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 Engineering

By

Mitchell Gene Grose

Dayton, Ohio

May, 2021

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

Name: Grose, Mitchell Gene

APPROVED BY:

Eric J. Balster, Ph.D.
Advisory Committee Chairman
Associate Professor and Chair,
Electrical and Computer Engineering

Member A., Ph.D.
Committee Member
Faculty A. Title, Faculty A
department

Member B., Ph.D.
Committee Member
Faculty B. Title, Faculty B
department

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. Eric J. Balster

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

Introduction Problem Statement; Literature Review

1.1 Problem Statement

1.1.1 This effort performs an ensemble study of ML architectures/parameters to best forecast 4 hours of turbulence conditions from prior measurements.

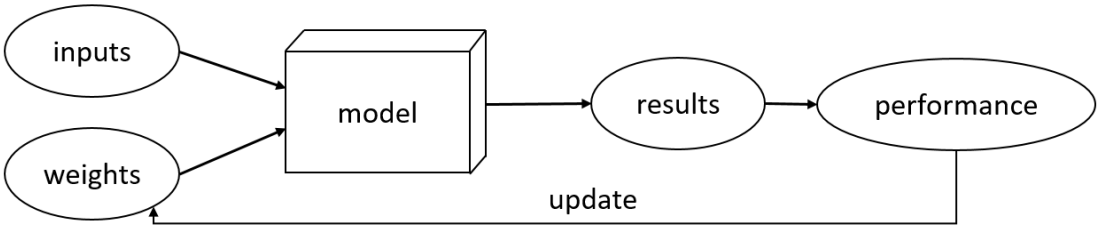
1.1.2 What is turbulence (C_n^2)? Why is it important to measure/model?

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

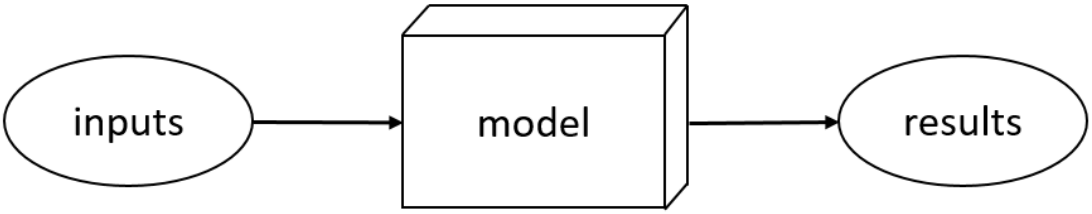
Figure 1.1: Effect of low-strength and high-strength C_n^2 conditions.

1.1.3 Review the fundamentals of machine learning.

Optimization algorithm!



(a) Training



(b) Testing

Figure 1.2: Simple example of machine learning model training and testing.

1.2 Literature Review

1.2.1 State of Turbulence C_n^2 Modeling

Physical Models

ML Models

Recently published literature predicts C_n^2 conditions at time t given weather inputs at time t.

1.2.2 General information about MLP/RNN architectures here? Does the in-depth review of the architectures go here?

CHAPTER II

Problem Setup

2.1 Physical Experimental Setup

2.1.1 Spatial Proximity

Show Google Maps screenshot of MZA office (location of weather station), Fitz Hall (location of DELTA), and Y-Tower (location of target). Overlay arrow from Fitz Hall pointing to the Y-Tower. Overlay text boxes with relevant location (LLA) information.

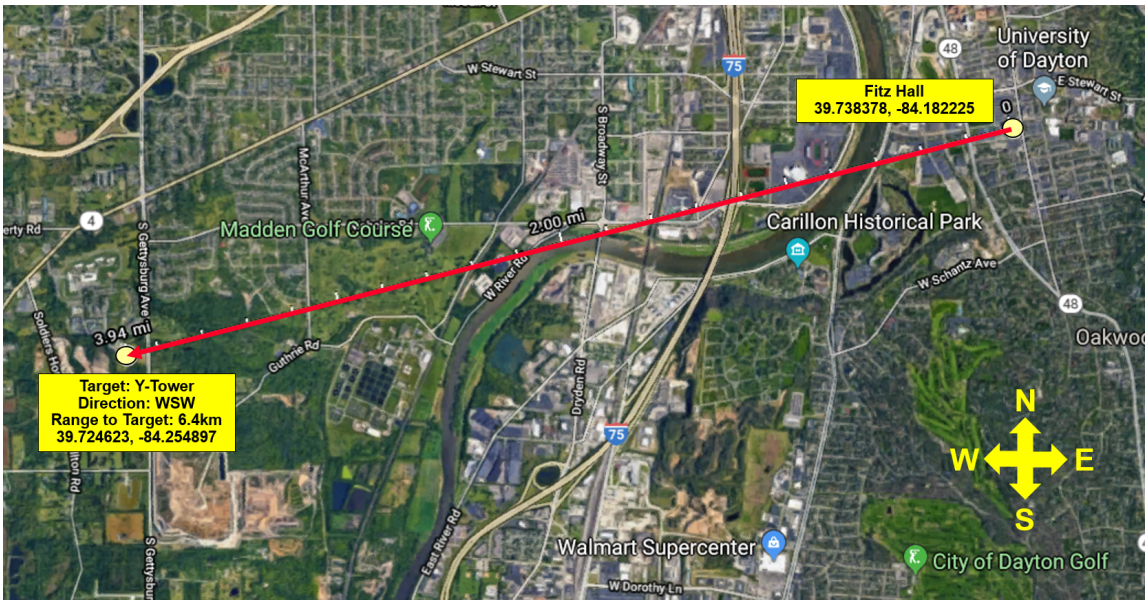
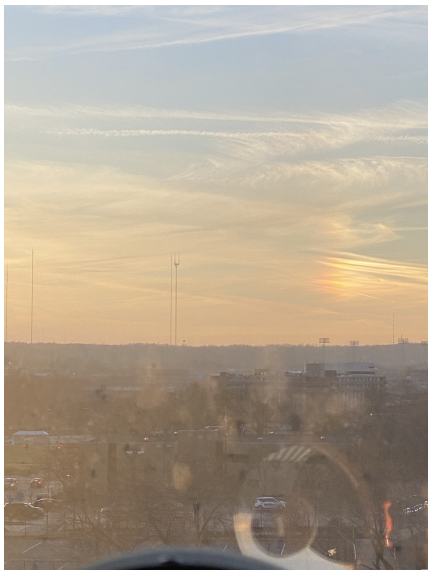


Figure 2.1: Fitz Hall to Y-Tower geometry. **NEED TO EXPAND MAP TO INCLUDE WEATHER STATION**

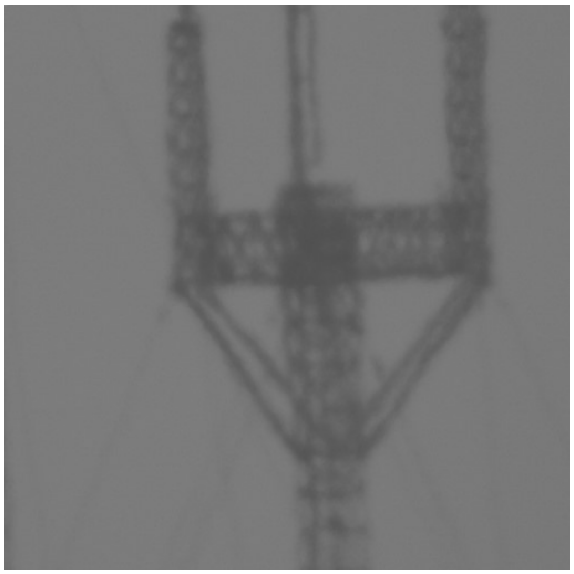
2.2 Modeling Experimental Setup

Finding best model to forecast 4 hours of turbulence conditions from prior environmental measurements.

(a) Telescope setup



(b) Wide view of target



(c) View of target through the DELTA telescope

Figure 2.2: DELTA setup for collection of minute-by-minute C_n^2 measurements.

2.2.1 Data Processing (Ch. 3)

2.2.2 Grid Search (Ch. 4)

Architectures, sequence length, number of layers, number of nodes per layer.

Statistical Analysis of Grid Search

2.2.3 Application of "Best" Model to Test Dataset (Ch. 5)

CHAPTER III

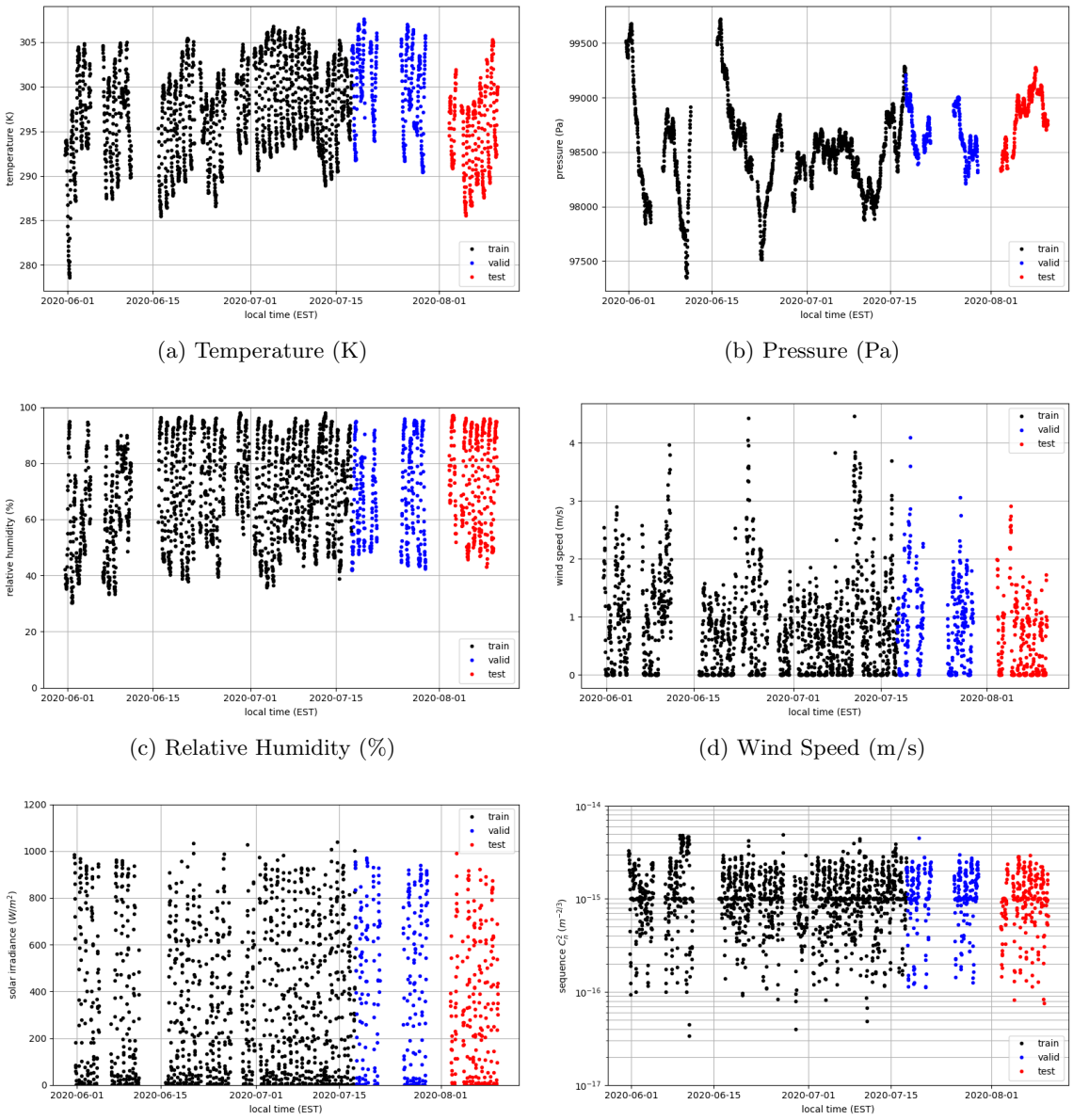
Dataset

3.1 Weather Sequences and Histograms

Temporal sequence plots presented in Figure 3.1.

3.2 Forecast Sequence and Histogram

3.3 Window-Averaging; Sub-Sampling; Parsing Data into Sequences/Forecasts



(a) Temperature (K)

(b) Pressure (Pa)

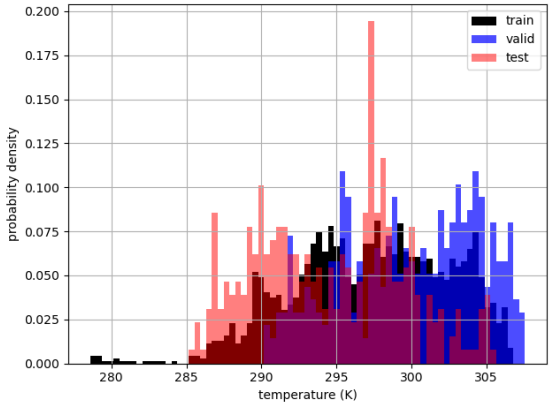
(c) Relative Humidity (%)

(d) Wind Speed (m/s)

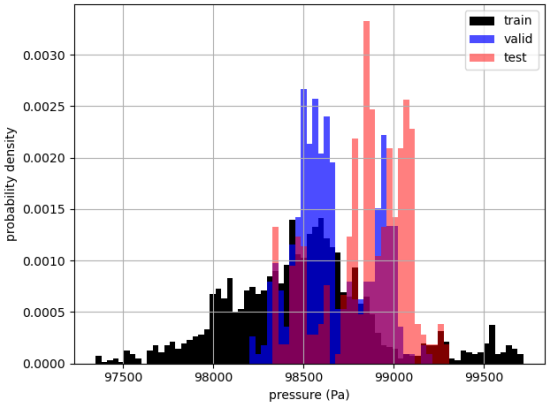
(e) Solar Irradiance (W/m^2)

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

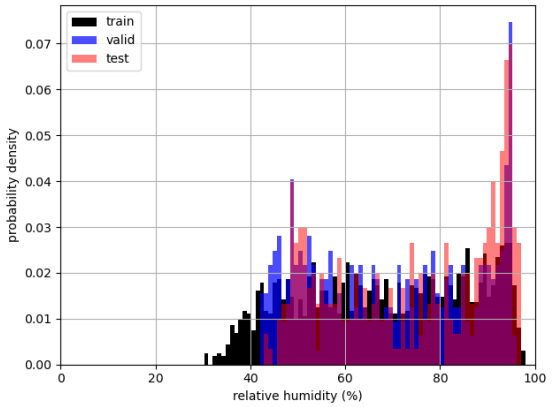
Figure 3.1: Sequence data as a function of time, parsed by train, validation, and test datasets drawn in black, blue, and red, respectively.



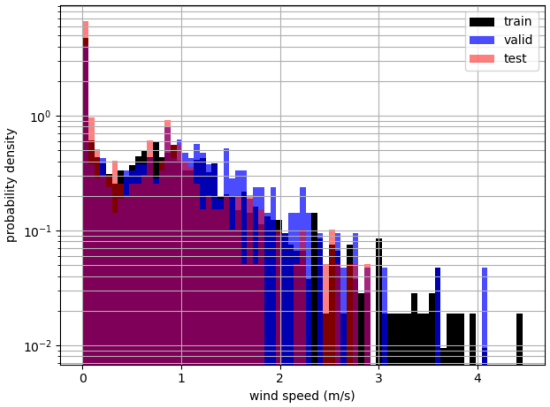
(a) Temperature (K)



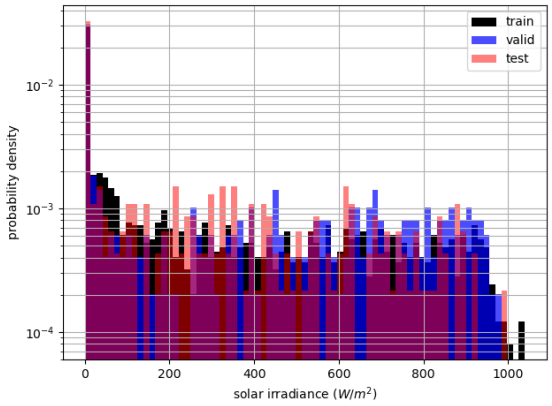
(b) Pressure (Pa)



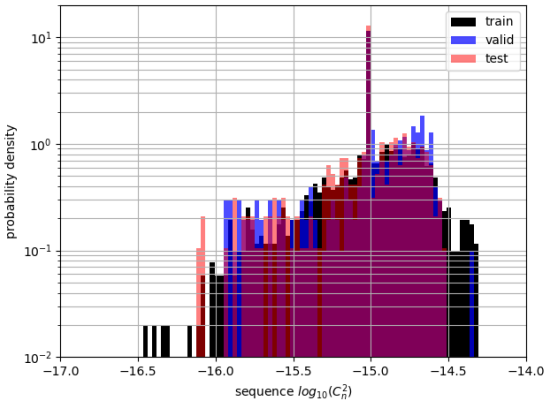
(c) Relative Humidity (%)



(d) Wind Speed (m/s)

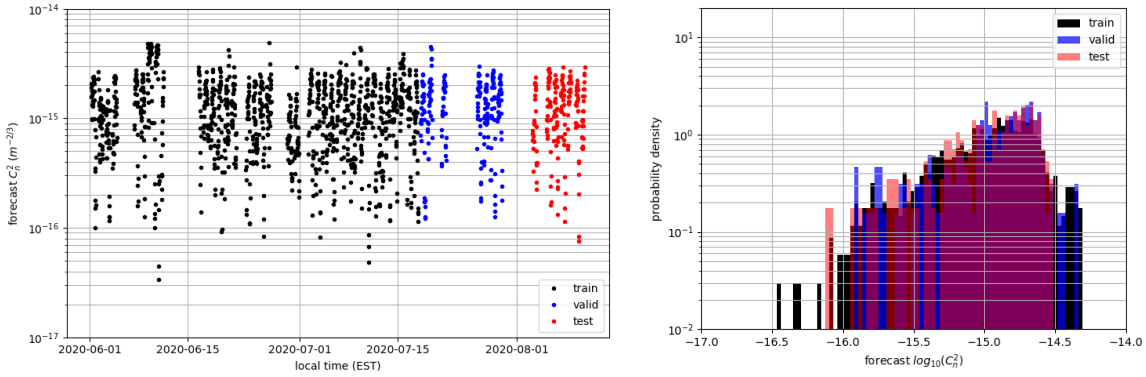


(e) Solar Irradiance



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

Figure 3.2: Sequence data in normalized histograms, parsed by the train, validation, and test datasets drawn in black, blue, and red, respectively.



(a) (b)

Figure 3.3: 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.

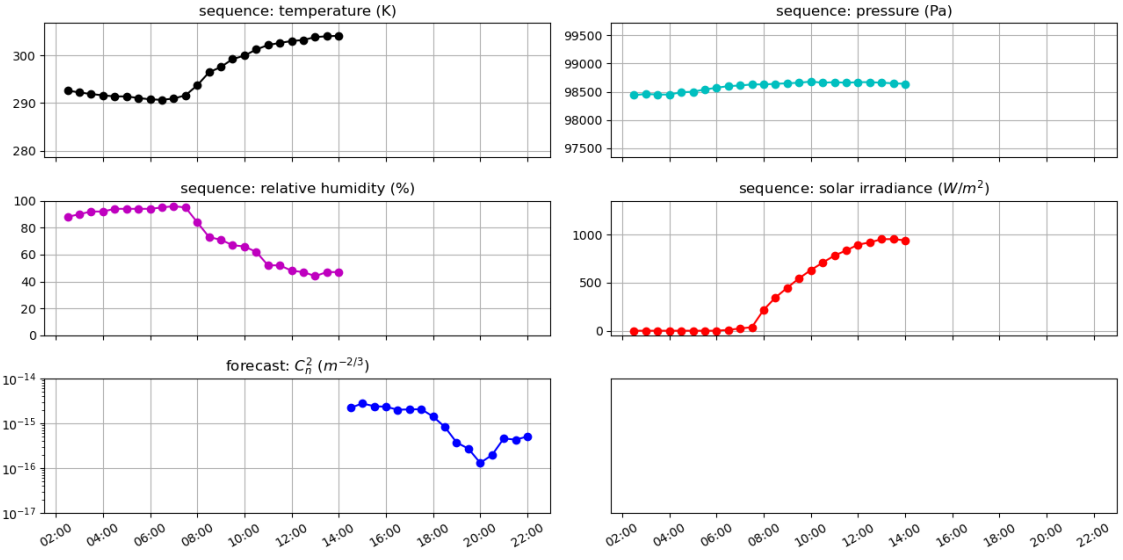


Figure 3.4: Example sequence and corresponding turbulence forecast. **NEED TO ADD WIND SPEED AND PRIOR TURBULENCE MEASUREMENTS**

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, then those parameters are used in the model applied to the test dataset for an 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, a model is trained with the first learning rate and it's performance on the validation dataset is recorded, then the model is trained again in the same fashion but with the second learning rate and the performance on the validation

dataset is recorded, and finally this is done again 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. The hyperparameters associated with best performing model are the best hyperparameters. 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 3.1, the available input sequence features (variables) are prior temperature, pressure, relative humidity, wind speed, solar irradiance, and C_n^2 measurements. The "input sequence features" parameter iterates over which fea-

tures (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, one of the most popular optimization algorithms used today (Go in depth about the algorithm here, or reference the background section? Reference paper on Adam, paper on correct implementation of regularization term, AdamW, and pytorch implementation). 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 there is no reason why I chose this specific value, 1e-3; I just wanted to employ a bit of regularization to avoid overfitting.

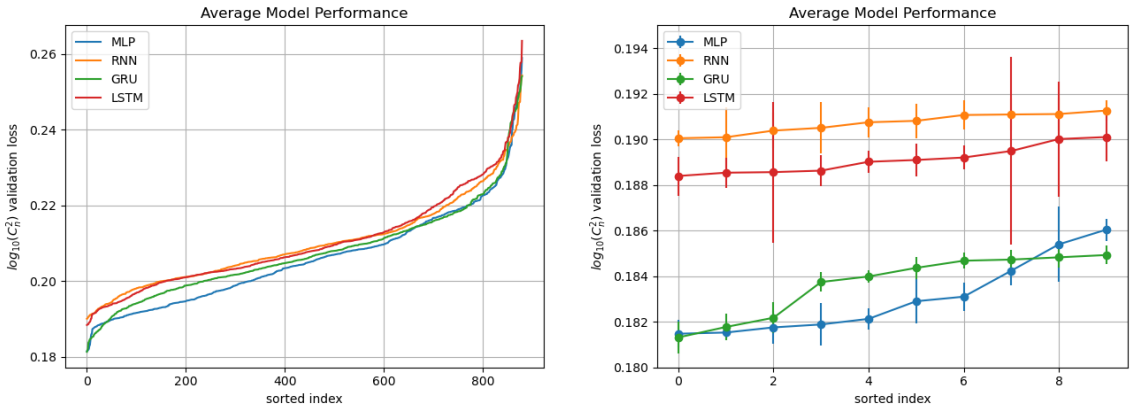
4.2 Results

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 (degrees of freedom $= N - 1$) of the RMSE scores are calculated per parameter combination yielding four arrays of 880 averages and standard

deviations. The standard error, or the standard deviation of the mean, is calculated by dividing the standard deviation by the square root of the number of samples, 10 in this case. From these statistics the best model 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, 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 shape



(a) Full set of iterations

(b) Top 10 iterations

Figure 4.1: Grid search results.

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 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. 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 for the MLP and GRU architectures in Figure 4.1b are smaller (better) than for 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.

4.2.2 Statistical Significance

The results presented in Figure 4.1 clearly show that specific models and 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

$$t = \frac{\bar{x_A} - \bar{x_B}}{\sqrt{\frac{Var(x_A)}{N_A} + \frac{Var(x_B)}{N_B}}}, \tag{4.1}$$

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

$$dof = \frac{\left[\frac{Var(x_A)}{N_A} + \frac{Var(x_B)}{N_B}\right]^2}{\frac{[Var(x_A)/N_A]^2}{N_A - 1} + \frac{[Var(x_B)/N_B]^2}{N_B - 1}}. \tag{4.2}$$

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 [1]. A very small p-value (≤ 0.05) means that the observed difference in means is very significant and the null hypothesis H_0 that the means are equal 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 Section 4.2.2 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 [2]. 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 4.1 and 4.2 is the 10 validation $log_{10}(C_n^2)$ RMSE scores from the very best GRU model. Sample B in Equations 4.1 and 4.2 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 similarly parsed by fundamental architecture. The MLP p-values are in

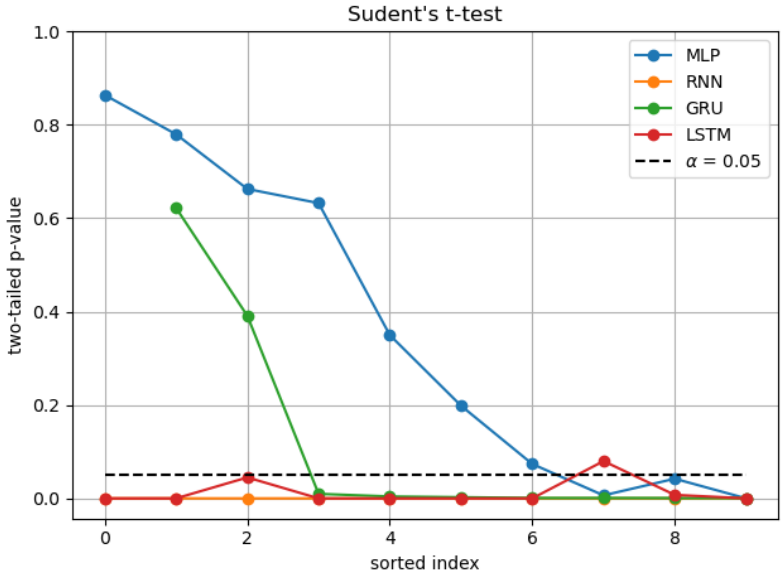


Figure 4.2: Grid search results.

blue, 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 zero is

omitted because performing the *Student's t-test* of 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.

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.

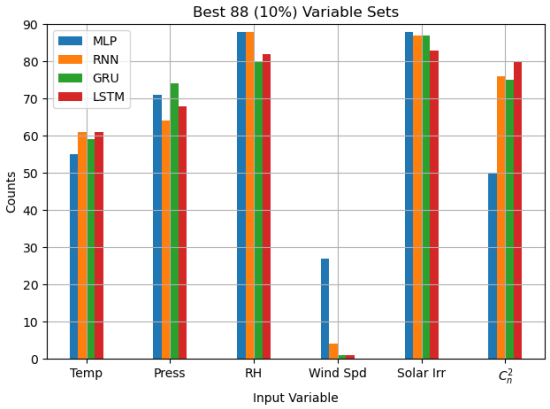
Table 4.1: Top GRU Model Parameters

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

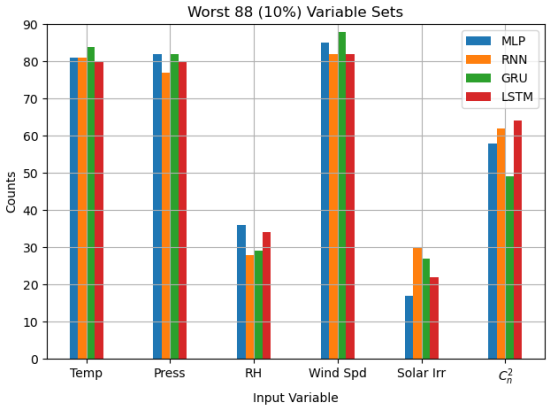
Similarly to Table 4.1, Table 4.2 summarizes the grid search parameters for the top seven MLP models.

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



(a) Best 10%



(b) Worst 10%

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

CHAPTER V

Test Results

5.1 GRU Test Dataset Performance Summary

After determining the best model 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 represent each model's

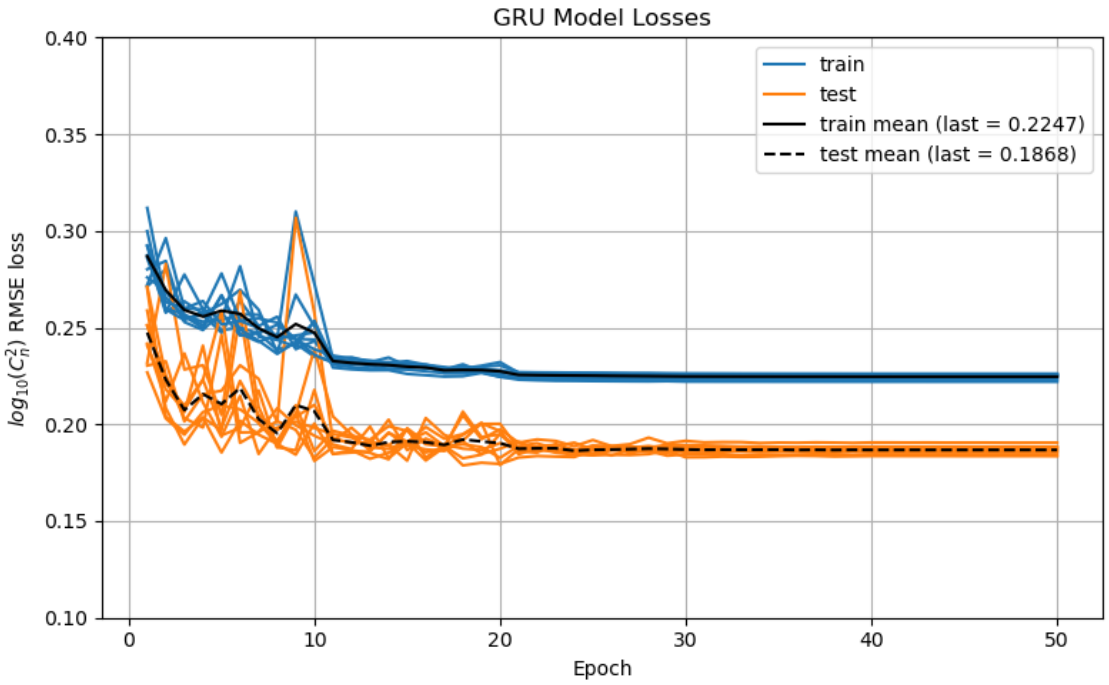


Figure 5.1: GRU train and test loss curves of 10 models.

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 (add student's t-test p-value here to show significance?), 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 10 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 10 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

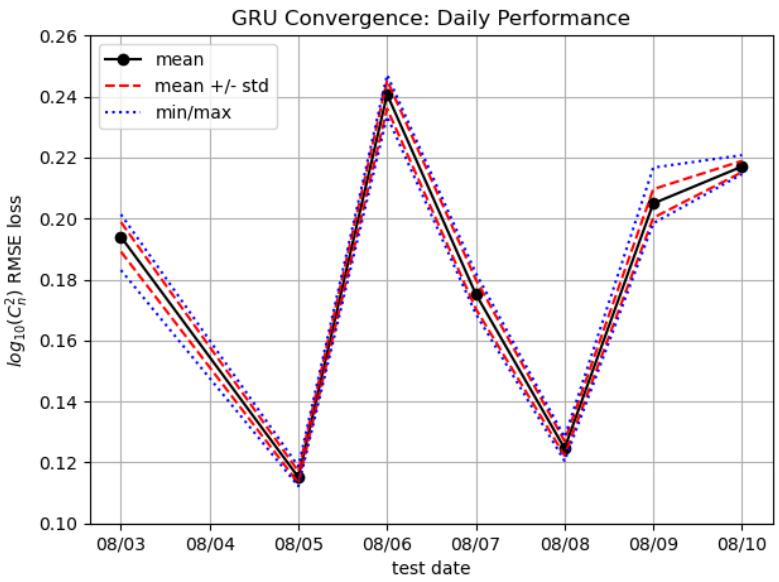


Figure 5.2: GRU daily summary performance: mean, standard deviation, and min/max.

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

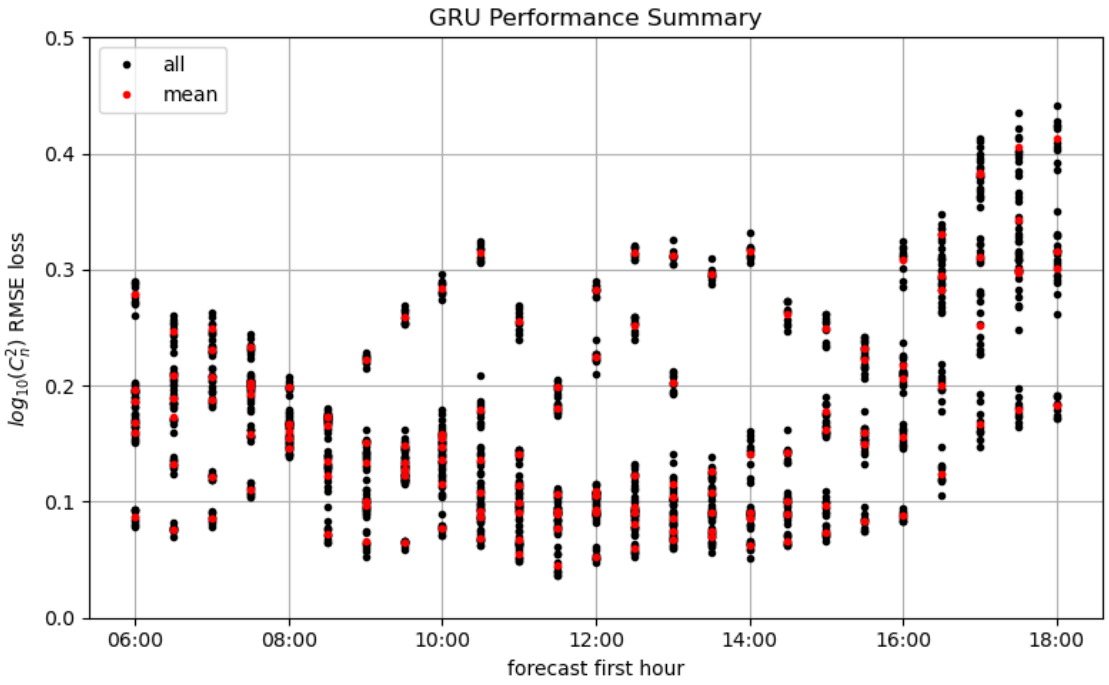


Figure 5.3: GRU hourly summary performance: individual and average.

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.

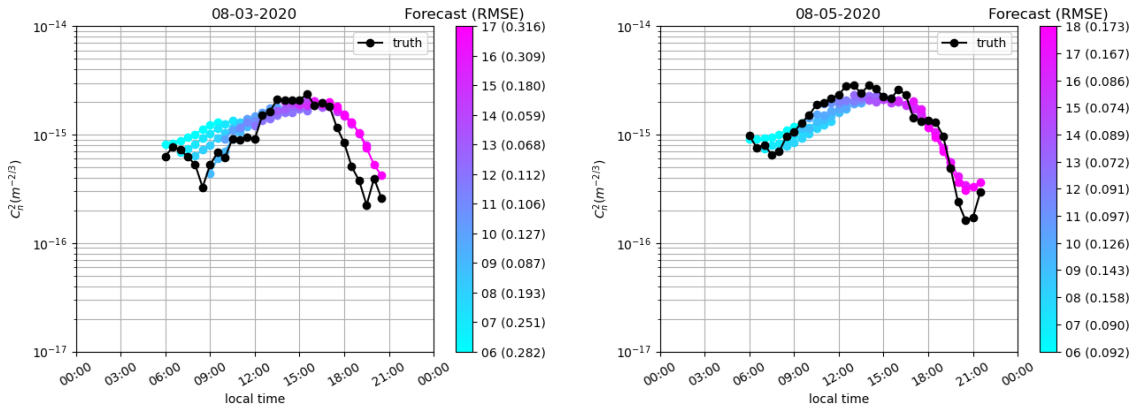
Figure 5.3 is a summary of model performance as a function of time of day. The trends indicate that best model performance starts around 0.2 (on the y-axis) in the beginning of the day, drops down to around 0.1 in the middle of the day, then rises in the evening to around 0.3. The morning forecasts trend 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.

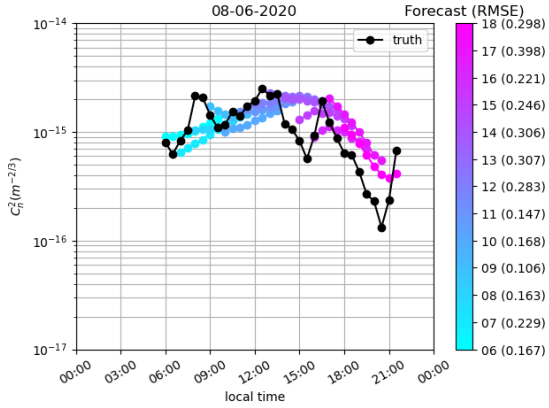
Each daily-C_n^2 plot contains multiple curves. The black curve represents the truth C_n^2 measured conditions. The other curves which transition in color from cyan to magenta represent the model forecasts throughout the day. The brightest cyan is the earliest forecast in the day, and the brightest magenta 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

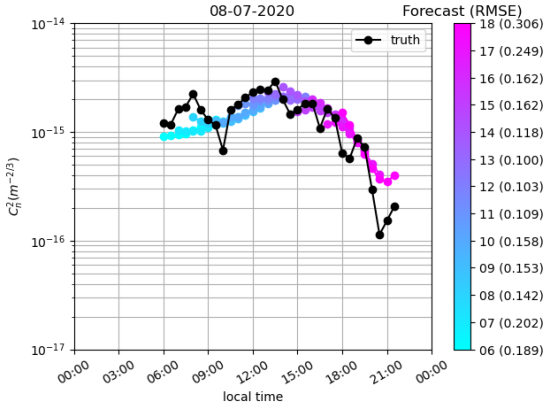


(a) August 03

(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.

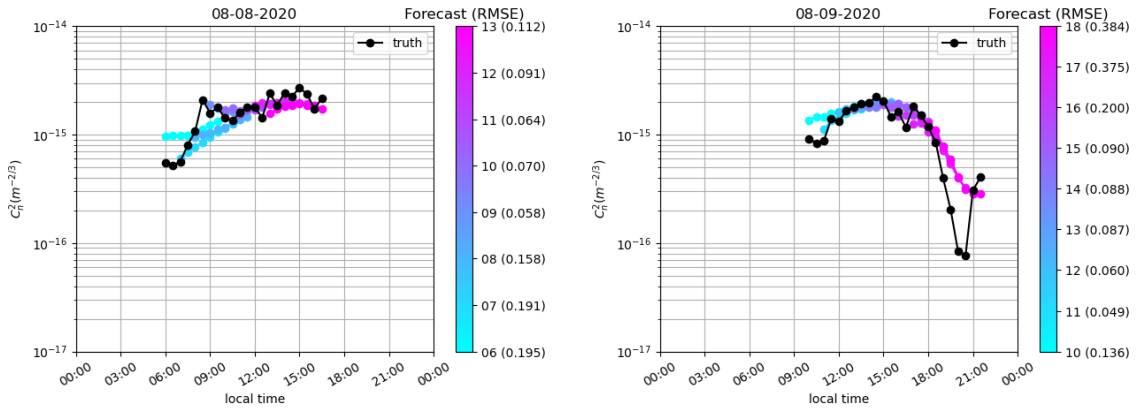
Model performance on 08/06 in Figure 5.4c is the worst of the seven test days (see Figure 5.2). The first three forecasts of the day at 06:00 - 08:00 have average to above-average error scores because the forecasts predict a steady rise in C_n^2 strength which generally is measured, but a weather-induced 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.

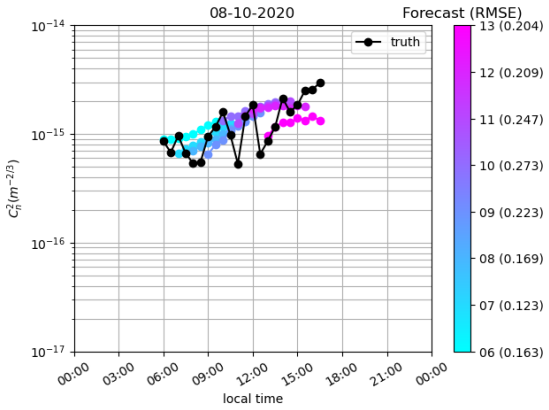
The final three test days, 08/08 - 08/10, are presented in Figures 5.5a, 5.5b, and 5.5c, respectively. On average over the 10-model ensemble study, 08/08 is the second best day of model performance. Throughout the day, the model generally predicts diurnal trend: a slow rise in C_n^2 strength until around 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 3.3a). 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.

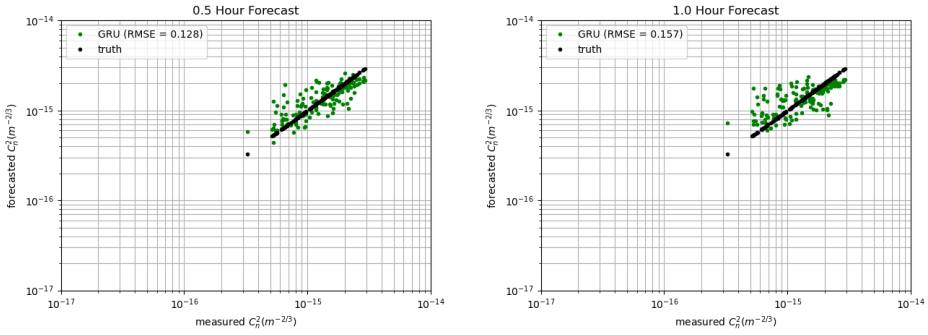
5.3 Forecast Analysis

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.

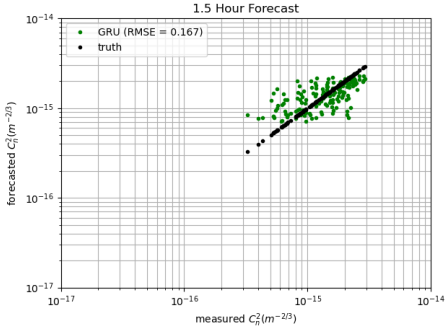
The scatter plot of the first forecast time, 30 minutes, in Figure 5.6a shows a strong clustering of the model forecasts about the measured C_n^2. Excluding one outlier, the range of measured C_n^2 the model is tasked to forecast is only from $5 \times 10^{-16}(m^{-2/3})$ to $3 \times 10^{-15}(m^{-2/3})$, 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

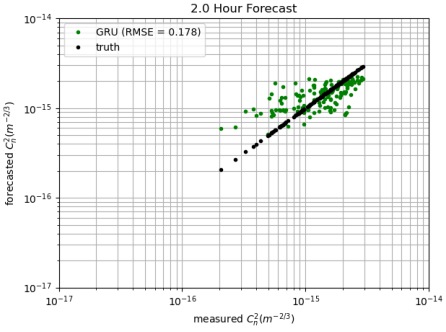


(a)

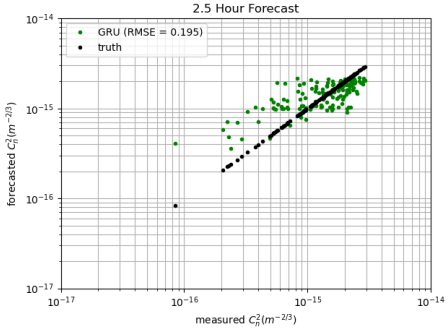
(b)



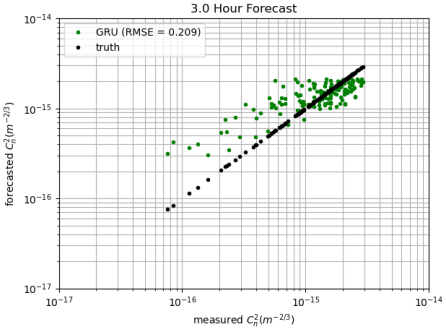
(c)



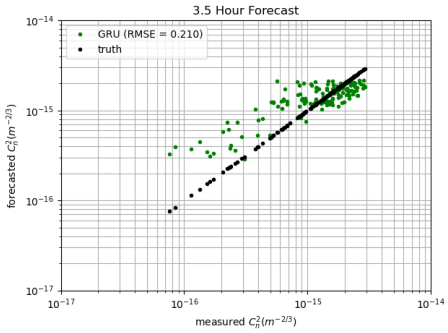
(d)



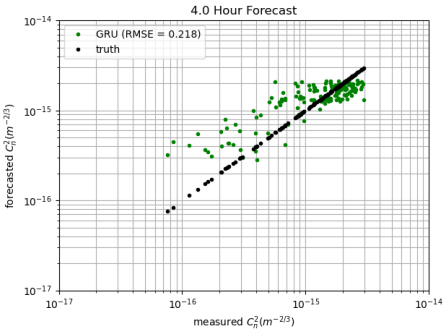
(e)



(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.

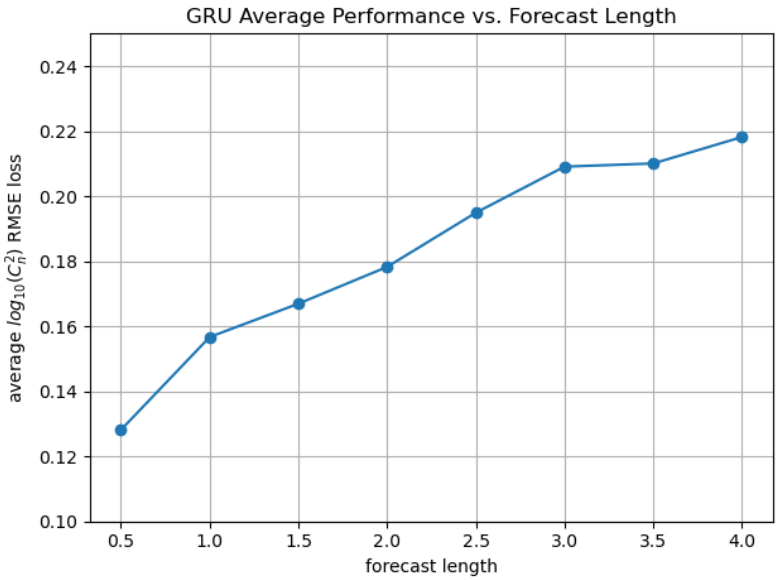


Figure 5.7: Average performance as a function of forecast length.

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.

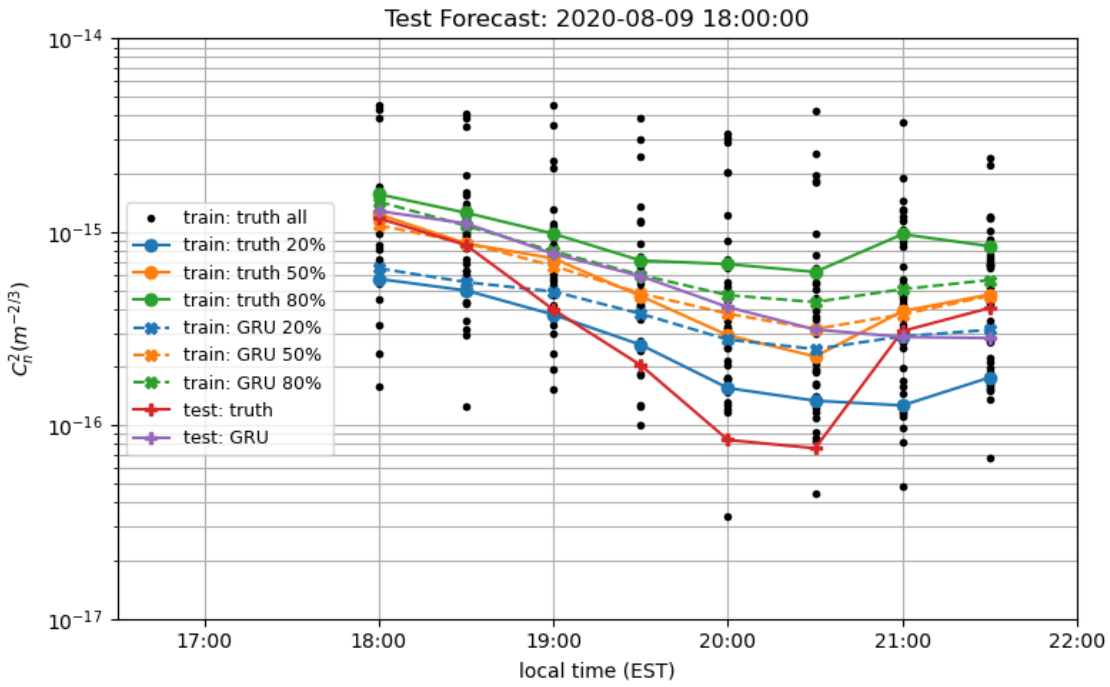


Figure 5.8: GRU performance analysis of the 08/09 18:00 forecast.

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

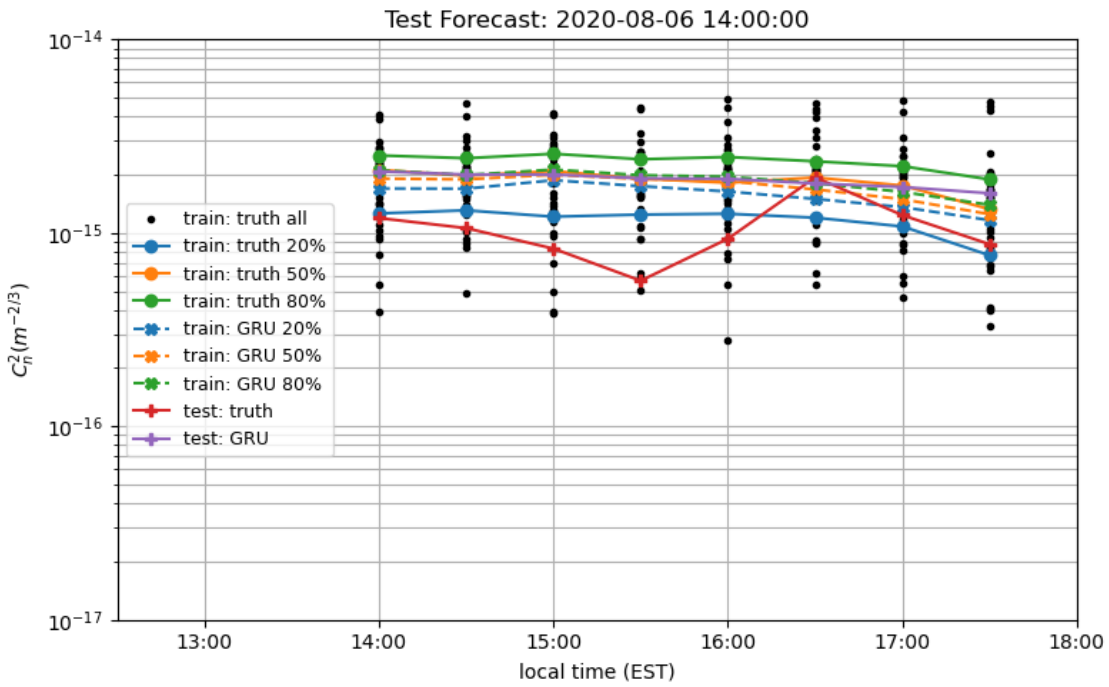


Figure 5.9: GRU performance analysis of the 08/06 14:00 forecast.

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

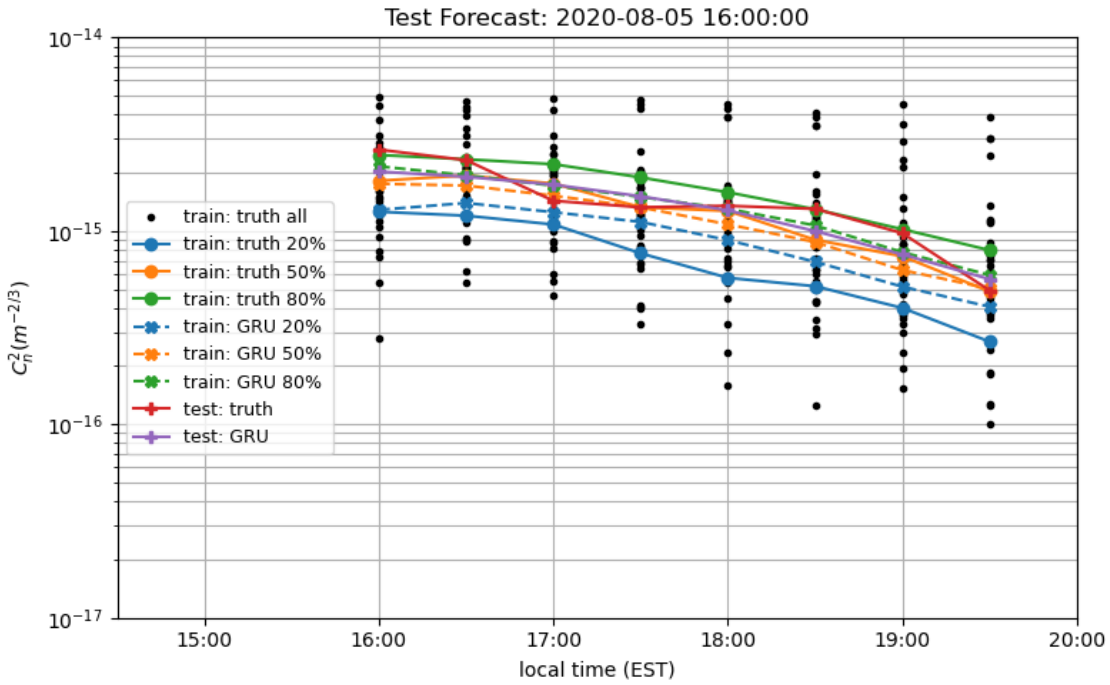


Figure 5.10: GRU performance analysis of the 08/05 16:00 forecast.

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.

CHAPTER VI

Future Work

BIBLIOGRAPHY

[1] 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.

[2] 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.