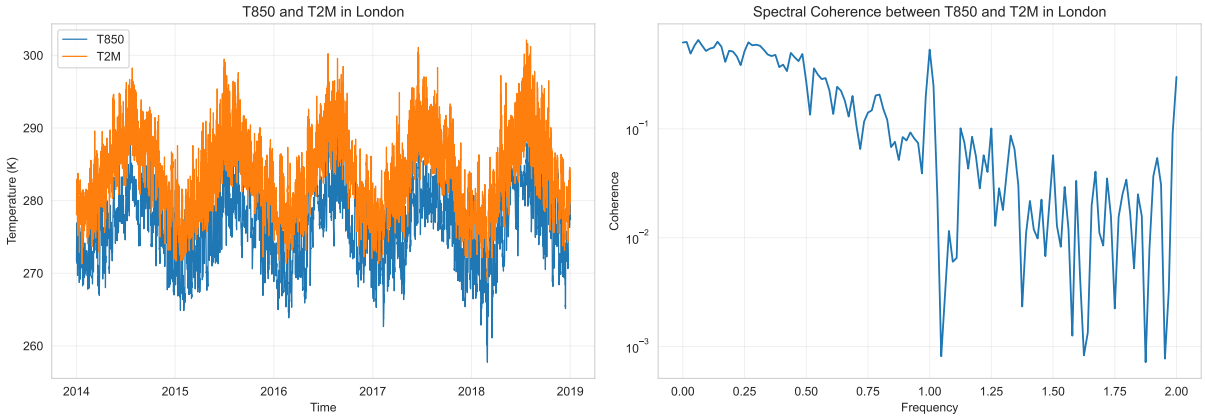


Figure 2.2: (left) The temperature at 850 hPa (T850) and at two metres (T2M) in London from 2014-2018. (right) Spectral coherence between T2M and T850 in London for the same period. Since measurements are taken every six hours, the frequency axis corresponds to cycles per day, with four observations per cycle.

dimensionalities that the target and the exogenous features take in the final embedding. The final embedding is then the concatenation of these latent representations to form $E \in \mathbb{R}^{L \times d}$ [44].

Adaptive Wavelet

Now that the data is embedded into a latent space, the Adaptive Wavelet layer projects it into learnable time-frequency representations. Specifically, a set of K learnable wavelet atoms $\{M_k \in \mathbb{R}^{d \times L}\}_{k=1}^{K}$ is defined, each parametrised to capture localised temporal and spectral structures. The embedding E is then projected onto each atom via element-wise multiplication to produce $P_k = E \odot M_k^\top$, creating a set of K coefficient matrices $P \in \mathbb{R}^{K \times L \times d}$. This decomposition retains the temporal structure of the original data while highlighting patterns across various timescales, and thus provides a rich, informative input for the next step: Multivariable Coherence Attention [44].

Multivariable Coherence Attention (MVCA)

Multivariable Coherence Attention (MVCA) operates on the set of wavelet coefficients $P \in \mathbb{R}^{K \times L \times d}$, produced by the Adaptive Wavelet layer. For each wavelet component, the coefficient matrix is linearly projected into queries (Q), keys (K), and values (V), similar to standard attention shown in Equation 2.3. Queries and keys are then transformed via a Fast Fourier Transform (FFT), and spectral coherence replaces the standard dot-product

similarity to compute attention weights. These coherence-based scores are normalised, scaled, and applied to the values, followed by a multi-layer perceptron (MLP) and residual connection to produce the final output [44].

Working in the frequency domain allows MVCA to capture how variables co-move across different timescales rather than simply comparing them step by step in time. This is particularly important for weather data, which exhibits multiple nested layers of seasonality and thus long-range dependencies.

For example, Figure 2.2 shows a side-by-side comparison of the time domain and frequency domain of the temperature measured at two metres (T2M) and at 850 hPa (T850) in London from 2014 to 2018. The time series plot (left) highlights that the two variables evolve together, especially on seasonal cycles, but it does not reveal at which timescales their relationship is strongest. On the other hand, the coherence plot (right) adds this information, showing that T2M and T850 are strongly related at low frequencies (seasonal to annual), while coherence weakens at higher frequencies (daily to multi-day variations). MVCA leverages these frequency-specific relationships by emphasising the components of the wavelet-transformed data that are most informative for forecasting [44].

Koopman Operator

Now, the Koopman Operator layer models how the MVCA embeddings $O \in \mathbb{R}^{K \times L \times d}$ evolve over time in a complex latent space. Specifically, this layer learns a linear transformation $K \in \mathbb{C}^{K \times K}$ that captures phase shifts for each wavelet component. The complexified input O_c is multiplied by K to produce the evolved embeddings $O_l = KO_c$, allowing the model to capture long-range temporal dependencies while also reducing the sequential error accumulation that is typical of other models like RNN [44].

Decoder

Finally, the decoder maps the evolved embeddings O_l from the Koopman Operator back to the dimensionality of the target forecast. This is implemented as a series of one-dimensional convolutional layers with GELU activations. The hidden size is progressively decreased to match the forecast horizon and is completed with an adaptive average pooling layer. This design allows the model to aggregate all of the temporal information that the model learned to produce predictions for each step ahead [44].

2.4 Multi-Output Modelling

While the preceding sections focused on time series forecasting and the specifics of Sonnet, the original Sonnet model was only experimented with for multivariable forecasting tasks. However, in this dissertation, I extend Sonnet to a multi-output setting (multivariate), where each model forecasts T850 jointly at several locations. This setup, illustrated on the right-hand side of Figure 2.1, can be interpreted in two similar ways. Within machine learning literature, it can naturally be viewed as a multi-task learning (MTL) problem, where each target series corresponds to a separate, learned task. However, within time-series forecasting literature, it can be seen as a global forecasting model (GFM), where instead of training one univariate model, a single global model learns across multiple series simultaneously. These two perspectives and their differences motivate many of the hypotheses and experiments throughout this paper.

Multi-task learning (MTL) is a machine learning approach in which multiple related tasks are learned jointly, rather than independently. First introduced by Caruana [1], MTL has since been widely used across domains like natural language processing [36], machine vision [6], and time series forecasting [18, 5]. The main idea behind it is that related tasks often share underlying structure, and thus by training them together, a single model can learn common temporal and cross-variable patterns between the series [1]. This is typically implemented via hard parameter sharing, where most or all of the model parameters are shared across tasks [7]. Caruana and many others have found that by training multiple tasks together with shared parameters, the risk of overfitting decreases, and thus it improves generalisation [1].

However, MTL also carries risks. Specifically, when tasks are not sufficiently related, joint modelling can cause negative transfer, where individual performance on each task degrades because of parameter sharing [15, 21]. The challenge, therefore, lies in striking the right balance between each of the modelled tasks.

On the other hand, Global Forecasting Models (GFM) embody the same principle within time series forecasting. When forecasting multiple time series, a GFM assumes that all series come from a single underlying data-generating process, rather than assuming a different process for each series [17]. Like multi-task learning, a GFM mitigates the risk of overfitting compared to a local, univariate-output model due to the larger sample size [24]. Likewise as well, many studies have shown these to be superior to local models [27, 37].

However, contrary to much of MTL literature, there is also much work that claims that GFMs achieve superior performance to local models even when the modelled series are

not sufficiently related or relevant to each other. For example, Sen et al. [17] proposed DeepGLO, which was trained to jointly forecast more than $100,000$ non-related time series and achieved a 25% reduction in error compared to other models. Additionally, Montero-Manso et al. [24] further provided a theoretical proof, showing that the forecasts of any univariate model can be exactly reproduced by a global model that includes the same series.

The natural question that arises in both MTL and GFMs is how to balance the contribution of each task or series during training. Assigning equal weights to all tasks is a common baseline, but this implicitly assumes that all tasks are equally important and equally difficult, which is often not true. Tasks can differ in scale, amount of noise, predictive difficulty, and, in our case, experimental importance. This imbalance can cause certain tasks to dominate the optimisation as they possess gradient conflict [15]. To address this, many have proposed weighted loss functions, in which each task's loss is scaled by a weight that reflects its relative contribution. In other words, each task's loss $\mathcal{L}_i(x)$ is scaled by a weight ϕ_i such that $\sum_{i=1}^{I} \phi_i = 1$. Thus, the total loss can be written as:

$$\mathcal{L}(x) = \sum_{i=1}^{I} \phi_i \, \mathcal{L}_i(x) \, . \tag{2.4}$$

There have been several different variants of weighted loss throughout different domains. For example, in classification tasks with imbalanced data, labels are often weighted inversely proportional to their frequencies to reduce bias toward majority classes [3]. Similarly, in MTL, task-specific weighting schemes have been proposed to dynamically update ϕ_i during training based on the given task's uncertainty [10], gradient magnitudes [9], and attention masks [16]. Other approaches, like Zhang et al. [46], treat the weights as learnable parameters that are jointly optimised with the model parameters.

2.5 Summary

Overall, this chapter has outlined the necessary background on time series forecasting, attention mechanisms, and Sonnet. It has also positioned the problem of extending Sonnet beyond its original multivariable setup into a multivariate setting, where forecasts are generated jointly across multiple spatial locations.

This chapter reveals a fundamental contradiction between multi-task learning (MTL) and global forecasting models (GFMs). MTL emphasises the need for relatedness across

tasks to avoid negative transfer, whereas GFMs often report gains even when series are heterogeneous. The Sonnet extension in this work lies between these two perspectives: it does not fully align with MTL, as all outputs contribute to a single weighted global loss rather than task-specific losses; nor is it a traditional GFM, since forecasts are restricted to a fixed subset of nine target locations, with other variables treated as exogenous predictors.

This middle ground motivates the following experiments in Chapter 3. By jointly forecasting T850 at spatially distributed locations, this paper investigates whether performance aligns more with MTL, where similarity and correlation across targets are critical, or with GFMs, which suggest joint modelling can succeed regardless of relatedness. Furthermore, I incorporate different weighted loss setups to encode my prior knowledge that the central location is the most informative, as well as to derive a proxy for the underlying latent spatial structure of the data.

Chapter 3

Methodology

In this chapter, I first introduce the datasets used for all experiments as well as how each dataset is split into training, validation, and testing sets. Then, I provide a detailed overview of all experiments and outline the process of extending Sonnet to handle multivariate forecasting tasks on these datasets.

3.1 Multi-Region Weather Datasets

All models are evaluated independently on five datasets corresponding to geographically and climatically diverse regions centred around the following cities: London, New York, Hong Kong, Singapore, and Cape Town. These areas were intentionally selected to span both hemispheres and to encompass different tropical, subtropical, Mediterranean, and oceanic climate zones. By testing each model across these different environments, I can evaluate how robust Sonnet performs under different weather phenomena and degrees of seasonality.

For each city c, its dataset $\mathcal{D}_c$ covers the 38 year period from 1980 to 2018, with measurements recorded every six hours (four time steps per day at 00:00, 06:00, 12:00, and 18:00). At each time step, five meteorological features are recorded: the temperature at 850 hPa measured in Kelvin (T850), the temperature at two metres measured in Kelvin (T2M), the zonal and meridional components at 10 metres (U10, V10), and the geopotential at 500 hPa (Z500).

These variables are measured at nine distinct locations extracted from a 3×3 grid neighbourhood around each city centre. Let $(\text{lat}_C, \text{long}_C)$ be the latitude and longitude of each city centre. Using a $r = 5.625°$ resolution grid, each grid cell represents the bilinearly

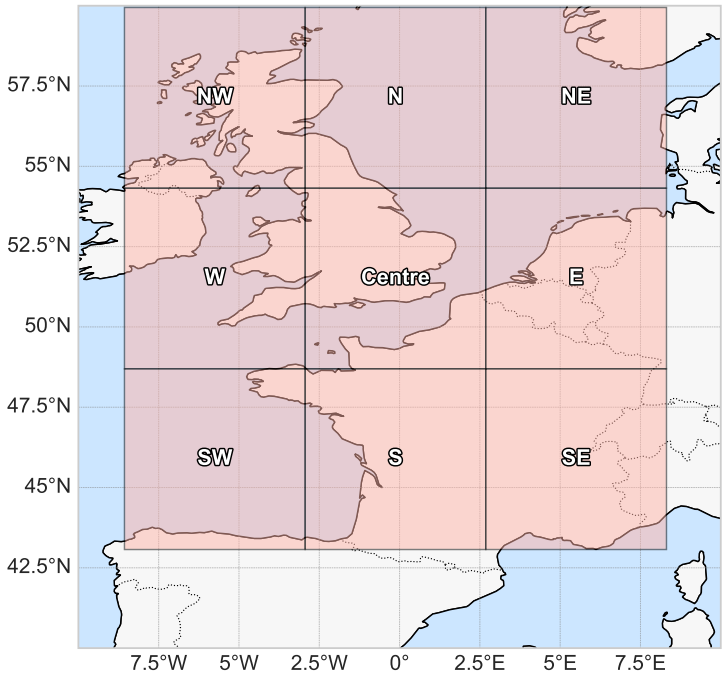


Figure 3.1: Map showing where each of the five features is measured in the London dataset. Each square represents the corresponding grid location, where each feature is sampled.

interpolated [22] value of the measured variable over a square region of size $r \times r$ centred at:

$$(\text{lat}, \text{long}) = (\text{lat}_C + \Delta_{\text{lat}}, \text{long}_C + \Delta_{\text{long}}) \tag{3.1}$$

where $\Delta_{\text{lat}}, \Delta_{\text{long}} \in \{-r, 0, r\}$ are the corresponding offsets in the directions: North (N), South (S), East (E), West (W), Northeast (NE), Northwest (NW), Southeast (SE), Southwest (SW), and Centre (C). Figure 3.1 illustrates the surrounding locations used in the London dataset, represented as distinct grid cells. The surrounding grid incorporates locations across the UK, Continental Europe, and the North Sea, which are all tied together by similar large-scale atmospheric patterns. Other work has shown that incorporating variables from surrounding locations, like this, can significantly improve forecasting performance. For example, Fowdur et al. [28] found that across a variety of machine learning models, incorporating surrounding locations as predictors reduced the mean absolute percentage error (MAPE) by 5% compared to models that relied only on a single location. Altogether, each dataset consists of 45 features (nine grids $\times$ five meteorological variables) across $58,440$ time steps per region. Each meteorological feature $f \in$ {T850, T2M, U10, V10, Z500}

measured at grid $g \in \{\text{N, S, E, W, NE, NW, SE, SW, C}\}$, will be referred to as $f_{(g)}$ for the remainder of the text.

The data are derived from WeatherBench [22], which is a lower-resolution regridded subset of data from ERA5 [20]. The data throughout this paper matches that of the weather experiments in Shu and Lampos's original Sonnet paper [44] to ensure comparability with prior results.

3.1.1 Train-Validation-Test Split

In order to train and evaluate different models, I split the data into training, validation, and testing sets. The training set is used to fit the model parameters. The validation set is used to tune the model hyperparameters. The testing set is reserved exclusively for the final evaluation and reporting of results.

Because weather conditions vary considerably across different years, I employ three distinct splits and report the average performance across all of them, as used by Shu and Lampos [44]. For example, the El Niño/La Niña Southern Oscillation (ENSO) is an irregular variation in winds and sea surface temperature of the Pacific Ocean, which is one of the main sources of year-to-year climate variability [12]. Due to teleconnections, this has significant impacts on weather patterns across the globe, including the five regions included in the datasets [12]. Relying on a single split could therefore overestimate or underestimate performance depending on the specific conditions (e.g. the phase of ENSO) of that particular year.

To illustrate, the average $\text{T850}_{(\text{C})}$ for each Winter in New York ranges from $274.3\,K$ to $277.3\,K$ with a standard deviation of $0.79\,K$, while Spring shows even greater year-to-year fluctuations with a standard deviation of $1.09\,K$. If a single test set happened to include one of these extreme years, then model performance could be biased. Thus, averaging across three distinct splits mitigates this risk and gives a more complete overview of each model's performance. This is consistent with the findings of Aziz et al. [35], who showed that employing multiple splits for time series models significantly reduces performance bias.

Because the dataset spans from 1980 to 2018, the test years are chosen as the final three: 2016, 2017, and 2018. For each test year, the validation set is defined as the immediately preceding year, while all earlier years form the training set. Choosing the validation set as the year directly before the test set means that none of the models are trained on the year directly before testing, which in many time series problems would be the most informative for predicting the next period. However, weather data exhibits strong annual seasonality,

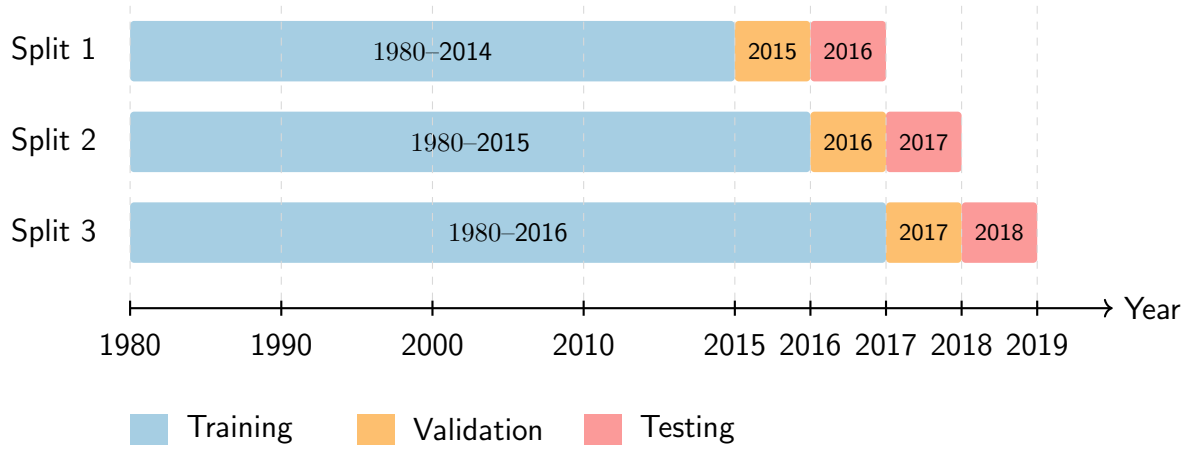


Figure 3.2: Training, validation, and testing splits used for each region's dataset and for all experiments

so most patterns repeat year to year. While minor year-to-year variations exist and climate change has been shifting some trends, their impact over a single year is relatively small. As a result, I do not expect this to have much of an effect on the final results.

Figure 3.2 summarises each of the three splits and shows how the training, validation, and testing sets are distributed across the varying years.

3.2 Experiments

For each region's dataset, described in Section 3.1, I experiment with extending the multivariable Sonnet architecture of Shu and Lampos [44] to the multivariate forecasting setting. In this setup, each model is tasked with jointly predicting T850 at all nine sub-locations within a region.

Although each model generates predictions for all sub-locations, the primary focus for addressing RQ1 and RQ2 (see Section 1.1) is how performance at the central point $\text{T850}_{(\text{C})}$ changes as the setting moves from multivariable to multivariate. The central point is chosen as the baseline for comparison amongst the nine predicted targets, because it is expected to be the most representative point of each region. This is because it is equidistant from the surrounding grids, which creates a symmetric and rich surrounding feature space that the other points do not fully possess.

The rest of this section presents four models: *MultiSonnet*, *Distance Weights*, *Learnable Weights*, and *Seasonal Persistence*. *MultiSonnet* does not directly bias any grid during

training, whereas *Distance Weights* and *Learnable Weights* are experimented with to improve the performance of $T850_{(C)}$ explicitly. *Seasonal Persistence* is used as a competitive, naive baseline that serves as comparison to the forecasts of the surrounding sub-locations that will help to address RQ3. For each model, the multivariate setup and specifics of each model must first be defined.

3.2.1 Multivariate Setup

To extend Sonnet into a multivariate forecasting setting, the input at each time step t consists of F features measured at the 3×3 neighbourhood of grid points described in Section 3.1:

$$\mathbf{X}_t \in \mathbb{R}^{L \times (G \times F)},$$

where L denotes the size of the look-back window, G the number of grid points in the neighbourhood, and F the number of meteorological features recorded at each grid point. The corresponding prediction target is:

$$\mathbf{Y}_t \in \mathbb{R}^{H \times G},$$

where H denotes the length of the forecast horizon. This formulation will now allow Sonnet to produce a joint trajectory of future values across all $G = 9$ spatial locations, rather than a single univariate series.

All experiments are tested across forecast horizons $H \in \{4, 12, 28\}$, corresponding to one day, three days, and one week, respectively. Given that the data is recorded at a six-hour resolution, this choice of horizons allows me to evaluate model performance on short, medium, and long-term forecasts. Longer horizons naturally accumulate more error at each step, so overall forecasting accuracy is expected to decrease as H increases. Forecasting long horizons at high-resolution data, for example, one week ahead at one hour resolution, would require 168 steps and is likely impractical. In contrast, the 6-hour resolution used in each region's dataset would only need 28 steps, and thus it provides a practical balance between forecast length and cumulative error.

The window of past observation, denoted L, contains a fixed number of past observations that the model sees during training for each step. For all experiments, it is fixed according to that of the original Sonnet model:

$$L(H) = \begin{cases} 28, & H \in \{4, 12\} \\ 56, & H = 28 \ . \end{cases}$$

Each of the three Sonnet variants (*MultiSonnet, Distance Weights, Learnable Weights*) employ this setup. They differ from the original Sonnet model in regard to the embedding, output projection, and training methods.

3.2.2 MultiSonnet: Equal Weight Baseline

To establish a baseline for multivariate forecasting, I extend Sonnet's univariate output projection $\mathbb{R}^d \to \mathbb{R}^1$ to a multivariate mapping $\mathbb{R}^d \to \mathbb{R}^G$, where d is the decoder embedding dimension. This allows the model to produce predictions simultaneously for all G grid points.

The training loss is adapted accordingly. Because the model now predicts a matrix $\mathbf{Y}_t \in \mathbb{R}^{H \times G}$, the loss must reflect the error across all time steps in the forecast horizon and across each of the spatial locations. In *MultiSonnet*, I treat all targets equally by averaging the mean squared error over all grid points and forecast steps. For a batch of T training windows, where each window is defined by its start time t, the loss can be written as:

$$\mathcal{L}(\theta) = \frac{1}{THG} \sum_{t=1}^{T} \sum_{h=1}^{H} \|\mathbf{y}_{t+h} - \hat{\mathbf{y}}_{t+h}\|_2^2 \ , \tag{3.2}$$

where $\mathbf{y}_{t+h}, \hat{\mathbf{y}}_{t+h} \in \mathbb{R}^G$ are the vectors of true and predicted values over the G spatial locations at forecast step h :

$$\mathbf{y}_{t+h} = \begin{bmatrix} y_{t+h,1} \\ y_{t+h,2} \\ \vdots \\ y_{t+h,G} \end{bmatrix}, \quad \hat{\mathbf{y}}_{t+h} = \begin{bmatrix} \hat{y}_{t+h,1} \\ \hat{y}_{t+h,2} \\ \vdots \\ \hat{y}_{t+h,G} \end{bmatrix},$$

and $\| \cdot \|_2$ denotes the Euclidean norm over the G grid points.

MultiSonnet is a straightforward extension of Sonnet to forecast each of the nine grid points, and thus it is considered the multivariate baseline for all future experiments. However, the uniform weighting in the loss function considers each of the grid points as equally important. Thus, it does not account for the practical importance of the central grid point $\text{T850}_{(\text{C})}$, which is the primary evaluation target. Therefore, equal weighting may

under-represent performance on the most critical central location. This motivates the introduction of a weighted loss function.

3.2.3 Distance Weights

To explicitly prioritise the central grid point $T850_{(C)}$, I modify the loss function, while keeping the same multivariate output projection $\mathbb{R}^d \to \mathbb{R}^G$. I introduce a new hyperparameter $\phi_C \in [0, 1]$ which controls the relative importance of $T850_{(C)}$ in the loss function. The remaining $G - 1$ grid points are scaled proportionally to their relative distance from the central point. As discussed in Section 3.1, each surrounding location was sampled uniformly from the centre of a $5.625°$ resolution grid and thus their relative distances to the central point are equal, such that

$$\phi_i = \frac{1 - \phi_C}{G - 1}, \quad i \neq c\,, \tag{3.3}$$

so that the sum $\sum_{g=1}^{G} \phi_g = 1$. The resulting weighted MSE loss that is minimised during training can be written as:

$$\mathcal{L}(\theta) = \frac{1}{TH} \sum_{t=1}^{T} \sum_{h=1}^{H} \left\| \mathbf{\Phi}^{1/2} (\mathbf{y}_{t+h} - \hat{\mathbf{y}}_{t+h}) \right\|_2^2, \tag{3.4}$$

where

$$\mathbf{\Phi} = \mathrm{diag}(\phi_1, \phi_2, \ldots, \phi_G) \in \mathbb{R}^{G \times G}$$

is a diagonal matrix of spatial weights, and $\mathbf{y}_{t+h}, \hat{\mathbf{y}}_{t+h} \in \mathbb{R}^G$ are the vectors of true and predicted values over the G grid points at forecast step h.

Incorporating a weight for the central grid point ϕ_C allows me to explore this trade-off between global accuracy across all grid points, as in *MultiSonnet*, and focused accuracy on the central location. When $\phi_C = 1/G$, the loss reduces to that of the *MultiSonnet* baseline. When $\phi_C > 1/G$, the model is encouraged to improve predictions at $T850_{(C)}$, potentially at the cost of the peripheral targets.

Additionally, I investigate the effect of modifying the embedding dimension d on multivariate performance. In the original multivariable Sonnet's [44] weather experiments as well as in *MultiSonnet*, the parameter α, which controls the relative space the targets and exogenous take up in the embedding dimension d, is fixed at 0.5 and d at 64. Thus, in the single-output model, $d = 64$ is split evenly between a single target and the remaining exogenous features. When extending to nine targets, this compresses more information

into the same space and thus potentially limits the model's representational capacity.

In the univariate model, the single target is embedded into 32 dimensions, and the 45 exogenous features into the remaining 32 dimensions. For τ targets and χ exogenous variables, these embeddings are obtained by linearly projecting the target and exogenous variables via trainable weight matrices $\mathbf{W}_y \in \mathbb{R}^{\tau \times (d/2)}$ for the target y and $\mathbf{W}_x \in \mathbb{R}^{\chi \times (d/2)}$) for the exogenous x [44]. Now that there are nine targets, the target weight matrix $\mathbf{W}_y$ maps all nine targets into the same 32-dimensional space, giving each target approximately $32/9 \approx 3.55$ dimensions. This compression was one of my original hypotheses for the decay in performance of *MultiSonnet* compared to the univariate baseline, as it may restrict the model's ability to capture complex interactions between targets.

To address this, I train models with both $d = 64$ and $d = 128$. A dimensionality of 128 was chosen because training tended to be more stable and efficient when the dimensionality was a power of two. Increasing this embedding dimension to 128 enlarges the target embedding space to 64 dimensions, giving each target approximately $64/9 \approx 7.11$ dimensions. In comparison, the exogenous weight matrix $\mathbf{W}_x$ maps the remaining exogenous features into the other 64 dimensions. This allows me to assess whether incorporating a larger embedding can better capture the joint dynamics of multiple spatial locations.

3.2.4 Learnable Weights

While *Distance Weights* explicitly prioritise the central grid point, it treats the relative importance of the remaining targets as uniform due to their equal distance from the central point. However, other work has suggested that a static weight scheme like this may not be ideal [46], since not all neighbouring locations are equally informative for forecasting the central point. To better capture these spatial relationships, I introduce *Learnable Weights* in which the weights of the peripheral targets are learned jointly with the model parameters, while keeping the central target weight fixed at $\phi_C = 0.5$.

By allowing the model to allocate attention to different grid points throughout training, the learnable weights serve as a proxy for the underlying spatial structure of the data. Grid points that are more informative for the central target naturally receive higher weights that reflect their empirical correlation and contribution to forecasting accuracy.

Preliminary experiments showed that, without constraints, the model tended to collapse most weights to zero, including the central grid point. Essentially, this ignores the majority of tasks. This collapsing is consistent with Caruana's observations in multi-task learning, where weighting mechanisms can lead the model to ignore specific tasks [1]. To combat this,

each learnable weight is constrained between 0.03 and 0.2 to ensure each task contributes to the loss and to maintain a balanced influence across each model.

For a diagonal matrix of spatial weights $\mathbf{\Phi} \in \mathbb{R}^{G \times G}$ and the other model parameters θ, the optimisation can be written as:

$$\min_{\theta, \mathbf{\Phi} \in \mathcal{C}} \mathcal{L}(\theta, \mathbf{\Phi}) = \frac{1}{TH} \sum_{t=1}^{T} \sum_{h=1}^{H} \left\| \mathbf{\Phi}^{1/2} \left(\mathbf{y}_{t+h} - \hat{\mathbf{y}}_{t+h} \right) \right\|_2^2 , \tag{3.5}$$

where $\mathcal{C}$ is the constraint set that ensures all weights contribute to the loss:

$$\mathcal{C} = \left\{ \mathbf{\Phi} = \mathsf{diag}(\phi_1, \dots, \phi_G) \;\middle|\; \phi_i \in [0.03, 0.2],\ \phi_c = 0.5 \right\} . \tag{3.6}$$

In addition to the weighted loss function, the embedding strategy is modified to better suit the multivariate forecasting framework. The original multivariable model uses separate linear projections to embed the single target and remaining exogenous features separately. This allows for a balanced representation between the single autoregressive signal of the target and the exogenous features. Now that there are nine targets, I experiment with jointly embedding the targets and exogenous variables. When multiple targets are predicted simultaneously, separate embeddings could limit the model's ability to capture interactions amongst targets and between targets and exogenous variables. A joint embedding removes this separation and may therefore provide the model with greater flexibility to represent dependencies across all inputs.

Overall, in comparison to the previous model variants, *Learnable Weights* extends the multivariate framework by combining location-adaptive weighting with a a modified embedding strategy. This design encourages the model to focus on the central location while still incorporating information from the surrounding locations.

3.2.5 Training

Before training, all feature values in the training, validation, and testing sets are normalised using the mean μ_{train} and standard deviation σ_{train} of the training set, such that each normalised feature $\tilde{x}$ is given by:

$$\tilde{x} = \frac{x - \mu_{\text{train}}}{\sigma_{\text{train}}} . \tag{3.7}$$

This allows all features to be on a comparable scale, which can improve training stability

and convergence. For example, in the London dataset, the variable $Z500_{(C)}$ has a mean of $54,509.5$ with a standard deviation of 1573.6, while $U10_{(C)}$ has a mean of 1.43 with a standard deviation of 3.7. Without normalisation, features like $Z500_{(C)}$ would dominate the optimisation due to their larger magnitude.

Each model was trained using the Adam optimiser, which updates the learning rate internally for each parameter [4]. The base learning rate of Adam, η, which scales these updates, was treated as a hyperparameter and linearly increased over the first five epochs to avoid getting trapped in local minima too early. During this warm-up phase, the base learning rate at epoch i can be given by:

$$\eta_i = \eta \left(f_0 + \frac{f_1 - f_0}{N} \cdot i \right), \quad i = 0, 1, \ldots, N, \tag{3.8}$$

where η is the base learning rate, f_0 and f_1 are the start and end factors of the warm-up, and N is the total number of warm-up epochs. For all experiments, f_0 is set to $1/3$, f_1 to 1, and N to 5. After the warm-up period, the base learning rate remains fixed at $\eta \cdot f_1$.

Adam is then used to minimise the loss functions shown in Equations 3.2, 3.4, and 3.5 by seeing the data in batches of 64. Each model is trained for a maximum of 100 epochs. In order to mitigate the risk of overfitting, training stopped early when the validation loss did not improve by $\delta = 0.0001$ in five epochs. Generally, as the forecasting horizon increases, the number of epochs that it takes during training decreases. Additionally, to account for the randomness in weight initialisation and data loading, the seed is fixed at 42 for all experiments. This value is used in the results presented by Shu and Lampos [44] and is commonly used in other work; fixing the seed mitigates the risk of results being driven by chance.

3.2.6 Hyperparameter Tuning

For each city and horizon, the models were tuned to optimise three key hyperparameters: the base learning rate of Adam η, the number of atoms each feature is decomposed into K (see Section 2.3), and the weight of $T850_{(C)}$ in the loss function ϕ_C for the *Distance Weights* experiments. To select an optimal combination of these hyperparameters, I perform a grid search using the following possible parameter values shown in Table 3.1.

For each experiment, a model is trained with every possible combination of (η, K, ϕ_C) and then performance is evaluated on the validation set. Specifically, I use the mean absolute error (MAE) between the predicted and actual values at the final step of the

Hyperparameter	Search Space
Learning rate (η)	0.005, 0.001, 0.0005, 0.0001, 0.00005
Number of Atoms (K)	8, 16, 32
Centre Weight (ϕ_C)	$0.\overline{1}$, 0.15, 0.2

Table 3.1: Hyperparameter search space for Grid Search. The Learning Rate and Number of Atoms were tuned in all models, whereas Centre Weight was tuned for Distance-based weights experiments.

horizon for the $T850_{(C)}$ (Equation 3.9). The final-step MAE is used because it is the primary evaluation metric, as described in the next section. Additionally, because the primary evaluation target for comparison is the central grid point, I consider the best hyperparameter combination to be that which performs best on the central grid point's forecast. Essentially, the weighted losses discussed in the previous sections bias training towards $T850_{(C)}$ and validation reinforces that.

Initial experiments were tuned using a Tree-structured Parzen algorithm implemented via Optuna [13], where each of the parameters is treated as continuous. For a fixed number of trials, the algorithm models the probability of a hyperparameter configuration x given a loss value y, $p(x|y)$, as well as the distribution of y, $p(y)$. This is done by fitting separate Gaussian Mixture Models to configurations below and above a chosen loss quantile. Candidate hyperparameters are then selected to maximise the expected improvement using the ratio of these densities [2]. This approach was initially preferred because, unlike grid search, it adapts and learns as more trials are run due to its recursive nature. Additionally, Optuna's framework allows for unpromising trials to be pruned before training has completed, which can shorten tuning time. However, it was not used in subsequent experiments due to the computational cost of achieving convergence. Additionally, treating the hyperparameters as continuous often led to overfitting on the validation set; performance would decline on the test set while still improving on the validation set as the iterations increased. To mitigate these, all final experiments were tuned with grid search over a fixed grid of discrete parameter values in order to reduce the computational cost and risk of overfitting to the validation set.

3.2.7 Evaluation

After finding the best hyperparameter combination (η^*, K^*, ϕ_C^*) from grid search, each model is finally evaluated on the test set. Although each model produces a full trajectory

of future values for steps $1, \dots, H$, evaluation is only carried out on the final step of the horizon H. This is because at each step, the model accumulates more error and thus averaging over each step would underestimate the difficulty of predicting the given horizon since it would include the easier-to-predict intermediate steps. Even though the Koopman Operator (see Section 2.3) reduces this sequential error accumulation with its single pass [44], training still shows a consistent performance drop over the horizon; the MAE at step H consistently remains higher than the average MAE over all steps $1, \dots, H$ for all model variants.

Specifically, I employ two different metrics to quantify the difference between the actual values and predicted values of each model: the mean absolute error (MAE) and the linear correlation (r). Each of these is intended to give a different overview of each model's performance.

The mean absolute error (MAE) gives an overview of the raw error in the units of the target variable, which in all experiments is Kelvin (K). For interpretability, because Kelvin and Celsius only differ by a constant, all errors expressed in K are equivalent to those in $^\circ C$. Given a matrix of predictions at the final step of the horizon, $\hat{\mathbf{Y}} \in \mathbb{R}^{T \times G}$ and of actual values $\mathbf{Y} \in \mathbb{R}^{T \times G}$, the MAE at grid point g is defined as:

$$\mathrm{MAE}_g(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{T} \sum_{t=1}^{T} \left| \mathbf{Y}_{t,g} - \hat{\mathbf{Y}}_{t,g} \right| , \tag{3.9}$$

where T is the number of steps in the test set.

The linear correlation ($r \in [-1, 1]$) gives a measure of accuracy by measuring the Pearson correlation coefficient between the predicted and actual values. Compared to the MAE, it allows for a fairer comparison of model performance between each of the five regions' datasets. This is because regions like Singapore naturally experience less variation in temperatures than other regions. For example, the standard deviation of $\mathrm{T850}_{(\mathrm{C})}$ in Singapore is $0.89\,K$, compared to New York's $6.28\,K$. Thus, the MAE of Singapore is likely to be much less than that of New York. Linear correlation alleviates this issue by normalising and scaling each series. For a grid point g, it can be defined as follows:

$$r_g(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{\sum_{t=1}^{T} (\hat{\mathbf{Y}}_{t,g} - \bar{\hat{\mathbf{Y}}}_g)(\mathbf{Y}_{t,g} - \bar{\mathbf{Y}}_g)}{\sqrt{\sum_{t=1}^{T} (\hat{\mathbf{Y}}_{t,g} - \bar{\hat{\mathbf{Y}}}_g)^2} \sqrt{\sum_{t=1}^{T} (\mathbf{Y}_{t,g} - \bar{\mathbf{Y}}_g)^2}} , \tag{3.10}$$

where $\bar{\mathbf{Y}}_g$ and $\bar{\hat{\mathbf{Y}}}_g$ are the means of the actual and forecasted values at grid g.

3.2.8 Seasonal Persistence

All of the previous experiments have focused on the performance of the central grid point $T850_{(C)}$, however, each model also produces forecasts for the surrounding grid points. In order to compare how each model performs on these points, I compare their performance to each other as well as to a *Seasonal Persistence* model.

Seasonal persistence is a naive method that forecasts the last known seasonal value (daily) of each target variable [31]. Because T850 exhibits strong seasonal patterns across each of the regions and sub-locations, as illustrated for London in Figure 2.2, it is expected that *Seasonal Persistence* will be a competitive baseline, especially at shorter horizons. However, it only considers each target independently and does not leverage exogenous predictors.

Although simple, Naive models like this are commonly used as a forecasting benchmark across literature. In many cases, they also have been shown to outperform more complicated deep learning models [31, 40]. Overall, considering that the target at the central grid $T850_{(C)}$ drives the hyperparameter tuning and weight in the loss functions, *Seasonal Persistence* provides a valuable baseline for the surrounding points, which will show if the models are learning meaningful spatial patterns beyond the central target.

3.2.9 Summary

In this chapter, I described the multi-city weather datasets, the rolling-year training, validation, and testing splits, and the extension of Sonnet to multivariate forecasting. Three model variants, *MultiSonnet*, *Distance Weights*, and *Learnable Weights*, were designed to jointly forecast T850 across nine spatial locations, with a focus on performance at the central grid point. Forecast accuracy was evaluated across short, medium, and long horizons using mean absolute error and linear correlation, while peripheral forecasts are compared to a seasonal persistence baseline.

These experimental choices provide a foundation for the next chapter, where the performance of each model is compared and analysed to answer the three key research questions on multi-output forecasting effectiveness.

Chapter 4

Results and Discussion

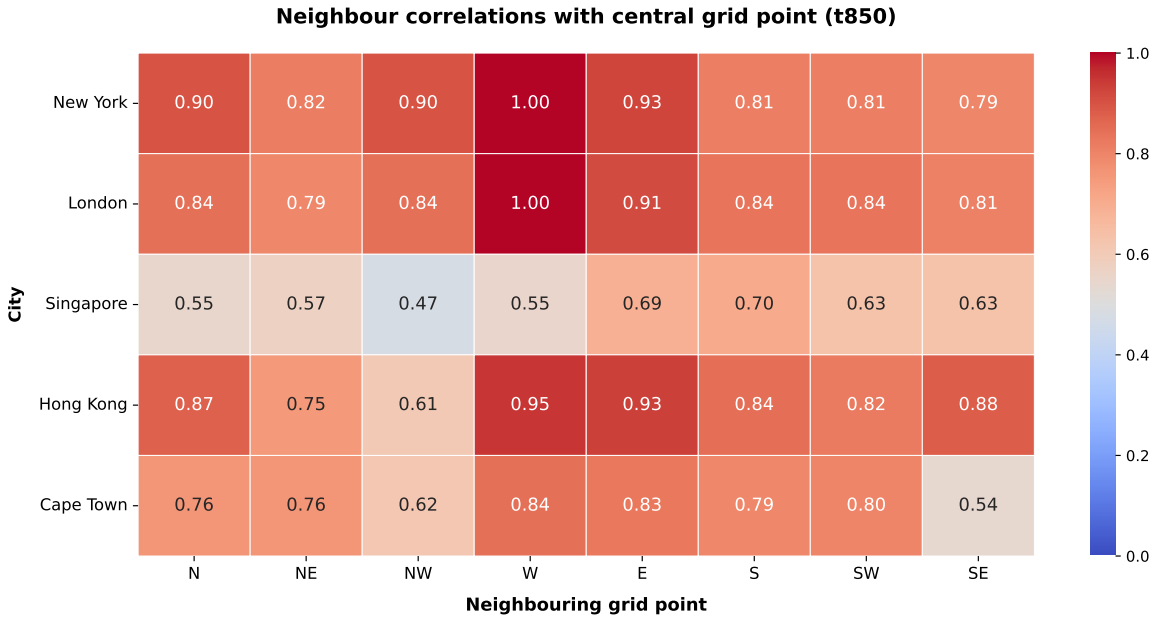
This chapter presents the results of the experiments discussed in Section 3.2, structured around answering the three key research questions defined in Chapter 1. First, I present a preliminary analysis of the correlations for each region's dataset, which motivates some of the intuition behind the experimental results. Next, I assess the feasibility of extending Sonnet to multivariate forecasting and how performance differs from the multivariable model (RQ1). Then, I examine the effectiveness of varying location-weighted losses to determine if the performance of the most informative target can be improved (RQ2). Finally, I conclude with an analysis of seasonal and region-specific patterns to investigate how spatial and temporal relations influence the effectiveness of multivariate forecasting (RQ3).

4.1 Preliminary Correlation Analysis

The experiments in Section 3.2 focus on forecasting temperature at 850 hPa (T850) across each region's local 3×3 neighbourhood of grid points. From the perspective of multi-task learning (MTL), each grid point can be viewed as a related forecasting task. Since MTL is most effective when related tasks share common structure (see Section 2.4), I first analyse the Pearson correlation coefficients between each region's central grid point and its surrounding eight sub-locations. Figure 4.1 presents a heat map of these correlations for each of the five regions.

There are apparent differences in the correlations across each of the regions. In London and New York, correlations are consistently high ($r > 0.79$) across all directions and approaching one in some cases, suggesting that the tasks are highly related. Because of

Figure 4.1: Pearson correlation coefficients between T850 at the central point and surrounding points across all time steps of each city.



the strong correlations, negative transfer appears less likely. However, the marginal benefit of forecasting all grid points may be minimal since the central grid point already captures the majority of the variation in the neighbourhood.

On the other hand, Hong Kong and Cape Town display a more diverse range of correlations, ranging from 0.45 to 0.95 for the former and 0.54 to 0.84 for the latter. In these areas, some neighbours move closely with the centre while others diverge substantially. This heterogeneity creates both opportunity and risk as the highly correlated neighbours may enable positive transfer, while weaker ones increase the potential for interference.

Singapore represents the most challenging case. Correlations range from 0.47 to 0.70, which means that no neighbour is strongly correlated with the centre. By Caruana's framework, such weak relationships increase the risk of negative transfer since the shared representations may be less useful across tasks [1]. However, from a global forecasting model perspective, this lack of strong correlation also implies that a single-output model would be particularly inadequate, as valuable information in the surrounding grid points would be missed by focusing solely on the centre.

These correlation patterns highlight why multi-output learning may behave differently across regions and motivate the analyses that follow.

4.2 Single-output Replication

In order to compare how the multivariate extension of Sonnet compares to the multivariable variant, I first attempted to replicate the results presented by Shu and Lampos [44]. To do so, I trained and tested the original univariate-output Sonnet model for $T850_{(C)}$ across each of the forecasting horizons. The replication results, shown in the first row of Table 4.1, are highly consistent with those presented by Shu and Lampos on the same task. Across each of the 15 city-horizon combinations, the average absolute difference in the reported linear correlations was 0.0019 (0.2526%) and from the reported MAEs was $0.0075\,K$ (0.5235%). The Pearson correlation coefficient between replicated and reported values for both MAE and linear correlation was 0.999, which confirms that this replication was successful. Minor deviations were expected due to different hardware and implementation. Thus, these results will serve as my comparison for *MultiSonnet*, *Distance Weights*, and *Learnable Weights*.

When replicating these results, I also tested changing the floating-point precision. The original Sonnet experiments used the highest floating-point precision (float32) for all models and training. Additional experiments were conducted to test whether decreasing the floating point precision to medium (bfloat16) during training and validation and then reverting to the highest precision for testing could improve performance. It was hypothesised that this approach might reduce the risk of overfitting and shorten training time. However, the results showed no significant differences in MAE across precisions ($p = 0.625$, Wilcoxon Test) or in linear correlations ($p = 1.0$, Wilcoxon Test). Additionally, there were no noticeable differences in training time. Therefore, all subsequent experiments and models used the highest precision for all training, validation, and testing.

4.3 Multivariate Performance (RQ1)

Now that I have an established multivariable baseline that is consistent with the original Sonnet results, I next evaluate how extending this model to jointly forecast nine targets under the same architectural framework affects performance. *MultiSonnet*, which considers each grid point uniformly, was trained and evaluated using the methods described in section 3.2.2. The average results across the three test seasons for $T850_{(C)}$ are presented in Table 4.1.

Relative to the univariate baseline, *MultiSonnet* underperforms at the central grid point of all cities, with the average MAE rising by approximately 2.7% and the linear correlation decreasing by about 1.5%. Additionally, across all cities except Singapore, the gap between

Table 4.1: Linear correlation (r) and mean absolute error (MAE) between the predicted and actual values for $T850_{(C)}$ for all cities and horizons H. Each value is the average across the three test seasons described in Section 3.1.1.

H	Model	London		Hong Kong		Cape Town		New York		Singapore	
		r	MAE	r	MAE	r	MAE	r	MAE	r	MAE
4	Replication	0.9098	1.7174	0.9651	0.6439	0.9101	1.6566	0.9661	1.3040	0.8373	0.3462
	MultiSonnet	0.9083	1.7419	0.9634	0.6703	0.9058	1.7495	0.9660	1.2898	0.8221	0.3690
	Distance Weights	0.9097	1.7346	0.9646	0.6523	0.9103	1.6806	0.9666	1.2948	0.8318	0.3514
	Learnable Weights	0.9122	1.7250	0.9671	0.6242	0.9164	1.6085	0.9661	1.2855	0.8374	0.3459
	Persistence	0.8035	2.5594	0.9119	0.9678	0.6439	3.3774	0.9234	1.9397	0.7430	0.4515
12	Replication	0.7400	2.9669	0.8415	1.2514	0.5465	3.5412	0.8743	2.4991	0.7488	0.4202
	MultiSonnet	0.7314	3.0011	0.8370	1.2705	0.5357	3.5651	0.8683	2.5446	0.7426	0.4274
	Distance Weights	0.7386	2.9402	0.8406	1.2734	0.5235	3.6087	0.8709	2.5259	0.7119	0.4530
	Learnable Weights	0.7288	2.9782	0.8412	1.2627	0.5419	3.5422	0.8717	2.4823	0.7506	0.4189
	Persistence	0.4819	4.4648	0.6048	2.0282	0.2311	5.1357	0.7291	3.7117	0.4862	0.6505
28	Replication	0.6772	3.2325	0.7954	1.4178	0.4728	3.7401	0.8535	2.6744	0.6948	0.4653
	MultiSonnet	0.6444	3.4510	0.7682	1.4851	0.4625	3.7868	0.8491	2.7175	0.6689	0.4785
	Distance Weights	0.6388	3.4905	0.7675	1.4735	0.4593	3.7414	0.8382	2.8487	0.6650	0.4833
	Learnable Weights	0.6833	3.2708	0.7851	1.4515	0.4462	3.8088	0.8469	2.7251	0.6998	0.4642
	Persistence	0.4698	4.5567	0.6019	2.1083	0.2272	5.2165	0.6893	4.0319	0.4007	0.6967

MultiSonnet and the replication substantially widens as the forecast horizon increases. Excluding Singapore, the average gap in MAE between Sonnet and *MultiSonnet* grows from $0.03\,K$ at $h = 4$ to $0.09\,K$ at $h = 28$, tripling in size.

Beyond its performance relative to replicated multivariable results, *MultiSonnet's* own accuracy also deteriorates significantly as the horizon increases. Excluding Singapore, at $h = 28$, *MultiSonnet's* MAE increases between $98\% - 122\%$ of its error at $h = 4$. This is illustrated for Cape Town in the top row of Figure 4.2. Clearly, *MultiSonnet's* ability to capture extremes diminishes as the forecast horizon increases. At $h = 4$, the model predicts extreme values and closely follows the trend of the actual values, but by $h = 28$, the forecasts appear smoothed and closely resemble a moving average. In other words, the model becomes less valuable as the horizon increases. However, even at $h = 28$, *MultiSonnet* still consistently outperforms *Seasonal Persistence* across all cities, reducing the MAE on average by $0.93\,K\,(28.5\%)$.

While the degradation is present amongst all cities, Singapore appears as an outlier to the magnitude of these changes, both relative to the multivariable replication and to increases in the forecasting horizon. The bottom plot of Figure 4.2 clearly illustrates how Singapore's relative performance across horizons does not decay to the extent of Cape Town and the other cities. In fact, its MAE only increases by 29% as we move from short

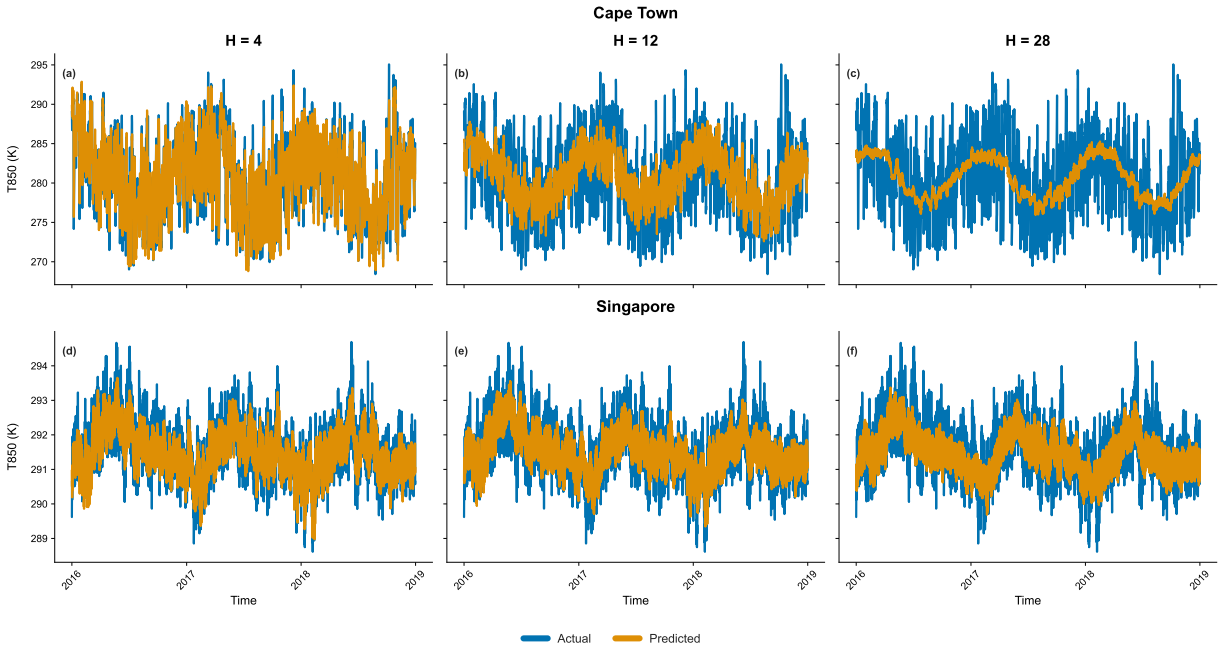


Figure 4.2: Plot of *MultiSonnet's* forecasts and actual values of $T850_{(C)}$ in Cape Town and Singapore across each of the test seasons and horizons.

to long-term forecasting.

One possible explanation lies in the correlation between the grid points (Figure 4.1), where Singapore's city centre is notably less correlated with surrounding targets than in the other four cities. I hypothesise that in spatially heterogeneous regions, like Singapore, the multivariate model can benefit from the diverse targets, which serve as complementary signals for Sonnet to learn broader atmospheric patterns that make it more robust to longer horizons. In contrast, in the other four cities, which have high spatial correlation, the additional outputs may have introduced redundant or noisy gradients and thus limited generalisation.

This suggests that when Sonnet must learn diverse patterns across grid points, as seen in Singapore's dataset, central-point performance naturally degrades. However, by jointly learning all locations, the model becomes more robust at longer horizons, effectively capturing broader trends even if extremes are smoothed. These results indicate that while the multivariate model effectively forecasts the surrounding grid points, it comes at a small but consistent cost to the performance of the central grid point, especially for specific cities and longer horizons.

To address the consistent degradation of central grid point accuracy, I next introduce

two loss-weighting schemes to bias training towards the central location.

4.4 Weighted Loss Comparison (RQ2)

Due to the apparent degradation in performance of *MultiSonnet* at the central grid point compared to the multivariable replication, I experimented with two variants of a weighted loss to place a greater weight on the central grid point ϕ_C during training. The *Distance Weights* strategy treats ϕ_C as a hyperparameter that is tuned with grid search, and the other eight weights are fixed proportional to their distance from the central point. The *Learnable Weights* strategy fixes $\phi_C = 0.5$ and then treats the other weights as learnable model parameters.

In addition, for the *Distance Weights* model, I explored increasing the embedding dimension from $d = 64$ to $d = 128$ for horizons $h = 4$ and $h = 12$ to evaluate whether a larger latent space would improve performance. The results showed only minimal changes: MAE decreased by $0.0049\,K$ (0.49%) and correlation increased by $0.0032\,(0.58\%)$. Horizon $h = 28$ was not evaluated due to the high computational cost. Given the negligible improvements relative to the increased training time, all reported *Distance Weights* model results use $d = 64$.

Relative to *MultiSonnet*, *Distance weights* improves the MAE and correlation in 53% of city-horizon combinations. Its gains in performance are most noticeable when $h = 4$, where it reduces the MAE on average by $0.0214\,K$ (2.3%). However, across all horizons, the average MAE increases by $0.0003\,K$, particularly due to a substantially worse performance in New York at $h = 28$.

Each *Distance Weights* model varies in the central point weight ϕ_C, which is tuned via a grid search over the set $\{0.\bar{1}, 0.15, 0.2\}$, where $0.\bar{1}$ corresponds to *MultiSonnet's* uniform weighting. From the grid search, which optimises the MAE at the central point, 80% of city-test year combinations selected $\phi_{\mathrm{C}} = 0.2$, the highest value in the search space, when $h = 4$. On the other hand, when $h = 28$, only 33.3% of combinations chose this highest value. This indicates that the model adjusts the contribution of surrounding points depending on the forecast horizon, implicitly reflecting the spatial structure of the data. At short horizons, the surrounding grid points contribute relatively little to forecasting the central point. As the horizon increases, surrounding points become more informative, potentially due to the model capturing broader atmospheric dynamics.

Overall, the *Distance Weights* model provides a simple mechanism to bias training

toward the central point and gives several short-term performance gains. Its effectiveness, however, depends on the forecast horizon. While higher central grid weights improve accuracy when $h = 4$, their benefit diminishes and can even harm performance at longer horizons. This suggests that fixed, distance-based weighting cannot fully capture how the relative importance of central versus surrounding points shifts over time.

In contrast, *Learnable Weights* consistently outperforms *MultiSonnet* in 12 out of 15 metrics, reducing the average MAE by $0.019\,K$ (2.7%) across all horizons and cities. Additionally, it outperforms the multivariable replication for short-term horizons in 80% of metrics, where it reduces the MAE on average by $0.016\,K$ (1.41%). Interestingly, Singapore once again stands out, where *Learnable Weights* outperforms both the multivariable replication and *MultiSonnet* across all metrics and horizons.

Unlike the distance-based approach, which depends on a fixed spatial weights, *Learnable Weights* adapts automatically to the horizon and city-specific structure in the data. This flexibility leads to more consistent improvements and reduces the need for manual tuning of ϕ_{C}. Overall, *Learnable Weights* emerges as the most effective variant for improving central point accuracy, particularly at shorter horizons and for modelling Singapore.

4.5 Spatial and Temporal Patterns (RQ3)

In order to understand the spatial relations in the data, I analyse the learned weight distribution from the *Learnable Weights* strategy as well as how the performance of the surrounding grid points $\{\mathrm{T850}_{(\mathrm{N})}, \mathrm{T850}_{(\mathrm{NW})}, \mathrm{T850}_{(\mathrm{NE})}, \mathrm{T850}_{(\mathrm{S})}, \mathrm{T850}_{(\mathrm{SW})}, \mathrm{T850}_{(\mathrm{SE})}, \mathrm{T850}_{(\mathrm{E})}, \mathrm{T850}_{(\mathrm{W})}\}$ changes relative to the central point and to different weightings. I then analyse temporal correlations in the data of the best model, *Learnable Weights*, to see how and where performance varies.

4.5.1 Spatial Patterns

Because the *Learnable Weights* model optimises central accuracy while allocating loss dynamically to the surrounding points, the resulting weights provide insight into which spatial dependencies the model finds most informative. In this sense, they act as an interpretable proxy for the spatial inductive biases that emerge during training. These are shown in Figure 4.3, which visualises the learned weight distributions taken over 45 iterations of grid search for each city.

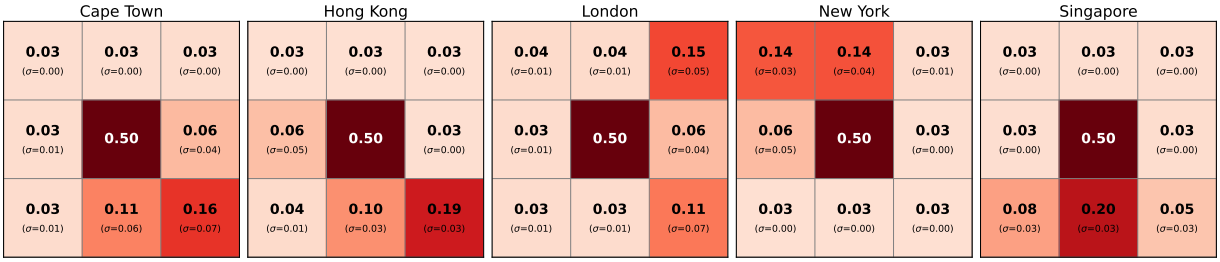


Figure 4.3: Learned Weight Distributions for each city. Each square represents a distinct grid area, analogous to that in Figure 3.1, where the surrounding weights are organised in the corresponding directions relative to the central point. Each distribution is taken from 45 iterations of hyperparameter tuning (15 iterations $\times$ 3 test years), where the bolded number represents the mean weight and σ the standard deviation.

Across all cities, three adjacent grid points consistently receive the largest share of the weight, together accounting for an average of 66.8% of the total weight assigned to the surrounding points, though their specific locations vary. In Singapore, where *Learnable Weights* performed the best, the learned weights align with the empirical correlation patterns shown in Figure 4.1. The highest weight is on its southern grid $\phi_S = 0.20$, which is also the location that has the highest correlation with the centre $r = 0.7$. Additionally, compared to the other cities, Singapore's volatility in the learned weight distribution is less, with an average standard deviation of 0.011, whereas the other cities range from 0.015 to 0.026.

Interestingly, in London and Cape Town, where *Learnable weights* underperformed relative to Singapore, the model puts the most weight on the most weakly correlated grids. For example, Cape Town's highest weight is on its southeastern grid ($\phi_{SE} = 0.16$), which is also its most weakly correlated ($r = 0.54$). For the other cities, the learned weights are not consistent with either the most correlated or the least correlated, but in the middle with higher volatility. Overall, this is suggesting that the model is capturing broader atmospheric dynamics beyond pairwise correlation. The higher volatility could potentially be due to the change in horizon, where the most informative neighbours could change over time.

Apart from the learned weight distributions, which highlight how the peripheral targets aid in forecasting the centre, each model also forecasts T850 at the surrounding locations. Table 4.2 shows the average performance for each model across the eight surrounding sub-locations $T850_{(g)}$ for all g in the set of directions {N, NW, NE, S, SW, SE, W, E}. Across all cities and horizons, each of the model variants, even those that bias the central point,

Table 4.2: Average mean absolute error (MAE) and Pearson Correlation coefficent (r) between the predicted and actual values of the surrounding grid points. Each value is the average over eight grid points (N, NW, NE, S, SW, SE, W, E) and three test years (2016, 2017, 2018). The best model for each city and horizon is shaded in grey.

H	Model	London		Hong Kong		Cape Town		New York		Singapore	
		r	MAE	r	MAE	r	MAE	r	MAE	r	MAE
4	Persistence	0.8333	2.4934	0.8955	1.0528	0.6186	3.5633	0.8865	2.3372	0.6921	0.4450
	MultiSonnet	0.9205	1.7056	0.9341	0.8864	0.8788	1.9469	0.9468	1.5997	0.7571	0.3730
	Distance Weights	0.9210	1.7012	0.9347	0.8738	0.8755	1.9753	0.9467	1.6151	0.7574	0.3726
	Learnable Weights	0.9175	1.7382	0.9289	0.8745	0.8685	2.0509	0.9413	1.6726	0.7523	0.3758
12	Persistence	0.6909	4.2399	0.5861	2.2076	0.2758	5.1033	0.6638	4.2455	0.3974	0.6384
	MultiSonnet	0.7627	2.9569	0.8204	1.3850	0.5455	3.5773	0.8270	2.9057	0.6471	0.4389
	Distance Weights	0.7653	2.9325	0.8134	1.4152	0.5353	3.6188	0.8276	2.8951	0.6409	0.4432
	Learnable Weights	0.7623	2.9422	0.8210	1.3978	0.5425	3.5825	0.8276	2.9267	0.6424	0.4447
28	Persistence	0.5014	4.5700	0.5692	2.2523	0.2539	5.2101	0.6401	4.4194	0.3404	0.6703
	MultiSonnet	0.6796	3.3975	0.7440	1.6262	0.5051	3.7415	0.8050	3.1442	0.5776	0.4807
	Distance Weights	0.6726	3.4514	0.7483	1.6011	0.5015	3.7358	0.7942	3.2267	0.5757	0.4825
	Learnable Weights	0.7059	3.2777	0.7621	1.5766	0.4941	3.7533	0.7997	3.1820	0.5842	0.4788

considerably outperform *Seasonal Persistence*. *MultiSonnet* performs the best relative to *Seasonal Persistence*, decreasing the MAE by $0.886\,K$ (29.3%) on average.

Contrary to the results presented in Table 4.1 for performance of the central grid, *MultiSonnet* and *Distance Weights* have considerably better performance in comparison to *Learnable Weights*, particularly at $h = 4$ and $h = 12$. Because *Learnable Weights* puts more weight on the central point $\phi_C = 0.5$, it is expected that the forecasts of the surrounding coordinates drop in accuracy, while the central point improves. However, it is interesting to see how, at $h = 28$, *Learnable Weights* achieves the best performance across three of the cities by significant margins. This suggests that, in some cases, the central location becomes increasingly informative for forecasting the surrounding grid as the forecast horizon increases.

This is further confirmed by analysing the prediction errors across the hyperparameter tuning for each *Learnable Weight* model. Across all iterations, the correlation between the MAE at the central point and the average MAE of the surrounding points is 0.9865. This signifies that as the model struggles with the central point, it almost always struggles with the neighbourhood, and when the central point improves, so does the surrounding grid.

Overall, these results indicate that improving the forecast of the central point can also benefit the surrounding locations, particularly at longer horizons. This reflects the fact that central and peripheral points are closely aligned, both spatially and through atmospheric

dynamics, so that enhancing the central prediction under a weighted-loss strategy can improve forecasts across the broader grid.

4.5.2 Temporal Patterns

To better understand how performance varies across cities and horizons, I next analyse temporal patterns in the forecast errors of the best model variant, *Learnable Weights*, across seasons and times of day.

Across all cities, forecasting errors exhibit clear annual seasonal patterns. In London and New York, errors peak in Winter, with average absolute errors roughly 31% and 21% higher than in Summer, respectively. Hong Kong shows the strongest seasonal variability, with winter errors nearly double those in Summer and substantially higher than Autumn. In contrast, Singapore and Cape Town's errors over time appear visually as white noise and show minimal annual variation as differences are below 10% between seasons.

Because Hong Kong demonstrates the largest seasonal contrasts, I examine model-specific differences there. At $h = 4$, *Learnable Weights* reduces Winter errors by 4.6% compared to *Distance Weights*, while Summer errors improve by 11.7%. At $h = 12$, however, Winter errors improve by 1.6%, and Summer errors increase slightly by 2.2%. This suggests that *Learnable Weights* can help to mitigate extreme seasonal conditions at short horizons, but diminish as the horizon increases.

The time of day each feature is recorded has little impact on the forecast errors. Most cities exhibit nearly uniform errors across hours (00:00, 06:00, 12:00, 18:00). Singapore shows a slight midday peak with errors about 10% higher at noon than at 00:00. At the same time, Hong Kong displays very minor hourly fluctuations relative to its larger seasonal swings.

Overall, annual seasonality is the primary driver of forecast error variation in New York, London, and Hong Kong, whereas Singapore and Cape Town maintain relatively stable accuracy through each year. These temporal error patterns complement the spatial analysis in the previous section by showing that model weaknesses are not only dependent on the degree of correlation between the grid points, but also on the seasonal variability within each region.

4.6 Summary

Overall, this chapter has shown that when extending Sonnet to multivariate forecasting, performance naturally degrades at the central target of each city. However, incorporating a weighted loss alleviates this performance degradation at short horizons and across all horizons for Singapore. Because of the differences in correlation between the centre and the surrounding grid points of Singapore, it is hypothesised that this spatial heterogeneity is what led to its superior performance when expanded to forecast all grid points jointly. Additionally, performance on the surrounding grids also performs well across cities and horizons compared to the naive *Seasonal Persistence* model, even when trained and tuned by biasing the central point.

Chapter 5

Conclusion

5.1 Summary of Findings

This dissertation investigated extending Sonnet [44], a state-of-the-art time series forecasting model, from its original multivariable, single-output design to a multivariate, multi-output framework for joint spatial forecasting. While Shu and Lampos's [44] experiments treated surrounding locations as exogenous inputs to predict a single central target (T850), this research investigated the trade-off involved in forecasting T850 simultaneously across a 3×3 spatial grid. Motivated by literature from multi-task learning (MTL) and global forecasting models (GFM), experiments were conducted on five distinct regional datasets to evaluate the impact on central-point accuracy and overall spatial performance.

The core finding is that the naive, multivariate extension, *MultiSonnet*, consistently degraded performance at the central grid point of all regions and horizons. However, the model gains the benefit of producing multiple forecasts simultaneously. This is evidenced by the fact that all multivariate model variants, including *MultiSonnet*, significantly outperformed *Seasonal Persistence* for the peripheral targets. Furthermore, compared to other model variants, *MultiSonnet* itself achieves the best performance on these peripheral points at short and medium horizons. However, the results confirm that distributing a model's focus across multiple forecasts inherently compromises its performance on the original, single-target objective, even though more locations are predicted.

To address the central grid performance loss, the *Learnable Weights* model was introduced, which uses a weighted loss function to bias training towards the central target, while treating the peripheral as learnable model parameters. This model successfully recovered central point performance, reducing the mean absolute error on average by 2.6% relative

to *MultiSonnet*, and even outperforming the original single-output Sonnet replication at short horizons. However, this central bias came at a cost to peripheral targets at short and medium horizons.

A surprising result was found when forecasting one week ahead, the largest of the tested horizons. The *Learnable Weights* model, even with its bias towards the central point, outperformed *MultiSonnet* on the average MAE of the peripheral targets in three of the five regions, with an average reduction of $0.0571\,K$. This suggests that over longer horizons in certain regions, the central grid point may be the most spatially informative location; thus, a model that is trained to prioritise it can consequently generate better forecasts for the entire region.

Analysis of the different regions and errors provided key insights into why performance degraded in some areas and improved in others. Most notably, Singapore's superior performance is evident relative to the other cities, for each model variant, and in comparison to the multivariable single-output model for *Learnable Weights*. The main hypothesis, supported by an analysis of the learned spatial weights, is that Singapore benefited from its heterogeneity and lower correlation between grid points. In contrast, the high homogeneity in the other regions introduced redundant information that potentially hindered learning compared to a solely multivariable model. This aligns with GFM literature, suggesting that the diversity in training data improves model generalisation.

Overall, this dissertation has demonstrated that Sonnet's architecture is highly adaptable for multivariate forecasting. Additionally, the *Learnable Weights* model successfully balances forecasts, and its trained weights reveal which surrounding locations the model finds useful for its predictions.

In conclusion, this dissertation provides a clear foundation for spatio-temporal forecasting with deep learning models. It shows that while a direct multivariate extension can harm performance on a primary task, strategic loss weighting can effectively recover and even enhance it.

5.2 Limitations

Throughout this project, I experienced several challenges.

A significant limitation was the time required to train and tune the models, particularly due to the limited access to GPU resources. Each experiment consisted of 45 final models (covering three horizons, five cities, and three test years). For each of these models, the

hyperparameters had to be tuned. For *MultiSonnet* and *Learnable Weights*, tuning involved a grid search of 15 configurations per model, whereas for *Distance Weights* experiments, the grid search involved 45 configurations per model. Each grid search took between 2 and 18 hours, depending on the number of other people using the same GPU. Consequently, the hyperparameter grid searches had to be constrained to a smaller parameter space, which may have prevented the identification of a more optimal set of hyperparameters for each model.

Furthermore, due to time constraints, the evaluation was primarily focused on the central grid point of each city as opposed to each of the nine grid points. This is due to the time it would have taken to train and tune individual models for each grid point.

5.3 Future Work

This research focused exclusively on point forecasts, producing a single best estimate for each target variable. However, weather forecasting is inherently probabilistic. Future work could explore integrating uncertainty quantification, such as Bayesian methods or Conformal Prediction, into the Sonnet framework. This could allow for a deeper analysis of how predictive uncertainty evolves with the forecasting horizon and is influenced by the different spatial weighting strategies.

Additionally, the sub-locations used for feature extraction in the five datasets were selected on a uniform 5.625° resolution grid, downsampled from the original ERA5 [20] database. It would be interesting to investigate how forecasting accuracy changes when including higher-resolution surrounding locations, as well as how incorporating prior knowledge beyond simple geographic proximity could improve the selection of surrounding points.

From a modelling perspective, a key next step would be to test more advanced multi-task learning techniques. For example, future work could explore dynamic weighting schemes like GradNorm [9], which automatically adjust the weights for each task's individual loss function during training based on how quickly each task is learning. This is a more advanced approach compared to the method presented in this dissertation, which utilises a single weighted loss for all targets.

Another interesting extension could be to apply this multi-output framework to a different meteorological variable, such as precipitation, to assess how the results hold up for a more sporadic feature.

Finally, given the performance observed in spatially heterogeneous regions like Singa-

pore, a critical future direction could be to test this framework on a much larger and more diverse set of global locations. This would further investigate the hypothesis that spatial heterogeneity is a key driver of multi-output performance. Training a single model on data from dozens of regions with varying correlation structures would truly test Sonnet's capacity as a GFM and clarify the conditions under which multi-output forecasting provides the greatest benefit.

Bibliography

[1] Rich Caruana. "Multitask Learning". In: *Machine Learning* 28.1 (1997), pp. 41–75. DOI: 10.1023/A:1007379606734.

[2] James Bergstra et al. "Algorithms for hyper-parameter optimization". In: *Advances in neural information processing systems* 24 (2011).

[3] Wenhao Huang et al. "Adaptive weight optimization for classification of imbalanced data". In: *International Conference on Intelligent Science and Big Data Engineering*. Springer. 2013, pp. 546–553.

[4] Diederik P Kingma. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[5] Liang Zhao et al. "Multi-task learning for spatio-temporal event forecasting". In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 1503–1512.

[6] Iasonas Kokkinos. "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[7] Sebastian Ruder. "An overview of multi-task learning in deep neural networks". In: *arXiv preprint arXiv:1706.05098* (2017).

[8] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[9] Zhao Chen et al. "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks". In: *International conference on machine learning*. PMLR. 2018, pp. 794–803.

[10] Alex Kendall, Yarin Gal, and Roberto Cipolla. "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7482–7491.

[11] Sima Siami-Namini and Akbar Siami Namin. "Forecasting economics and financial time series: ARIMA vs. LSTM". In: *arXiv preprint arXiv:1803.06386* (2018).

[12] Costas A Varotsos, Arthur P Cracknell, and Maria N Efstathiou. "The global signature of the el Niño/La Niña southern oscillation". In: *International Journal of Remote Sensing* 39.18 (2018), pp. 5965–5977.

[13] Takuya Akiba et al. "Optuna: A next-generation hyperparameter optimization framework". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 2623–2631.

[14] Vitor Cerqueira, Luis Torgo, and Carlos Soares. "Machine learning vs statistical methods for time series forecasting: Size matters". In: *arXiv preprint arXiv:1909.13316* (2019).

[15] Ting Gong et al. "A comparison of loss weighting strategies for multi task learning in deep neural networks". In: *IEEE Access* 7 (2019), pp. 141627–141632.

[16] Shikun Liu, Edward Johns, and Andrew J Davison. "End-to-end multi-task learning with attention". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 1871–1880.

[17] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. "Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting". In: *Advances in neural information processing systems* 32 (2019).

[18] Zekai Chen et al. "Multi-task time series forecasting with shared attention". In: *2020 international conference on data mining workshops (ICDMW)*. IEEE. 2020, pp. 917–925.

[19] Eranga De Saa and Lochandaka Ranathunga. "Comparison between arima and deep learning models for temperature forecasting". In: *arXiv preprint arXiv:2011.04452* (2020).

[20] Hans Hersbach et al. "The ERA5 global reanalysis". In: *Quarterly journal of the royal meteorological society* 146.730 (2020), pp. 1999–2049.

[21] Isabelle Leang et al. “Dynamic task weighting methods for multi-task networks in autonomous driving systems”. In: *2020 IEEE 23rd International conference on intelligent transportation systems (ITSC)*. IEEE. 2020, pp. 1–8.

[22] Stephan Rasp et al. “WeatherBench: a benchmark data set for data-driven weather forecasting”. In: *Journal of Advances in Modeling Earth Systems* 12.11 (2020), e2020MS002203.

[23] Ashis Kumar Mandal et al. “Comparative study of univariate and multivariate long short-term memory for very short-term forecasting of global horizontal irradiance”. In: *Symmetry* 13.8 (2021), p. 1544.

[24] Pablo Montero-Manso and Rob J Hyndman. “Principles and algorithms for forecasting groups of time series: Locality and globality”. In: *International Journal of Forecasting* 37.4 (2021), pp. 1632–1653.

[25] Martin G Schultz et al. “Can deep learning beat numerical weather prediction?” In: *Philosophical Transactions of the Royal Society A* 379.2194 (2021), p. 20200097.

[26] Haoyi Zhou et al. “Informer: Beyond efficient transformer for long sequence time-series forecasting”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 12. 2021, pp. 11106–11115.

[27] Amedeo Buonanno et al. “Global vs. local models for short-term electricity demand prediction in a residential/lodging scenario”. In: *Energies* 15.6 (2022), p. 2037.

[28] Tulsi Pawan Fowdur, Rosun Mohammad Nassir-Ud-Diin Ibn, et al. “A real-time collaborative machine learning based weather forecasting system with multiple predictor locations”. In: *Array* 14 (2022), p. 100153.

[29] Masooma Ali Raza Suleman and S Shridevi. “Short-term weather forecasting using spatial feature attention based LSTM model”. In: *IEEE Access* 10 (2022), pp. 82456–82468.

[30] Jerald A Brotzge et al. “Challenges and opportunities in numerical weather prediction”. In: *Bulletin of the American Meteorological Society* 104.3 (2023), E698–E705.

[31] Antônio Augusto Rodrigues de Camargo and Mauri Aparecido de Oliveira. “Analysis of the application of different forecasting methods for time series in the context of the aeronautical industry”. In: *Engineering Proceedings* 39.1 (2023), p. 74.

[32] Vaia I Kontopoulou et al. “A review of ARIMA vs. machine learning approaches for time series forecasting in data driven networks”. In: *Future Internet* 15.8 (2023), p. 255.

[33] Safwan Mahmood Al-Selwi et al. “LSTM inefficiency in long-term dependencies regression problems”. In: *Journal of Advanced Research in Applied Sciences and Engineering Technology* 30.3 (2023), pp. 16–31.

[34] S Thundiyil et al. “Transformers for modeling long-term dependencies in time series data: a review”. In: *2023 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*. IEEE. 2023, pp. 1–5.

[35] Azlan Abdul Aziz et al. “Repeated time-series cross-validation: A new method to improved COVID-19 forecast accuracy in Malaysia”. In: *MethodsX* 13 (2024), p. 103013.

[36] Shijie Chen, Yu Zhang, and Qiang Yang. “Multi-task learning in natural language processing: An overview”. In: *ACM Computing Surveys* 56.12 (2024), pp. 1–32.

[37] Matthias Groß and Lukas Hans. “Leveraging potentials of local and global models for water demand forecasting”. In: *Engineering Proceedings* 69.1 (2024), p. 129.

[38] Yuxuan Shu and Vasileios Lampos. “DeformTime: capturing variable dependencies with deformable attention for time series forecasting”. In: *arXiv preprint arXiv:2406.07438* (2024).

[39] Yuxuan Wang et al. “Timexer: Empowering transformers for time series forecasting with exogenous variables”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 469–498.

[40] Nico Beck, Jonas Dovern, and Stefanie Vogl. “Mind the naive forecast! a rigorous evaluation of forecasting models for time series with low predictability: N. Beck et al.” In: *Applied Intelligence* 55.6 (2025), p. 395.

[41] Mohammad Liton Hossain, SM Shams, and Saeed Mahmud Ullah. “Time-series and deep learning approaches for renewable energy forecasting in Dhaka: a comparative study of ARIMA, SARIMA, and LSTM models”. In: *Discover Sustainability* 6.1 (2025), pp. 1–25.

[42] Mei Huang and Qian Ai. “Multi-task learning load time series situational prediction based on gated recurrent neural networks considering spatial correlations”. In: *Frontiers in Energy Research* 12 (2025), p. 1476613.

[43] Habib Irani et al. “Time Series Embedding Methods for Classification Tasks: A Review”. In: *arXiv preprint arXiv:2501.13392* (2025).

[44] Yuxuan Shu and Vasileios Lampos. “Sonnet: Spectral Operator Neural Network for Multivariable Time Series Forecasting”. In: *arXiv preprint arXiv:2505.15312* (2025).

[45] Shakir Showkat Sofi and Ivan Oseledets. “A case study of spatiotemporal forecasting techniques for weather forecasting”. In: *GeoInformatica* 29.2 (2025), pp. 275–298.

[46] Han Zhang et al. “Quantum-Enhanced Multi-Task Learning with Learnable Weighting for Pharmacokinetic and Toxicity Prediction”. In: *arXiv preprint arXiv:2509.04601* (2025).

Appendix A

Appendix

A.1 Code

All of the code of this dissertation was done in Python and extensively using PyTorch for modelling. All of the code can be found at: https://github.com/tbluth4/Dissertation_2

A.2 AI Usage Statement

Throughout this dissertation, ChatGPT was used as a supplementary tool. Its use was limited to refining wording and phrasing for clarity, assisting with code debugging during experiments, enhancing the visual quality of specific figures, and providing guidance on LaTeX syntax, including figure placement, table organisation, and equation formatting.