FORECASTING ATMOSPHERIC TURBULENCE CONDITIONS FROM PRIOR ENVIRONMENTAL PARAMETERS WITH ARTIFICIAL NEURAL NETWORKS: AN ENSEMBLE STUDY

Thesis

Submitted to

The School of Engineering of the

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree of

Master of Science in Electro-Optics and Photonics

By

Mitchell Gene Grose

Dayton, Ohio

May, 2021

University of Dayton

FORECASTING ATMOSPHERIC TURBULENCE CONDITIONS FROM PRIOR ENVIRONMENTAL PARAMETERS WITH ARTIFICIAL NEURAL NETWORKS: AN ENSEMBLE STUDY

Name: Grose, Mitchell Gene

APPROVED BY:

Edward Watson, Ph.D.
Advisory Committee Chairman
Graduate Professor, Electrical and Computer Engineering

Vijayan K. Asari, Ph.D.
Committee Member
Professor, Electrical and Computer Engineering

Yakov Diskin, Ph.D.
Committee Member
Adjunct Professor, Electrical and Computer Engineering

Robert J. Wilkens, Ph.D., P.E.
Associate Dean for Research and Innovation
Professor
School of Engineering

Eddy M. Rojas, Ph.D., M.A., P.E.
Dean, School of Engineering

ABSTRACT

FORECASTING ATMOSPHERIC TURBULENCE CONDITIONS FROM PRIOR ENVIRONMENTAL PARAMETERS WITH ARTIFICIAL NEURAL NETWORKS: AN ENSEMBLE STUDY

Name: Grose, Mitchell Gene
University of Dayton

Advisor: Dr. Edward Watson

This is where you should start your abstract.

For name of person(s) to whom you are dedicating your thesis

ACKNOWLEDGMENTS

Write your acknowledgements here.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

Background and Problem Statement

1.1 Turbulence (C_n^2) Background

1.1.1 What is Turbulence (C_n^2)?

Optical (atmospheric) turbulence is caused by naturally occurring small variations in air temperature (< 1°C) which cause random variations in wind velocity (eddies) which we view as turbulent motion in the atmosphere. The temperature differences cause small differences in atmospheric density and therefore in refractive index. These index changes are moved around by the random variations in wind eddies and can accumulate to cause significant inhomogeneities in the index profile of the atmosphere which can cause the wavefront of a beam to significantly change in propagation. The effects are high spatial frequency beam spreading, low spatial frequency beam wander, and intensity variations (scintillation) [1]. Atmospheric turbulence strength is characterized by the refractive-index structure constant, C_n^2 ($m^{-2/3}$). Turbulence strength is highly variable with season, time of day, geographical location, altitude, and local weather patterns. Generally, C_n^2 varies in value from 1×10^{-12} (very strong) to 1×10^{-17} (very weak).

Atmospheric turbulence can have many adverse affects on optics in imaging and laser systems. All imaging through a non-negligible distance of the Earth's atmosphere is impacted by atmospheric turbulence, leading to a blurring effect in an image. An example of this is shown in Figure 1.1. Both images in Figure 1.1 are of the same target on the same day but during low-strength turbulence (left) and high-strength turbulence (right) conditions. There are many developed and developing technologies which are highly impacted by the impact of atmospheric turbulence.

(a) Low-strength (C_n^2)   (b) High-strength (C_n^2)

Figure 1.1: Effect of low-strength and high-strength C_n^2 conditions.

In imaging applications, the image degradation in Figure 1.1b due to strong turbulence can significantly impact feature extraction and target recognition. The twinkling of celestial objects in the night sky is due to atmospheric turbulence. Observatories (astronomy) are frequently built at high altitude geographical locations in part to avoid light pollution, but also to reduce amount of turbulence degradation by shortening the distance the incoming light must propagate through the atmosphere. In laser propagation applications, the degradation of the wavefront can significantly reduce the power of a beam on the target from a high energy laser (HEL) weapon system, or disrupt the transfer of signal in optical communication.

Atmospheric turbulence is important to measure and model because the atmospheric turbulence impact can be compensated with adaptive optics (AO). The strength of the atmospheric turbulence is proportional to the amount of adaptive optics compensation needed to correct for the aberrations. Using astronomy as an example, the basic components of an adaptive optics system consists of a wavefront sampler, a wavefront sensor (WFS), a corrector such as a deformable mirror, and a control computer to perform real-time numerical calculations. Light from an astronomical object is captured by the optical system consisting of a telescope and imager (camera). Part of the light is sampled by the WFS and this data is sent to the computer to calculate the necessary corrections in the deformable mirror to sharpen the image [1]. A similar system exists for a laser propagation setup. Another application of the measurement of turbulence an optical system which requires parameter tuning at a specific strength of C_n^2 for further experimentation or use. Finally, the use of a laser weapon system might be best for turbulence conditions weaker than a specific strength so knowledge of the current conditions can decide whether to use the laser system or traditional weapons.

1.1.2 State of Turbulence (C_n^2) Modeling

There is no theoretical (physical) model for turbulence which is accurate for the many cases, but many have been developed for specific cases. One such model is the Hufnagel-Valley boundary model calculates C_n^2 as a function of altitude [2]. Another is the submarine laser communications (SLC)-Night model which associates C_n^2 strengths with specific altitude ranges [1].

Another approach to modeling turbulence (C_n^2), specifically at the surface level, is by the use of machine learning. This approach acts as a surrogate model of the relationship between the local environmental (weather) and C_n^2 measurements. Typical weather measurements are air temperature, pressure, relative humidity or dew point, wind speed and direction, solar irradiance, and time-correlated atmospheric turbulence (C_n^2) typically with a scintillometer [3, 4, 5]. The machine learning approach primary employs the artificial neural network (ANN) in a multi-layer perceptron (MLP) optimized by a gradient descent algorithm [4, 5]. **TALK ABOUT MORE LITERATURE?**

This type of approach associates a set of environmental (weather) measurements with a time-correlated C_n^2 measurement and is most useful as a surrogate model to predict C_n^2 for comparison with measurements. This technique illustrates the capability of a model, but from a utility perspective is only useful if current C_n^2 measurements are desired but a sensor is not deployed. A more useful model *forecasts* future C_n^2 instead of making time-correlated predictions. One approach to this is to train a regression model on time-correlated weather and C_n^2 measurements, then apply it to weather forecasts from a numerical weather prediction (NWP) data source to yield a C_n^2 forecast.

The work in this thesis develops a novel machine learning approach to forecast future daytime C_n^2 conditions from *prior* environmental (weather) and C_n^2 measurements. Deep learning is used to create a low-altitude model capable of forecasting 4 hours of future daytime C_n^2 conditions with estimates every 30 minutes using no more than 16 hours of prior environmental measurements. This technique is ideal for any application that uses future C_n^2 predictions and bypasses the need to download and format NWP data for model inference.

1.2 Thesis Overview

This section provides a brief chapter-by-chapter overview of the work in this thesis.

1.2.1 Machine Learning Review (Ch. 2)

Chapter II steps through a review of machine learning modeling. First, the basics of model training and testing is summarized. Next, the fundamental machine learning architectures used throughout this work are described in detail. Finally, other vital components to machine learning modeling such as optimization are reviewed.

1.2.2 Data Processing (Ch. 3)

Chapter III first summarizes the weather and C_n^2 data sources and their spatial relationship to each other. Chapter III then details the data preprocessing techniques including confidence filtering, window averaging, and interpolating. Finally, the methodology is described to format the data into model input/output pairs and parse into train, validation, and test datasets.

1.2.3 Grid Search (Ch. 4)

Chapter IV details the technique used to find the best architecture and parameters to best forecast C_n^2: the grid search. A statistical test, the *Student's t-test*, is described then employed as a significance test of the grid search results. The top performing models in the grid search are then compared and a single model is selected.

1.2.4 Test Dataset Model Evaluation (Ch. 5)

Chapter V applies the selected model to the test dataset and illustrates performance. This Chapter then details an analysis on the overall model performance to illustrate general model capabilities, and then analyzes three specific forecasts to understand why the model performs well and poorly in specific scenarios.

1.2.5 Summary and Future Work (Ch. 6)

Chapter VI summarizes this thesis and details items for future work encountered throughout this effort.

CHAPTER II

Machine Learning Modeling Review

PyTorch reference [6]. Colah's blog reference [7].

2.1 Fundamentals of Training and Testing

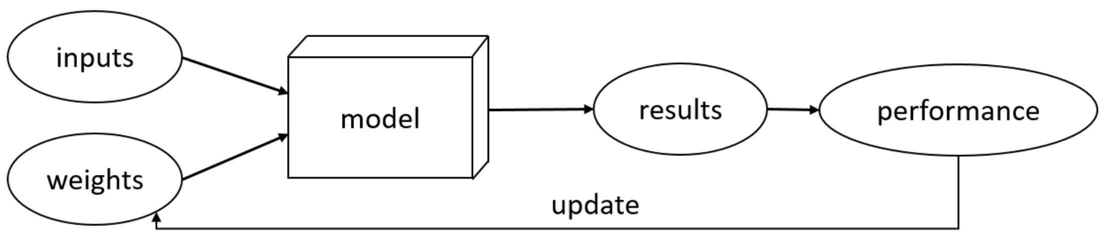


Figure 2.1: Simple example of machine learning model training .

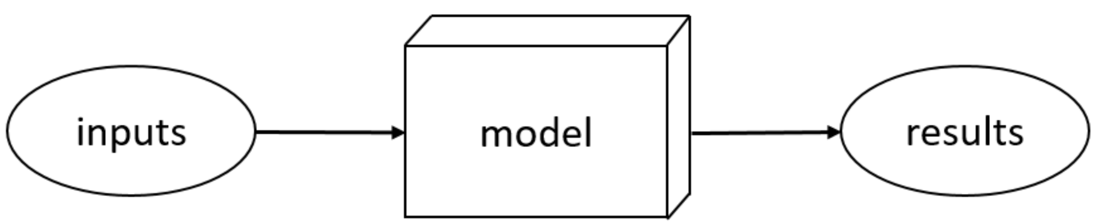


Figure 2.2: Simple example of machine learning model testing.

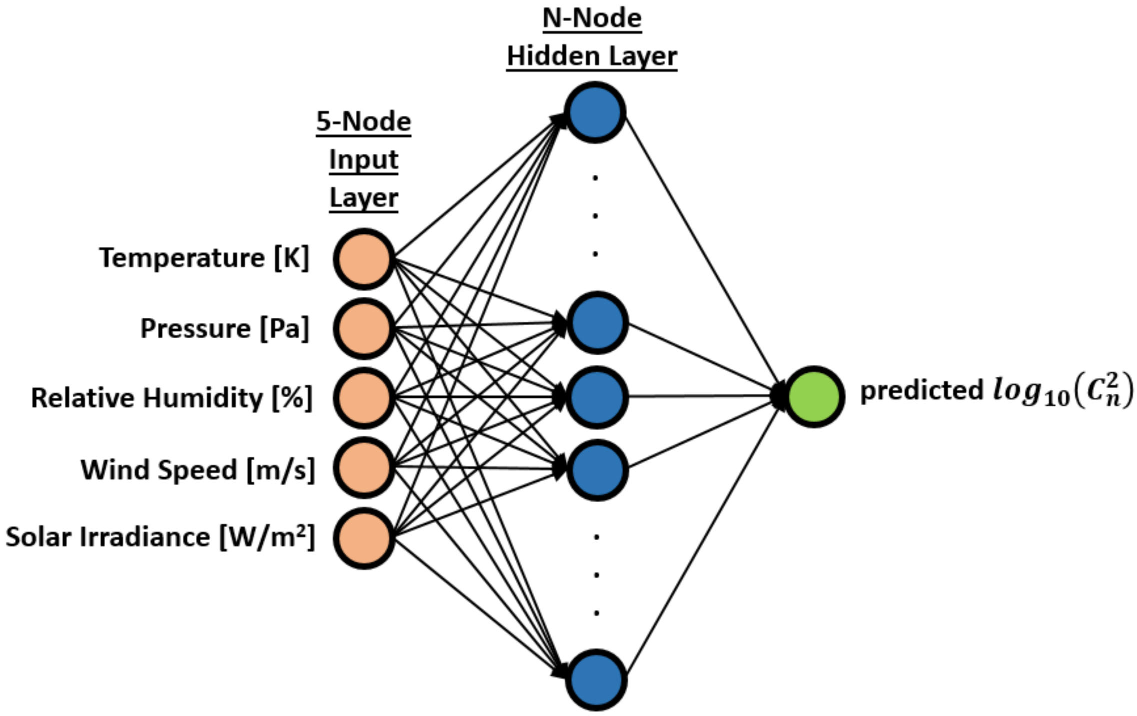


Figure 2.3: Simple MLP (multi-layer perceptron) to predict $log_{10}(C_n^2)$ from weather inputs.

2.2 MLP and RNN Architectures

2.2.1 MLP

$$y = \sigma\left(xW + b\right) \tag{2.1}$$

where x is the input, W is the learnable weights and b is the learnable bias.

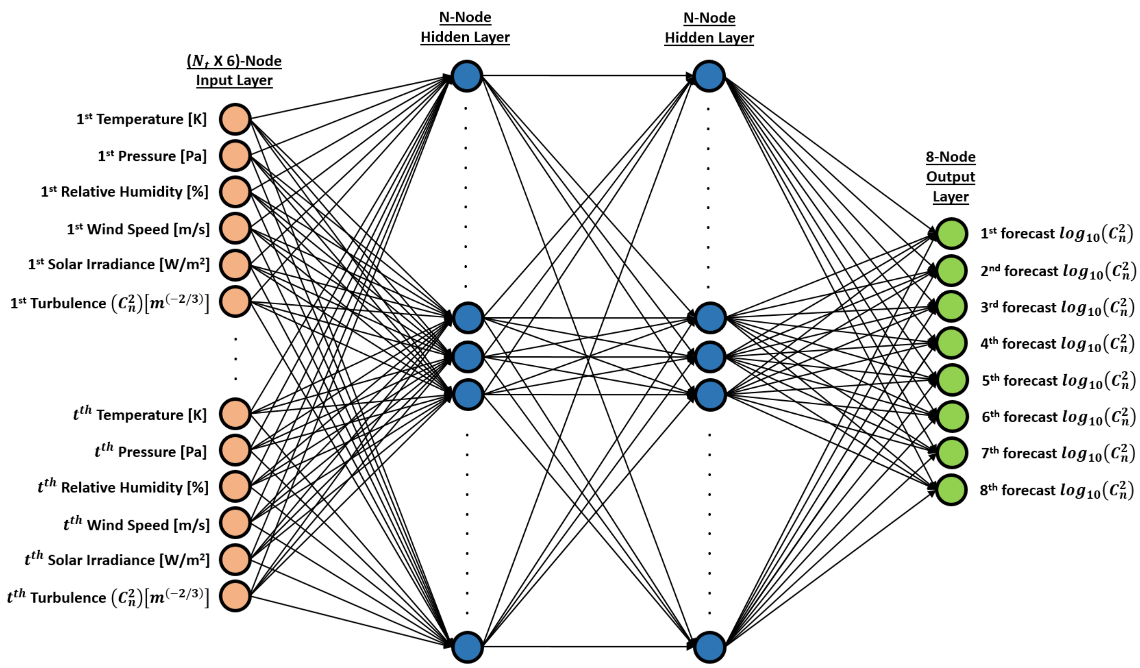


Figure 2.4: MLP to forecast eight time steps of $log_{10}(C_n^2)$ from multiple time steps of weather inputs.

2.2.2 RNN

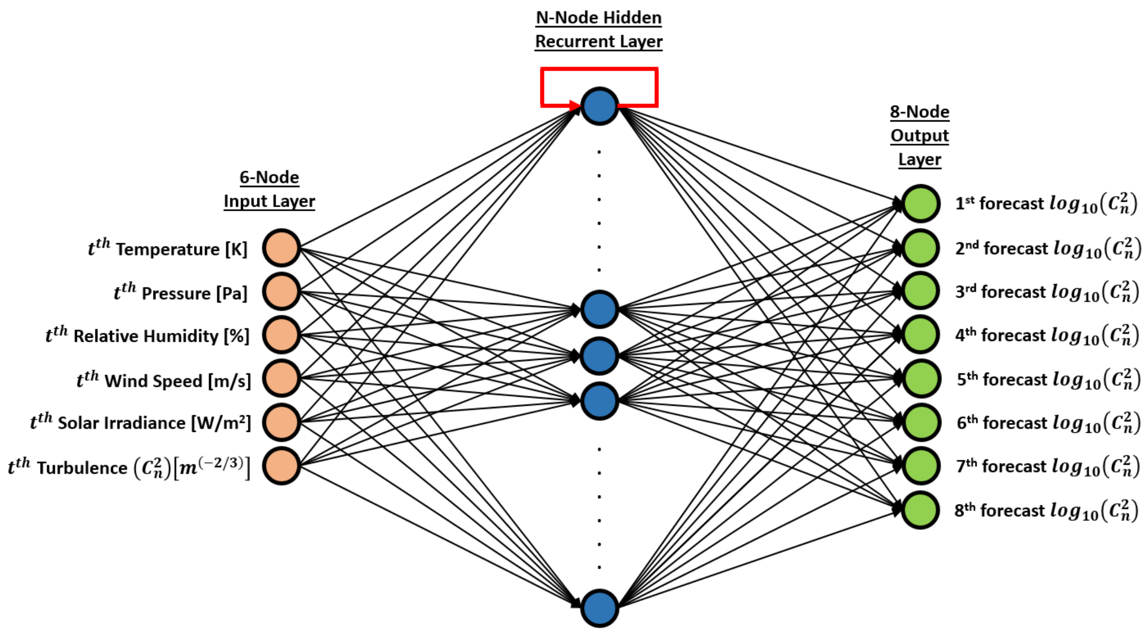


Figure 2.5: Single-layer RNN architecture to forecast eight time steps of $log_{10}(C_n^2)$ from multiple time steps of weather inputs processed by the recurrent layer.

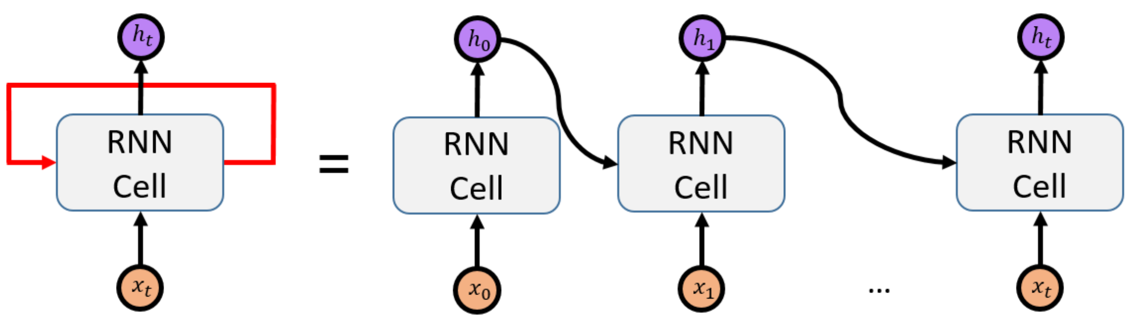


Figure 2.6: Unrolled single-layer RNN

A recurrent neural network (RNN) is a neural network that operates on a variable-length sequence. The RNN processes the sequence input with a hidden state h whose activation

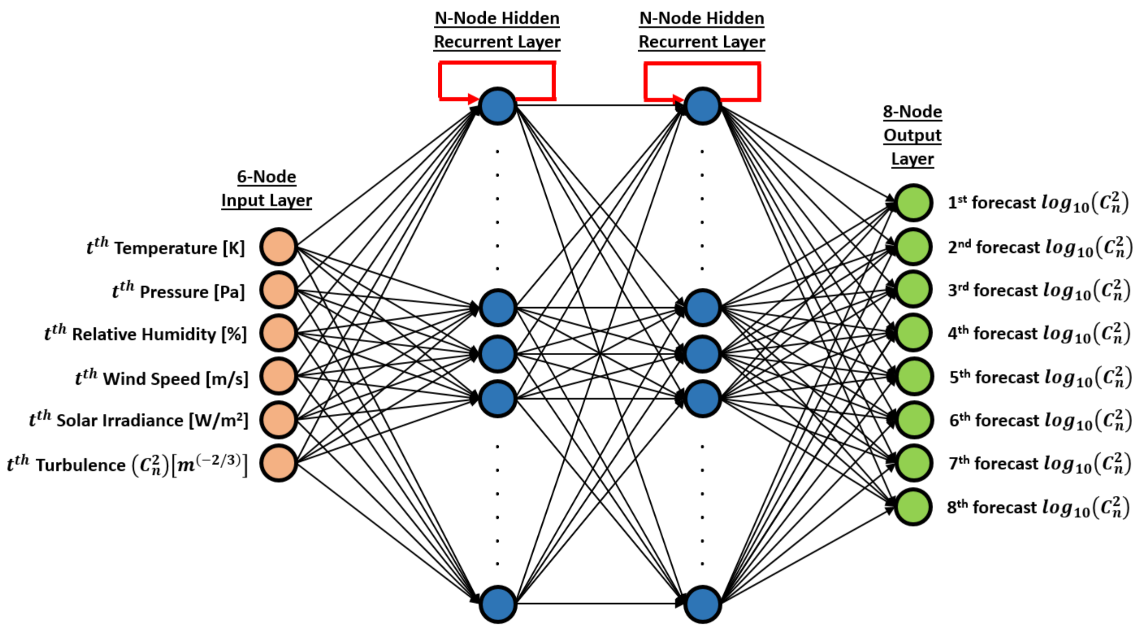


Figure 2.7: Multi-layer RNN architecture to forecast eight time steps of $log_{10}(C_n^2)$ from multiple time steps of weather inputs processed by the recurrent layers.

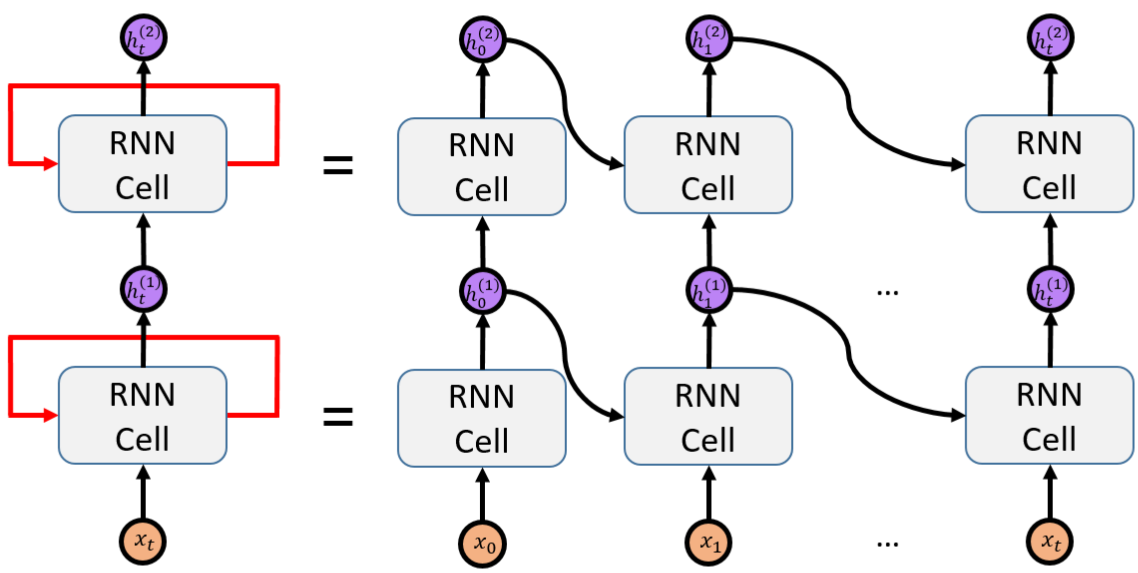


Figure 2.8: Unrolled multi-layer RNN

(value) at each time t is dependent on the activation of the previous time. At each time step t the hidden state h_t of the RNN is updated by

$$h_t = \tanh\left(W_{ih}x_t + b_{ih} + W_{hh}h_{(t-1)} + b_{hh}\right) \tag{2.2}$$

where x_t is the input at time t, $h_{(t-1)}$ is the hidden state of the previous layer at time $t-1$ or the initial state at time 0 (zero), W_{ih} is the learnable input-hidden weights, b_{ih} is the learnable input-hidden bias, W_{hh} is the learnable hidden-hidden weights, and b_{hh} is the learnable hidden-hidden bias [6].

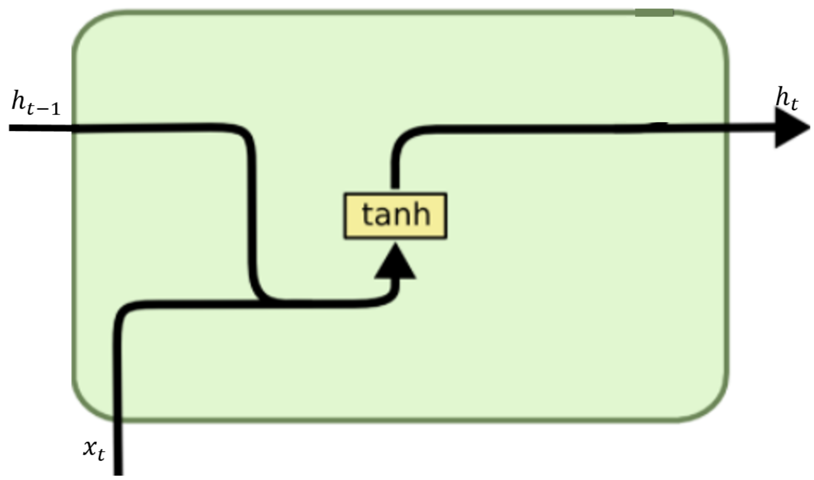


Figure 2.9: Simple RNN cell

2.2.3 LSTM

For each element in the input sequence, each layer computes the following function:

$$i_t = \sigma\left(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}\right) \tag{2.3}$$

$$f_t = \sigma\left(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}\right) \tag{2.4}$$

$$g_t = \tanh\left(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}\right) \tag{2.5}$$

$$o_t = \sigma\left(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}\right) \tag{2.6}$$

$$c_t = f_t \odot c_{(t-1)} + i_t \odot g_t \tag{2.7}$$

$$h_t = o_t \odot \tanh\left(c_t\right) \tag{2.8}$$

where h_t is the hidden state at time t, c_t is the cell state at time t, x_t is the input at time t, $h_{(t-1)}$ is the hidden state of the layer at time $t-1$ or the initial state at time 0 (zero), and i_t, f_t, g_t, and o_t are the input, forget, cell, and output gates, respectively. W_{ii}, W_{if}, W_{ig}, and W_{io} are the learnable input-hidden weights for the input, forget, cell, and output gates. σ is the sigmoid function, and $\odot$ is the Hadamard (element-wise) product [6].

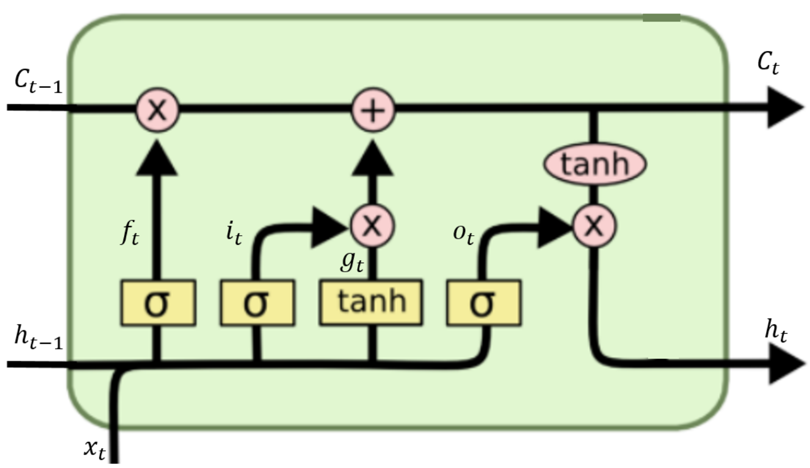


Figure 2.10: LSTM-RNN cell

2.2.4 GRU

For each element in the input sequence, each layer computes the following function:

$$r_t = \sigma\left(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}\right) \tag{2.9}$$

$$z_t = \sigma\left(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}\right) \tag{2.10}$$

$$n_t = \tanh\left(W_{in}x_t + b_{in} + r_t \odot \left(W_{hn}h_{(t-1)}b_{hn}\right)\right) \tag{2.11}$$

$$h_t = (1 - z_t) \odot n_t + z_t \odot h_{(t-1)} \tag{2.12}$$

where h_t is the hidden state at time t, x_t is the input at time t, $h_{(t-1)}$ is the hidden state of the layer at time $t-1$ or the initial hidden state at time 0 (zero), and r_t, z_t, and n_t are the reset, update, and new gates, respectively. W_{ir}, W_{iz} and W_{in} are the learnable input-hidden weights for the reset, update, and new gates, respectively, and are of shape $numlayers \times 3hiddensize \times inputsize$. W_{hr}, W_{hz} and W_{hn} are the learnable hidden-hidden weights for the reset, update, and new gates, respectively, and are of shape $numlayers \times 3hiddensize \times inputsize$. b_{ir}, b_{iz}, and b_{in} are the learnable input-hidden biases of the reset, update, and new gates, respectively. b_{hr}, b_{hz}, and b_{hn} are the learnable hidden-hidden biases of the reset, update, and new gates, respectively. σ is the sigmoid function, and $\odot$ is the Hadamard (element-wise) product [6].

2.3 Optimization and Loss Function

Neural networks are a collection of nested functions that are executed on some input data. These functions are defined by *parameters* (weights and biases). The training of a neural network is the process of updating the parameters to reduce the loss between network outputs and truth values. Training happens in two steps. The first step is forward propagation in which the network makes a prediction by passing input values through its functions

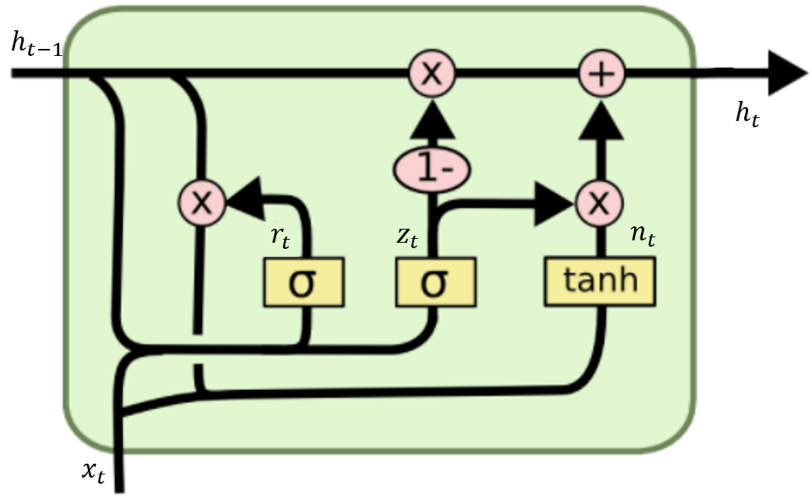


Figure 2.11: GRU-RNN cell

(applying the weights and biases). The second step is backward propagation where the network adjusts its parameters proportionate to the error in its prediction. The network does this by moving backwards through the network from the output, collecting the derivatives of the error with respect to the parameters of the functions and optimizing the parameters using gradient descent. The derivatives of the error with respect to the parameters are called gradients. PyTorch's automatic differentiation engine *autograd* handles this entire process [6].

Follow 3Blue1Brown video but with my own images and notations. Start with a 1x1 then move to a 2x2 network? "These chain rule expressions give you the derivatives that determine each component in the gradient that helps minimize the loss of the network by repeatedly stepping downhill." - 3Blue1Brown

2.4 Chapter Summary?

CHAPTER III

Dataset

Data preprocessing is an essential step in producing an effective machine learning model. The data for training must be representative of the problem being modeled, and the validation and test sets should be representative of the train dataset. Failing to satisfy these conditions can lead to an ineffective model, validation results that lead to poor model and hyperparameter selection, and testing results that inaccurately represent a model's capability. This Chapter steps through the preprocessing applied to the C_n^2 and weather data including filtering, window averaging, formatting into input sequences and forecasts, and parsing into train, validation, and test datasets.

3.1 Measurement Collection Setup

3.1.1 Weather Measurements

The entire dataset uses time correlated weather and C_n^2 measurements. The weather measurements are from a Davis Instruments Vantage Pro2 Plus weather station [8] deployed in front of an office building from 12 April 2020 through 10 August 2020. The weather variables recorded by the station and considered in this work are temperature, pressure, relative humidity, wind speed, and solar irradiance. Table 3.1 lists the technical specs of the weather station including the measurement resolution, accuracy, and archive interval.

The variable resolution refers to the minimum difference measurable, for example relative humidity can distinguish between 90% and 91%, but not between 90% and 90.5%. The resolution of wind speed, 0.45 m/s, is 1 mph (mile per hour) resolution converted to m/s. Variable accuracy is the variable error per measurement, so for example a measurement of relative humidity of 90% is has an error range of ± 2%. Finally, the archive interval indicates

Table 3.1: Davis Instruments Vantage Pro2 Plus Weather Station Measurements

Variable	Resolution	Accuracy ±	Archive Interval
Temperature	0.1°C	0.3°C	1 min
Pressure	0.1 mb	1.0 mb	1 min
Relative Humidity	1%	2%	1 min
Wind Speed	0.45 m/s	5%	1 min
Solar Irradiance	1 W/m^2	5%	1 min

how often the variable is reported. The archive interval is different from the measurement frequency. The anemometer measures wind speed every 2.5 to 3 seconds but the average over the archive interval, 1 minute, is reported. The archive interval of this dataset is 1 minute, the shortest interval available from the weather station.

The location of the weather station is on a storm drain surrounded by grass, about 20 feet to the southwest of a single-story office building. The weather station is not obscured by any trees or shrubbery in very close proximity. To the south of the weather station about 180°field of view is completely open space. The wind measurements are taken by an anemometer about 3 meters above ground level (AGL). The other measurements are recorded about 2.5 meters AGL. Figure 3.1 illustrates the location of the weather station and labels the key sensors and solar panel. The image in Figure 3.1 is taken from the southeast to illustrate the proximity of the station to surrounding geography to the northwest. To the right of the image is the office building about 20 feet away.

3.1.2 Turbulence (C_n^2) Measurements

The other component of the dataset is minute-by-minute measurements of C_n^2 as measured by an MZA DELTA (Delayed Tilt Anisoplanatism) turbulence profiler [9]. The

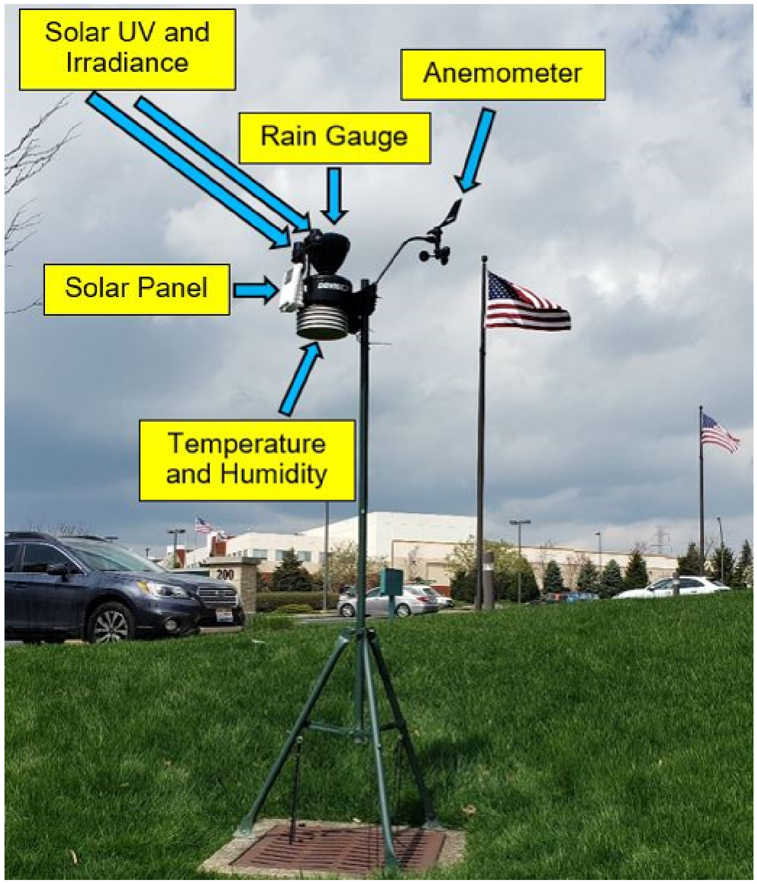
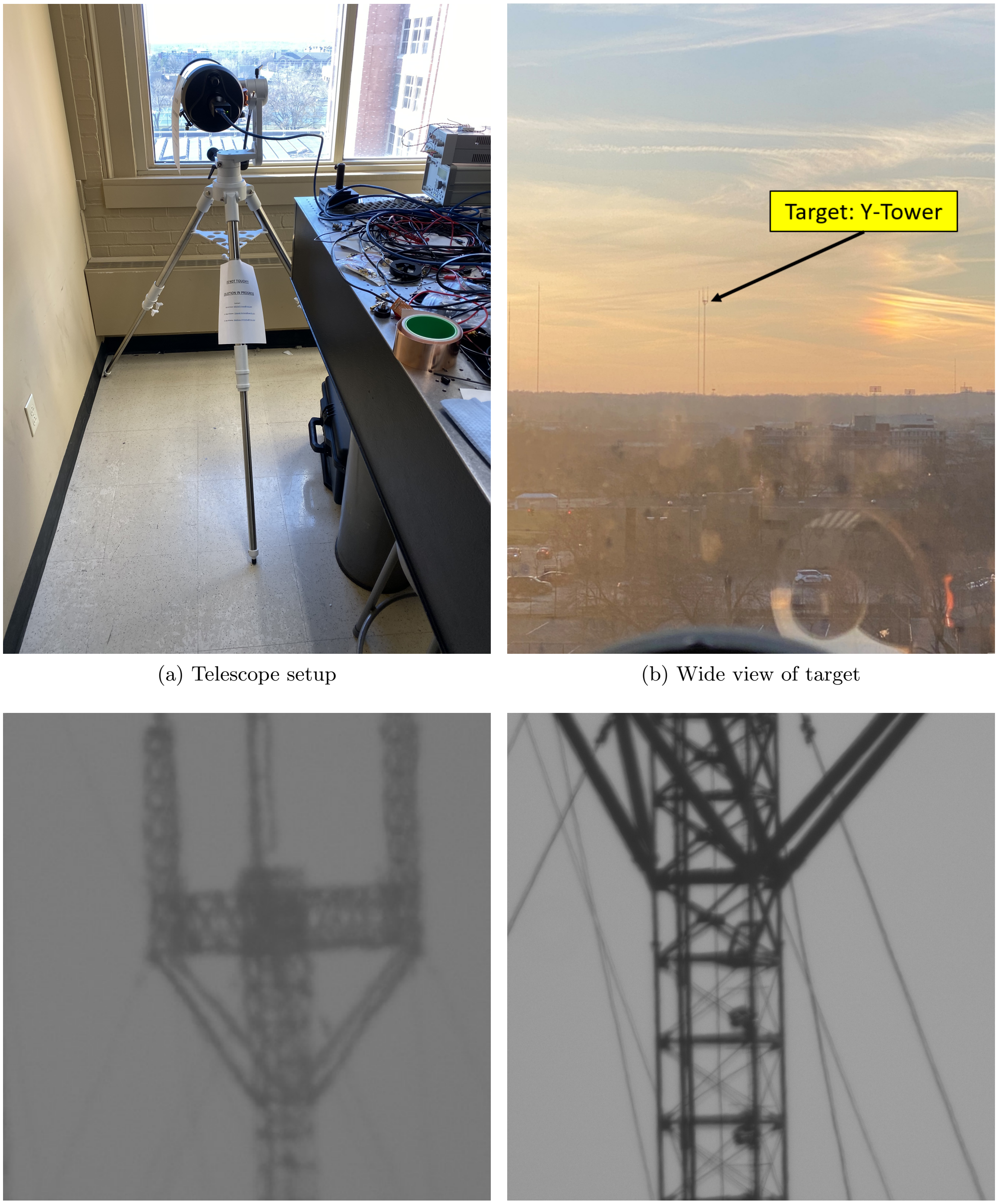


Figure 3.1: Weather station deployment

DELTA is a passive imaging sensor that uses a monochrome camera attached to a 6-inch telescope with 1.5 m focal length. The DELTA calculates differential jitter as a function of angular separation to measure C_n^2 by collecting 300 frames (images) of a target of opportunity at 100 Hz. It's desirable for the target to have high contrast features like edges and corners throughout the image to make measurements of differential jitter at variety of separations. Separations of 0.5 to 20 aperture diameters are recommended, so for a 6" telescope aperture the smallest feature separation would be 3" and largest 10' apart in the target plane.

The C_n^2 measurements in this work span from 12 April 2020 through 10 August 2020. During this time the DELTA was deployed on the 5th floor of Fitz Hall at University of Dayton in Dayton, Ohio. Figure 3.2a is a picture of the DELTA's deployment in Fitz Hall. The target the DELTA observes throughout the collection is the *WRGT/WKEF TV Dayton* tower to the southwest of the sensor. The range to the target is 6.4km. The platform is approximately 249 meters above mean sea level (AMSL), and the part of the target being imaged is estimated to be 566 meters AMSL. Figure 3.2b illustrates a wide field-of-view (WFOV) picture of the target being imaged by the DELTA. Figure 3.2c illustrates an narrow field-of-view (NFOV) image of the target during turbulent conditions. Figure 3.2d is a single frame from a DELTA image set during an evening neutral event (low C_n^2 strength). The sharp contrast in foreground and background and the abundance of edges and corners (features) make the target in Figure 3.2d excellent for the DELTA. The tower target is dubbed the "Y-Tower" because from the view of the DELTA the target looks like a "Y" as shown in Figures 3.2c and 3.2d.

As stated above, the DELTA calculates differential jitter as a function of angular separation to measure C_n^2. It's the calculation of C_n^2 at 10 locations along the observation path that make the DELTA a turbulence profiler in comparison with another sensor, like a Scintillometer, which reports only a single C_n^2 for the entire propagation path. The locations, or screens, of these C_n^2 measurements are at 5%, 15%, ..., 85%, 95% along the propagation path. In this work the profiling action is not utilized. Rather, the measurements along the path at each minute are uniform-path averaged for a single C_n^2 measurement per collection that is representative of the entire path. Although a uniform-path average is applied, the propagation path's geometry with respect to the terrain is highly relevant to the C_n^2 measurements at each screen and the *type* of propagation path this work investigates.



(a) Telescope setup

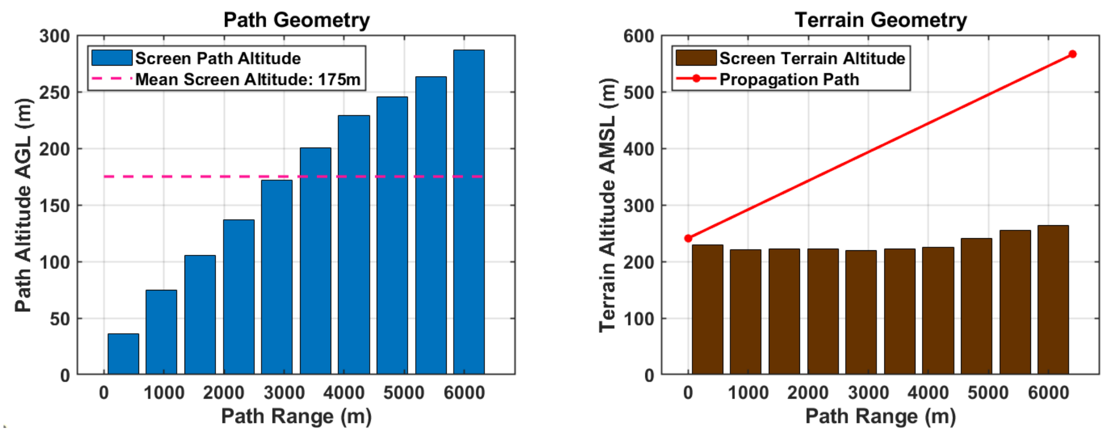
(b) Wide view of target

(c) Narrow view of the target

(d) Target as seen by the DELTA

Figure 3.2: DELTA setup for and target view for collection of minute-by-minute C_n^2 measurements.

Figure 3.3a illustrates the geometry of the propagation path. The y-axis is the path altitude AGL, and the x-axis is the path range. Each blue bar represents a DELTA screen at the aforementioned normalized locations along the propagation path. The average screen altitude, 175m, is drawn as a magenta dashed line. Similarly, Figure 3.3b illustrates with



(a) Path geometry (b) Terrain geometry

Figure 3.3: DELTA propagation path geometry

brown bars the altitude AMSL of the terrain at the DELTA screens. The propagation path is illustrated as the red line. Note that the angle of the propagation path with respect to the terrain bars in Figure 3.3b is misleading because the x-axis and y-axis limits on the plot are not scaled the same.

The DELTA screens AGL in Figure 3.3a illustrate that the altitude as a function of propagation path linearly increases for nearly the entire path. The average path altitude, 175m, is between the 5th and 6th screen. The height of the brown bars in Figure 3.3b illustrate that the altitude of the terrain as a function of the propagation path does not significantly change over the 6.4 km. This path illustrates that this work builds a machine learning model which forecasts C_n^2 at an average altitude of 175m AGL over 6.4 km. This geometry is unique as many long-term C_n^2 collections do not include these significant altitude changes and average altitude.

3.1.3 Spatial Relationship of Platforms and Target

The location of the DELTA (C_n^2) platform is approximately 8.87 km to the west of the Vantage Pro2 Plus (weather) platform, and the Y-Tower being imaged by the DELTA is approximately 6.40 km to the southwest of the DELTA platform. The combination of these paths puts the Vantage Pro2 Plus platform approximately 15.05 km from the Y-Tower. Figure 3.4 illustrates the spatial relationship between the three locations of relevance, where each yellow circle marker are approximate locations. Above each circle is a label of the

Figure 3.4: Spatial relationship between the DELTA platform, Vantage Pro2 Plus (weather) platform, and the Y-Tower (DELTA) target.

location being marked and its latitude and longitude. The red arrow pointing from the DELTA platform marker to the Y-Tower marker represents the DELTA propagation path being processed to a C_n^2. The blue arrows point from the Vantage Pro2 Plus platform marker to the DELTA platform marker and Y-Tower target marker.

Figure 3.4 illustrates one of the significant challenges of this work. There is significant spatial separation of the C_n^2 path and weather measurements. Turbulence by nature is a highly variable process and adding a gap between 8.87 km and 15.05 km from the weather measurements to the C_n^2 path adds another layer of complexity. The weather measurements at the particular platform location may or may not be correlated with the C_n^2 measurements to the west, and the degree of correlation is vulnerable to change in time. For example, on a cloudless day, the solar irradiance trends are likely highly correlated with the C_n^2 trends, but on a day with scattered clouds the C_n^2 measured at a particular minute might be in the sunshine but the correlated solar irradiance measurement is in the clouds. This modeling effort is especially subject to the spatial separation since the experiment is performed in Ohio, a state known for high weather variability on a day-to-day and even hour-to-hour timescale. Data processing to focus on trends is meant in part to dampen the effect of high frequency events that are not well correlated between the C_n^2 path and weather measurements platform.

A theoretical advantage to modeling with the simple RNN, GRU, and LSTM is to combat against the significant spatial separation of the weather measurements and C_n^2 path. Weather tends to flow from west to east, so from Figure 3.4 the C_n^2 measurements will be impacted by weather that is measured at a later time at the weather platform. These architectures process the input sequences as a time-series, thus it's theorized the networks could capture relationships between the measured C_n^2 and weather conditions that are not temporally synced.

3.2 Data Preprocessing

3.2.1 Filtering, Window Averaging, and Interpolating

Filtering

The MZA DELTA reports two metrics for each measurement: image confidence and C_n^2 confidence. The confidence levels each range from 0% to 100% and as the names suggest report the quality of the image sequence used for the calculation of C_n^2 and the quality in the C_n^2 calculation, i.e., the quality of the differential jitter measurements as a function of angular separation. The C_n^2 measurements can be easily quality-filtered by only keeping measurements which satisfy two user specified thresholds. In this work the thresholds are set to 50% and 70% minimum image and C_n^2 confidence, respectively. Figure 3.5 illustrates the image and C_n^2 confidences of each measurement as a function of local time-of-day throughout the experiment as blue and orange markers, respectively. The black solid and dashed black

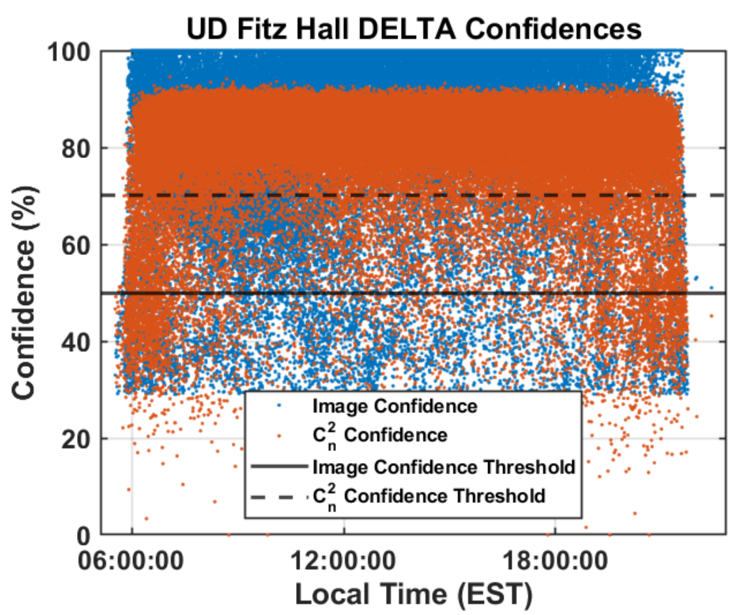


Figure 3.5: DELTA image and C_n^2 confidences.

lines represent the image and C_n^2 thresholds. A C_n^2 measurement is removed if an orange marker is below the dashed black line *or* if a blue marker is below the solid black line. The majority of the C_n^2 confidences throughout the day are above the specified threshold. Most of the measurements removed due to C_n^2 confidences are in the early morning and late evening when target illumination is less than ideal resulting in poor differential jitter measurements. The image threshold filtering is more uniform across the entire day. A high C_n^2 confidence is more important to a measurement than a high image confidence, so the threshold for the image confidence is lower. Of the 76,186 measurements that passed the confidence thresholds, the average image and C_n^2 confidences are over 91% and 83%, respectively.

The Vantage Pro2 Plus weather station does not report data quality metrics on a per-variable basis. Evaluation of weather measurements in plots does not raise questions of data quality, both in the raw measurement value for a check of nonphysical values, and measurement trends for a check that the temporal rate of change of the measurements is realistic.

Window Averaging

After quality filtering, the weather and C_n^2 measurements are window averaged. A written returns an array of window averaged datetimes (dates + times) and corresponding window averaged measurements given four inputs. The four inputs are an input array of datetimes, an input array of measurements, a window width, and an interval size. The window width determines the temporal width the function uses to calculate an average. For example, given a window width of five minutes the window will look at $\pm$ 2.5 minutes around the current time being averaged. Any measurements $\geq$ to the current time minus

2.5 minutes and < the current time plus 2.5 minutes is included in the average. The datetime returned for this example window averaged measurement is exactly the middle of the window. The interval size determines the temporal step each iteration. For example, given an interval size of one minute, the function will perform a window average about minute 07:05 on a given day, then step to perform a window average about minute 07:06. This iteration stops when the window includes a datetime beyond the last datetime in the array of measurements.

This modeling effort focuses on learning the relationship between the trends of prior environmental measurements and future C_n^2 measurements. To achieve this the window average uses a width of 30 minutes (± 15 minutes) and interval of 1 minute. This wide window significantly smooths the weather and C_n^2 measurements to dampen high-frequency events and amplify the trends. This more significantly impacts the C_n^2 measurements which are highly stochastic by nature.

Interpolating

After window averaging both sets of measurements, each is linearly interpolated to datetimes from 12 April 2020 through 10 August 2020 sampled every 30 minutes. The linear interpolation is performed by an open-source robust implementation from *NumPy* [10]. For each set of interpolations (weather and C_n^2), only interpolations less than 30 minutes are kept. Those that do not pass this condition are simply set to NaN (Not a Number) to preserve the arrays of measurements and datetimes sampled every 30 minutes. After this step the data is reduced to 5809 measurements at a 30 minute sample rate.

3.2.2 Formatting into Sequences and Forecasts

Nighttime C_n^2 as Input Sequences

A limitation of the experimental setup described in Section 3.1.2 is the passive imager DELTA lacking an illuminated target. As a result, C_n^2 measurements before 06:00 and after 22:00 are entirely absent. Without a method to fill the nighttime C_n^2 this modeling technique is unable to incorporate prior C_n^2 measurements as an input variable to the model. It's hypothesized that including prior C_n^2 as an input will improve modeling performance as it will provide the model information about the trend of C_n^2 leading up to the forecast. The model is expected to learn that it's first forecast timestamp should follow the leading trend. If results indicate the inclusion of prior C_n^2 improves forecasting, then a technique for filling missing nighttime data is necessary for real-world applications where a target may not be illuminated during nighttime but morning forecasts are still required.

The data after filtering, window averaging, and interpolating as described in Section 3.2.1 is the basis of the nighttime C_n^2 filling. Nighttime measurements are classified as an index where the measurement is before 08:00 or after 20:00, and whose C_n^2 measurement already labeled as NaN. This technique generously includes times from late evening into early morning, but only those which are already missing measurements. Of the 5809 data points, 2065 (35.5%) satisfy these conditions. Without this filling, over a third of the dataset would be removed from consideration for further processing. Of the 2065 filled measurements, 124 fall into early morning and late evening at 07:00 and 21:00, illustrating the need to be generous with the classification of "nighttime." The effect of missing data is further amplified when complete sequences of data must be formatted in next steps.

The average of the C_n^2 dataset is approximately $9 \times 10^{-16} (m^{-2/3})$ so rounding sets the C_n^2 fill-value for the qualifying indices to a constant $1 \times 10^{-15} (m^{-2/3})$. Every input sequence used for training will have a constant C_n^2 and solar irradiance measurement of $0W/m^2$ during nighttime, thus it's expected the model will learn the difference between a constant sequence C_n^2 and a sequence of varying C_n^2 leading up to the forecast.

Formatting into Sequences and Forecasts

The next step in the data processing is to format into input sequences and output forecasts. As part of the grid search described in Section 4.1, multiple input sequence lengths are formatted to investigate the amount of prior weather and C_n^2 measurements to most effectively forecast future C_n^2 conditions. The forecast length of 4 hours (eight 30-minute time steps) is held constant for every input sequence length. Sets of sequences and forecasts are formatted for input sequence lengths of 4 hours (8 time steps), 8 hours (16 time steps), 12 hours (24 time steps) and 16 hours (32 time steps). The logic of the formatting follows.

Loop through the entire formatted dataset one timestamp at a time. The sequence is every input variable (temperature, pressure, relative humidity, wind speed, solar irradiance, and nighttime filled C_n^2) up to the desired sequence length. For example with a 4 hour sequence length, the sequence is the first 8 timestamps. Then, the forecast is the measured C_n^2 for the next 4 hours (8 timestamps) immediately following the last timestamp in the input sequence. If any part of the input sequence or output forecast contains a NaN then the sequence/forecast set is not kept. As a result of this filtering, a single NaN can cause many sequence/forecast sets to be removed because they're filtered regardless of how many NaNs are in the sequence/forecast and their location. This process loops through the entire dataset until the last timestamp of the forecast is beyond the last timestamp of the dataset. This process is repeated for each of the desired sequence lengths: 4 hours, 8 hours, 12 hours, and 16 hours. Once each is formatted, only the sequence/forecasts pairs common to all sequence lengths are kept to maintain consistency in training, validating, and testing. A total of 1255 sequence/forecast pairs are created for each of the sequence lengths.

3.2.3 Final Data Preparations for Modeling

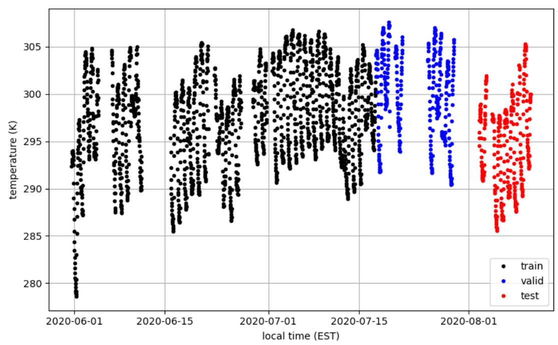
Parse Data into Train, Validation, and Test Sets

Proper parameter search to find the best model requires an unbiased evaluation of the models trained during the search. Once a model and parameters are selected, the model must be evaluated again without bias to form conclusions about the model's capability. To achieve this, the formatted dataset must be broken into three parts: train, validation, and test datasets. The ensemble of models for the parameter search are trained exclusively with the train dataset. The evaluation of the parameter search models is based on validation dataset performance. The best model as evaluated on the validation dataset is trained on the train and validation datasets concatenated together, then applied to the test dataset for a final evaluation.

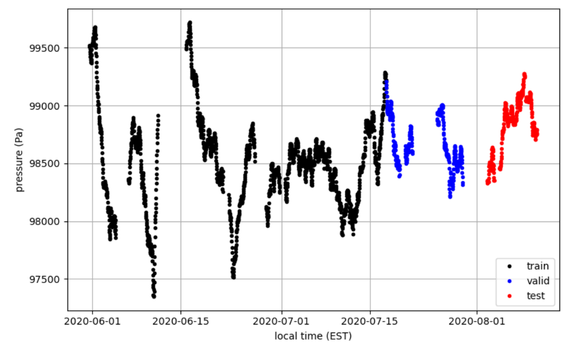
Modeling future C_n^2 conditions from prior environmental measurements is dependent on the time of day and year due to daily and seasonal weather fluctuations. This feature is amplified to a significant challenge in the context of a real-world application of this problem: forecasting C_n^2 conditions days or weeks beyond the last day of training examples. In this scenario the model is tasked with extrapolating forecasts given input weather conditions that are not representative of the train dataset. The input conditions could be unique in their temporal evolution or the measurements are beyond the dynamic range of the training data. An example of significant model extrapolation that could lead to poor performance is training a model on data from summertime then performing inference on data from wintertime.

As stated above, both the weather and C_n^2 measurements for this work range from 12 April 2020 through 10 August 2020, springtime through summertime. Following the timeline of model training and deployment, the train, validation, and test datasets are parsed from the entire dataset in the order listed. The ensemble of models for the parameter search is trained on the train dataset which temporally spans from the beginning of the data to a selected endpoint. The models are evaluated on the validation dataset which temporally spans from immediately after the train dataset endpoint to another selected endpoint. The train and validation datasets are concatenated and applied to a test dataset which starts immediately after the validation dataset endpoint.

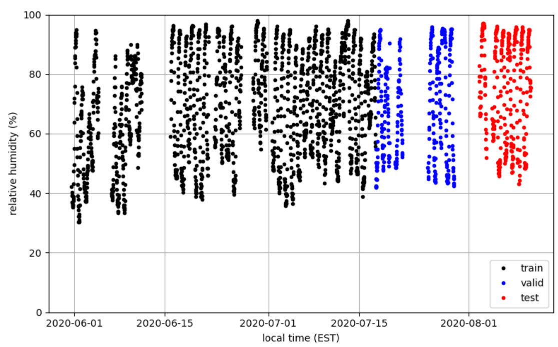
The train, validation, and test datasets are selected on two criteria in this work. The first is the real-world scenario of deploying sensors for 1-2 months before an experiment where forecast C_n^2 conditions are relevant. Experiments of this nature are often scheduled a many months in advance at minimum so sensor deployment 1-2 months before is realistic. This deployment time increases the range of unique weather and C_n^2 conditions for model training examples before inference on experiment days. The second criteria is bounding the dataset to summertime to avoid model extrapolation. This scenario idealizes the model development and evaluation since evolution of weather conditions is an inherent problem, but analysis of significant model extrapolation is not within the scope of this work. Obeying this criteria, the full dataset for training through testing spans 01 June 2020 through 10 August 2020. Figure 3.6 illustrates the input sequence variables as a function of local time for the selected dataset. Temperature, pressure, relative humidity, wind speed, solar irradiance, and



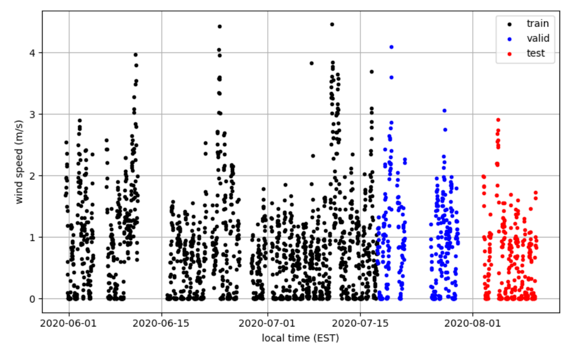
(a) Temperature (K)



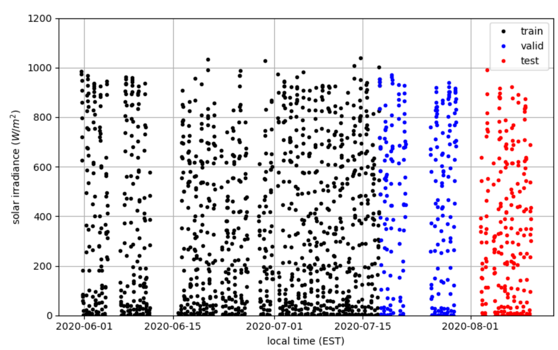
(b) Pressure (Pa)



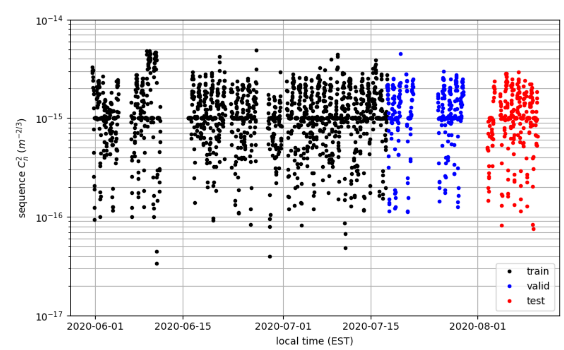
(c) Relative Humidity (%)



(d) Wind Speed (m/s)



(e) Solar Irradiance (W/m^2)



(f) Turbulence C_n^2 ($m^{-2/3}$)

Figure 3.6: Sequence data as a function of time, parsed by train, validation, and test datasets drawn in black, blue, and red, respectively.

nighttime-filled C_n^2 are in Figures 3.6a - 3.6f, respectively. The plotted measurements are after the data processing described above, so they're at 30-minute intervals after filtering, window averaging and interpolating. Furthermore in each plot, the train, validation, and test datasets are plotted with black, blue, and red markers, respectively. The train dataset spans from 01 June through 17 July. The validation dataset spans from 18 July through 01 August. Finally, the test dataset spans from 02 - 10 August. Of the 1255 sequence/forecast pairs, 933 (74.3%) are in the train dataset, 174 (13.9%) in the validation dataset, and 148 (11.8%) in the test dataset.

The measurements in Figure 3.6 illustrate many interesting features. First for temperature in Figure 3.6a, the positive result of only using data from June through August is shown. Across the entire dataset, except for the very first day, the temperature measurements are within about 20 Kelvin. Further, the validation dataset (blue) is a great representation of the train dataset (black). The test dataset (red) has a few cold days in the beginning, but then warms up again. This cold snap is not ideal, but is well within the dynamic range of the train and validation datasets. Pressure in Figure 3.6b shows that long, large scale measurements are made in June but in July and August the measurements are on a shorter scale. The validation and test pressure datasets are well within the dynamic range of the train dataset, but do not illustrate clearly similar trends.

Relative humidity in Figure 3.6c is a very stable weather measurement across the train, validation, and test datasets. The measurements never drop below 30%, and the validation and test datasets are within the dynamic range of the train dataset. The oscillatory nature of relative humidity is shown in Figure 3.6c and appears consistent between the three datasets. Wind speed in Figure 3.6d is not a stable measurement. An oscillatory nature is not apparent, and wind speed is a significant driver of weather events as shown by the spikes in measurements, especially in the train dataset. Commonality between the three datasets in Figure 3.6d are not clearly extracted.

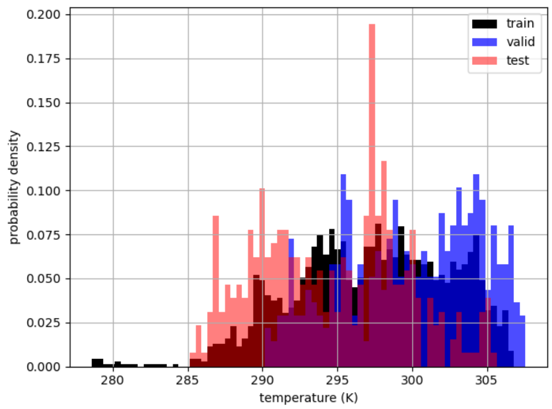
Solar irradiance in Figure 3.6e is another stable weather measurement across the three datasets. The model is expected to learn the temporal aspect of solar irradiance, i.e., a measurement of 0 (zero) indicates nighttime, and also the variation from a standard diurnal trend is due to cloud conditions which is typically associated with C_n^2 conditions which also deviate from a standard diurnal trend. Solar irradiance is directly impacted by the seasonal weather change (Earth's axial tilt). The highest solar irradiance measurements for a day are in summertime, specifically around the summer solstice. The highest measurement of the day gradually decreases from the summer solstice to a minimum around the winter solstice. This effect is not clearly shown in Figure 3.6e since only summertime is considered.

Sequence C_n^2 in Figure 3.6f illustrates stability and the impact of nighttime filling with $1 \times 10^{-15}(m^{-2/3})$. First, the test and validation datasets are strongly representative of the train dataset. Besides a few instances in the whole dataset, the maximum C_n^2 does not increase above $3 \times 10^{-15}(m^{-2/3})$. Daily oscillation of C_n^2 is apparent in Figure 3.6f and is the standard diurnal trend. The low-strength C_n^2 conditions are much more variable than high-strength conditions. There are several examples of C_n^2 weather than $1 \times 10^{-16}(m^{-2/3})$ but they are spread through the entire dataset. These events are associated with particularly strong evening neutral events. The line of markers at $1 \times 10^{-15}(m^{-2/3})$ in Figure 3.6f is the nighttime padding. These padded measurements are strongly correlated with the $0W/m^2$ solar irradiance measurements in Figure 3.6e.

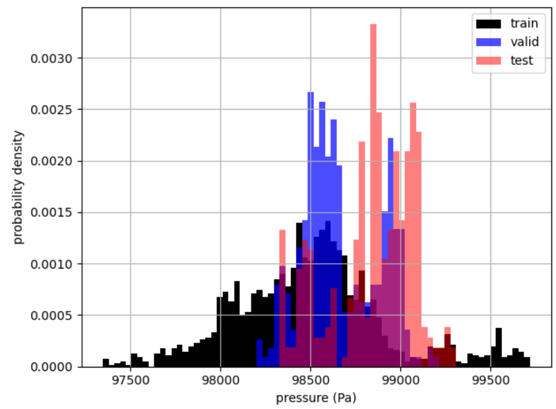
The relationship of the train, validation, and test datasets is further shown in Figure 3.7 which plots probability densities of each input sequence variable. Temperature, pressure, relative humidity, wind speed, solar irradiance, and nighttime filled C_n^2 are plotted in Figures 3.7a through 3.7f, respectively. Similarly to the temporal plots, the black bars in Figure 3.7 represent the train dataset, blue bars the validation dataset, and red bars the test dataset. The bars in each plot of Figure 3.7 represent how the *distribution* of the three datasets relate to each other. They also explicitly show how the dynamic range of the validation and test datasets relate to the dynamic range of the train dataset.

Temperature in Figure 3.7a shows that the distribution of the validation dataset is near the high-temperature portion of the train dataset since the blue bars are to the right of the plot. There are even blue bars at the very high temperatures where there are no black bars which is indicative that there are measurements in the validation dataset which are hotter than anything in the train dataset. This is an example where the model is forced to extrapolate given input measurements its never encountered in training. The test dataset (red bars) is at the lower end (left side) of the temperature distributions indicative that colder temperatures are part of the test measurements. These low temperature measurements are still within the bounds of the train dataset dynamic range since there are black bars to the far left of Figure 3.7a. The features of these distributions are consistent with the features noted in the temporal plot in Figure 3.6a.

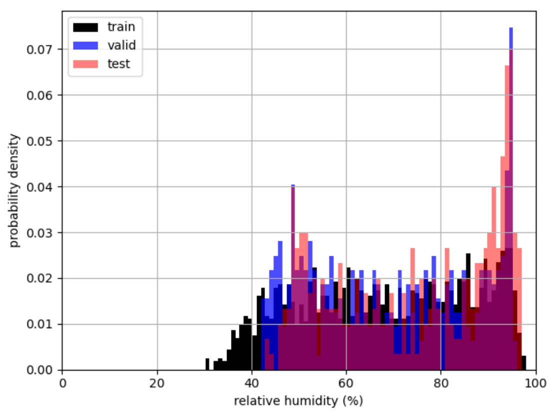
Pressure in Figure 3.7b shows that the temperature distribution is most populated around 98,500 Pa and looks similar to a Gaussian distribution. The validation dataset is mostly distributed around the center of the train distribution, an encouraging feature. The test dataset is similarly distributed but a scale factor higher around 99,000 Pa. There is also a significant amount of validation dataset pressure measurements around this value.



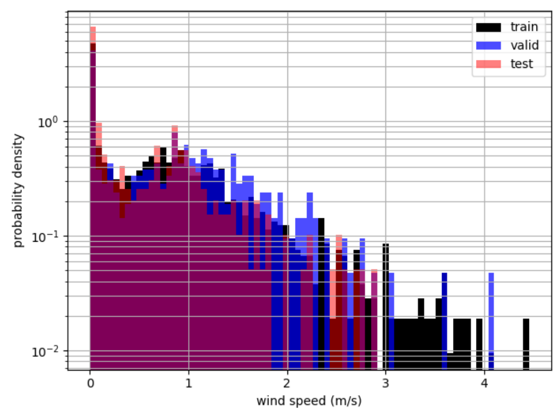
(a) Temperature (K)



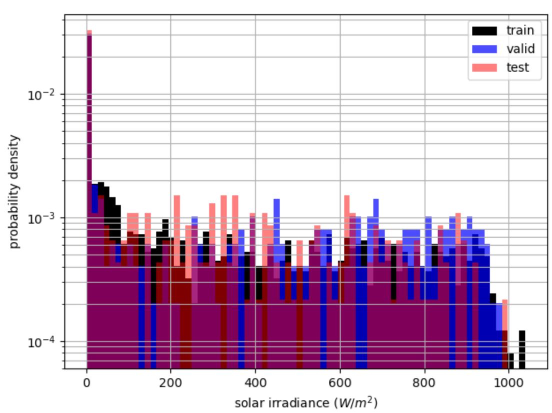
(b) Pressure (Pa)



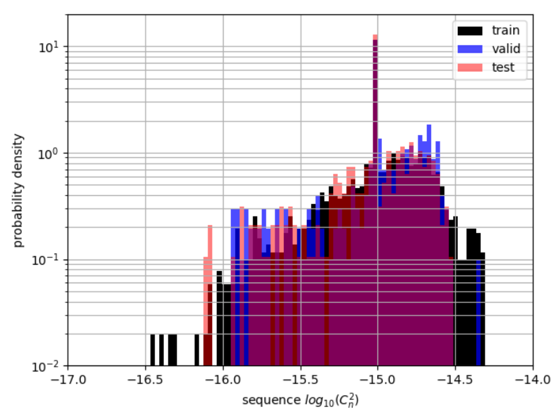
(c) Relative Humidity (%)



(d) Wind Speed (m/s)



(e) Solar Irradiance



(f) Nighttime-filled $log_{10}(C_n^2)$

Figure 3.7: Sequence data in normalized histograms, parsed by the train, validation, and test datasets drawn in black, blue, and red, respectively.

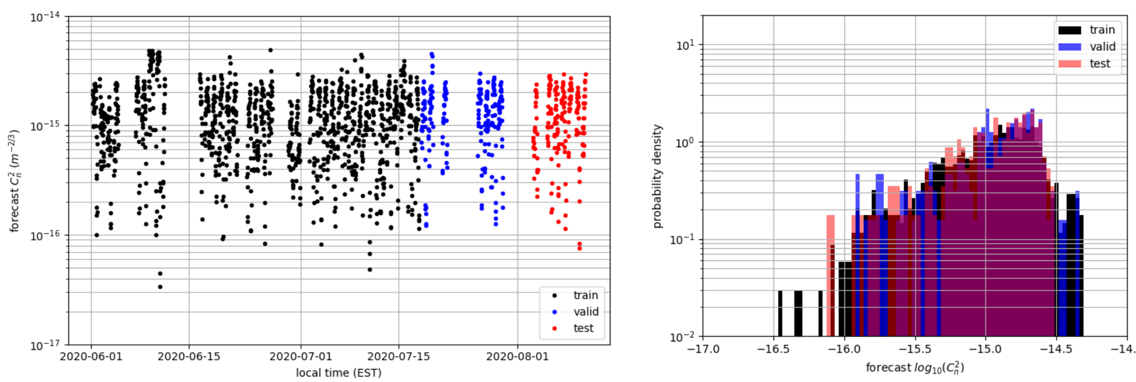
Relative humidity in Figure 3.7c illustrates the stability between the three datasets first described above. The distribution of the three bar sets are very similar in shape and magnitude. The spike in probability just below 100% and the slight rise around 50% is characteristic of all three datasets. The distribution between is likewise very similar. Figure 3.7c illustrates an ideal case for the datasets to be highly representative of each other.

Wind speed in Figure 3.7d is the first histogram to show comparison that is more encouraging than the temporal plots in Figure 3.6. The shape of the three distributions in Figure 3.7d are very similar from wind speeds of 0 m/s to about 2 m/s. The spike in measurements at 0 m/s is highly consistent among the three distributions. The sharp drop off back to another increase around 1 m/s, and the gradual decline in probability from 1 m/s to 2 m/s is also consistent. The validation distribution continues to follow the train distribution beyond 2 m/s, but the test measurements become far more sparse. Figure 3.7d indicates that below 2 m/s the validation and test distributions are highly representative of each other. However, the temporal trends of wind speed are most relevant to this problem and that is not illustrated in the histogram. As stated earlier, the temporal trends in wind speed are not clear in Figure 3.6d. The validation distribution is within the train distribution, and the test distribution even more so.

The solar irradiance distributions in Figure 3.7e are highly representative of each other. The spike in probability density at $0W/m^2$ is nearly identical across the three distributions. Furthermore, the shape of the distributions up to $1000W/m^2$ are also very similar. Some gaps exist in the test dataset, but the general distribution is representative. Most importantly, the validation and test distributions are within the dynamic range of the train distribution.

Finally, nighttime-filled $log_{10}(C_n^2)$ in Figure 3.7f again illustrates very similar distributions between the three datasets. For the majority of the measurements from about -15.75 through -14.5 the distributions are nearly identical. The spike in probability at -15 is due to the nighttime filling. The dynamic range of the validation and test datasets are mostly well within the bounds of the train dataset. Generally, the relationship of the distributions in Figure 3.7f is an ideal scenario.

The other part of the sequence/forecast pair is the forecasts. Figure 3.8 illustrates the temporal and histogram plots in Figures 3.8a and 3.8b. These plots are nearly identical to input sequence C_n^2 Figures 3.6f and 3.7f with the primary difference being the lack of nighttime measurement filling at $1 \times 10^{-15}(m^{-2/3})$. This is shown as the missing line of



(a) (b)

Figure 3.8: Turbulence (C_n^2) forecasts as a function of time and as a normalized histogram. The train, validation, and test datasets are drawn in black, blue, and red, respectively.

markers in Figure 3.8a and the missing spike in Figure 3.8b. The three datasets are very similar both temporally and in their distributions, again a highly desirable quality. An interesting feature about the three datasets in Figure 3.8a is that the validation dataset has a measurement above $3 \times 10^{-15}(m^{-2/3})$ but the test dataset does not. Furthermore the validation dataset does not have any measurements weaker than $1 \times 10^{-16}(m^{-2/3})$ but the test dataset has two. The histogram in Figure 3.8b reflects this. These features are the most significant difference between the three datasets and potentially could impact the parameter selection process and final model evaluation. The parameter selection process is evaluated on a dataset which does not have any extremely deep C_n^2 events. Therefore the best model could be a model which does not learn to handle these deep neutral events at all which will result in poor performance on the deep neutral event in the test dataset. Furthermore, the validation dataset has an event of abnormally strong C_n^2. This could also lead the best model to handle these events well but will not be reflected in the final model evaluation since an event of this sort is not in the test dataset.

Overall, the sequences and forecasts presented in Figures 3.6, 3.7, and 3.8 show that the train, validation, and test datasets are strong representations of each other, and there are no significant changes in weather patterns on a day-to-day and seasonal basis. Also, a few measurements of temperature is the only instance of a validation or test measurement being beyond the dynamic range of the train dataset. This small outlier does not propagate to the test dataset because the final model is trained on the combination of train and validation datasets. These are highly desirable characteristics for a machine learning study.

Data Normalization

With the sequence/forecast pairs for training, validation, and testing, the final step in data processing is to normalize the data for training and inference. The normalization parameters are derived from the train dataset, but are applied to the train, validation, and test datasets. In this work each input variable and output C_n^2 from the train dataset are min-max ranged between 0 and 1 using

$$\hat{x} = \frac{x - x_{min}}{x_{max} - x_{min}}, \tag{3.1}$$

where x is the variable considered, x_{min} is the minimum of x, x_{max} is the maximum of x, and $\hat{x}$ is the normalized variable. To ensure stability, data normalization for C_n^2 is applied to the $log_{10}(C_n^2)$ since values associated with turbulence are most often smaller than $1 \times 10^{-12}(m^{-2/3})$. The normalization parameters x_{max} and x_{min} are saved then Equation 3.1 is applied to the validation and test datasets. Any measurements outside of x_{max} or x_{min} will normalize to beyond 0 and 1.

Throughout this work evaluation of model performance is done in $log_{10}(C_n^2)$ space so model outputs must be unnormalized by

$$x = \hat{x} \cdot (x_{max} - x_{min}) + x_{min} \tag{3.2}$$

and compared with the unnormalized measurement C_n^2 forecasts.

3.3 Chapter Summary?

CHAPTER IV

Grid Search

This chapter walks through the methodology used to select the best model and parameters to forecast 4 hours of C_n^2 given prior environmental measurements, then the statistical analysis performed as a justification for the model selection.

4.1 Methodology

Given a problem to model with machine learning there are model hyperparameters to adjust in infinitely many combinations, each of which can impact model performance. Just a few examples of these hyperparameters are the optimization algorithm, the learning rate of the optimization algorithm, the number of layers in a model architecture and the number of nodes per layer. Given the many combinations of a model's hyperparameters, a careful method to determine the best combination must be employed. The definition of the "best" model is the model which results in the best performance when applied to the validation dataset, a subset of the entire dataset specifically held from training for hyperparameter tuning. By holding back the validation dataset, an unbiased optimization of the hyperparemeters can be performed. The model and parameters which performs best on the validation dataset is then applied to the test dataset for another unbiased evaluation of the final model.

There are many methods of hyperparameter optimization, but common methods are grid search and random search. Grid search, or a parameter sweep, is an exhaustive search through manually specified hyperparameters. If iterating over only one hyperparameter, for example three different learning rates, three models are trained with the three learning rates. A model is trained with the first learning rate and it's performance on the validation dataset is recorded, then another model is trained but with the second learning rate and the performance on the validation dataset is recorded. This is done once more for the third learning rate. The performance of the models with the three learning rates are compared and the best learning rate is the learning rate used by the model that performed best. This method can quickly explode in computation time as the number of hyperparameters to iterate increases. For example, iterating over two hyperparameters of three values each results in nine combinations of hyperparameters. The number of combinations is defined as the multiple of the number of values across each hyperparameter, so if there are five hyperparameters with 1, 2, 3, 4, and 5 values, then the total number of combinations is $1 \times 2 \times 3 \times 4 \times 5 = 120$ combinations.

The other common method, the random search, replaces the exhaustive grid search by randomly selecting hyperparameters within defined bounds. An algorithm randomly selects the hyperparameters and models are trained with the different combinations then applied to the validation dataset to evaluate performance. A benefit of the random search is the selection of parameter combinations that might not be defined in a grid search.

In this work the grid search is employed because prior knowledge of hyperparameters is unknown, thus the exhaustive grid search is necessary to explore a wide range of combinations. This grid search iterates over five parameters. The outermost parameter is the four fundamental architectures: MLP, simple RNN, GRU, and LSTM. The MLP is a common machine learning architecture and serves as a baseline. The simple RNN, GRU, and LSTM are variants of the general RNN and are searched to find if a specific variant is better or worse than the others when applied to this problem. The next parameter is the input sequence variables used by the model. From Section 3.2.3, the available input sequence features (variables) are prior temperature, pressure, relative humidity, wind speed, solar irradiance, and C_n^2 measurements. The *input features* parameter iterates over which features (variables) to train the model. Since there are six available features, and each feature can only be used or not used, there are $2^6 = 64$ total combinations. However, a threshold is set to train on a minimum of four features which reduces the total number of input feature combinations to 22. This threshold is set to discourage the model from memorizing one or two features, and for a significant reduction in computation time.

The third search parameter is the *input sequence length*. Independent of the input features (variables), in a single sequence/forecast the amount of information available to the model is dependent on the time-length of the input sequence. Whether the model performs best with only 4 hours of input data or 16 hours of input data is highly relevant information. Thus, four lengths of the input sequence are searched: 4, 8, 12, and 16 hours. These lengths are chosen to be $1\times$, $2\times$, $3\times$, and $4\times$ the 4 hour forecast length. Note that the train, validation, and test datasets are carefully formatted so the same forecasts are trained, validated, and tested regardless of the input sequence length. This avoids an instance where a 4 hour input sequence might exist for a particular forecast but a 16 hour input sequence is not available due to missing data. The fourth parameter searched is the number of hidden layers in each architecture: 1 or 2. The fifth and final searched parameter is the number of hidden nodes per hidden layer: 10 through 50 in steps of 10. The final two parameters essentially search over the number of parameters in the model with some variation in the interaction of those parameters. Each fundamental architecture in total iterates over $22 \times 4 \times 2 \times 5 = 880$ combinations of parameters. Due to the stochastic nature of model training, a single model could perform significantly different than another model trained with the same parameters, thus a total of 10 models are trained per combination per fundamental architecture to ensure the stability of results. In this search a total of $880 \times 4 \times 10 = 35,200$ models are trained.

For each model trained in the grid search the following parameters are fixed: mini-batch size, optimization algorithm, initial learning rate, learning rate decay (step and decay factor), and weight decay. The mini-batch size, the number of training examples used per model parameter update, is set to 32 yielding a total of 30 parameter updates (933/32) per epoch (iteration through the entire dataset). The optimization algorithm is AdamW[11] [12]. The initial learning rate is 0.01 and decays by a factor of 10 every 10 epochs. This results in learning rates 0.01, 0.001, 1e-4, 1e-5, and 1e-6 from epochs 1 - 10, 11 - 20, 21 - 30, 31 - 40, and 41 - 50, respectively. The high initial learning rate is to ensure suitably-high gradients are back-propagated through the model to allow each model 10 epochs (300 total parameter updates) to escape any local minima. This training method consistently leads to a strong model convergence in only a few seconds. The weight decay is a regularization technique applied to the optimization algorithm and is 0.001.

4.2 Grid Search Results

The analyses of the grid search are performed independently on each architecture. From the four grid searches, four 2d-arrays of RMSE loss scores, the performance metric between validation truth and output $log_{10}(C_n^2)$, are recorded for analysis. The 2d-arrays are shape 880×10 for 880 parameter combinations and 10 models each. From these four 2d arrays, the average and standard deviation of the RMSE scores are calculated per parameter combination yielding four arrays of 880 averages and standard deviations. The standard deviation, the square root of the average of the squared deviations from the mean, is calculated by

$$\sigma = \sqrt{\frac{\sum (x - \bar{x})^2}{N - ddof}} \tag{4.1}$$

where σ is the standard deviation, x is the data, $\bar{x}$ is the average of the data, N is the number of data points, and $ddof$ is the delta degrees of freedom. Using $ddof = 1$ provides an unbiased estimator of the variance of the infinite population. It is standard practice to use $ddof = 1$. $ddof = 0$ should be used in the situation of measuring the variance of a distaribution whose mean $\bar{x}$ is known a priori rather than being estimated from the data [13].

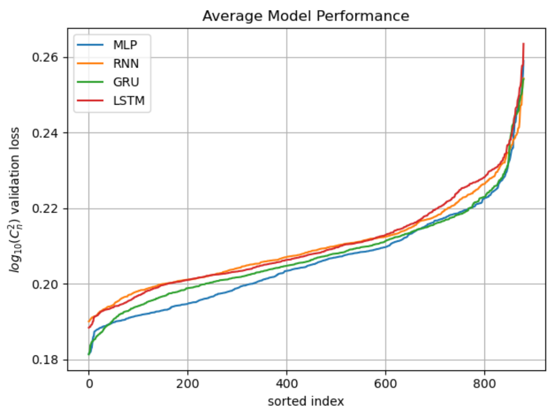
If the average, $\bar{x}$ in Equation 4.1, were calculated many times with different sets of sampled data the values of $\bar{x}$ would themselves have a standard deviation. This is called the *standard error* of the estimated mean $\bar{x}$. Assuming the underlying distribution is Gaussian, the *standard error* is given approximately by

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{N}}, \tag{4.2}$$

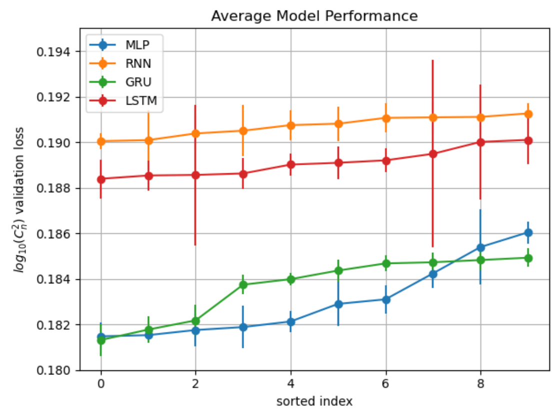
where $\sigma_{\bar{x}}$ is the standard error of the mean, σ is the standard deviation from Equation 4.1, and N is the number of samples used the calculate $\bar{x}$ [13]. In the calculation of the standard error of the average loss scores, N is 10, the number of models trained for each combination of the grid search. From these statistics the best model as applied to the validation dataset is determined and significance quantified.

4.2.1 Results Sorting

The four arrays of average RMSE scores are sorted from best to worst and the sort indices are applied to the array of grid search parameters. From these sorted arrays the best model and its parameters are extracted. Figure 4.1 illustrates the validation average $log_{10}(C_n^2)$ RMSE loss as a function of the sorted index. Figure 4.1a plots all 880 sorted scores for each fundamental architecture. Figure 4.1b plots only the first ten to focus on the best performers. In each plot MLP is drawn in blue, simple RNN in orange, GRU in green, and LSTM in red. The curves in each figure are monotonically increasing because the sorted losses are plotted. Figure 4.1b additionally plots the standard error. The general



(a) Full set of iterations



(b) Top 10 iterations

Figure 4.1: Grid search results.

shape of the sorted loss curves are highly correlated from architecture to architecture and their slopes are consistent from sorted index 100 through 800. On either side, between sorted indices 0 through 100, and 800 through 880, the curves exponentially increase. This is an indication that there are a general set of modeling parameters which are notably better than all the others, and likewise a set that are far worse than the others.

In terms of model performance, the curves in Figure 4.1a generally indicate that over the grid search space the MLP dominates the ensemble of RNN architectures. Throughout the sorted indices, but most importantly at the beginning (left) of the sorted indices, the MLP (blue) and GRU (green) architectures perform better by a large margin compared with the simple RNN (orange) and LSTM (red) architectures. This is further shown in Figure 4.1b which illustrates the first ten sorted indices of Figure 4.1a. The loss curves illustrate that on average the best ten models of the simple RNN is the worst of the four architectures and the best ten models of the LSTM models are a small scale factor better. This result is interesting since the MLP is the baseline architecture and the ensemble of RNNs are designed to handle time series data. Another notable feature of the top ten LSTM models in Figure 4.1b is the large standard error of the mean calculated from Equation 4.2. This is an indication of significant variance in performance over the 10 models trained per grid search iteration. During brief analysis two LSTM models trained on the same parameters could illustrate impressive then poor performance. This high variability is not desirable and leads to high average RMSE, but does reveal the capability of the LSTM if the right set of training parameters can consistently result in a good model.

Further evaluation of the loss curves in Figure 4.1b shows that the best ten MLP and GRU models are very similar, even crossing each other twice. Of the top ten MLP and GRU models, the very best of each (sorted index 0) show that the GRU is slightly better with a validation average $log_{10}(C_n^2)$ of 0.181316 vs. 0.181476 for the MLP. The standard errors are also very similar, 0.000697 vs. 0.000605, for the GRU and MLP, respectively. Thus the consistency of model convergence for one model is not notably better than the other. Generally the standard error bars of the MLP and GRU architectures in Figure 4.1b are smaller (better) than the bars of the simple RNN and especially the LSTM architectures. These results illustrate that of the four architectures, the MLP and GRU are proving to be best suited for this problem as evaluated on the validation dataset.

4.2.2 Statistical Significance

The results presented in Figure 4.1 clearly show that specific architectures and corresponding parameters perform better on the validation dataset than others. However, from the top performing models there is little discrepancy in performance metric which introduces ambiguity into which model and parameter combination is the very best. To sort through these similar models is the *Student's t-test* which is a test of whether two sample means are different to a specific level of significance. Performing tests of significance on the results in Figure 4.1b statistically distinguishes the models to show if a model is significantly better than another.

Student's t-test Foundation

The *Student's t-test* is a statistical test of the null hypothesis H_0 that two samples have equal means. The alternative hypothesis H_a is that samples do not have equal means. The foundation of the test is as follows. Take one set of measurements, then some event happens, then take another set of measurements. Did the event, like a change in a control parameter, make a difference? In this work the measurements are model performances on the validation dataset and the event is a change in the model parameters. There are several variations of the *Student's t-test* including the *independent t-test* for equal and unequal variances, and the *dependent t-test* for paired samples. The *independent t-test* for unequal variances is used in this work because the samples are independent and the variances are not assumed to be equal. This specific test is known as *Welch's t-test* and given samples x_A and x_B defines the statistic t as

$$t = \frac{\bar{x_A} - \bar{x_B}}{\sqrt{\frac{Var(x_A)}{N_A} + \frac{Var(x_B)}{N_B}}}, \tag{4.3}$$

where $\bar{x_A}$, $Var(x_A)$ and N_A are the sample A mean, variance and size, respectively. Likewise, $\bar{x_B}$, $Var(x_B)$ and N_B are the sample B mean, variance and size. The two-tailed

p-value or significance of this value of t is calculated with dof degrees of freedom

$$dof = \frac{\left[\frac{Var(x_A)}{N_A} + \frac{Var(x_B)}{N_B}\right]^2}{\frac{[Var(x_A)/N_A]^2}{N_A-1} + \frac{[Var(x_B)/N_B]^2}{N_B-1}}. \tag{4.4}$$

The p-value is a number between zero and one and is the probability that $|t|$ (hence two-tailed) could be this large or larger just by chance under the assumption that the null hypothesis H_0 is correct [13]. A very small p-value (≤ 0.05) means that the observed difference in means is very significant and the null hypothesis H_0 is rejected at the 5% significance level. This does not, however, prove that the null hypothesis H_0 is false or the alternative hypothesis H_a is true. A low p-value means *either* that the null hypothesis is true and a highly improbable event has occurred *or* that the null hypothesis is false.

Student's t-test Results

The *Student's t-test* described in above is robustly implemented as a function from *SciPy*, a Python-based open-source software for mathematics, science, engineering, and most importantly in this case, statistics [14]. Using the *ttest_ind* function and setting parameter *equal_var* to False performs the *Welch's t-test* given two arrays of measurements.

The *t-test* is performed on each of the top ten models in Figure 4.1b where sample A in Equations 4.3 and 4.4 is the 10 validation $log_{10}(C_n^2)$ RMSE scores from the very best GRU model. Sample B in Equations 4.3 and 4.4 iterates through the 10 validation RMSE scores of the rest of the models in Figure 4.1b. This results in p-values of the best GRU model evaluated against the next nine best GRU models and the best ten MLP models, simple RNN models, and LSTM models. A significance level (α) of 0.05 is predefined to reject the null hypotheses H_0 if the $p-$value is $\leq \alpha$. Figure 4.2 summarizes these results by plotting the two-tailed p-value as a function of the sorted index, the same x-axis as in Figure 4.1b. The p-values are again parsed by fundamental architecture. The MLP p-values are in blue,

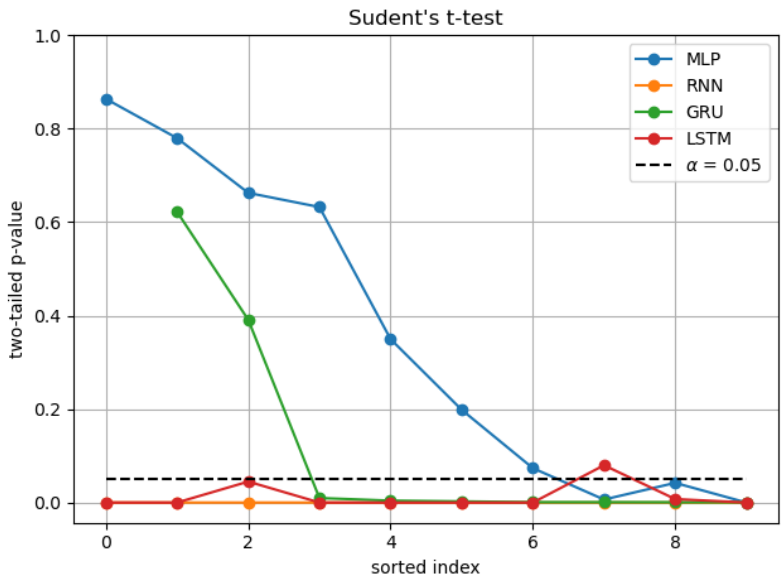


Figure 4.2: *Welch's t-test* of the grid search results

simple RNN in orange, GRU in green, and LSTM in red. The black dashed line in Figure 4.2 is the p-value threshold $\alpha = 0.05$. The GRU p-value at sorted index 0 (zero) is omitted because performing the *Welch's t-test* on the best GRU model against itself is irrelevant.

Any markers below the black dashed line (α) in Figure 4.2 represent models whose mean performance is statistically different from the best GRU model at the 5% level, a rejection of the null hypothesis H_0. Since the best GRU model is the best performing model in the entire grid search, the models below the α line are statistically worse at the 5% level. Any markers above the α line represent models whose performance is not statistically worse, i.e., the null hypothesis H_0 is not rejected at the 5% level. A total of ten markers in Figure 4.2 are above the α line: two are the next two best GRU models, seven are the seven best MLP models, and one is the eighth best LSTM model. The LSTM model which does not reject the null hypothesis is a result of the high variance in the model illustrated by the large standard error bars in Figure 4.1b at sorted index 8 on the x-axis. Likewise, the standard error bars at sorted index 3 in Figure 4.1b also correspond with the p-value that is nearly to the α line in Figure 4.2.

The other markers above the α line from MLP and GRU are consistent with the average RMSE scores in Figure 4.1b. The first three GRU scores in Figure 4.1b hover around 0.182 with the first three MLP scores. Then the GRU scores step up to nearly 0.184 and steadily increase. The MLP scores steadily rise until the seventh sorted index where there is a notable step above 0.184. The indices of these steps, three for GRU and seven for MLP, also are the first models where the null hypothesis is rejected at the 5% level. This is illustrated in Figure 4.2 by the GRU (green) marker at sorted index 3 being the first GRU model below the α line and the MLP marker at sorted index 7 being the first MLP model below the α line.

4.3 Evaluation of Best Models

4.3.1 Comparative Analysis

From the results in Figure 4.2, the top three GRU models and top seven MLP models are selected for further evaluation because they fail to reject the *Welch's t-test* null hypothesis. The LSTM model which also does not reject the null hypothesis is not considered for further evaluation because its rejection is explained by the very high standard error. Mentioned above, the indices to sort the validation average $log_{10}(C_n^2)$ RMSE scores are used to sort the corresponding grid search parameters. Table 4.1 summarizes the grid search parameters associated with the top three GRU models. The three rows in Table 4.1 (ignoring the header) represent the best GRU model (GRU 1), then the next two best GRU models (GRU 2 and GRU 3). The columns in Table 4.1 sort the grid search parameters into

Table 4.1: Top GRU Model Parameters

Model	Sequence Features	Sequence Length (hours)	Layers	Nodes
GRU 1	Press, RH, SI, C_n^2	12	2	40
GRU 2	Press, RH, SI, C_n^2	12	2	50
GRU 3	Press, RH, SI, C_n^2	12	2	30

"Sequence Features" which is the input sequence features (variables) used by the model, "Sequence Length (hours)" that is the length of input sequence data used by the model and is reported in number of hours, "Layers" which is the number of layers used by each model, and "Nodes" which is the number of nodes per layer in each model.

The summary in Table 4.1 is remarkably consistent. The input sequence features used by all three of the best models are pressure, relative humidity, solar irradiance and C_n^2, the sequence length of each model is 12 hours, the number of GRU layers for each model is two, and the number of nodes per GRU layer is 40, 50, and 30 in order of the top three GRU models. The consistency of the parameters from model-to-model indicates a specific type of model has been found to best perform on the validation dataset. It is also encouraging that the best GRU model uses 40 nodes per layer, then 50, and finally 30. If the best GRU used 50 nodes, for example, a concern would arise of whether the bounds of the grid search were wide enough, i.e., the number of nodes covered in the grid search have gone higher than 50.

Similarly to Table 4.1, Table 4.2 summarizes the grid search parameters for the top seven MLP models. Evaluation of Table 4.2 again yields consistency of the best MLP models. For each of the models the input sequence features are temperature, pressure, relative humidity, and solar irradiance. The input sequence length is 16 hours for the first four models, then

Table 4.2: Top MLP Model Parameters

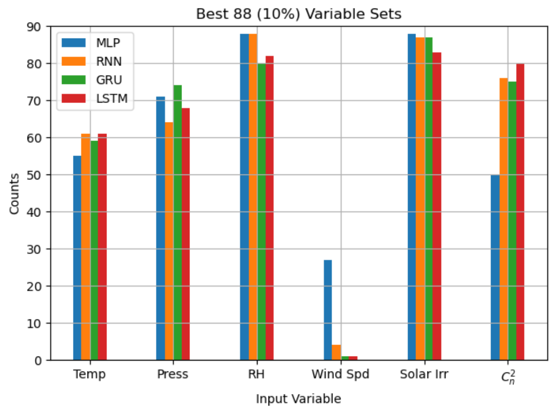
Model	Sequence Features	Sequence Length (hours)	Layers	Nodes
MLP 1	Temp, Press, RH, SI	16	2	30
MLP 2	Temp, Press, RH, SI	16	2	40
MLP 3	Temp, Press, RH, SI	16	2	50
MLP 4	Temp, Press, RH, SI	16	2	20
MLP 5	Temp, Press, RH, SI	12	2	40
MLP 6	Temp, Press, RH, SI	12	2	30
MLP 7	Temp, Press, RH, SI	12	2	50

12 hours for the last three. The number of hidden MLP layers is two for all seven models. Finally, the number of nodes is 30, 40, 50, then 20 for the 16 hour sequence lengths (first four models), then 40, 30, and 50 for the 12 hour sequence lengths (last three models).

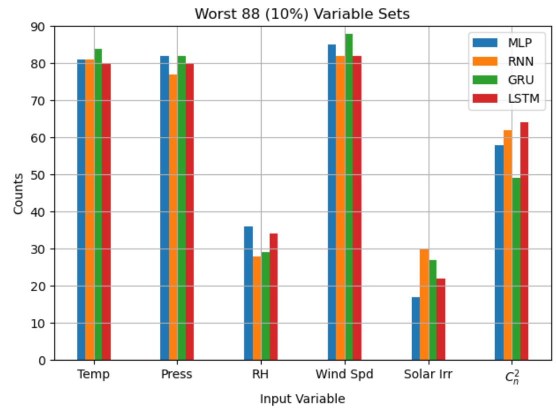
The consistency of input sequence features in the best MLP models is encouraging because like the best GRU models, a specific type of MLP model has been found to perform best on the validation dataset. It's also encouraging that the number of hidden layers in the MLP model is consistently 2 across all seven models. The ordering of the sequence lengths and number of nodes per hidden layer in Table 4.2 illustrates that using 16 hours of input sequences and at least 20 nodes per layer is better than using 12 hours of input sequences and at least 30 nodes per layer. Most importantly, though, is shows that that using 12 hours of input sequences and at least 30 nodes is better than 16 hours of input sequences with only 10 nodes. This is an indication that using only 10 nodes per layer in the MLP likely does not yield the model complexity to perform well on the validation dataset.

There are two major distinctions between the best GRU and MLP models summarized in Tables 4.1 and 4.2. First, the input sequences each only use four features (variables), but the features used by each architecture are different. Both architectures use pressure, relative humidity, and solar irradiance, but the GRU architecture uses C_n^2 and MLP uses temperature. It's interesting that the use of temperature, one of the most fundamental weather variables and oscillatory in character, is not used by the best GRU models. It's more interesting, however, that the best MLP models do not use prior C_n^2 measurements as an input. It's expected that the prior C_n^2 measurements would be a significant driver of the model outputs since generally the trend of recent measurements would continue in the future, so it's curious that the best MLP models deviate from this expectation. The second major distinction is the input sequence lengths. The best GRU models only require 12 hours of inputs, whereas the four best MLP models require 16 hours of inputs. From a practical perspective, requiring four less hours of input sequences is highly desirable in a real-world application.

In addition to the different input features used by the best GRU and MLP models, another curious result is that only four input features are used by each architecture when up to six variables are available. To investigate further, the "Sequence Features" column in Tables 4.1 and 4.2 are expanded to evaluate more models, the best 88 (10%) and worst 88 (10%). The simple RNN and LSTM are also included in this analysis. The number of times each input feature (variable) is used by the best 88 and worst 88 models is recorded per architecture. Figure 4.3 illustrates bar charts of these results. Each category on the x-axis is an input feature, and the y-axis is the number of counts. There are four bars for each category representing the fundamental architectures. The blue bars are MLP, orange bars are simple RNN, green bars are GRU, and red bars are LSTM. Figure 4.3a is the counts of each input feature for the best 88 models of each architecture, and Figure 4.3b is the counts for the worst 88 models of each architecture.



(a) Best 10%



(b) Worst 10%

Figure 4.3: Best 10% and worst 10% variable sets.

Evaluation of best 88 models of each architecture in Figure 4.3a illustrates several interesting results, but one stands out in particular: wind speed is rarely used as an input feature in the best 10% of models. Specifically, the ensemble of RNN architectures almost never use wind speed: the simple RNN uses wind speed four times, and the GRU and LSTM models only use wind speed once each. The best 88 MLP models, however, use wind speed just under 30 times. Concurrent evaluation of the worst 88 models of each architecture in Figure 4.3b further reveals information about wind speed: all four fundamental architectures use wind speed in nearly all of the worst 88 models. The ensemble of RNNs almost universally ignoring wind speed in the best models, combined with the almost universal inclusion in the worst models, is strong evidence that the exclusion of wind speed as an input feature for the best GRU and MLP models is not by chance. Rather, this is conclusive insight that prior wind speeds are not beneficial to the forecasting of C_n^2 in this problem. It is likely that this is due to the large spatial gap between the weather station and C_n^2 path.

For the MLP, the wind speed counts in Figures 4.3a and 4.3b offer some insight into how the MLP is learning compared with the ensemble of RNN architectures. The processing of the RNNs essentially forces the model to learn the input sequences as a function of time, whereas the MLP, which uses the same information in a flattened layer, is free to learn any relationships. Specifically the MLP is not bounded to learn the temporal relationship between the input variables, thus is is reasonable to theorize that the MLP is more prone to learning relationships that are not as physical.

Two more categories of input features stand out in Figures 4.3a and 4.3b: relative humidity and solar irradiance are nearly universally used in the best 88 models of each architecture, and are seldom used in the worst 88 models of each architecture. This result reveals the opposite conclusion about wind speed: relative humidity and solar irradiance are strong drivers of C_n^2 and the spatial separation between weather measurements and C_n^2 path does not significantly impact the models' performances, i.e., these two variables are reliably transferred across space.

Temperature and pressure in Figures 4.3a and 4.3b are two input features whose counts require careful evaluation. In the best 88 models, temperature has an average around 57 - 58 counts across the four architectures, but has 80+ counts for each architecture in the worst 88 models. Similarly, pressure has around 68-69 counts on average in the best 88 models, and around 80 counts on average in the worst 88 models. The counts in the best 88 models suggest that pressure is not as sensitive to the problem as relative humidity and solar irradiance, and temperature is even less sensitive. The counts in the worst 88 models initially suggest that temperature and pressure are generally poor input features, but the very low counts of relative humidity and solar irradiance influence that conclusion, possibly incorrectly. Because a minimum of four input variables are required by the grid search's bounds, the low counts for relative humidity and solar irradiance already account for the maximum two dropped variables, which means that temperature and pressure by default will often get used as an input to the worst 88 models. This argument is not valid for wind speed because the counts are extremely low in the best 88 models *and* the counts are extremely high in the worst 88 models. As a final note about temperature and pressure, if the minimum number of input features was set to three instead of four it's possible they would be entirely excluded from the best GRU models based on the counts in Figure 4.3a, leaving only relative humidity, solar irradiance, and C_n^2.

The final input feature of discussion in Figures 4.3a and 4.3b is C_n^2. This is another input feature (wind speed) in which there is a strong difference between MLP and the ensemble of RNN architectures. For the best 88 models, C_n^2 is utilized 75+ times by the ensemble of RNN architectures, but only 50 times by the MLP. Most importantly, though, is that *none* of the best seven MLP architectures from Table 4.2 use C_n^2. This is further evidence of the fundamental difference between the processing of the ensemble of RNNs and the MLP. It's expected that prior C_n^2 measurements is a strong driver of future C_n^2 and the ensemble of RNNs strongly learn this relationship. The MLP has not learned this relationship as strongly possibly because C_n^2 is only relevant to forecasting C_n^2 if it is processed temporally.

The counts in Figures 4.3a and 4.3b have been shown to behave as a top-level input sensitivity analysis. Many useful conclusions have been derived about how the architectures use the inputs. Further evaluation would require a significant level of time-series analysis which is beyond the scope of this effort.

4.3.2 Selection of the Best Model

The results presented in this Chapter have shown the best performing model as applied to the validation dataset. By *Welch's t-test* it has been shown to be significantly better than all but nine other models at the 5% level. The best model (GRU) has been evaluated against the next two best GRU models to show strong consistency in the model parameters. This same evaluation and result has been shown for the seven MLP models. The next step in this work applies a model to the test dataset so a single model is chosen. This single model is "GRU 1" in Table 4.1. It has two GRU layers with 40 nodes each and uses 12 hours of pressure, relative humidity, solar irradiance, and C_n^2 as input features. This model is chosen for many reasons.

1. It results in the lowest average RMSE as applied to the validation dataset.

2. The next two best GRU models, which are not significantly different in performance, share the same input features, input sequence length, and number of layers.

3. This model uses 40 input nodes and the next two best GRU models, which are not significantly different in performance, use 50 and 30 nodes which shows strong consistency.

4. This model requires only 12 hours of input sequences compared with MLP's requirement of 16 hours for the best models. In a real-world application this is highly desirable.

In addition to the reasons to use the selected GRU model, no evidence has been shown to support the use of an MLP model instead.

4.4 Chapter Summary?

CHAPTER V

Test Dataset Model Evaluation

5.1 GRU Test Dataset Performance Summary

After determining the best model in Chapter IV as applied to the validation dataset, the train and validation datasets are combined into an updated train dataset. The selected model is trained on the updated train dataset and then applied to the test dataset. Like in the grid search, 10 individual models are trained to evaluate convergence. Figure 5.1 illustrates the model training process by plotting the $log_{10}(C_n^2)$ RMSE loss (evaluated between forecast and measured truth) as a function of training epoch. The blue curves

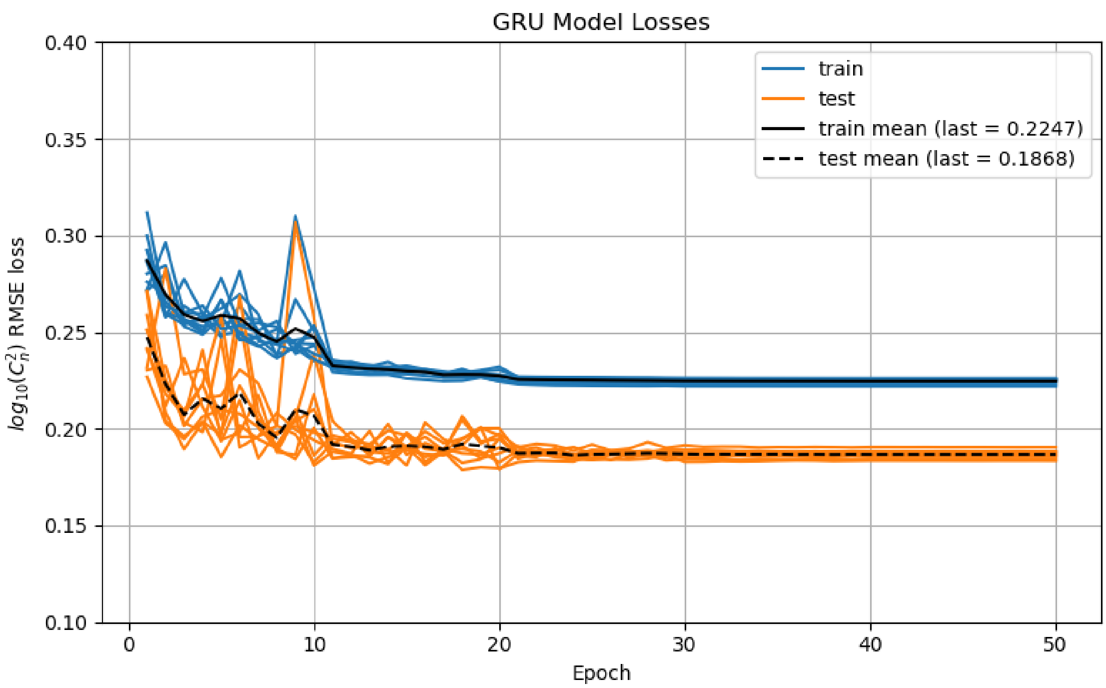


Figure 5.1: GRU train and test loss curves of 10 models.

represent each model's performance on the train dataset and the orange curves represent the performance on the test dataset. The solid and dashed black lines are the per-epoch average of the train and test dataset performances, respectively. The loss curves in Figure 5.1 indicate that convergence rates of the models as applied to the train and test datasets are consistent with each other, an indication that the test dataset is a good representation of the train dataset. This is further illustrated by bumps in the loss scores in both train and validation loss curves like the single blue and orange spike at epoch 9. The per-epoch variability in the test dataset is higher than the train dataset since the test dataset is much smaller and thus more prone to changes in loss score with an update in model parameters. The 50th (last) epoch loss scores averaged over the 10 models are reported in the legend as 0.2247 and 0.1868 for the train and test dataset, respectively. The standard deviations of the 50th epoch loss scores for the train and test datasets are 0.0013 and 0.0020, respectively. As a point of comparison, the best MLP model was also trained then applied to the test dataset 10 times. The average 50th epoch loss scores are 0.2457 and 0.1938 with standard deviations of 0.0027990 and 0.0020191 for the train and test dataset, respectively. These results indicate that as applied to the test dataset, on average the chosen GRU model is better than the best MLP model by a large margin, further validating the selection of the GRU model. The standard deviations of the test dataset loss scores are nearly identical, indicating the stability of the models are similar.

Another way to visualize model convergence is to apply the ten individually-trained models to the test dataset parsed by day. The test dataset spans 08/03 and 08/05 - 08/10, so each model is evaluated on these seven days. The $log_{10}(C_n^2)$ RMSE loss scores as a function of test day are illustrated in Figure 5.2. Evaluated over the ten models, in black is the mean, red the mean $\pm$ the standard deviation, and blue the minimum and maximum scores. The day-to-day change in average error score, illustrated by the large jumps in

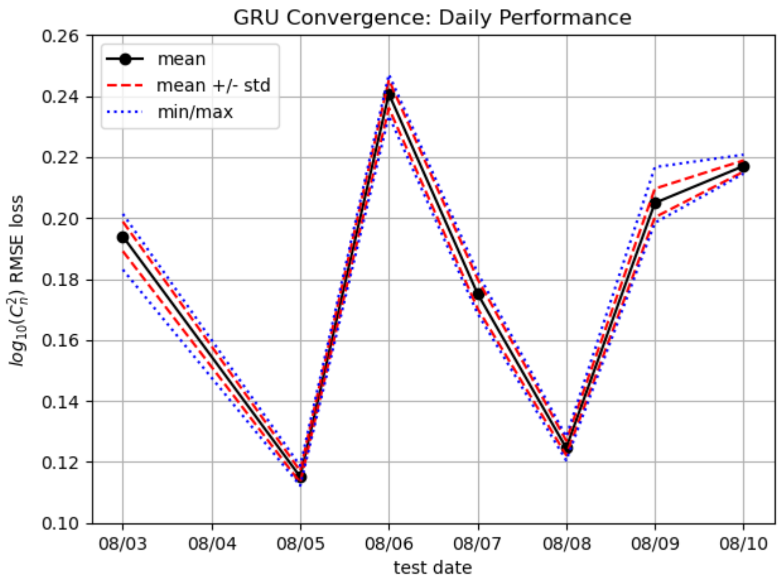


Figure 5.2: GRU daily summary performance: mean, standard deviation, and min/max.

error score like from 08/05 to 08/06, indicate that model performance varies by daily C_n^2 conditions. However, the tightness of the red and blue curves about the black markers indicate the models are converging to consistent solutions when evaluated on a daily basis. The largest difference between minimum and maximum loss scores is just less than 0.02 on 08/03 which is still a small variation in model performance for a given day. On 08/05 and 08/08 the red and blue lines are on top of the black markers indicating the model convergence is highly consistent when applied to these two days. These day-by-day results further confirm the test dataset loss curves in Figure 5.1 which indicate the models converge to a consistent solution when applied to the entire test dataset.

Another visualization of average model performance from the 10-model ensemble is presented in Figure 5.3 which is the $log_{10}(C_n^2)$ RMSE loss score of the test dataset parsed by the first timestep in each forecast. To make the plot in Figure 5.3, the 148 forecasts in the test dataset are sorted by the time of day of the first timestamp in each 4-hour forecast, the 30-minute forecast. The loss scores of each model applied to every test dataset forecast whose timestamp corresponds with a given time of day is plotted as the black markers. The average of the 10 loss scores are plotted as red markers for each test dataset forecast whose first timestamp is at the given time. For example, evaluating the 06:00 (far left) time of

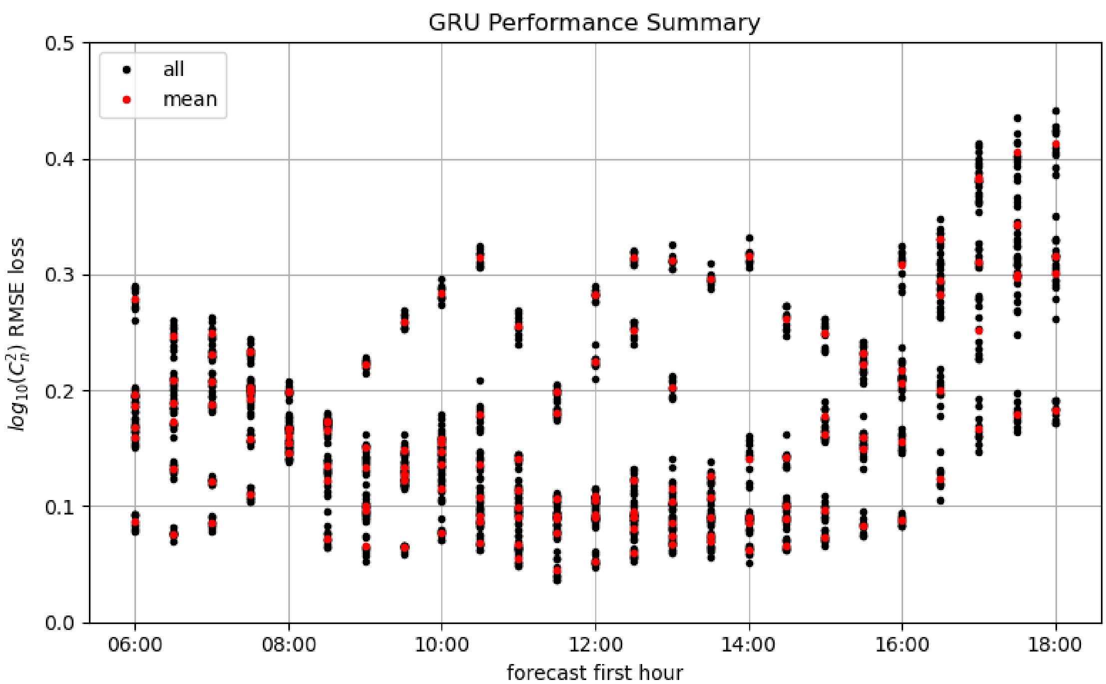


Figure 5.3: GRU hourly summary performance: individual and average.

day in Figure 5.3 shows six red markers. This means that in the test dataset there are six forecasts whose first timestamp is at 06:00, and each red marker is the average loss (from the 10-model ensemble) for each of the six forecasts. The black dots at 06:00, of which there are 60 markers, are the model-by-model loss scores for each of the six forecasts. Still evaluating the 06:00 time of day, there is one red marker around 0.275 on the y-axis that is on top of a cluster of black points. This individual red marker is the average of the black markers which surround it. Thus, this cluster of black markers and their average, the red marker, is an evaluation of the ensemble model performance on one of the six forecasts whose first timestamp is at 06:00. Overall, Figure 5.3 shows 148 red markers for the 148 forecasts in the test dataset, and 1480 black markers for the 10 models per 148 forecasts.

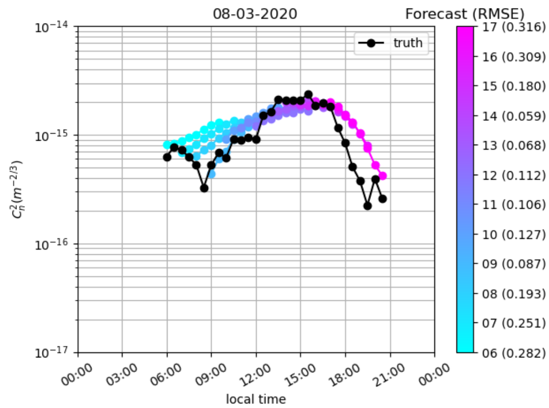
Figure 5.3 is a summary of model performance as a function of time of day. The trends indicate that best model performance starts around 0.2 (on the y-axis) in the beginning of the day, drops down to around 0.1 in the middle of the day, then rises in the evening to around 0.3. The morning forecasts trend to be around the overall test dataset performance, about 0.18 to 0.20. On average, best performance is in the middle of the day between 11:00 and 15:00 since most of the black and red markers are clustered around 0.1 on the y-axis. In this window a few forecasts are consistently outliers (indicated by the red markers also being an outlier from the trend) in terms of error score. These individual forecasts are of interest for analysis. The high error scores at the end of the day, 16:00 to 18:00, is indicative that there are features of these late-day forecasts which are present in the measurements but are consistently not being captured by the ensemble of models. Generally, Figure 5.3 indicates the model performs best in the middle of the day, slightly worse in the morning, and generally poorly in the late afternoon and evening.

5.2 Daily C_n^2 Forecasts

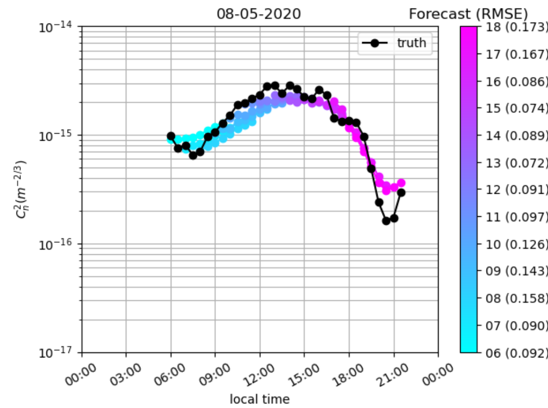
After the ensemble study of the GRU models applied to the test dataset, a single GRU model is applied to the test dataset and carefully evaluated in this section. Figures 5.4 and 5.5 are daily-C_n^2 plots of the test forecasts and measured truth as a function of local time. Figures 5.4a, 5.4b, 5.4c, and 5.4d plot the forecasts on 08/03 and 08/05 - 08/07, respectively. Figures 5.5a, 5.5b, and 5.5c plot the forecasts on 08/08 - 08/10, respectively.

Each daily-C_n^2 plot contains multiple curves. The black curve represents the truth C_n^2 measured conditions. The other curves which transition in color from cyan to magenta represent the model forecasts throughout the day. The brightest cyan is the earliest forecast in the day, and the brightest magenta is the latest forecast in the day. The forecasts in between are appropriately colored to smoothly transition between the two end colors. The colorbar of each plot in Figures 5.4 and 5.5 illustrate the curve color scheme by labeling the forecast's first timestamp next to the color with which it's associated. Additionally, the $log_{10}(C_n^2)$ RMSE loss score between each forecast and measured truth is also labeled in parenthesis next to the colorbars. For example in Figure 5.4a, the earliest forecast's (furthest left) first timestamp is at 06:00 and is correspondingly labeled in the colorbar as 06. The color of this earliest curve is brightest cyan which is consistent with the label on the colorbar. The error score for this forecast is 0.282. Likewise, the last forecast of the day in Figure 5.4a is at 17:00 and is the brightest magenta curve. The colorbar label is 17 and is the top color in the colorbar. The loss score for this individual forecast is 0.316. Note that in these daily-C_n^2 plots only the forecasts whose first timestamp is at the top of the hour are illustrated. This is done purely for aesthetics to avoid cluttered plots.

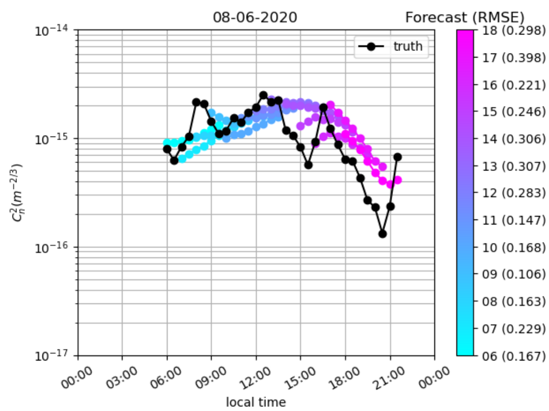
The daily-C_n^2 plots in Figure 5.4 illustrate many unique features. Starting with 08/03 in Figure 5.4a, the morning (cyan) forecasts all forecast a similar trend in C_n^2: a steady rise. Interestingly, the magnitude of the forecasted conditions seems to be different by a scale-factor between the first three forecasts at 06:00 - 08:00. This illustrates the impact of including prior C_n^2 conditions as an input sequence into the model. At the 07:00 and 08:00 forecasts, the model is given information that the most recently measured C_n^2 is trending down in magnitude so the forecasts, while still trending upward, lowers in overall magnitude because the model has learned that its forecasts should start near the most



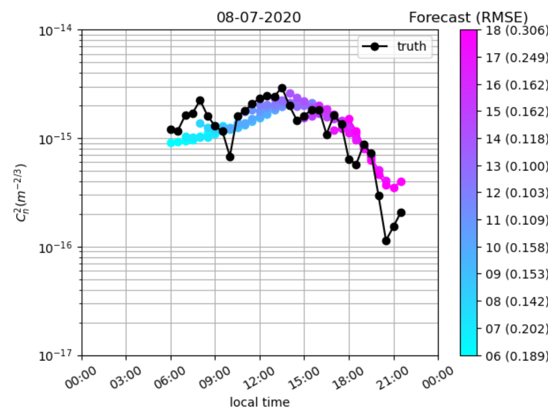
(a) August 03



(b) August 05



(c) August 06



(d) August 07

Figure 5.4: August 3 and 5 - 7 daily C_n^2 forecasts.

recently measured conditions. This feature is consistently seen throughout all test forecasts. Another interesting feature is the evening forecasts in Figure 5.4a. The 15:00 - 18:00 model forecasts predict C_n^2 to significantly lower in magnitude starting around 18:00 until 20:30. The measured conditions significantly lower, but temporally earlier than predicted. This is a rare case where the model gets the evening neutral event magnitude mostly right but misses the temporal component. Generally, the model on this day forecasts a mostly standard diurnal trend, but the measurements are weather impacted. At 10:00 there is a slight drop in measured turbulence, then from 10:30 - 12:00 the measured conditions are steady around $9 \times 10^{-16}(m^{-2/3})$. A standard diurnal trend occurs from 12:30 to 17:00, and the model accurately forecasts this.

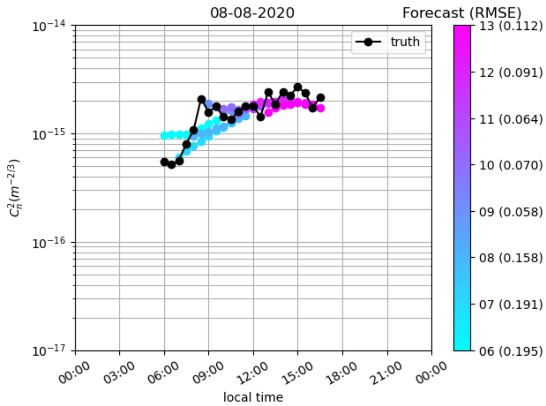
Forecasts on 08/05 in Figure 5.4b are consistent with a standard diurnal trend. Specifically, the forecasts accurately predict the weak morning neutral event indicated by the drop in C_n^2 around 07:00 to 08:00 then a steady rise in strength until 12:00. The forecasts predict slightly lower C_n^2 strength than measured, but the difference is small. The forecasts then accurately predict the drop in C_n^2 strength starting around 15:00. These forecasts are accurate temporally and in strength of neutral event until the deepest part around 21:00. The measured C_n^2 strength continues to drop but the model does not reach this depth. However, the model does accurately forecast the slight rise in C_n^2 at the very end of the day. Every forecast in Figure 5.4b has an error score lower than the error score of 0.1868 from the 10-model ensemble average as applied to the entire test dataset. More specifically, 8 of the 12 forecasts have error scores less than 0.01. These scores indicate that 08/05 is an illustration of great model performance for an entire day.

Model performance on 08/06 in Figure 5.4c is the worst of the seven test days (see Figure 5.2). The first three forecasts of the day at 06:00 - 08:00 have average to above-average error scores because the forecasts predict a steady rise in C_n^2 strength which generally is measured, but a weather-induced rise in C_n^2 strength at 08:00 is not captured. The 09:00 forecast, which is the lowest error score of the day at 0.106, benefits from knowledge of the prior C_n^2 measurements as an input. The forecast starts high, around $2 \times 10^{-15}(m^{-2/3})$, drops down in C_n^2 strength then continues upward in C_n^2 strength with the other forecasts which is accurate with the truth measurements. Forecast performance is poor again starting at 14:00 because another weather-induced turbulence event has occurred, but this time it's a 2-hour drop in C_n^2 strength which the forecasts beforehand do not predict. The 15:00 forecast is the first to see the prior measured C_n^2 conditions which include the weather event and so the model tries to compensate for the suddenly-lower C_n^2 strength by starting lower then rising for a few timestamps to reach the conditions it expects around this time. The 16:00 forecast captures the first timestamp, but misses the very high strength C_n^2 at 15:30, then sharp and consistent drop in C_n^2 as part of the evening neutral event from 15:30 till 20:30. The 17:00 and 18:00 forecasts also miss this long and deep evening neutral event. Overall, 08/06 illustrates where the model struggles: highly-weather impacted conditions.

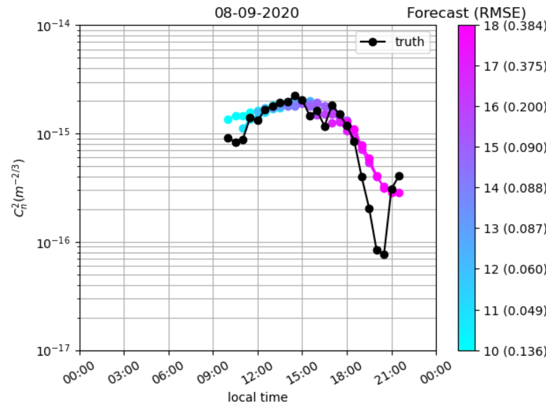
The fourth test day, 08/07 in Figure 5.4d, illustrates a unique combination of forecasts. In the morning, the model forecasts a slow rise in C_n^2 strength until around 13:00 which is generally consistent with measurements, but does not predict the weather-induced rise in C_n^2 at 08:00 then subsequent drop at 10:00. These two events are very short, only consisting of a single timestamp (30 minutes). After this mini-event, the measured conditions are generally diurnal trend through 19:00. There are a few dips in measured C_n^2 strength but they are high frequency and bounce back to a normal diurnal trend which the model accurately forecasts. The forecasts from 11:00 through 14:00 all have error scores below 0.15, well below the average on the entire test dataset of 0.1868 from the ensemble results. The forecasts at 17:00 and 18:00 have high error scores because the model does not accurately predict the extremely sharp drop in C_n^2 strength from $7 \times 10^{-16}(m^{-2/3})$ at 19:30 to about $1 \times 10^{-16}(m^{-2/3})$ at 20:30, only 1 hour later. These late forecasts accurately predict the slight rise in turbulence at 21:00 and 21:30 after the deepest part of the evening neutral event, but the error scores are high because the model could not reach the depth. The 18:00 forecast is a good example of the model accurately forecasting the temporal nature of turbulence conditions but missing the very low magnitude of the neutral event.

The final three test days, 08/08 - 08/10, are presented in Figures 5.5a, 5.5b, and 5.5c, respectively. On average over the 10-model ensemble study, 08/08 is the second best day of model performance. Throughout the day, the model generally predicts diurnal trend: a slow rise in C_n^2 strength until around 15:00 where conditions steady then start to decrease again. The 09:00 through 13:00 forecasts all yield low error scores with 4 of them below 0.01. The first three forecasts from 06:00 through 08:00 have error scores around the ensemble average due to two factors. The 06:00 forecast first predicts turbulence conditions around $1 \times 10^{-15} (m^{-2/3})$, which is the nighttime filled C_n^2 strength, when the measured conditions are actually low. Then the 06:00 forecast altogether misses the weather-induced sharp rise in turbulence conditions from 07:00 to 08:30. The 07:00 and 08:00 forecasts also miss this high-frequency event. It's not until the 09:00 forecast which has the information about the prior turbulence conditions that the forecasts accurately settle into the diurnal trend.

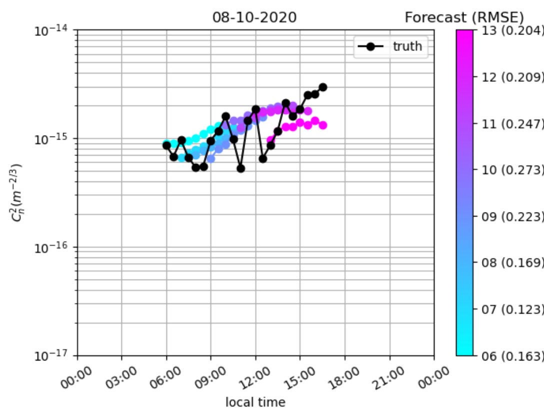
From the 10-model ensemble study, model performance on 08/09 ranks 4/7 on average and is also one of the most variable from model-to-model (Figure 5.2). Analysis of the forecast C_n^2 conditions in Figure 5.5b illustrate how an average over the entire day can be misleading. The first forecast at 10:00 has an error score of 0.136 which is below the 10-model overall average, then the 11:00 - 15:00 forecasts each boast error scores less than 0.01.The measured conditions are mostly diurnal and is accurately forecasted by the model. The only deviations are the very slight morning neutral event from 10:00 to 11:00 then a weather-induced slight drop in C_n^2 strength around 16:00. The two forecasts which raise the day's average error score are at 17:00 and 18:00. These forecasts accurately predict the temporal nature of the evening neutral event, but strongly miss the sharp decline and depth of the event. From 18:00 to 20:00, C_n^2 strength drops from just over $1 \times 10^{-15} (m^{-2/3})$ to $8 \times 10^{-17} (m^{-2/3})$, over 1 order-of-magnitude change. In fact, this evening neutral event is



(a) August 08



(b) August 09



(c) August 10

Figure 5.5: August 8 - 10 daily C_n^2 forecasts.

just one of only a handful of cases in the entire dataset in which the evening neutral event dips below $1 \times 10^{-16}(m^{-2/3})$ (count number of individual evening neutral events that reach this threshold? I think there are 7 including test dataset from Figure 3.8a). The 17:00 and 18:00 forecasts yield error scores of 0.375 and 0.384, respectively, which are two of the highest error scores in the entire test dataset. Beside this uncharacteristically deep neutral event, model performance on 08/09 is well better than average.

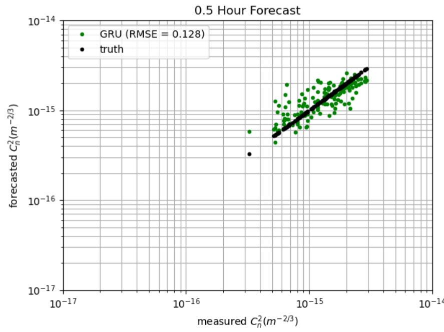
The last day of the test dataset, 08/10, is the second worst day of model performance from the 10-model ensemble average in Figure 5.2. The morning forecasts, which begin at 06:00, generally all predict standard diurnal trends: a slow increase in C_n^2 strength through about 15:00 then a decline. The measured conditions, however, are highly variable both temporally and in C_n^2 strength. Between 06:00 and 15:00 there are five instances of a drop in C_n^2 strength. Likely, at least four of them are weather-induced (possible the 08:00 is morning neutral event). The model forecasts a diurnal trend which follows the trend of the high-strength C_n^2 points over this time period. The 12:00 and 13:00 forecasts predict C_n^2 strength to lower as the afternoon moves into early evening, but the measured conditions actually rise to the highest point in the entire day from 15:30 to 16:30. The 13:00 forecast starts at a lower C_n^2 strength than the surrounding forecasts due to the model having the input sequence information about the very low C_n^2 strength at 12:30. This day again illustrates the the model's inability to accurately forecast high-frequency weather-induced C_n^2 fluctuations. Instead, the model has learned a trend in C_n^2 based on the time of day. The model has also learned to adjust its starting C_n^2 strength based on prior C_n^2 measurements, then settle back into the expected trend.

5.3 Forecast Analysis

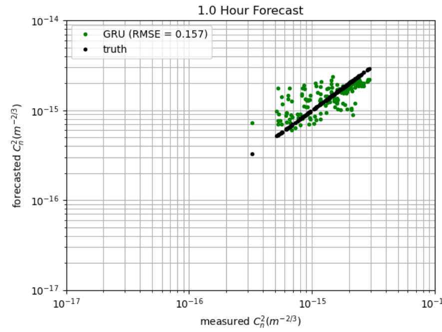
The model forecasts and corresponding truth measurements from Figures 5.4 and 5.5 are rearranged by forecast length. Figure 5.6 presents a scatter plot of measured C_n^2 on the x-axis and forecasted C_n^2 on the y-axis for each time in the 4-hour model forecasts: 0.5 hour, 1 hour, 1.5 hour, and so on through 4 hours. In each scatter plot, the black markers are the truth values plotted as measured C_n^2 vs. measured C_n^2 to yield the linearly arranged markers. The green markers are the model forecast at the scatter plot's designated forecast time plotted as forecasted C_n^2 vs. measured C_n^2. These scatter plots are visually evaluated by how closely the green markers cluster around the black markers. A set that is well clustered around the black points is better than a set that is not well clustered. If the model was perfect, the black markers would directly overlay the green markers. As a quantitative metric of model performance at each forecast hour the average $log_{10}(C_n^2)$ RMSE loss is reported in each legend.

The scatter plot of the first forecast time, 30 minutes, in Figure 5.6a shows a strong clustering of the model forecasts about the measured C_n^2. Excluding one outlier, the range of measured C_n^2 the model is tasked to forecast is only from $5 \times 10^{-16}(m^{-2/3})$ to $3 \times 10^{-15}(m^{-2/3})$, a narrow window. This forecast time also benefits from being closest to the most recently measured C_n^2 and other input sequence variables. Thus, it is expected for this forecast time to do well overall. The RMSE loss score is 0.128, well below the 10-model ensemble average over the whole dataset of 0.1868. The scatter plot of the second time of the model forecasts, 1 hour, in Figure 5.6b shows a slightly worse performance than the 30 minute forecast time. The set of measured points looks very similar to that of 30 minutes, but the clustering of the green model markers about the black truth markers is not as tight. The error score of 0.157 confirms this visual analysis.

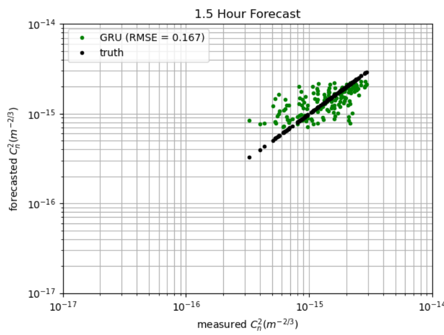
The 1.5 and 2 hour forecasts in Figures 5.6c and 5.6d start to clearly show a different set of measured C_n^2 conditions illustrated by the additional black markers toward the middle and lower left corner of the scatter plots. Most of the green model markers have a reasonable cluster around the mid- and high-strength C_n^2 black markers. However, the model struggles to capture measurements below $7 \times 10^{-16}(m^{-2/3})$ as illustrated by the black markers continuing left on the x-axis below this strength, but the green markers staying above $7 \times 10^{-16}(m^{-2/3})$ on the y-axis in Figure 5.6c. In Figure 5.6d is a similar trend, but



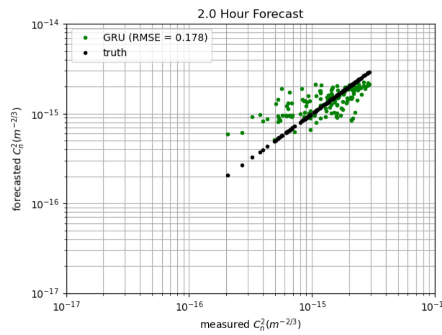
(a)



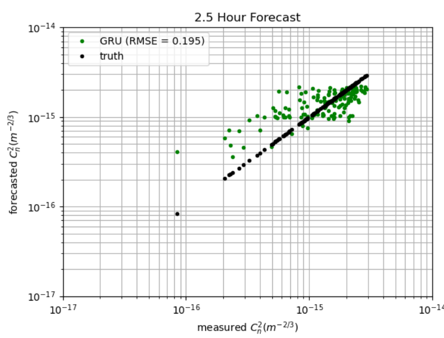
(b)



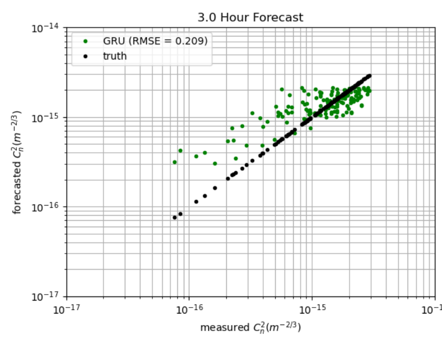
(c)



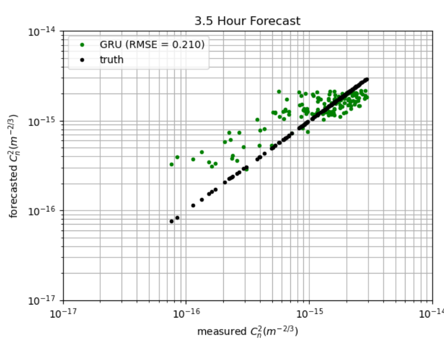
(d)



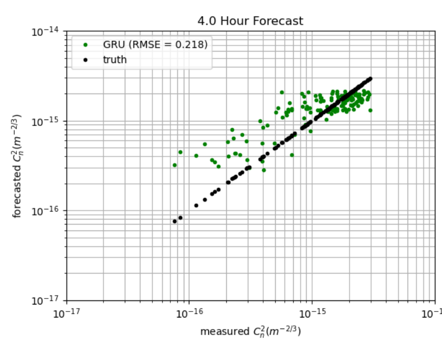
(e)



(f)



(g)



(h)

Figure 5.6: Scatter plots.

a couple of model points are beginning to drop into lower strength C_n^2 conditions. The average error scores continue to climb, reaching 0.167 and 0.178 at the 1.5 and 2 hour forecast times, respectively.

The 2.5 and 3 hour forecasts in Figures 5.6e and 5.6f further shown a different set of measurements the model attempts to forecast. Most notable are the increased number of low-strength C_n^2. As forecast time extends further out the number of evening neutral events included in the examined sets also increases. The clustering of the higher-strength C_n^2 conditions continues to slightly deteriorate. The clustering of the green model markers is not as tight about the black truth markers, and more importantly the slope of these green markers is angling more with respect to the slope of the black markers. At low-strength C_n^2 evening neutral events the model is still struggling to forecast C_n^2 that reaches the measured strength. One positive element of these scatter plots is that there are no green markers which are wildly off from it's corresponding black marker. For example, there are no green modeled markers at $2 \times 10^{-15}(m^{-2/3})$ when it's black marker is nearing $1 \times 10^{-16}(m^{-2/3})$. This means the model has generally learned the turbulence conditions well, understanding when to forecast high, medium, and low-strength C_n^2. The error scores for the 2.5 hour and 3 hour forecasts are 0.195 and 0.209, respectively, which follows the trend of increasing error with forecast length.

Finally, the 3.5 and 4 hour forecasts in Figures 5.6g and 5.6h continue to illustrate the trend shown so far: the set of measured C_n^2 to forecast further evolves to include more evening neutral event low-strength conditions. Interestingly, the group of green model markers around the higher-strength C_n^2 is very well clustered with itself, but is at a significant angle with respect to the black truth markers' angle. The model's struggle to capture the deep neutral events is most clearly observed in Figures 5.6g and 5.6h. The model still has learned when to forecast low-strength C_n^2, but cannot reach the required depth. The average loss scores for the 3.5 and 4 hour forecasts are 0.210 and 0.218, respectively.

Summarizing the scatter plot error scores from Figure 5.6 is Figure 5.7 which plots the loss scores as a function of forecast length. The plot illustrates that as the forecast length increases so does the model performance error. This is an expected result as any forecasting application is expected to decline in performance as forecast time increases.

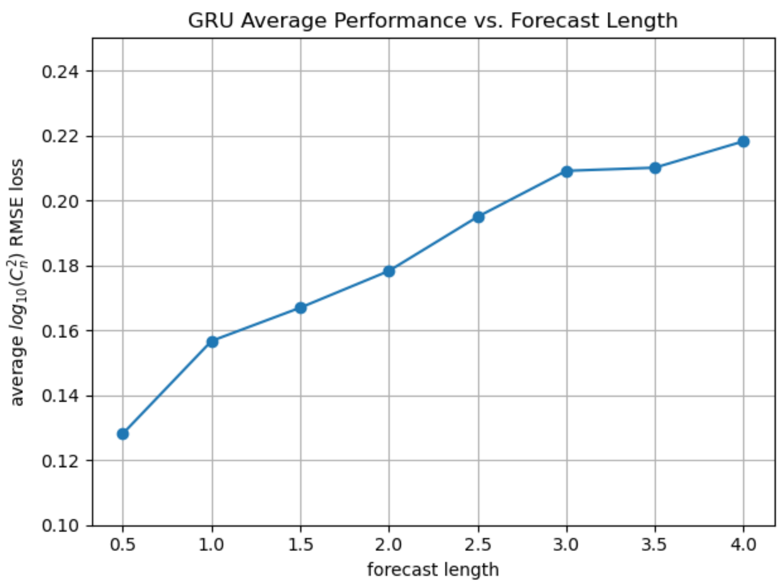


Figure 5.7: Average performance as a function of forecast length.

5.4 Individual Forecast Analysis

The final analysis of this model is to understand why it performs well and poorly in specific cases. The section uses the cumulative distribution function (CDF) to statistically illustrate the model's strengths and weaknesses. The CDF is a tool which sorts the given data by magnitude then assigns each sorted data point a percentile from zero (noninclusive) to one (inclusive). The percentile represents the percentage of data points that are less than or equal to the data point's magnitude.

5.4.1 2020/08/09 18:00

A single trend has been abundantly clear throughout the model analysis: the model struggles to capture the deep evening neutral events. In Section 5.2, Figure 5.5b, one of the worst model forecast in terms of RMSE (0.384) is the 08/09 18:00 forecast. This is the latest forecast of the day (and all days) and the deep neutral event was noted above to be one of the deepest in the entire dataset, making it an excellent candidate for further analysis which goes as follows.

First, every forecast and input sequence from the train dataset is gathered whose first timestamp matches the forecast in question, 18:00. There are a total of 40 train forecasts whose first timestamp is 18:00. Next, the 40 train forecasts are sorted by measured (truth) C_n^2 magnitude at each timestamp in the 4-hour forecast, so in this case that yields a 40 x 8 should this be in non-plain text? array where each column is sorted by magnitude. The CDF is then computed per timestamp, yielding eight CDFs in this case where percentiles are associated with specific C_n^2 strengths. Finally, the CDFs of C_n^2 values are interpolated to subjectively selected percentiles of 20%, 50%, and 80% for each timestamp, yielding a 3 x 8 array of C_n^2 values. The next step in this analysis applies the trained model to the 40 input sequences associated with the forecasts just evaluated to CDF 20%, 50%, and 80%. The exact same process just described - sorting the C_n^2 values per timestamp, calculating the CDF, then interpolating to the desired percentiles - is performed on the 40 model forecasts.

Figure 5.8 illustrates the results of this analysis. C_n^2 is plotted as a function of local time of day, where the only forecasts considered are those whose first timestamp is at 18:00.

The black markers represent the train dataset truth (measured) C_n^2 at each timestamp, thus at 18:00, 18:30, and so on through 21:30 there are 40 black markers each. The solid blue, orange, and green lines with circle markers represent the CDF 20%, 50%, and 80%, respectively, of the truth C_n^2 at each timestamp. The dashed blue, orange, and green lines with "X" markers represent the CDF 20%, 50%, and 80%, respectively, of the model forecast C_n^2 at each timestamp. Finally, the solid red and purple lines with "+" markers represent the truth C_n^2 and model's prediction of the test forecast whose first timestamp is 08/09 18:00.

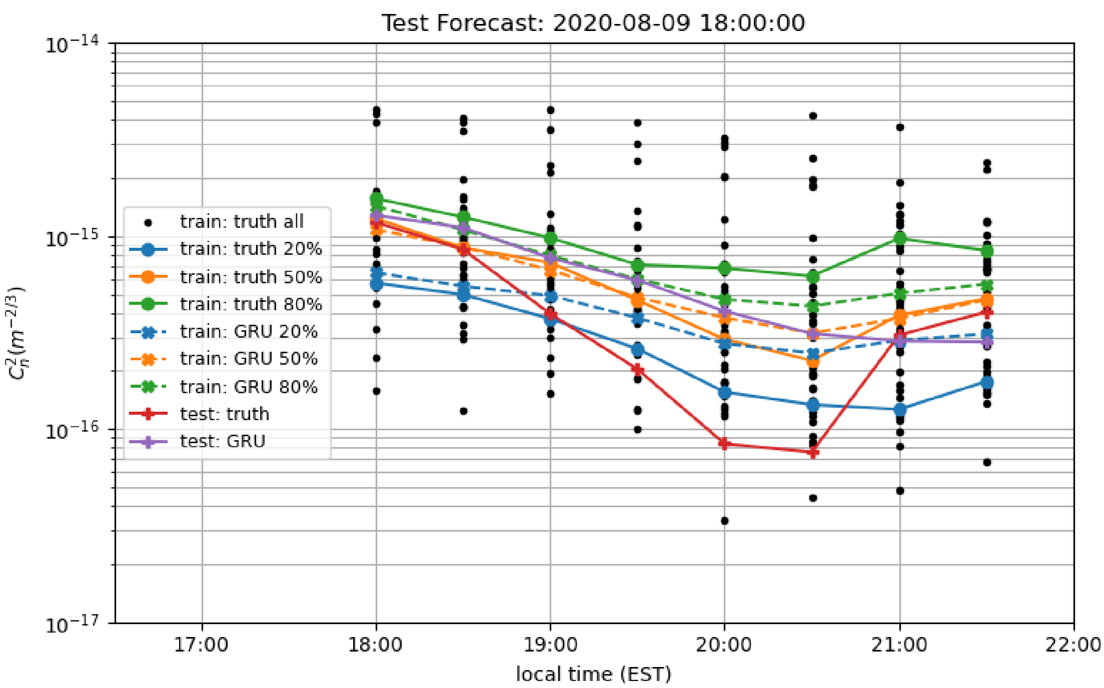


Figure 5.8: GRU performance analysis of the 08/09 18:00 forecast.

The analysis in Figure 5.8 reveals that the model has generally learned the ability to forecasting C_n^2 from prior weather measurements very well, but lacks the ability to forecast extreme scenarios like the deep neutral event on 08/09. This is illustrated in the plot by comparing the solid and dashed lines of the same colors. Comparing the solid and dashed orange lines, which represent the CDF 50% of the truth and model forecasts, shows that the model has learned the relationship between input sequences and output forecasts because the two lines are nearly on top of each other. However, the dashed blue line is generally a scale factor above the solid blue line, and similarly the dashed green line is a scale factor below the solid green line. This indicates that even when the model is applied to the train dataset, a biased result, the model cannot forecast very high and low C_n^2 conditions from the measurements. Figure 5.8 also reveals why the model performs so poorly on the 08/09 18:00 forecast. The solid red line which is the truth starts almost at the truth CDF 80% line (solid green) at 18:00 then proceeds to fall below the truth CDF 20% line (solid blue) at 19:30, 20:00, and 20:30. The model's 08/09 18:00 forecast (solid purple line) starts very closely with the truth (solid red line) but does not match but steep decline in C_n^2 strength. Interestingly, the model's forecast does fall to the CDF 20% of the model's output on train data (dashed blue line) but takes the entire 4-hour forecast to do so. This is an indication that the model has learned that C_n^2 is expected to be low, but does not have the ability to forecast the very deep events.

5.4.2 2020/08/06 14:00

Another example of poor model performance on the test dataset is the 08/06 14:00 forecast in Figure 5.4c. This forecast is chosen for further analysis because it is an example of poor performance (RMSE is 0.306) that does not incorporate an entire neutral event. Rather, this forecast is in the middle of the day when C_n^2 trends are typically very diurnal. Figure 5.9 illustrates the same analysis described above. The most revealing aspect of

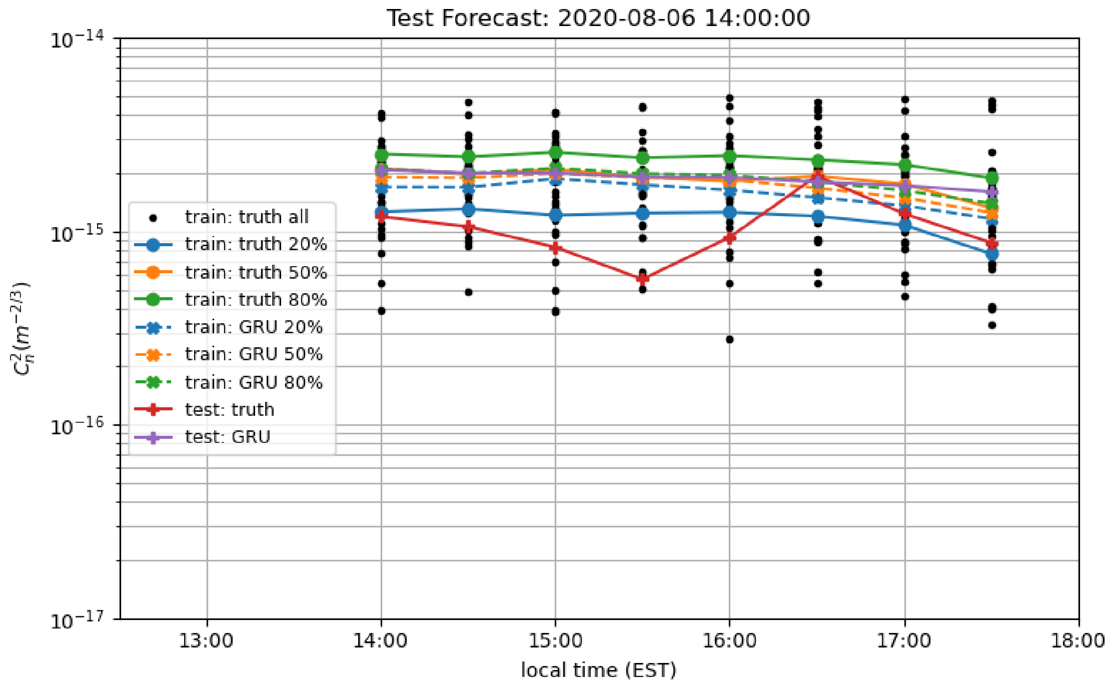


Figure 5.9: GRU performance analysis of the 08/06 14:00 forecast.

Figure 5.9 is that the bounds of the truth CDF 20% and 80% (solid blue and green lines) are very narrow, generally separated by a factor of two. Furthermore, the train dataset model CDF 20% and 80% (dashed blue and green lines) are nearly on top of each other. The proximity of the CDF percentile lines are so close that the CDF 50% lines are covered throughout most of the plot. This evaluation further shows that the model has generalized the C_n^2 conditions similarly to the 08/09 18:00 forecast analysis above. In this case the bounds are especially narrow since the model has learned to generalize and even the bounds of the measured conditions at this time are narrow. Thus, Figure 5.9 clearly shows that the model's performance on the 08/06 14:00 forecast is poor because the measurements throughout this time are in the extremes of all conditions from training. Specifically, the first 5 timestamps from 14:00 through 16:00 of the truth (solid red line) are below the truth

CDF 20% line (solid blue). Especially at 15:30, the C_n^2 measurement is the second lowest of any measurement in the train dataset evidenced by only one black marker being below the red marker.

5.4.3 2020/08/05 16:00

The final test forecast analyzed using this CDF method is the 08/05 16:00 forecast in Figure 5.4b. This forecast is chosen to illustrate an example of a model's good forecast with respect to RMSE score, a low 0.086 in this case. Like the two prior examples, the bounds

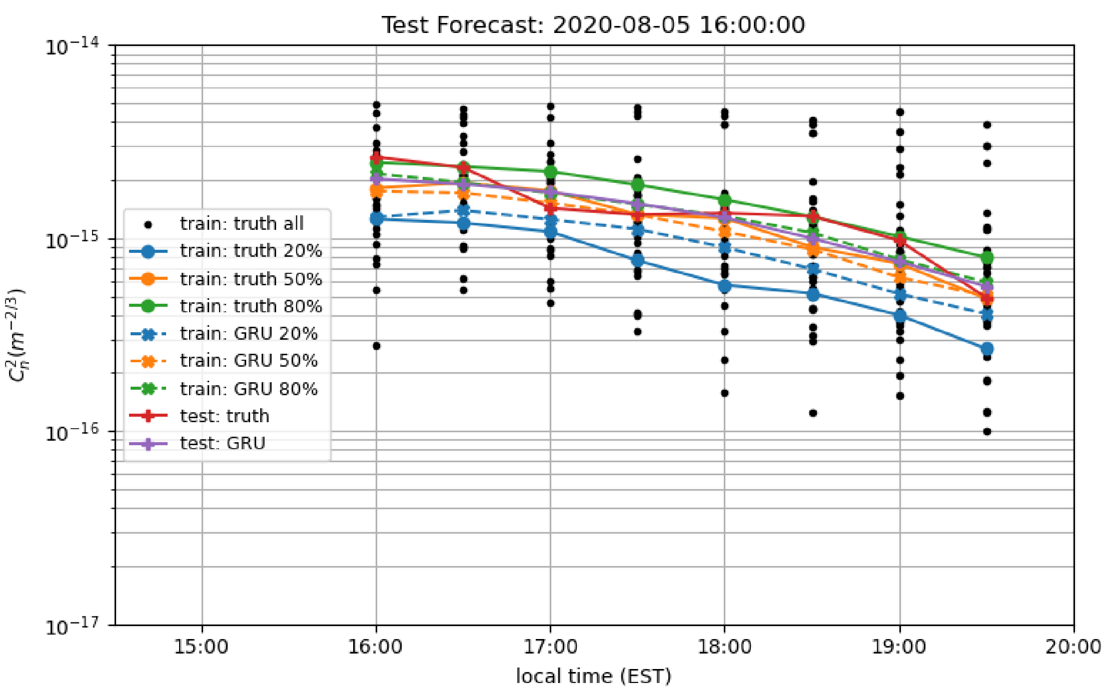


Figure 5.10: GRU performance analysis of the 08/05 16:00 forecast.

between the truth CDF 20% and 80% lines is narrow throughout the forecast. There is a slight increase in width from about a factor of two at 16:00 to more than a factor of three at

19:30 but is not significant over the course of the forecast. Also like the prior examples, the model forecast CDF 20% and 80% lines (dashed blue and green) are again narrower than the truth CDF lines, further indication of the model's generalization. The model forecast CDF 50% line (dashed orange) is nearly overlaying the truth CDF 50% line (solid orange) also further indicates that the model has learned the general C_n^2 forecast at this time of day from the train dataset and is effectively applying it to the test dataset. Model performance on this test forecast is good because the truth (red line) mostly oscillates about the model output CDF 80% values throughout the forecast, and the model forecast nearly overlays this same CDF line. Since the measurements are within the learned bounds of the model's learned relationship, the model therefore is effective in this forecast.

5.4.4 Conclusion of Analyses

These analyses have resulted in an understanding of where and why the model performs well and poorly. Capturing the depth of evening neutral events is a common issue with this model as illustrated by the daily-C_n^2 plots in Figures 5.4 and 5.5, and by the scatter plots in Figure 5.6. The analysis of the 08/09 18:00 forecast, an extreme example, shows that the model has generally learned to forecast an evening neutral event from this time, but only is able to forecast a limited range of C_n^2 strengths. In the analysis of the mid-day conditions on 08/06 14:00 in Figure 5.9 it is shown that the model has learned to forecast similar C_n^2 strengths mostly around $2 \times 10^{-15}(m^{-2/3})$ with a very narrow window. Significant deviations, like high frequency weather induced events, are difficult to forecast and is missed entirely in the case of the 08/06 14:00 forecast analysis. Finally, in Figure 5.10 is has been shown that the model performs very well when the forecast predicts a smooth transition of C_n^2 conditions which is the case for many of the forecasts in the test dataset.

5.5 Chapter Summary?

CHAPTER VI

Summary and Future Work

6.1 Summary

6.2 Future Work

1. Improve data preprocessing techniques to fill small gaps in weather or C_n^2 measurements.

2. Model C_n^2 at specific DELTA bins (screens), like the first or second, to isolate the forecasting to a particular part of the propagation path.

3. How are these RNN models impacted by input weather sequences that are out of range of the train dataset? This is relevant to deploying a model to test during a time of significant seasonal weather change.

4. Investigate whether the model performance declines if some data is missing in a sequence and instead padded by interpolation or some other technique.

5. Further investigation into the sensitivity input feature counts (sensitivity analysis)

6. Extend the grid search to include different number of epochs, learning rates (and decays), and fewer input variables

7. Explore shorter (like 2 hour) and longer (like 8 hour) forecasts!

BIBLIOGRAPHY

[1] R. Tyson, *Principles of Adaptive Pptics.* Boca Raton, FL: CRC Press, 2011.

[2] G. C. Valley, "Isoplanatic degradation of tilt correction and short-term imaging systems," *Appl. Opt.*, vol. 19, no. 4, pp. 574–577, Feb 1980. [Online]. Available: http://ao.osa.org/abstract.cfm?URI=ao-19-4-574

[3] C. Jellen, J. Burkhardt, C. Brownell, and C. Nelson, "Machine learning informed predictor importance measures of environmental parameters in maritime optical turbulence," *Appl. Opt.*, vol. 59, no. 21, pp. 6379–6389, Jul 2020. [Online]. Available: http://ao.osa.org/abstract.cfm?URI=ao-59-21-6379

[4] C. Su, X. Wu, T. Luo, S. Wu, and C. Qing, "Adaptive niche-genetic algorithm based on backpropagation neural network for atmospheric turbulence forecasting," *Appl. Opt.*, vol. 59, no. 12, pp. 3699–3705, Apr 2020. [Online]. Available: http://ao.osa.org/abstract.cfm?URI=ao-59-12-3699

[5] Y. Wang and S. Basu, "Estimation of optical turbulence in the atmospheric surface layer from routine meteorological observations: an artificial neural network approach," in *Laser Communication and Propagation through the Atmosphere and Oceans III*, A. M. J. van Eijk, C. C. Davis, and S. M. Hammel, Eds., vol. 9224, International Society for Optics and Photonics. SPIE, 2014, pp. 300 – 307. [Online]. Available: https://doi.org/10.1117/12.2063168

[6] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[7] C. Olah, "Understanding lstm networks." [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[8] "Vantage Pro2 Plus Technical Specs." [Online]. Available: https://www.davisinstruments.com/vantage-pro2/tech-specs/

[9] "Atmospheric Characterization Devices." [Online]. Available: https://www.mza.com/products.php?page=productsAtmosChar

[10] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern,

M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[12] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019.

[13] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. USA: Cambridge University Press, 2007.

[14] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

APPENDIX A

More Stuff at the End

This is Appendix A. Appendix A should span multiple lines so I can see if the margins are correct. We need to ensure that we have 1.5 inches on the LHS margin.

APPENDIX B

Even More Stuff after the End

This is Appendix B. Appendix B should span multiple lines so I can see if the margins are correct. We need to ensure that we have 1.5 inches on the LHS margin.