8-2023

Physics-Guided Deep Learning for Solar Wind Modeling at L1 Point

Robert M. Johnson
Utah State University

PHYSICS-GUIDED DEEP LEARNING FOR SOLAR WIND MODELING AT L1 POINT

by

Robert M. Johnson

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Data Science

Approved:

Soukaina Filali Bourbrahimi, Ph.D. Major Professor	Mario Harper, Ph.D. Committee Member
Mike Taylor, Ph.D. Committee Member	D. Richard Cutler, Ph.D. Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2023

ABSTRACT

Physics-guided Deep Learning for Solar Wind Modeling at L1 point

by

Robert M. Johnson, Master of Science

Utah State University, 2023

Major Professor: Soukaina Filali Bourbrahimi, Ph.D.
Department: Computer Science

Solar wind modeling is categorized into empirical and physics-based models, both of which predict the properties of solar wind in different parts of the heliosphere. Empirical models are relatively inexpensive to run and have shown great success at predicting the solar wind at the L1 Lagrange point. Physics-based models provide more sophisticated models based on magnetohydrodynamics (MHD) that are computationally expensive to run. In this paper, we propose to combine empirical and physics-based models by developing a physics-guided neural network for solar wind prediction. We show the variability of physics-guided loss across multiple deep-learning models. We then discuss the strengths of physics-guided neural networks under various constraints to inform us more about the characteristics of the input and output data. My main contributions in this thesis are:

1. The putting forth of a novel physics-guided loss function for solar wind prediction;

2. A discussion on the data, and how it can best be used;

3. And finally and most importantly, showing that while physics-guided loss functions can provide certain networks with aid in physics related problems, they are not a silver bullet, and are not always guaranteed to improve performance.

(62 pages)

PUBLIC ABSTRACT

Physics-guided Deep Learning for Solar Wind Modeling at L1 point

Robert M. Johnson

Neural networks are adept at finding patterns that are too long and too small for humans to find in data. Usually, this power is used to generate predictions with greater accuracy than most alternative models. However, we can also use this power to understand more about the data we train these networks on. We do this by changing the data that the networks train on and the data they are tested on. This allows us to both control the maximum length of a pattern and to compare data between different groups, in our case, different solar cycles. This thesis is our attempt to understand solar wind data better. We do this by proposing a physics based framework and comparing the results of different inputs and outputs through different networks. These results show three major things: 1, that training networks using the physical law of Ohm's law for an ideal plasma can improve network performance predictions; 2, that the specific characteristics of different solar cycles make them more suitable for training or testing; and 3, that while physics guided loss functions can be helpful in certain situations, they are no silver bullet to improved predictions.

ACRONYMS

ACE	Advanced Composition Explorer
$B_{x,y,z}$	the three components of the magnetic field
CME	coronal mass ejections
CNN	convolutional neural network
FAIR	findability, accessibility, interoperability, and reusability
GPS	global positioning system
GRU	gated recurrent unit network
IMP	Interplanetary Monitoring Platform
JAXA	Japanese Aerospace Exploration Agency
LSTM	long short-term memory network
MHD	magnetohydrodynamic
NASA	national air and space administration
NN	neural network
OMNI	OMNI is a project name by NASA, not an acronym.
PGNN	physics-guided neural network
ResNet	residual neural network
SEP	solar energetic particles
$V_{x,y,z}$	the three components of the velocity field
WSA-Enlil	Wang-Sheely-Arge Enlil

ACKNOWLEDGMENTS

I would like to thank my advisor, Soukaina Filali Bourbrahimi, for helping me to find a project on which I could make a difference.

I would like to thank my father, Randy Johnson, for helping to foster my curiosity and desire to learn about the world, and for pushing me to take a computer science course a semester earlier than I expected.

I would like to thank my brother, Topher Johnson, for going on his mission between those semesters, causing my father to give me the aforementioned advice.

I would like to thank my mother, Shauna Johnson, for being both patient and consistent with me, helping me to get the grades I needed to attend this prestigious university.

I would like to thank the university for providing an environment in which I could thrive and excel.

Finally, I would like to thank my Fiance, Emily Williams, for helping me to stay consistent in my thesis writing.

Robert M. Johnson

CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 What is solar wind

The sun is a ball of hot plasma, powered by nuclear fusion reactions in its core. The energy from these reactions propagates outwards until the photosphere, where the energy radiates outward as light. Some of this energy is instead transferred into kinetic energy, which is released when particles obtain enough kinetic energy to reach escape velocity and leave the sun. This charged stream of plasma is then known as solar wind. The solar wind plasma can be considered an ideal plasma, or in other words, the electrostatic force controls the movements of the plasma more than the processes of ordinary gas kinetics [2].

1.2 Why is it important to understand solar wind

The high-velocity particles of solar wind can have a variety of effects when they approach the Earth. Some are relatively harmless and beautiful, such as the northern lights visible when the solar wind interacts with Earth's atmosphere. Others are more dangerous, like causing cancer for orbiting astronauts. Others have a larger, though less life-threatening effect of causing shorts and other damage to satellites [3]. Knowing how to best prepare for these events requires being able to predict when they will happen and how intense they will be. This is why one of the latest United States executive orders related to space weather studies urges scientists to direct attention to develop a response plan to severe space weather conditions [4].

In addition to these obvious benefits, understanding the phenomena that lead to solar wind events can also help us understand the sun itself. Finding the difference between the slow and fast solar wind has helped us to find the causes of them, such as coronal holes [5]. As we understand more about the wind the sun sends our way, we have the opportunity

Fig. 1.1: Solar wind traveling to Earth at the speed of 300 kilometers per second *(Image Courteousy from [1])*

to understand more about why those things are sent and more about what processes send them.

1.3 What are neural networks

One technology that has improved our ability to predict future events is the neural network. Comprised of many parameters and the linear algebra holding them together, neural networks have revolutionized the field of data science. As our world generates more data that is accurately captured and kept, we have been able to learn more about how to build better neural networks that can see patterns invisible to the human eye.

Neural networks work by taking an input, then performing linear algebraic processes on that input and on the results of those processes. At the end of that loop, a prediction is generated. That prediction can then be compared to the truth, and the difference between the two can be pushed backwards into the neural network to improve the next prediction that goes through it. This process can be continued until a desired accuracy threshold is reached. Using this technology, computers have been able to identify objects in a picture, translate languages, and compile information for human consumption.

1.4 Chapter organization

The rest of the paper is organized as follows: Chapter 3 describes the data used in this

study, Chapter 4 defines the proposed Ohm's law constraint, followed by a description of the methods and experimental results in Chapter 5 and Chapter 5.3 respectively. An analysis of the data available, partitioned by solar cycle, is put forth in Chapter 7. Chapter 8 continues this usage of using the training of neural networks with optimized hyperparameters to understand solar wind data by comparing the effects of varying the *span* and *prior* of the input space. Chapter 6 considers the interconnectedness of our inputs and outputs by performing a small amount of feature selection on both our input values and our target values. Finally, Chapter 9 concludes the paper and shows potential directions for future works.

CHAPTER 2

Related Works

Since the discovery of solar wind over half a century ago, many studies have been conducted to predict solar wind. One of the highly adopted physics-based solar wind forecasting models is that of the Wang-Sheely-Arge Enlil (WSA-Enlil) hybrid model - a three-dimensional magneto-hydrodynamic (MHD) model. WSA-Enlil starts by taking the input state of the sun at a given time and runs it through relevant MHD equations. The model then propagates the result outward from the sun and forward in time to forecast solar wind speeds three to four days in advance [6].

Several incremental works that build on the WSA-Enlil have been proposed. The increment is done either by slightly improving the accuracy of the model or by making the MHD equation computations run faster. Yu et al. propose an improved WSA-Enlil by approximating the MHD computation through key assumptions. The improved WSA-Enlil was reduced from a three-dimensional problem into a one-dimensional problem without compromising significant accuracy [7]. Yu et al. focused their modeling only on predicting the slow solar wind since fast solar wind is much more easily predicted by looking at the sizes of the sources of the fast solar wind - namely, coronal holes [5]. Finally, Liu et al. developed a new forecasting model that relies on dark areas of the sun to produce solar wind forecasts [8].

The advent of large neural networks has also allowed other researchers to improve over WSA-Enlil by using more data-driven methods, as opposed to purely physics-based systems used previously. Yang et al. achieved better results at 2.5 solar radii by using a three-layer fully connected network to establish the relationship between the polarized magnetic field and the electron density and solar wind velocity [9]. The same group also improved upon their work by enforcing self-consistent boundary conditions. Hemapriya et al. developed a convolutional neural network online model trained on solar images to gain a comprehensive

knowledge of the solar activity prior to predicting solar wind velocities [10]. Leitner et al. found the distribution of solar wind to be quasi-invariant. Due to this quasi-invariant nature, we propose to adopt a physics-guided neural network (PGNN) as a novel approach for solar wind prediction. Specifically, our paper contributions are summarized below:

1. We model the solar wind prediction problem as a multivariate time series prediction task and propose a novel loss function based on Ohm's law for an ideal plasma.

2. We train multiple state-of-the-art deep learning forecasting models and show the superiority of our physics loss.

3. We explore multiple data normalizations and assess their effect on our model.

4. We made our source code open-source in a project website[1] that meets the principles of Findability, Accessibility, Interoperability, and Reusability (FAIR) [11].

[1] https://sites.google.com/view/solarwindprediction/

CHAPTER 3

Data

Without collected data, we cannot check if our predictions are correct. In this chapter, we show the origin of our data, the form of our data, and the sources of our data. We also cover how we pre-process the data, some summary statistics about the data, and talk about the normalization methods used to prepare the data for consumption by neural networks.

3.1 Origin

Our data originates from the NASA OMNI multi-spacecraft data set of near-Earth solar wind parameters [12]. Table 3.1 summarizes the seven OMNI parameters that we used for prediction. B_x, B_y, and B_z are the three components of the magnetic field measured by a three-axis teslameter (Gauss meter). The velocity sensors measure the three-dimensional velocity distribution functions of electrons and ions (V_x, V_y, and V_z) [13]. Finally, the electric field parameter E refers to the force exercised by the physical field that surrounds electrically charged particles.

3.2 Form

The data set consists of 5-minute time resolution multivariate time series data, which we preprocess by averaging into a 1-hour time resolution. We used 12 hours (*prior*) as the number of hours prior to the solar wind event. The *prior* is the interval of time in the future when our model should be able to predict the parameters of the ambient solar wind. In other terms, we investigated the possibility of predicting the ambient solar wind characteristics from the electrical field, velocity, and magnetic field characteristics 12 hours before its occurrence. We considered a *span* of 24 hours, which corresponds to the number of hours we observe the solar wind characteristics. figure 3.1 illustrates the time series of the seven solar wind physical parameters we chose to forecast for the year 1992.

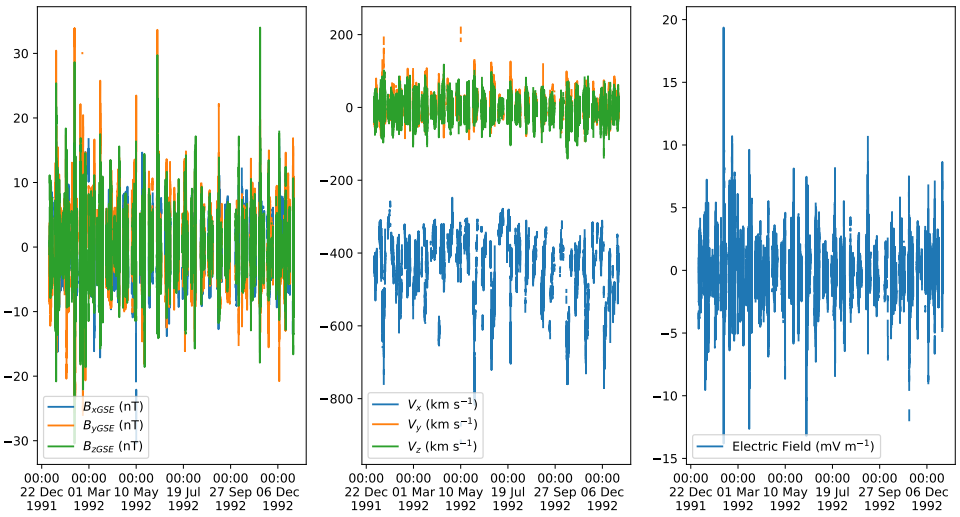


Fig. 3.1: OMNI time series data snapshot for the year 1992

3.3 Data Sources

Our data comes from three satellites: Geotail, a joint mission of NASA and the Japanese Aerospace Exploration Agency (JAXA); ACE, NASA's Advanced Composition Explorer; and Explorer 50, also known as IMP-8, which was the final in a series of the Interplanetary Monitoring Platform. Geotail is stationed in high elliptical Earth orbit, launched there in July of 1992. ACE was launched in August 1997 and began operations in January 1998. ACE maintains its position in the Earth's L1 Lagrange point, which is

Table 3.1: OMNI features and metadata

Feature	Unit	Description
E	mV/m	Electric field
Vx	km/s	X component of the velocity
Vy	km/s	Y component of the velocity
Vz	km/s	Z component of the velocity
Bx	nT	X component of the magnetic field
By	nT	Y component of the magnetic field
Bz	nT	Z component of the magnetic field

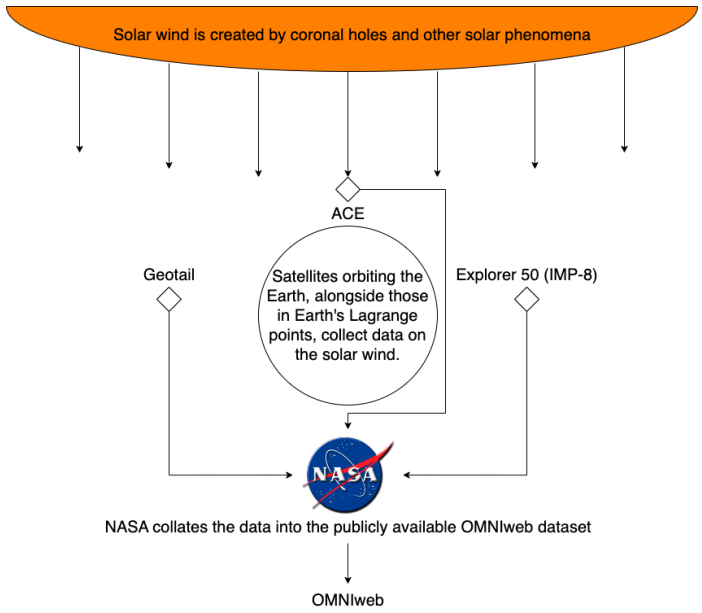


Fig. 3.2: The process by which NASA collects Solar Wind data and makes it available for public use.

where the forces due to the gravity of the sun and the Earth are balanced to allow for the satellite to orbit the Sun with the same period as the Earth. Explorer 50, also known as IMP-8, was launched in October 1973 and was able to detect solar wind for 7 to 8 days of every 12 days in orbit. In October 2001 Explorer 50 was terminated as an independent mission, and the last useful data from it was acquired in October 2006 [14].

3.4 Data Pre-processing

While these satellites have provided our data at a resolution of every minute and every five minutes, this resolution can have holes without data that are not good for neural network use. At the same time, a resolution this detailed can make the size of neural networks grow out of control. Due to this, we performed data preprocessing to obtain a resolution of one hour, detailed in algorithm 1.

3.5 Exploratory Data Analysis

Table 3.2 contains various statistics about the data used in this study. The magnetic field parameters are given in nanoTeslas (nT), the electric field parameter in millivolts

Algorithm 1 Data Preprocessing

Input: OMNIdata O, with a time resolution of 1-minute intervals, and offset l between the starts of input planes

Output: Preprocessed data P, with input planes of multivariate time series of length t

define data iteration variable $i_d = 0$
hours = time of last observation - time of first observation
Define history H such that it contains an entry $H_{i,j}$ for each hour i and each parameter j
for each hour i_h from 0 to hours **do**
    observationCount = 0
    hourData = 0 for each data entry
    while $i_h = O[i_d]_h$ **do**
        hourData += $O[i_d]$
        i_d+ = 1
        observationCount += 1
    end while
    if observationCount > 0 **then**
        H_{i_h} = hourData $\div$ observationCount
    else
        H_{i_h} = NaN
    end if
end for
Define list S, a list of length hours - t where True means we can start an input plane from that index
for i_s, iterator through S **do**
    if none of S from $i_s - l$ to i_s and H has no NaN values in i_s to $i_s + t$ **then**
        $S[i_s]$ =True
    end if
end for
Define P as a stack of n input planes of length l where n is the sum of S
for each true value in S, with i_s as the index, and i_p as the number of iterations of this loop **do**
    $P[i_p] = H[i_s : i_s + t]$
end for
return P

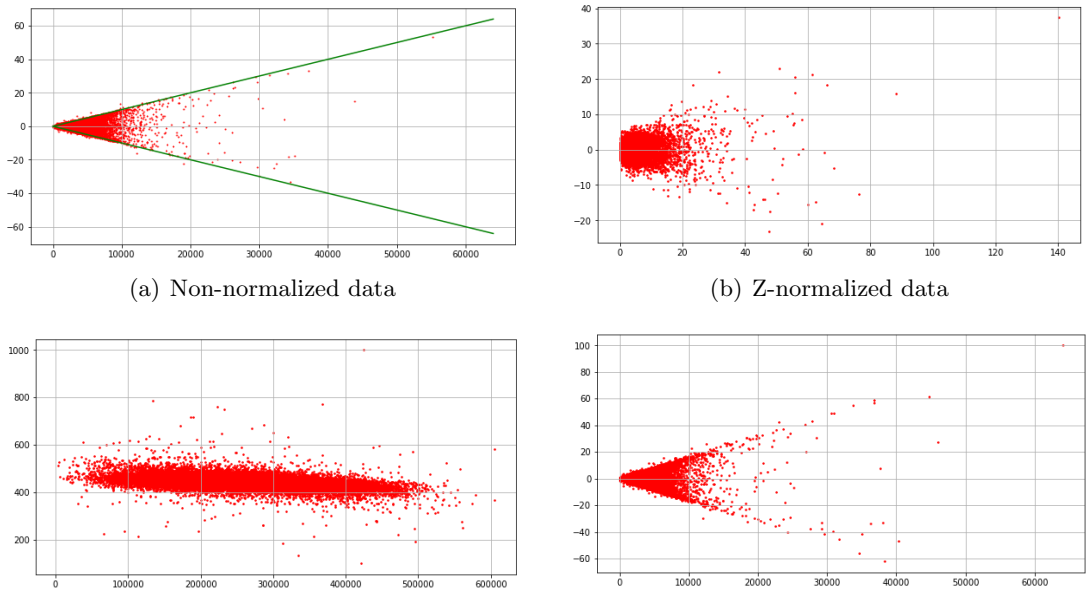
Variable	Total Mean	Total Standard Deviation	SC 22 Mean	SC 22 Deviation
E	0.006893	1.431	-0.02834	1.526
B_x	0.01461	3.520	0.1071	3.983
B_y	0.03156	3.987	-0.02035	4.319
B_z	0.01316	2.914	0.09527	3.033
V_x	-429.5	101.4	-441.3	105.5
V_y	-1.405	22.44	-2.838	21.45
V_z	-1.916	20.10	-0.7085	22.83
Variable	SC 23 mean	SC 23 deviation	SC 24 mean	SC 24 deviation
E	0.03096	1.604	0.006136	1.146
B_x	-0.06340	3.743	0.06063	3.016
B_y	0.1301	4.283	-0.05330	3.427
B_z	-0.01369	3.134	0.002621	2.553
V_x	-438.5	103.7	-414.6	93.59
V_y	-2.707	23.45	0.3380	21.26
V_z	-0.9008	19.89	-3.353	18.47

Table 3.2: Standard deviations and means of several solar wind parameters throughout different solar cycles

per meter (mV/m), and the velocity is in kilometers per second (km/s). All non-velocity parameters tend to stay close to zero, which is to be expected since the solar wind does not carry a very unbalanced charge.

3.6 Data normalization

Neural networks are highly sensitive to variations in the input data [15]; therefore, data normalization is an important step before model training. In this paper, we considered three data normalization techniques in addition to the raw non-normalized data. figure 3.3(a) shows the raw electric field (E) data as a function of $||v \times B||^2$. Figures 3.3(b), 3.3(c), and 3.3(d) show the same data when normalized using z-norm, 100 to 1000 norm, and the



(a) Non-normalized data

(b) Z-normalized data

(c) 100-1000 normalized data

(d) Max-100 normalized data

Fig. 3.3: Scatterplots of E as a function of $||\hat{V} \times \hat{B}||^2$ under different data normalizations. The green lines show the relationship $|E| - \alpha||\hat{V} \times \hat{B}||^2 = 0$ which follows the adapted Ohm's Law.

max-100 data normalization respectively.

3.6.1 No normalization

To establish a coherent baseline, we considered raw non-normalized multivariate time series as the first data product for training our models. The main limitation of using raw data stems from the different orders of magnitude of the input physical parameters. This difference in orders of magnitudes means that the loss is more affected by targets with larger values than by targets with smaller values for the same percent error. The raw non-normalized data follows physics Ohm's law ($|E| = ||\hat{V} \times \hat{B}||^2$) which constrains the range of possible values (see equation 4.3). To give a general sense of Ohm's law, the linear constraints are plotted in green as shown in Figure 3.3.

3.6.2 Z-normalization

A common form of normalization consists of fitting the data to a Gaussian distribution.

The normalized values correspond to the number of standard deviations away from the mean. Standardization is achieved by subtracting the mean of all the data and dividing it by the standard deviation. In an ideal context, about 68% of the data reside in the range [-1,1]. The limitation of standardization is that it does not retain the inter-series relationships.

$$x' = \frac{x - \mu}{\sigma} \tag{3.1}$$

3.6.3 Min-max normalization

One of the common data normalizations in the deep learning community is the min-max normalization from 0 to 1, which is achieved using Equation 3.2. We used a min-max normalization with $x_{\min} = 100$ and $x_{\max} = 1000$. Our motivation for choosing this range stems from our proposed physics loss that has a zero as a reference point. The [100-1000] range assures time series data values that are not null. Since the min-max normalization includes an additional step, the linear physics constraint is not well maintained after normalization. Figure 3.3(c) shows that while the strict less-than relationship is not as clearly visible as it is in Figure 3.3(a), the data does still cluster together.

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{3.2}$$

3.6.4 Max-normalization

Max-normalization is achieved by expressing data points as a factor of the maximum value in the time series as shown in Equation 3.3. The normalization maintains the relationship between pairs of data points and the reference point (zero). This common reference point allows the proposed physics-based loss to keep the data relationship invariant. Figure 3.3(d) shows the data distribution under max-normalization. The figure shows that conservative linear data constraints similar to Figure 3.3(a) could easily be applied.

$$x' = \frac{x}{x_{\max}} * 100 \tag{3.3}$$

CHAPTER 4

Ohm's Law Constraint

In this work, we propose a guided neural network that relies on a fundamental underlying relationship in the data, known as Ohm's law for an ideal plasma as defined in Equation 4.1 [2]. This chapter goes through the derivation of an equation that is usable as a loss function from Ohm's law for an ideal plasma and then defends the validity of that equation in the dataset chosen for this project.

4.1 Derivation

$$J = \sigma(E + V \times B), \tag{4.1}$$

where J is the current vector field, σ is the electrical conductivity scalar, E is the electric vector field, V is the velocity vector field, and B is the magnetic vector field. Since the solar wind is made up of about equal parts electrons and protons, $J \approx 0$. Equation 4.2 shows the modified Ohm's law for solar wind.

$$-E \approx V \times B. \tag{4.2}$$

One of our data-level limitations is that while the velocity and magnetic field parameters are expressed as the three vector quantities that they are, OMNI data only provides one scalar for the electric field. In order to combine all the parameters into Equation 4.2 we relaxed the equality into an inequality property. We used the knowledge that the magnitude norm of the product of three orthogonal component vectors is greater than or equal to the magnitude of the individual component vectors [16]. Since Equation 4.2 is a vector equality, the norm of the product of the velocity field and the magnetic field must be greater than the magnitude of any single component vector of the electric field as defined in Equation

4.3.

$$|E| - \alpha||\hat{V} \times \hat{B}||^2 \leq 0, \tag{4.3}$$

where α is a constant to allow for unit conversion.

4.2 Testing the validity of the equation

The validity of this equation is visible in Figure 3.3(a), where $\alpha||\hat{V} \times \hat{B}||^2$ is plotted on the x-axis and E is plotted on the y-axis. Each red point is a specific hour from our dataset before trimming into our desired size of input plane. Plotted in green is the equality relation.

While there are visible red dots outside of the green lines in Figure 3.3(a), this figure doesn't communicate the number of points in the red region well. Our dataset contains nearly two hundred and fifty thousand hours of useable data; the number of points that lie outside of the green bounds is quite small in comparison.

CHAPTER 5

Methodology

In this chapter, we take the modified Ohm's law equation 4.3 and prepare it to be used for generating loss for neural networks. The neural network architectures to be used are then proposed and justified. Finally, we propose a modified R-squared metric to allow for easier graphing.

5.1 Ohm's law-guided Neural Network for Solar Wind Prediction

To develop a physics-constrained neural network, we rely on a loss function that penalizes the network based on violations of physics laws, and a physics model input [17]. As discussed in Chapter 4, our data follows Ohm's law for an ideal plasma that we modified to Equation 4.3. Since there is a negativity constraint, we use a rectified linear unit (ReLU) as shown in Equations 5.1-5.2.

$$\mathcal{L}_{PHY}(\hat{Y}) = ReLU(H(\hat{Y}, Y)) \tag{5.1}$$

$$H(E, V, B) = -(|E| - \alpha||\hat{V} \times \hat{B}||^2) \tag{5.2}$$

In addition to the physics loss, we used the Root Mean Square Error (RMSE) as the traditional loss used for continuous values as defined in Equation 5.3. To find a balance between the two losses, we use a hyper-parameter λ. Our proposed combined physics-constrained loss function is shown in Equation 5.4.

$$\mathcal{L}_{RMSE}(\hat{Y}, Y) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{5.3}$$

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_{RMSE}(\hat{Y}, Y) + \lambda * \mathcal{L}_{PHY} \tag{5.4}$$

The value of this total loss function $\mathcal{L}$ is then backpropagated to the network using the publicly available ADAM optimizer in the PyTorch library, as proposed by [18]. This leaves our cross-fold validation sequence as dictated in Algorithm 2

Algorithm 2 Grid search with 3 fold cross validation

Input: OMNIdata O, collated into two-dimensional planes composed of d time series of length l.

Output: The training loss over all 700 epochs, as well as the result of the single tuple of predictions on the test data

```
Divide O into O_1, O_2, O_3 with 30% of the data each, and O_t with the remaining 10% for testing
for each set of hyperparameters and network in grid do
    for i in [1,2,3] do
        initiate model using current set of hyperparameters
        with O_j such that O_j = O_{1,2,3} − O_i
        for 700 epochs do
            calculate the predictions of the model on O_j
            calculate the loss using RMSE
            if Hyperparameters include physics loss then
                calculate physics loss and add to RMSE Loss
            end if
            backpropagate the loss through the network using Adam
            print the loss to file
        end for
        calculate validation loss on O_i
        print validation loss to file
    end for
    generate predictions using the trained model on O_t
    calculate test R-squared metric
    print R-squared metric value to file
end for
```

5.2 Baselines

We consider five deep learning baselines in our study that we trained with our proposed $\mathcal{L}$ loss and without physics loss $\mathcal{L}_{phy}$. In this section, we outline the details of our baselines.

5.2.1 Time-based CNN

Convolutional Neural Networks (CNNs) work on data of sequential nature (e.g., images

and maps) that have an underlying dependence between contiguous data points. Since our multivariate time series data are sequences, we used a modified CNN for the prediction. The CNN modifications involve using two types of kernels: $1 \times n$ kernels that modify the univariate time series across the time dimension, and $m \times n$ kernels with dimension m being equal to the number of univariate time series contained in the multivariate time series. These changes allow our CNN to be aware of how the variables change with time, as well as how each variable relates to all other variables.

5.2.2 ResNet

Residual Neural Network model (ResNet) is a deeper model that learns how to modify the existing data to match their correct probability distribution. ResNet models utilize skip connections to jump over some layers to avoid the vanishing gradients problem. To augment the ResNet model for time series data, we used the same kernel shapes as used in the time-based CNN. The other benefit of these skip connections is that it allows our data to move toward the final step more easily; since each input parameter is reflected in an output parameter, this means that our output parameters are more likely to be more correct initially.

5.2.3 RotateNet

RotateNet network takes the multivariate time series input and makes a neural model that learns how to distinguish between the different geometric transformations (rotations) performed on the normal multivariate time series matrix. RotateNet is the first method to discover multivariate time series anomalies by constructing a self-supervised classification model [19]. The auxiliary expertise learned by the model generates feature detectors that effectively forecast the next time steps.

5.2.4 LSTM

Long Short-Term Memory (LSTM) is a memory-based model that can handle the complexity of sequence dependence among the input variables by maintaining a state (memory)

across very long sequences. One of the ways the model addresses the exploding and vanishing gradient problem is by having a forget mechanism that ignores data points based on a probabilistic model.

5.2.5 GRU

Gated Recurrent Unit (GRU) model has been developed to improve the LSTM model by adding extra gates to allow the network to extract important bits from sequences [20]. GRU is simpler and faster to train than the LSTM model as they do not require a memory unit. The GRU model generally performs better on short time sequences compared to the LSTM model.

5.3 Experimental Setup & Results

To evaluate our proposed baselines with our proposed $\mathcal{L}$ loss and without physics loss $\mathcal{L}_{phy}$, we split the data into training, validation, and testing sets and used the R-squared measure as defined in Equation 5.5. Before training our models, we performed a grid search over the architectures and hyper-parameters of the baseline models. The grid searches were run on data with a 60/30/10 split, where the first partition is used as training data, the second as validation data, and the final withheld partition as testing data. After seven hundred epochs of training, each network was evaluated on the testing data. We repeated the process three times and averaged the scores to find the optimal hyperparameters with which we could start our experiments detailed in Chapters 6, 7, and 8.

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{5.5}$$

One obstacle presented by using R-squared is the potentially unbounded lower values. The second term in equation 5.5 cannot be less than zero, as both the top and bottom are squared real values. This puts a hard upper limit at 1 but means that a bad enough prediction engine can have arbitrarily bad values. These values can be so large as to completely destroy any graph they are shown on, so we propose equation 5.6

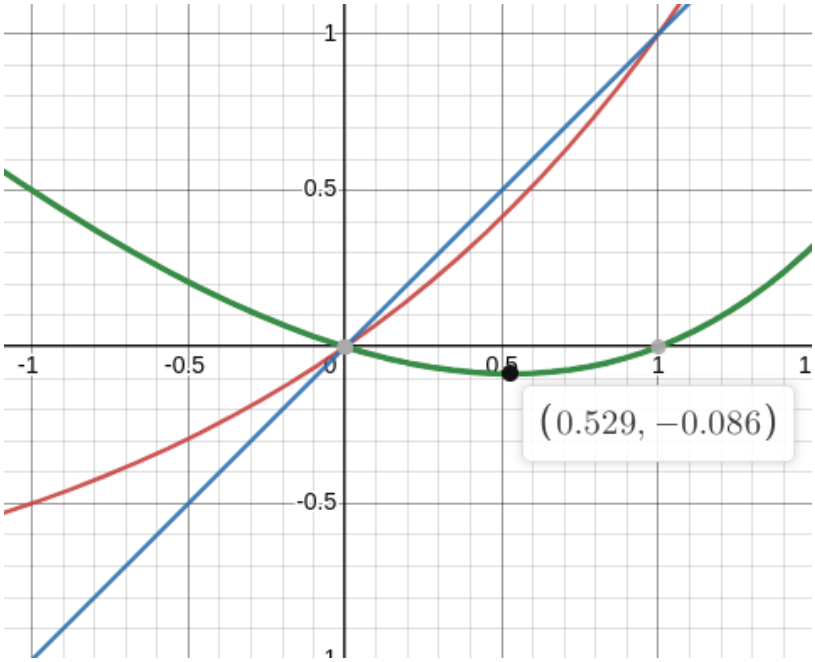


Fig. 5.1: A graph of modified R-squared shown in red versus the input R-squared metric in blue. The difference between the two is plotted in green, with the maximum difference shown as a point. Graph provided by Desmos.com.

$$R_m^2 = 2^{1-\frac{\sum_i (y_i-\hat{y}_i)^2}{\sum_i (y_i-\bar{y})^2}} - 1 \tag{5.6}$$

By using the standard R-squared metric as the exponent of 2, shifted down by one, we retain a decent approximation of the standard R-squared metric on the range [0,1], maintain the sign of all values, and change the prior infinite lower bound to a lower bound of -1 as the standard R-squared metric approaches negative infinity. While this may cost the natural intuition on the negative values, it allows us to see the complete range of R-squared values possible, and those given to us by the networks. In Figure 5.1, our modified R-squared is plotted, alongside the original R-squared and the difference between the two. The largest difference between the positive input values and their respective outputs is roughly 8%, at about 5.29. As the inputs go off to negative infinity, our modified values approach negative one.

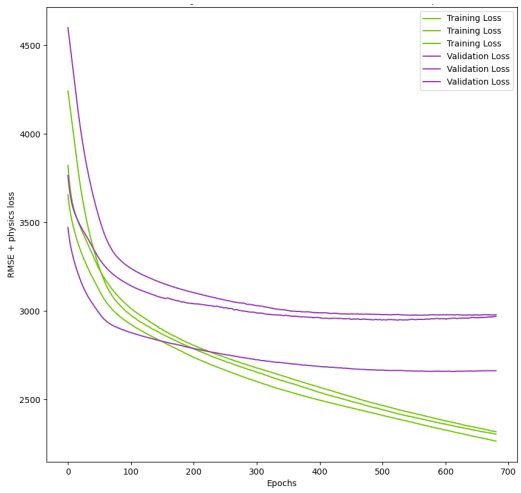


Fig. 5.2: Training and Validation loss over 700 epochs for a time-based convolutional neural network.

5.4 A check on the validity of our routine

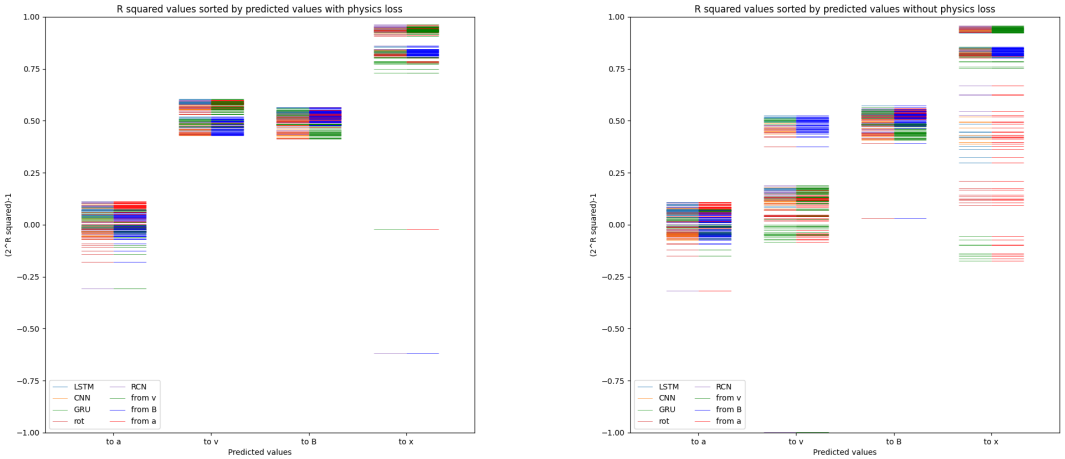
Figure 5.2 shows the training and validation loss of a single convolutional neural network over the seven hundred epochs that we train it on. This graph contains no major surprises; at the beginning of the training cycle, loss is higher, and it decreases as more training is done. As training continues, the validation accuracy of the model starts to plateau, showing that the model has learned to its capacity. More training is likely to introduce overfitting, where the model learns the particulars of the training data set instead of the general trends that are present in both the training and the validation sets.

CHAPTER 6

Feature and Target Selection

Feature selection is a powerful tool that helps explain which inputs best explain differences in output; usually, it is used for problems with a single output. In our case, we also had multiple targets used for regression, so we applied a similar process to the targets our networks were trained on. We selected three categories of inputs: all seven available components of Ohm's law, designated a; the 3-vector of velocity v; and the 3-vector of the magnetic field B. We also selected four categories of outputs: the prior three inputs, as well as just the earthbound component of velocity, shown in these figures as x. We also show here the difference in using physics based loss for all of these options. We then trained each of our five networks on each possible pairing of inputs, outputs, and physics loss ten times, for a total of 1200 individual training and testing cycles. This chapter analyzes the results of this experiment.

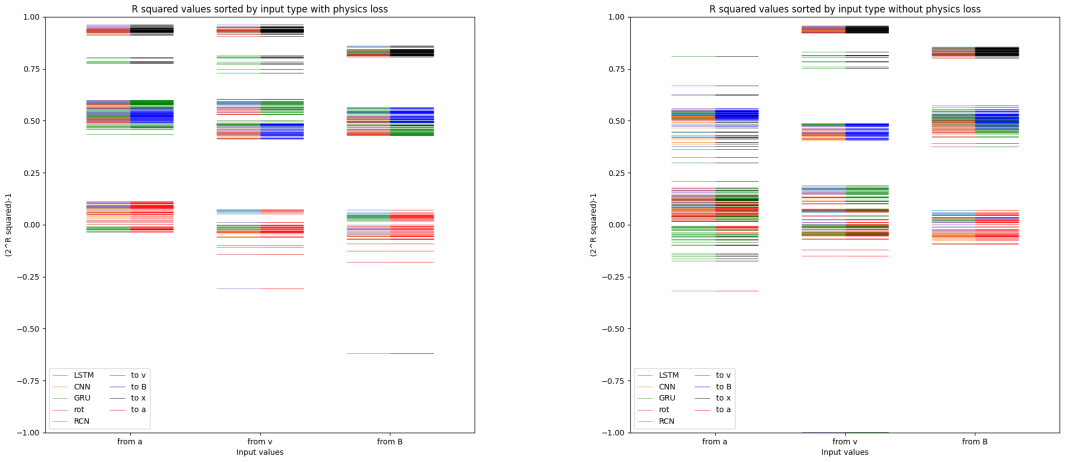
Figures 6.1 to 6.3 each contain all results generated in this process. Due to the four input parameters (input set, target set, network, if physics loss is enabled) and one output parameter (modified R-squared, covered in section 5.3), we are faced with the difficult prospect of showing a five-dimensional space on a two-dimensional page. To overcome this challenge, we segregate the values based on if physics loss was enabled, with the enabled loss on the left and with no physics loss on the right; then plot the transformed R-squared values on the y-axis and place each input parameter on the x-axis on separate graphs. To give the graphs a measure of homogeneity, we plot each R-squared value as a line composed of two parts, each part being a color corresponding to one of the non-axis values. These colors are consistent throughout all three figures; V_x, shown as x, is always plotted in black; all parameters discussed in the introduction, or a, is always plotted in red; etc. After we analyzed the data, we performed a one-sided T-test on the data when grouped by network, for a sample of 50 runs. The p-values from these tests are plotted in Figure 6.4.



(a) With physics loss

(b) Without physics loss

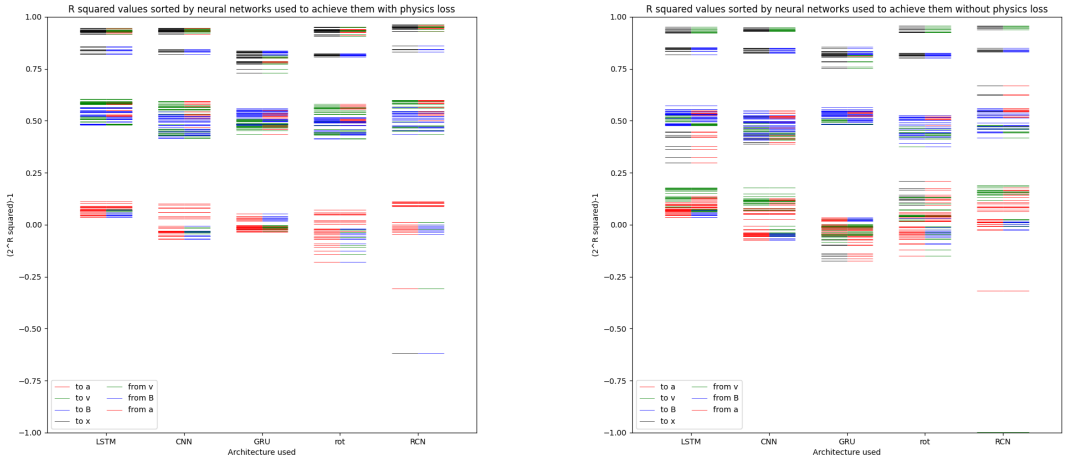
Fig. 6.1: Event plots showing the transformed R-squared values of our sequence analysis, grouped by target.



(a) With physics loss

(b) Without physics loss

Fig. 6.2: Event plots showing the transformed R-squared values of our sequence analysis, grouped by inputs.



(a) With physics loss

(b) Without physics loss

Fig. 6.3: Event plots showing the transformed R-squared values of our sequence analysis, grouped by network.

As is easily visible in Figure 6.1 and quite apparent in the others, our networks did best when asked to predict the only output set with only a single value: x. When attempting to predict a smaller set of output variables, the final linear layers in every neural network can carry more information related to the desired output. Networks that have to predict the three vector quantities of B or v have to fit the information for three variables into the latent space, while networks predicting a need to divide the latent space between all seven regression targets. However, this division of latent spaces is only strictly harmful when the networks predicting x have access to the physics based loss. When that extra penalization is not applied, the networks predicting B or a continued to do well due to the correlation between the output fields taking on a similar role. This is also visible when predicting v from B; it appears that the self-correlation from the B field also performs the function of the physics based loss when considered as an input. The opposite is visible in networks that learn to predict v and x; without the physics loss to help provide the relationship between the output parameters, these networks perform considerably worse without any other changes.

Figure 6.2 continues the pattern of either extremely stark effects or hardly any effects at all. Notable is the lack of high values predicting x from a; this continues the counter-intuitive pattern from Chapter 8 when more data often meant less accuracy. It is visible here that physics based loss can help networks pick out which bits of an input space are more important.

As a more comprehensive analysis of Figure 6.2, the drop in accuracy of predicting x from v and a with physics loss as opposed to predicting x from B is expected, since both a and v contain x, while B does not. Another expected variation is the switch of networks predicting B and those predicting v over the columns corresponding to predicting from B to predicting from v; naturally, networks with input of the same type as their output will usually outperform networks trained on data only related to the data they are trying to predict. Also visible is the rather large difference between networks predicting v from inputs of v discussed in the analysis of Figure 6.1.

Figure 6.3 is perhaps most interesting due to its *lack* of variability over columns and physics based loss. While the effects discussed in the prior paragraphs, namely, the wider distribution of lower points when not using physics loss, and the improvements in networks predicting x from a, are still visible, not very much else changes between columns and between networks informed by the physics loss and those without. Four of the five network architectures have an upper band from roughly 0.95 to 0.9; all have a high band from 0.85 to 0.8, a middle band from 0.6 to 0.4, and a low band from 0.1 to -.25. The only exceptions are the lack of an upper band for GRU networks, and the extremely low values attributable to the fragility of residual-based neural networks.

When we consider the differences and variability discussed in the prior paragraphs, the results shown in Figure 6.4 are exactly as we expect. Predicting v from v and a as well as predicting x from a benefit from the additional structure provided by the physics based loss, where the other target and input combinations do not benefit from it. This is a very interesting phenomenon that shows that a given physics based loss equation can perform well in one situation while performing quite poorly in another. All seven of the

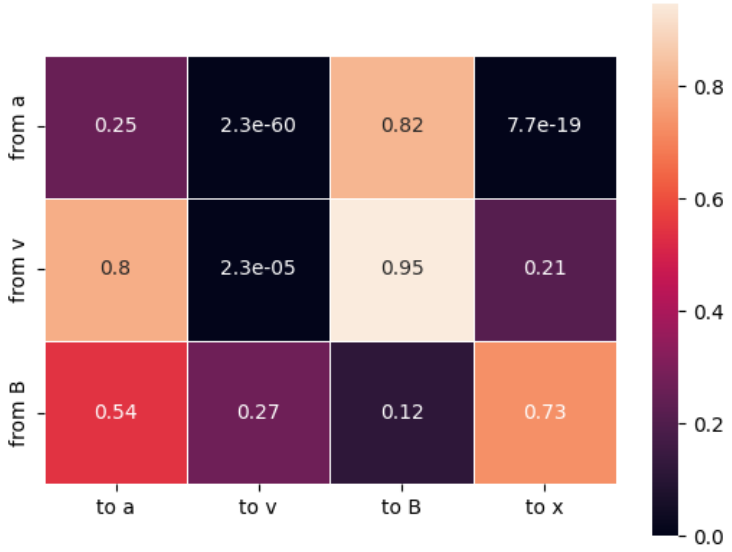


Fig. 6.4: The p-values measuring the effectiveness of Ohm's law physics-based loss (n=50).

predicted and input values are subject to Ohm's law for an ideal plasma, but Ohm's law is only *useful* in niche circumstances. It is the opinion of the authors that when considering physics-guided neural networks, the set of input and output parameters should be iterated over in the same initial grid search used to optimize networks for the rest of the experiments planned.

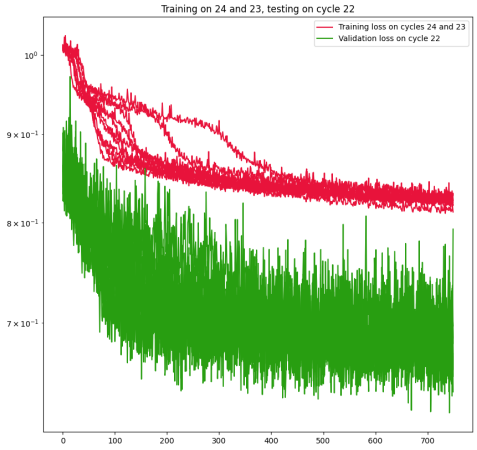
CHAPTER 7

Case Study: Solar Cycles

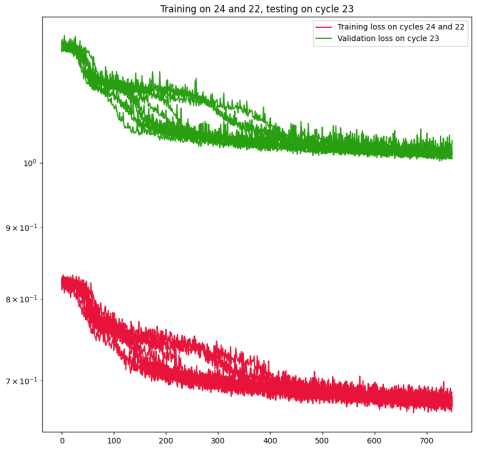
To start our analysis of the data used in our solar wind prediction, we modified our cross validation routine to perform on separate solar cycles instead of randomly shuffled data. This was done with all seven parameters as input and as output. We then picked the network hyper-parameters that produced the best results most consistently, trained the networks that used them, and compared the results. These parameters were: an LSTM with no physics-weighted loss on Z-normalized inputs; a time-based CNN with a loss function with physics weight 0.3 on input data normalized from 100 to 1000; a ResNet trained with a physics loss factor of 0.3 on input data normalized from 100 to 1000; a GRU network trained on data normalized to have a maximum magnitude of 100 with a physics loss factor of 0.0001; and a rotateNet implementation that was trained on data normalized from 100-1000 with a physics loss factor of 0.003. The results of these experiments are shown in Figures 7.1, 7.2, 7.3, 7.4, and 7.5 respectively. The rest of this chapter will analyze these figures.

7.1 Baseline neural networks Vs. those in this experiment

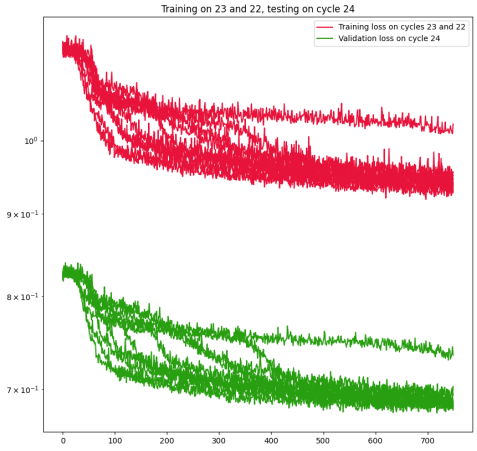
Before we can start to analyze these runs, it is helpful to understand the more typical nature of how neural network training and testing works, so that we can contrast this experiment to what should be expected. A good example of this is shown in Figure 5.2. As the routine performs the work of each epoch, the network is slowly improved using the information available from the loss obtained from the training set. Using this loss, the network learns the patterns inherent in the training set. For the beginning epochs, the patterns learned tend to be general patterns that apply to the testing set; but as time goes on, the network starts to learn more of the specific patterns of the training set that do not appear in the testing set. Given sufficient epochs, the training routine will cause the model



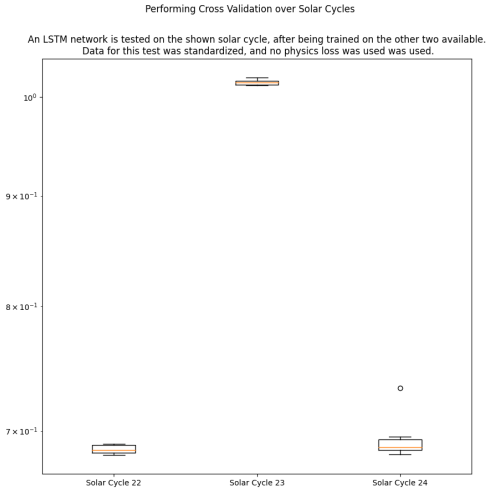
(a) Testing on solar cycle 22



(b) Testing on solar cycle 23

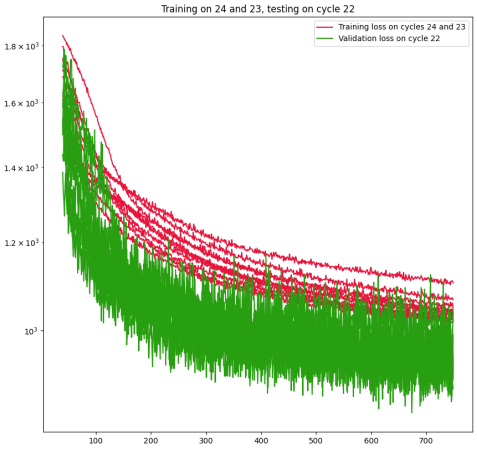


(c) Testing on solar cycle 24

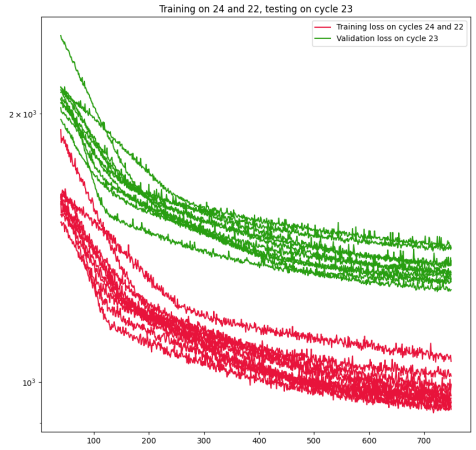


(d) Box plot of the various final testing losses over ten runs.

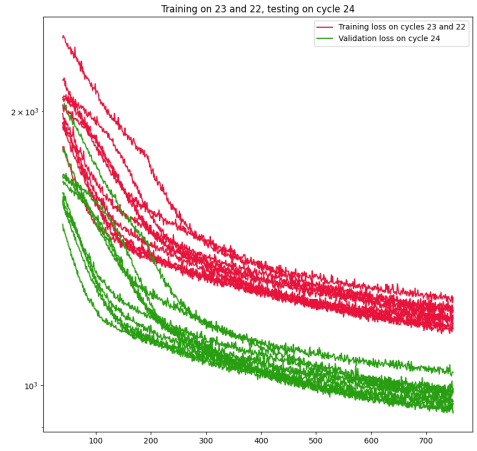
Fig. 7.1: Results of an experiment in which an LSTM with hyperparameters derived from earlier experiments was tested on the captioned solar cycle, having been trained on the remaining two. This was repeated ten times to establish a mean. Subfigure 7.1(d) contains a box plot of the final testing accuracies of the runs, graphed on the same axis to allow for comparison. All figures are log scaled.



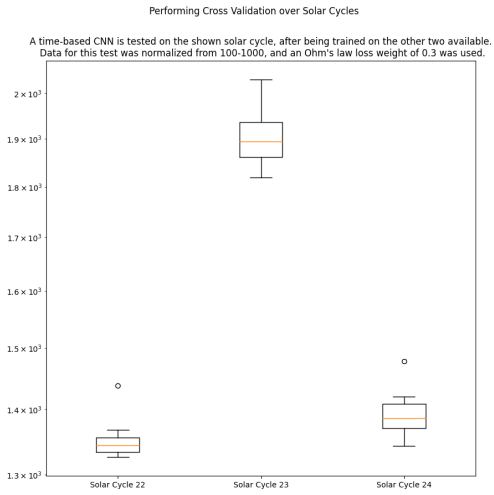
(a) Testing on solar cycle 22



(b) Testing on solar cycle 23

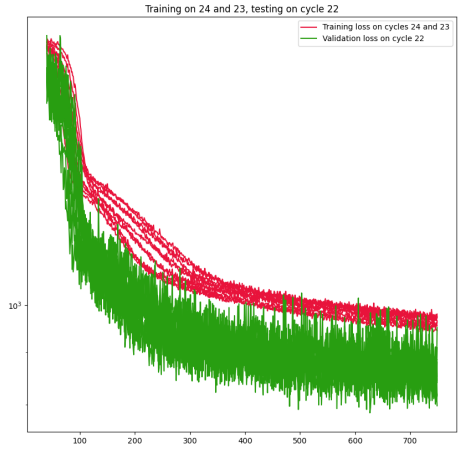


(c) Testing on solar cycle 24

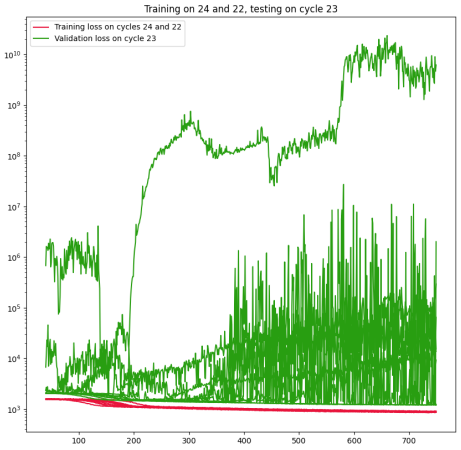


(d) Box plot of the various final testing losses over ten runs.

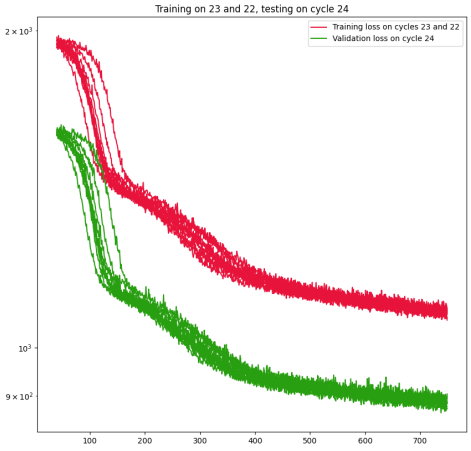
Fig. 7.2: Results of an experiment in which a time-based CNN with hyperparameters derived from earlier experiments was tested on the captioned solar cycle, having been trained on the remaining two. This was repeated ten times to establish a mean. Subfigure 7.2(d) contains a box plot of the final testing accuracies of the runs, graphed on the same axis to allow for comparison. All figures are log scaled.



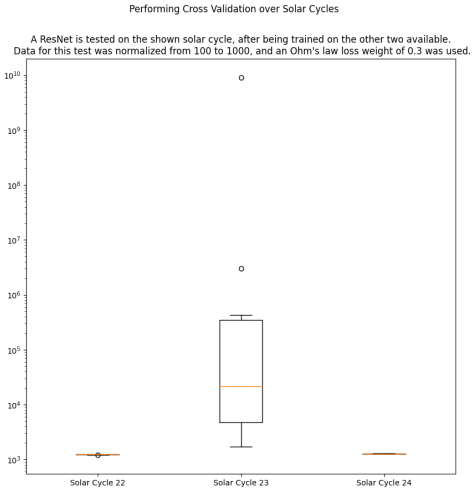
(a) Testing on solar cycle 22



(b) Testing on solar cycle 23

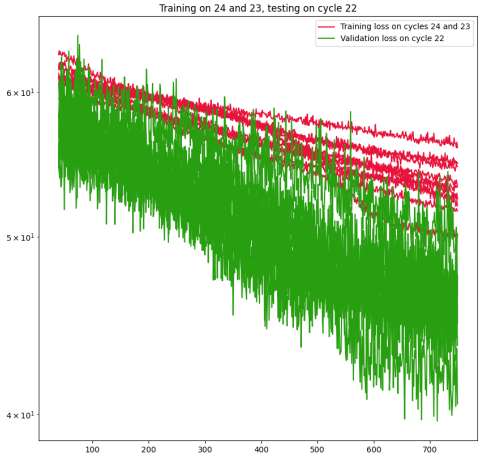


(c) Testing on solar cycle 24

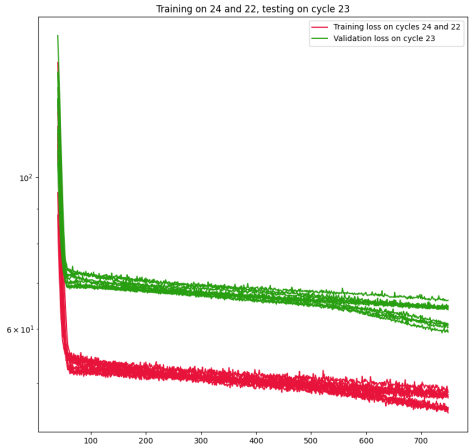


(d) Box plot of the various final testing losses over ten runs

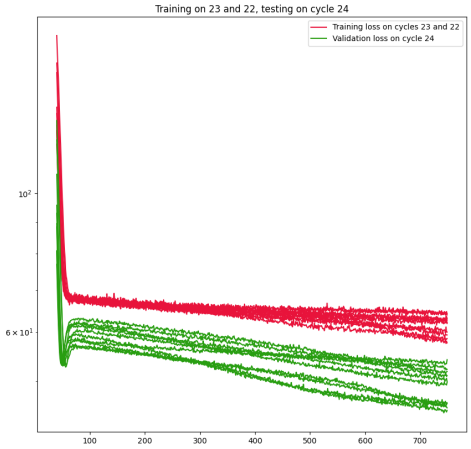
Fig. 7.3: Results of an experiment in which a ResNet with hyperparameters derived from earlier experiments was tested on the captioned solar cycle, having been trained on the remaining two. This was repeated ten times to establish a mean. Subfigure 7.3(d) contains a box plot of the final testing accuracies of the runs, graphed on the same axis to allow for comparison. All figures are log scaled.



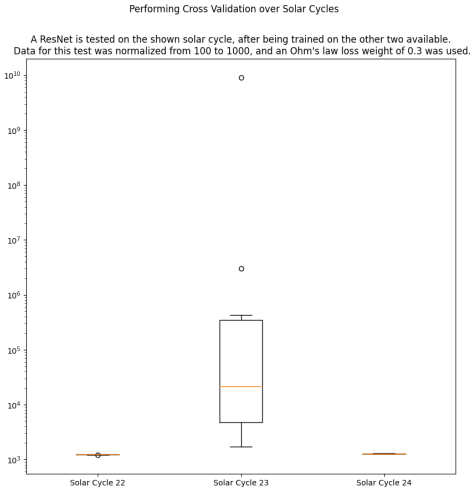
(a) Testing on solar cycle 22



(b) Testing on solar cycle 23

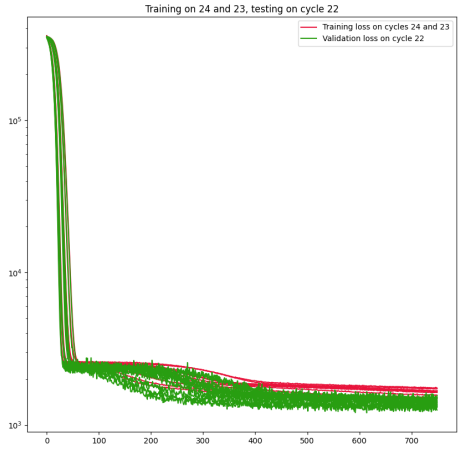


(c) Testing on solar cycle 24

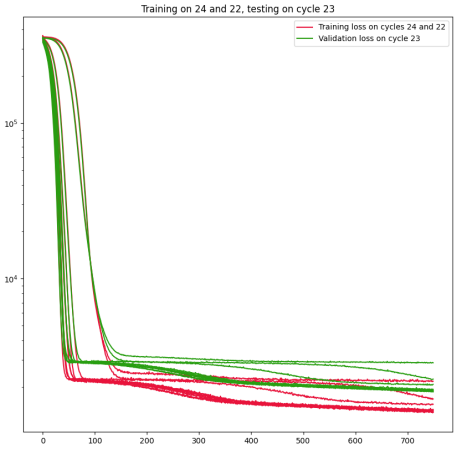


(d) Box plot of the various final testing losses over ten runs

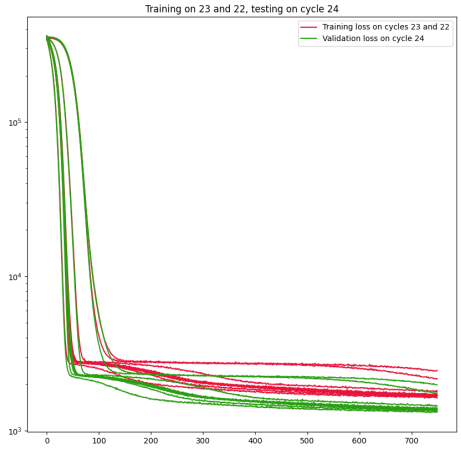
Fig. 7.4: Results of an experiment in which a GRU network with hyperparameters derived from earlier experiments was tested on the captioned solar cycle, having been trained on the remaining two. This was repeated ten times to establish a mean. Subfigure 7.4(d) contains a box plot of the final testing accuracies of the runs, graphed on the same axis to allow for comparison. All figures are log scaled.



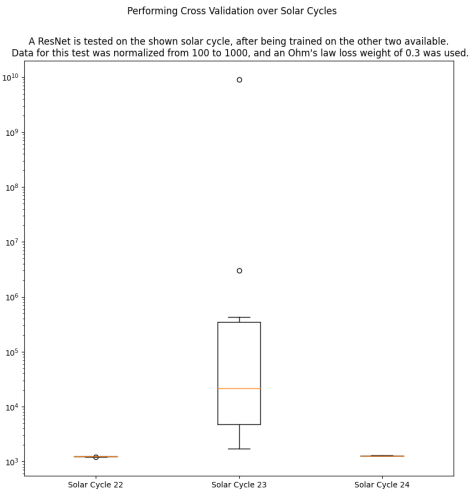
(a) Testing on solar cycle 22



(b) Testing on solar cycle 23



(c) Testing on solar cycle 24



(d) Box plot of the various final testing losses over ten runs

Fig. 7.5: Results of an experiment in which a RotateNet with hyperparameters derived from earlier experiments was tested on the captioned solar cycle, having been trained on the remaining two. This was repeated ten times to establish a mean. Subfigure 7.5(d) contains a box plot of the final testing accuracies of the runs, graphed on the same axis to allow for comparison. All figures are log scaled.

to overfit the data, where the network eschews the general patterns of the input space in favor of the more specific patterns only found in the training set. This phenomenon leads to the generally true assumption that testing accuracy is lower than training accuracy.

This key relationship is broken when performing this test; while networks trained on solar cycles 22 and 24 and tested on solar cycle 23 keep this behavior, networks trained on 23 and 24 and tested on 22 as well as networks trained on 22 and 23 and tested on 24 have better testing accuracies than training. This is visible across all five experiments. Also visible across all five experiments is the erratic nature of the testing accuracy on solar cycle 22, and the strong relationship between training and testing losses of networks trained on solar cycles 22 and 23 that are tested on solar cycle 24. We will now analyze the reasons for each of these in detail.

7.2 Networks trained on solar cycles 22 and 24, then tested on solar cycle 23.

Neural networks take data from a high dimensional input space and propagate forward into a different output space. To have an accurate neural network, it must be trained on a representative portion of the input space. When the training set does not form a representative sampling of the input space, the network must extrapolate to attempt to predict the output of an input not sampled previously. The results of the experiments discussed in this section lead us to consider the spaces of each variable in our input; kernel density estimates for all seven active variables over all three solar cycles are plotted in Figures 7.6, 7.7, 7.8, 7.9, 7.10, 7.11, and 7.12. Densities from solar cycle 24 are plotted in red, densities from solar cycle 23 are plotted in green, and densities from solar cycle 25 are plotted in blue. Plotted normally, these estimates look identical, but when plotted on a log scale, we can start to see the outliers that make such a difference. In fact, out of the seven variables shown, solar cycle 23 has both the maximum and the minimum value for five of them. This means that networks trained like those in this section have to make predictions on data that they have not been trained on. This leads to massive errors, which are easily visible in the subfigure b of Figures 7.1 through 7.5.

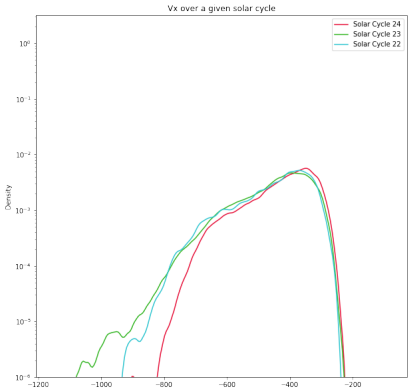


Fig. 7.6: A kernel density estimate of the distribution of the X component of the velocity field.

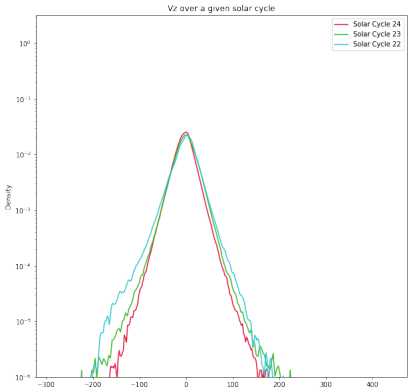


Fig. 7.8: A kernel density estimate of the distribution of the Z component of the velocity field.

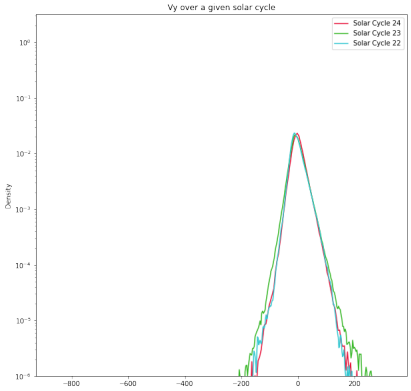


Fig. 7.7: A kernel density estimate of the distribution of the Y component of the velocity field.

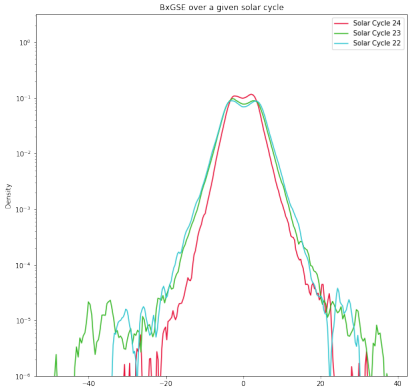


Fig. 7.9: A kernel density estimate of the distribution of the X component of the magnetic field.

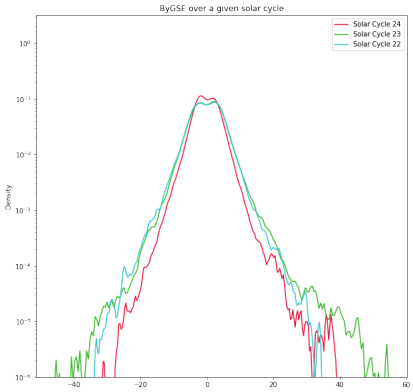


Fig. 7.10: A kernel density estimate of the distribution of the Y component of the magnetic field.

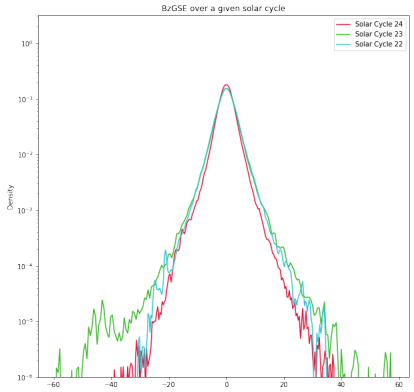


Fig. 7.11: A kernel density estimate of the distribution of the Z component of the magnetic field.

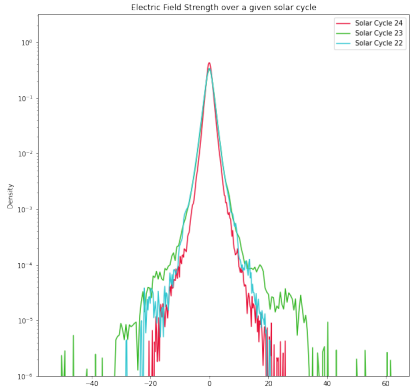


Fig. 7.12: A kernel density estimate of the distribution of the Electric Field strength.

A logical explanation for the distributions shown in Figures 7.6 through 7.12 comes from the number of sunspots observed throughout each cycle, shown in Figure 7.13. Sunspots are holes in the sun's photosphere, which allow the highly pressurized matter contained within it to escape. This phenomenonis one of the main sources of the fast solar wind. Due to these particles moving faster, a larger magnetic field is created, and since a higher flow rate means the number of particles passing through a given space is larger, the electric field is stronger as well. The reason that solar cycle 22 does not exhibit the same issues is mostly

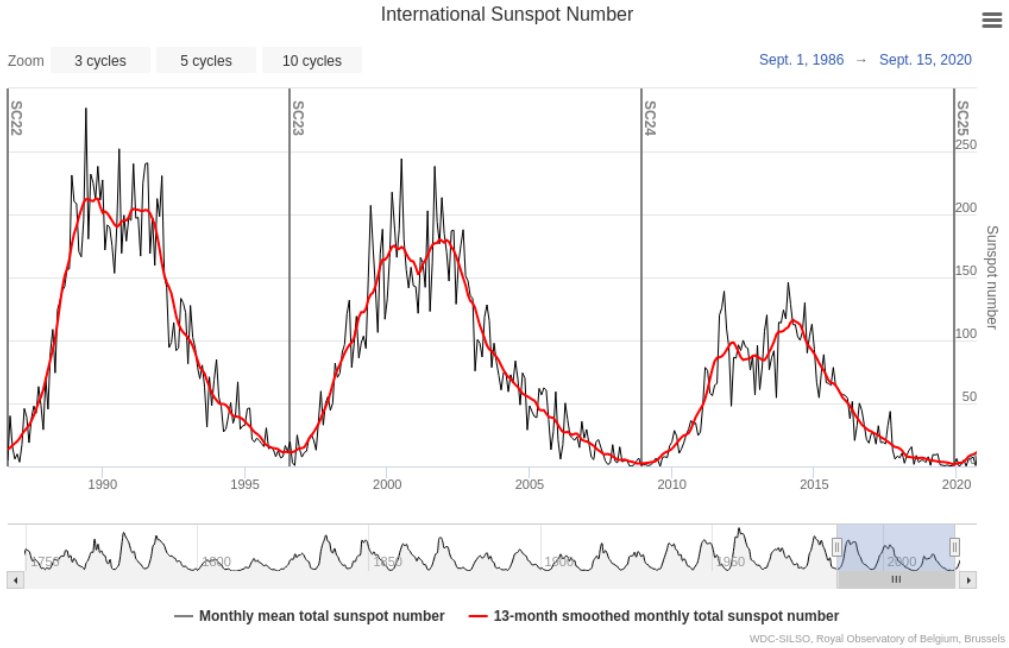


Fig. 7.13: A plot of the mean number of sunspots in a given month, from the beginning of solar cycle 22 in 1986 to the end of solar cycle 24 in 2020. A 13-month average is shown in red. Image courtesy of spaceweatherlive.com

due to the amount of data available from said cycle, discussed more in section 7.3.

7.3 Networks trained on solar cycles 23 and 24, then tested on solar cycle 22.

While section 7.2 showed the dangers of a training set that is not representative of the input space, networks trained like those in this section show the pitfalls of a testing set that is not representative of the output space. The danger with networks trained like those in this section is easily apparent when viewing how many complete tuples are available in each solar cycle, where each tuple includes at least one measurement of each variable over the space of an hour. While solar cycles 23 and 24 have roughly one hundred thousand tuples each, solar cycle 22 only manages about 440 thousand. This difference is exacerbated when we consider that for our purposes we require a streak of twenty-four complete tuples, along with one more complete tuple twelve hours away to use as a target. With these constraints, cycles 23 and 24 have roughly 4.8 thousand data, while cycle 22 comes in with just over a thousand. This is visualized in Table 7.1 This disparity means that data collected from

solar cycle 22 has trouble capturing all the patterns in the output space. This means the loss, which uses an arithmetic mean to standardize results across the validation experiment, is more sensitive to the patterns that are found in the available data for solar cycle 22.

7.4 Networks trained on solar cycles 22 and 23, then tested on solar cycle 24.

The results of the networks discussed in this section show us the incredible relationship between the data in solar cycle 24 and the data in solar cycle 23, and to a lesser extent, 22. Where validation accuracy for the networks from section 7.3 was noisy and highly variable, networks trained on cycles 22 and 23 and validated on solar cycle 24 kept a very firm relationship, with an increase in training accuracy reflected in real-time by the validation accuracy. Despite all training runs and validation runs sharing the colors red and green respectively, it is not difficult to see which validation line should go with which training line. This points to a key relationship between the underlying distributions of each cycle: the data from solar cycle 24 follows the same patterns as solar cycle 23, and to a lesser extent 22, but with values that are much closer to the mean. Due to the factors discussed in this section, as well as those discussed in sections 7.2 and 7.3, we recommend training in this way for future networks. Solar cycle 23, augmented with cycle 22, provides a more complete sampling of the input space than any other mixture, and solar cycle 24 provides a good test to see if the network is correctly learning the general patterns present in the physical phenomena, or is merely overfitting to patterns available only in the sampled data.

Table 7.1: Counts of hours with available data. Also shown are counts of non-overlapping input planes of given numbers of hours.

Solar Cycle	Total valid hours	6 hours	12 hours	24 hours	36 hours	48 hours
24	101710	14175	7345	4876	3553	2785
23	101096	14110	7328	4853	3582	2819
22	43439	5130	2054	1094	582	407
Total	246245	33415	16722	10823	7717	6011

CHAPTER 8

Sequence Analysis

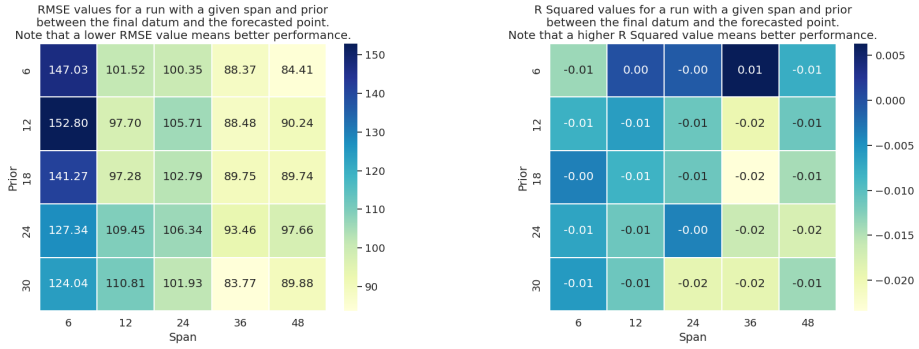
In addition to understanding the progression of the data from one solar cycle to the next, we seek to understand the width of our input space and the relationship between the input space and different lengths of output space. This process is called sequence analysis. This chapter will explain the procedure we used to do this, and then go through the results of each network, as well as the best results achieved overall.

8.1 Procedure

In this experiment, we varied the *prior* (the length of time between the input plane and the output vector) and the *span* (the length of the input plane). The *prior* was given possible values of 6, 12, 18, 24, and 30. The *span* was chosen from the numbers 6, 12, 24, 36, and 48. Each combination of *span* and *prior* had each network trained on it ten times, using three-fold cross-validation each time. The three instances of each network were then tested on ten percent of the data that had been withheld prior to training, and the average of these three was calculated for each of the ten runs of a given network architecture. The minimum test RMSE for each architecture class is shown in part (a) of Figures 8.1 through 8.4. The maximum R-squared value from each architecture class is shown in part (b) of the same figures. Figure 8.6 shows the maximum over all five networks and the mean of the maximums from each individual network.

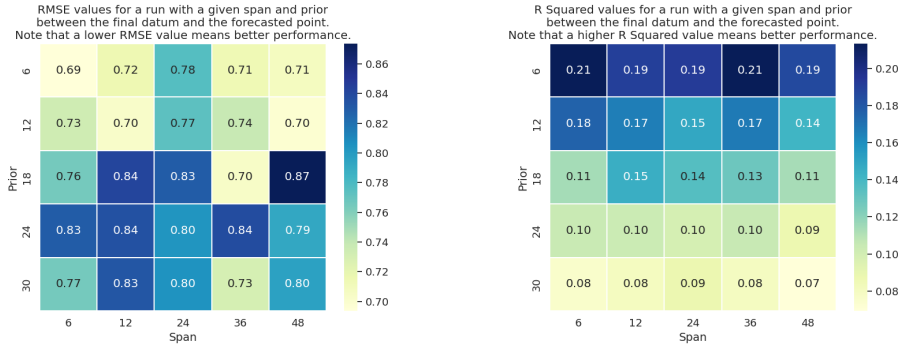
8.1.1 GRU

Figure 8.1 shows the results of the sequence analysis when performed on a GRU network. This network was designed at its inception to learn the patterns between different points in a time series, and this shows in the RMSE results. As the length of the input plane grows, there are more opportunities for the network to find the patterns; this decreases the



(a) RMSE (b) R-squared

Fig. 8.1: Analysis of the effect of different *priors* and different *spans* on our best-performing GRU network, in both mean squared error, and R-squared metrics. Note that while a low MSE is preferred, a higher R-squared means better performance.



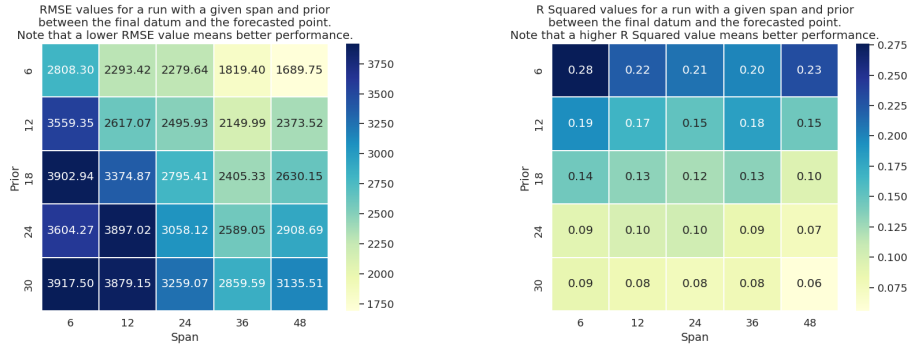
(a) RMSE (b) R-squared

Fig. 8.2: Analysis of the effect of different *priors* and different *spans* on our best-performing LSTM network, in both mean squared error, and R-squared metrics. Note that while a low MSE is preferred, a higher R-squared means better performance.

error. The relative stability of the error would seem to indicate that the size of the prior is not as important to this network, but when we look at the values of the R-squared metric, we can see that this network is mostly guessing close to the mean.

8.1.2 LSTM

Figure 8.2 shows a stark contrast to Figure 8.1; instead of being roughly invariant over different *priors*, our LSTM architecture much preferred shorter *priors* and was reasonably resilient against a change in *span*. This is likely due to the design of an LSTM; since



(a) RMSE (b) R-squared

Fig. 8.3: Analysis of the effect of different *priors* and different *spans* on our best-performing ResNet, in both mean squared error, and R-squared metrics. Note that while a low MSE is preferred, a higher R-squared means better performance.

an emphasis is made on having the ability to remember information for an indeterminate amount of steps, the LSTM is much better equipped to deal with long or short *spans*, whereas the GRU preferred longer *spans*. This is born out in 8.2(b), which shows that our network does have some real predicting power. As the input plane is separated from the output vector, our accuracy predictably lowers. An experimental artifact remains in 8.2(a); Since we performed our grid search optimization on a *span* of 24 hours, the hyperparameters we chose to use in this experiment favored the *span* of 24 hours.

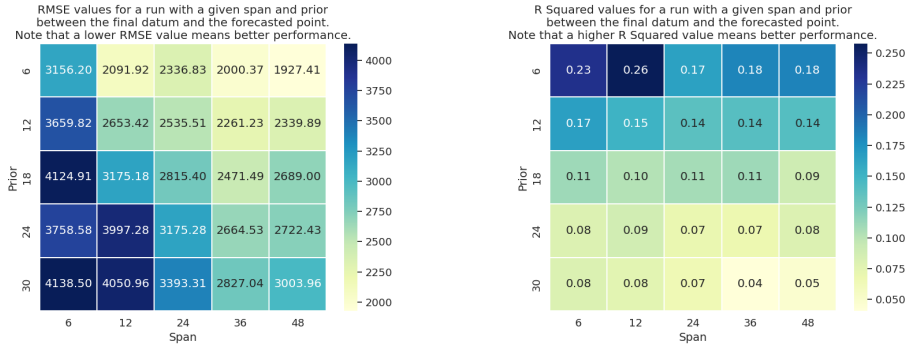
8.1.3 ResNet

Figure 8.3 shows a much more typical sequence analysis. As the network receives less data, it is less able to fit well enough to predict correct outcomes; and as the input plane moves away from the predicted vector, the relationship between the two becomes fainter. The higher R-squared value in the upper right corner is likely due to a data artifact; since we only use input and output planes with full data, many possible data points that a *span* of 6 can fit inside comfortably must be rejected for higher *spans*. This idea is validated by Table 7.1.

8.1.4 RotateNet and time-based CNN

(a) RMSE (b) R-squared

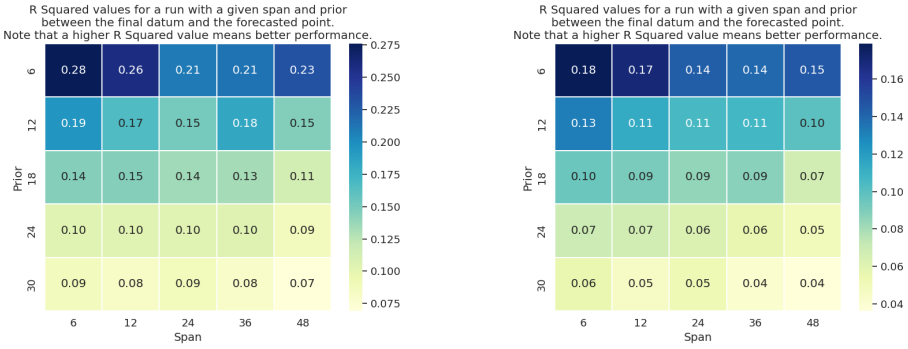
Fig. 8.4: Analysis of the effect of different *priors* and different *spans* on our best-performing rotational resNet, in both mean squared error, and R-squared metrics. Note that while a low MSE is preferred, a higher R-squared means better performance.



(a) RMSE (b) R-squared

Fig. 8.5: Analysis of the effect of different *priors* and different *spans* on our best-performing convolutional neural network, in both mean squared error, and R-squared metrics. Note that while a low MSE is preferred, a higher R-squared means better performance.

The information contained in Figures 8.4 and 8.5 is best viewed in relation to 8.3. As in Figure 8.3(b), Figure 8.4(b) has a maximum in the upper left corner, corresponding to the lowest *prior* as well as the lowest *span*. This is likely due to the rotational ResNet not converging as fast as the more standard ResNet architecture discussed previously. Due to this slower convergence rate, the extra data added in by minimizing the *span* was a much bigger bonus to the network, outweighing the larger input space provided by a larger *span*. This is also shown in Figure 8.5 to a lesser degree.



(a) Best Overall

(b) Mean of Network Best

Fig. 8.6: Analysis of the effect of different *priors* and different *spans* on our best-performing networks overall.

8.1.5 Best Overall

Figure 8.6 contains the best R-squared values over all 50 runs for a given *span* and *prior* in 8.6(a), and the mean of the best scores for the given *span* and *prior* across all network architectures. This shows our best runs overall, as well as gives us a network-agnostic look at the attributes of the data. Easily visible is the much larger importance of the length of the *prior* vs the length of the *span*; this implies that our data are highly non-stationary, and small changes can make a big difference. Also visible is a slight trend towards smaller *spans*; but while on a per-network basis this may have been more due to extra training data being available, when considered alongside the extreme power of a shorter *prior*, it seems to imply that data further away from the predicted vector is simply worth less.

CHAPTER 9

Conclusion

In this work, we propose an adjusted Ohm's law inequality that we used for building a new physics loss to guide deep learning models for the task of solar wind forecasting. To our best knowledge, this is the first effort to build a hybrid physics and deep learning model for solar wind prediction. Our findings support our hypothesis that physics loss helps fine-tune model prediction in certain cases, though it is important to note that as is tradition in data science, it is no silver bullet offering improvement in every situation. In this work, we considered five baseline models that all showed improvement when informed with Ohm's law physics loss. As a future direction to this work, we would like to explore the inclusion of the WSA-ENLIL physics model forecasts as an input to our hybrid physics-guided model. We also recommend to researchers also looking to use this data to use solar cycle 24 as their validation and test set, with solar cycles 22 and 23 as training data due to the small amount of data in solar cycle 22 and the wide domain of data available in solar cycle 23. This approach will allow more data to be used in training since there is no more concern about learning about the test set due to overlapping data contained within the training set. The data that become available due to testing in this matter should also aid in attaining the higher accuracies of the shorter spans due to having more data to learn from.

9.1 Future work

In this study, we used relatively simple and low-parameter networks to allow us to consider a variety of inputs, outputs, and hyperparameters. Future work is possible by constructing better networks more aptly suited to this data. It may also be prudent to decrease the size of each time step, to allow for a larger input space while keeping the networks focused on a closer time window, as was shown to be effective in Chapter 8. If random sampling for testing and training data is not desired, this study argues that data

from solar cycle 23 should be in the training set, with testing sets coming from more recent data. Finally, this study argues that when considering physics based loss, feature and target selection should be considered early in the process due to the volatility shown in Chapter 6.

Another avenue for future work is the inclusion of solar data attached to the OMNI data. One weakness of our experiment was that our networks had to look at the past data to attempt to understand the state of the sun, then move that state forward, then propagate that state earthward to predict the current solar wind characteristics. However, as more missions are completed and better instrumentation is put in space, we can obtain better data of the actual state of the sun to feed into our networks to allow them to skip needing to glean the state of the sun from past data. It would also be helpful to attempt to predict more than a single snapshot window; since fast solar wind travels quicker than slow solar wind, understanding that fast wind is coming would prefer a smaller prior for prediction, as opposed to a longer prior which would better fit a period of mostly slow solar wind.

A more theoretical avenue of further research would be *why* physics-guided loss functions only provide benefit at certain times, such as how Ohm's law-based loss was proven near useless when any information about the magnetic field was included in the inputs or outputs. Understanding this relationship could aid future researchers to know when to spend resources developing physics informed loss functions, as opposed to spending those same resources to improve the data-based methods of prediction.

REFERENCES

[1] S. Martin, "Solar winds travelling at 300km per second to hit earth today," https://www.express.co.uk/news/science/1449974/solar-winds-space-weather-forecast-sunspot-solar-storm-aurora-evg, 2021, accessed: 2022-05-01.

[2] A. H. Boozer, "Ohm's law for mean magnetic fields," *Journal of plasma physics*, vol. 35, no. 1, pp. 133–139, 1986.

[3] S. S. Board, N. R. Council *et al.*, *Severe space weather events: Understanding societal and economic impacts: A workshop report*. National Academies Press, 2009.

[4] J. Eastwood, E. Biffis, M. Hapgood, L. Green, M. Bisi, R. Bentley, R. Wicks, L.-A. McKinnell, M. Gibbs, and C. Burnett, "The economic impact of space weather: Where do we stand?" *Risk Analysis*, vol. 37, no. 2, pp. 206–218, 2017.

[5] Y. S. Shugai, "Analysis of quasistationary solar wind stream forecasts for 2010–2019." *Russian Meteorology & Hydrology*, vol. 46, no. 3, pp. 172 – 178, 2021. [Online]. Available: https://dist.lib.usu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=asn&AN=151541530&site=ehost-live

[6] D. Emmons, A. Acebal, A. Pulkkinen, A. Taktakishvili, P. MacNeice, and D. Odstrcil, "Ensemble forecasting of coronal mass ejections using the wsa-enlil with coned model," *Space Weather*, vol. 11, no. 3, pp. 95–106, 2013.

[7] M. Owens, M. Lang, L. Barnard, P. Riley, M. Ben-Nun, C. J. Scott, M. Lockwood, M. A. Reiss, C. N. Arge, and S. Gonzi, "A computationally efficient, time-dependent model of the solar wind for use as a surrogate to three-dimensional numerical magnetohydrodynamic simulations." *Solar Physics*, vol. 295, no. 3, pp. 1 – 17, 2020. [Online]. Available: https://dist.lib.usu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=asn&AN=142764714&site=ehost-live

[8] B. Luo, Q. Zhong, S. Liu, and J. Gong, "A new forecasting index for solar wind velocity based on eit 284 Å observations." *Solar Physics*, vol. 250, no. 1, pp. 159 – 170, 2008. [Online]. Available: https://dist.lib.usu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=asn&AN=33235306&site=ehost-live

[9] Y. Yang and F. Shen, "Modeling the global distribution of solar wind parameters on the source surface using multiple observations and the artificial neural network technique." *Solar Physics*, vol. 294, no. 8, p. N.PAG, 2019. [Online]. Available: https://dist.lib.usu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=asn&AN=138578239&site=ehost-live

[10] H. Raju and S. Das, "Cnn-based deep learning model for solar wind forecasting." *Solar Physics*, vol. 296, no. 9, pp. 1 – 25, 2021. [Online]. Available: https://dist.lib.usu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=asn&AN=153080952&site=ehost-live

[11] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne *et al.*, "The fair guiding principles for scientific data management and stewardship," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[12] N. Papitashvili, D. Bilitza, and J. King, "Omni: a description of near-earth solar wind environment," *40th COSPAR scientific assembly*, vol. 40, pp. C0–1, 2014.

[13] T. Mukai, S. Machida, Y. Saito, M. Hirahara, T. Terasawa, N. Kaya, T. Obara, M. Ejiri, and A. Nishida, "The low energy particle (lep) experiment onboard the geotail satellite," *Journal of geomagnetism and geoelectricity*, vol. 46, no. 8, pp. 669–692, 1994.

[14] "Nasa - nssdca - spacecraft - details." [Online]. Available: https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1973-078A

[15] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, "Spectrally-normalized margin bounds for neural networks," *Advances in neural information processing systems*, vol. 30, 2017.

[16] A. Bresler, G. Joshi, and N. Marcuvitz, "Orthogonality properties for modes in passive and active uniform wave guides," *Journal of Applied Physics*, vol. 29, no. 5, pp. 794–799, 1958.

[17] A. Karpatne, W. Watkins, J. S. Read, and V. Kumar, "Physics-guided neural networks (PGNN): an application in lake temperature modeling," *CoRR*, vol. abs/1710.11431, 2017. [Online]. Available: http://arxiv.org/abs/1710.11431

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] I. Golan and R. El-Yaniv, "Deep anomaly detection using geometric transformations," *Advances in neural information processing systems*, vol. 31, 2018.

[20] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," 2014. [Online]. Available: https://arxiv.org/abs/1409.1259

APPENDIX

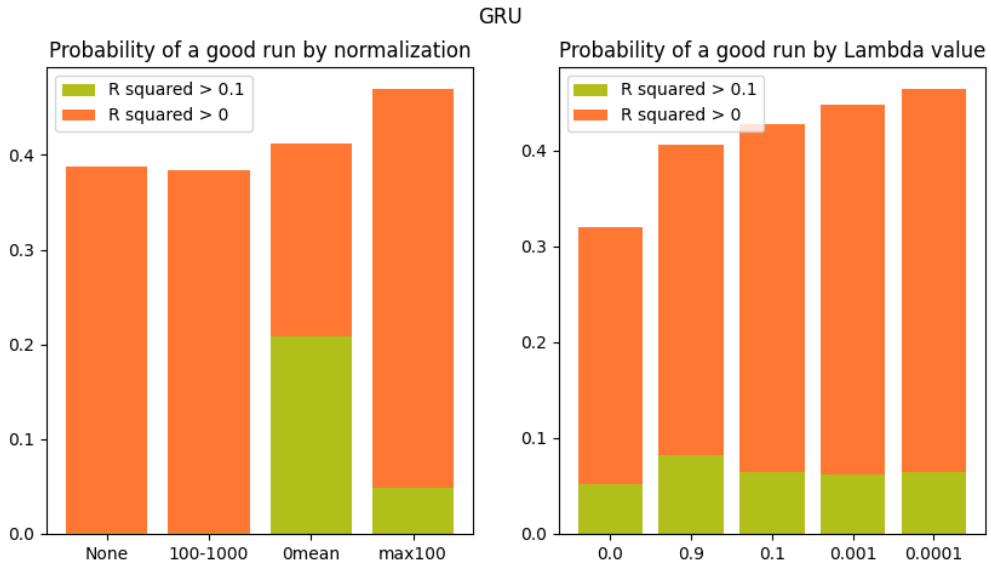


Fig. 1: Appendix: Probabilities of achieving a given R-squared threshold with GRU network. *(The maximum R-squared achieved by this network was 0.158)*

Included here are the results and analysis published when I presented in the Pattern Recognition and Remote Sensing workshop (PRRS2022) held in Montreal, Canada. These results helped to inform the final experiments shown in this thesis, so they may feel incomplete when compared to the work we were able to do later. This later work includes both the modified R-squared metric and the tests of statistical significance. As detailed in Chapter 6, these neural network training circumstances fail to achieve statistical significance under the modified R-squared metric.

These experiments were performed in the following way: we defined a grid of hyperparameters, including normalization and physics loss constant. Each of these runs was tested on data withheld from the training cycle, and the R-squared metric was noted. After all runs were completed, runs with R-squared greater than 0.1 were plotted in green, and the remaining runs that had an R-squared greater than 0 were plotted in orange. These counts were then divided by the total number of runs to obtain a probability of a run above the given threshold.

Figure 1 provides the best argument in this section for the usage of our physics-based loss. However, in the context of this entire section it is more likely that we merely had a

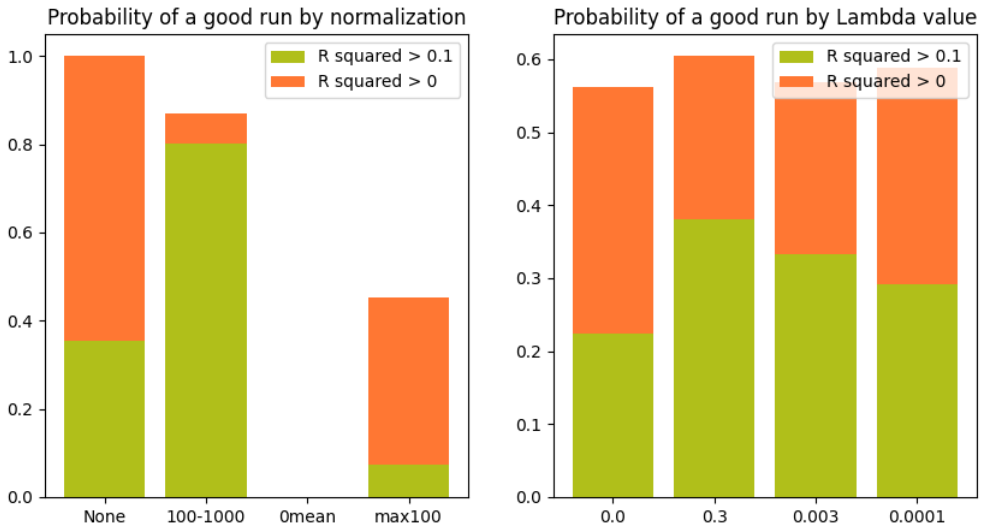


Fig. 2: Appendix: Probabilities of achieving a given R-squared threshold with a time-constrained ResNet. *(The maximum R-squared achieved by this network was 0.175)*

lucky couple of runs. It is also worth noting that while we started with five values of lambda in this particular experiment, fewer values were used in successive experiments due to time constraints.

Figure 2 shows that while our physics-based loss did not help the ResNet architecture beat the average using the R-squared metric, our loss did help it to achieve a score higher than 0.1. A lambda value of 0.3 shows the most improvement, but this is likely due to an influence on learning rate, more easily visible on the other half of the figure. The magnitude of neural network loss roughly corresponds to the magnitude of the data. This loss is then multiplied by the learning rate before being backpropagated through the network to update the weights and biases in it. What we found, and later improved in our work, was to take this in to account.

Figure 3 shows the R-squared results of the time-based CNN which is relatively a simple network with few parameters. We note that among the four normalization methods, the 100-1000 min-max normalization achieved the best results for a positive R-squared and R-squared ¿ 0.1. We also note that although the positive R-squared probability is similar for the non-physics constrained CNN (i.e., $\lambda \neq 0$) and the physics constrained CNN for

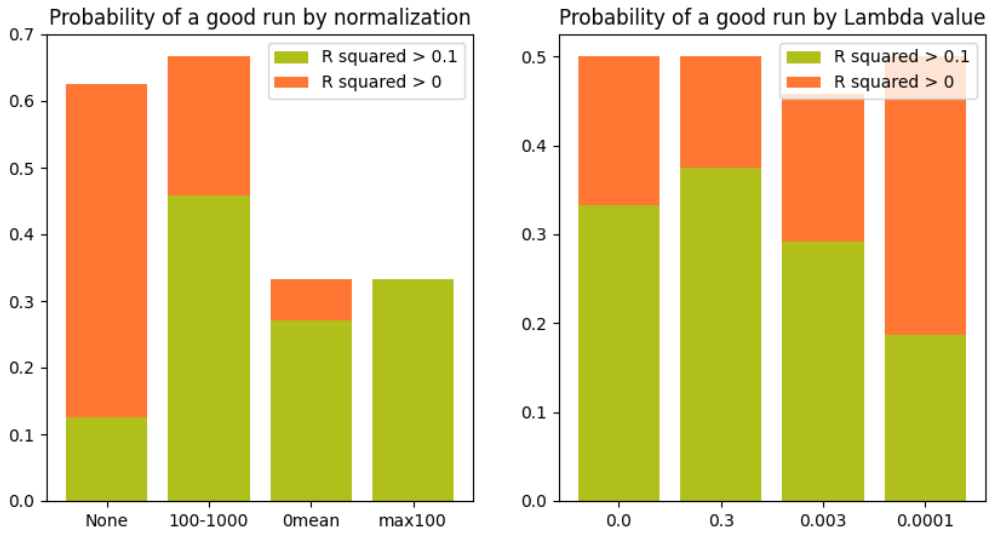


Fig. 3: Appendix: Probabilities of achieving a given R-squared threshold with a time-based CNN. *(The maximum R-squared achieved by this network was 0.178)*

$\lambda = 0.3$, the constrained model provides better r squared probabilities greater than 0.1.

Figure 4 shows the results of the RotateNet which requires significantly fewer parameters than the ResNet model (roughly one-third of the number of parameters). ResNet performed the best using Z-normalization (zero mean). Similar to CNN and ResNets, RotateNet did not perform well with low values of λ. The network performed well either under a high λ or an unconstrained loss (i.e., $\lambda = 0$).

Figure 5 shows the results of our experiments when trained using the LSTM baseline. The LSTM baseline performed the best when trained with Z-normalized data and outperformed the other data normalizations by a large margin. While the LSTM did not need physics loss to produce a decent result, the physics loss provided to be useful in producing a better quality result that has a higher probability of R-squared being greater than 0.1.

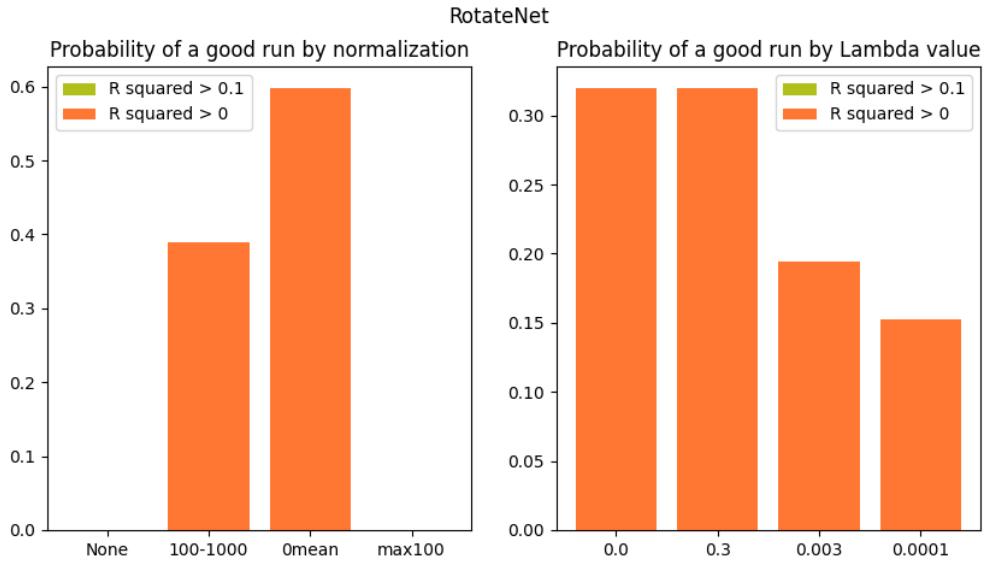


Fig. 4: Appendix: Probabilities of achieving a given R-squared threshold with a rotating ResNet implementation. *(The maximum R-squared achieved by this network was 0.087)*

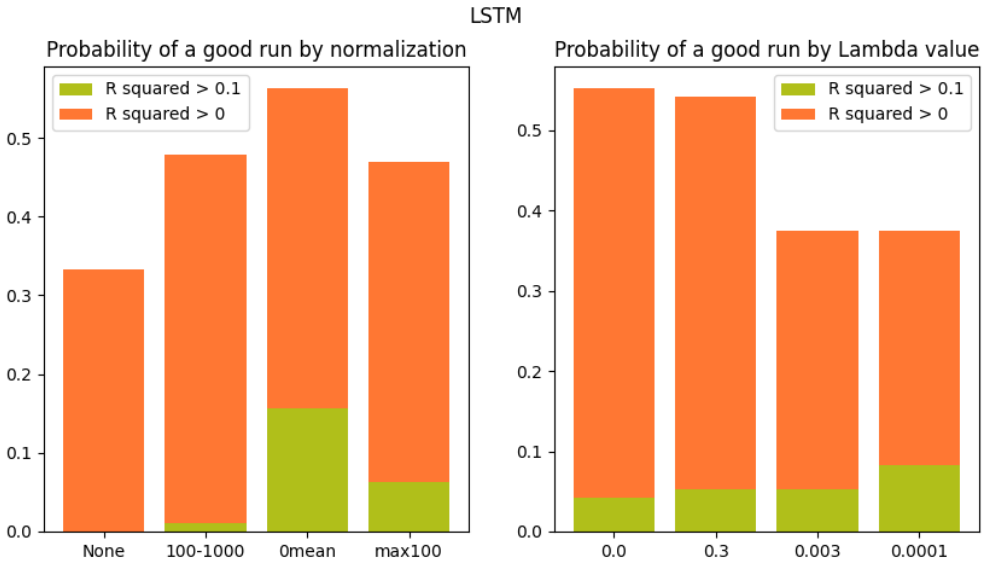


Fig. 5: Appendix: Probabilities of achieving a given R-squared threshold with an LSTM network. *(The maximum R-squared achieved by this network was 0.173)*