2023

PROBABILISTIC SHORT TERM SOLAR DRIVER FORECASTING WITH NEURAL NETWORK ENSEMBLES

Joshua Daniell
West Virginia University, jddaniell@mix.wvu.edu

PROBABILISTIC SHORT TERM SOLAR DRIVER FORECASTING WITH NEURAL NETWORK ENSEMBLES

Joshua Daniell

Thesis submitted
to the Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University

in partial fulfillment of the requirements for the degree of

Master of Science in
Aerospace Engineering

Piyush Mehta, Ph.D., Chair
Jason Gross, Ph.D.
Hang Woon Lee, Ph.D.

Department of Mechanical and Aerospace Engineering

Morgantown, West Virginia
2023

Keywords: uncertainty estimates, probabilistic forecasting, machine learning, model ensembles, space weather, thermosphere, satellite drag

ABSTRACT

Probabilistic Short Term Solar Driver Forecasting with Neural Network Ensembles

Joshua Daniell

Commonly utilized space weather indices and proxies drive predictive models for thermosphere density, directly impacting objects in low-Earth orbit (LEO) by influencing atmospheric drag forces. A set of solar proxies and indices (drivers), $F_{10.7}$, $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$, are created from a mixture of ground based radio observations and satellite instrument data. These solar drivers represent heating in various levels of the thermosphere and are used as inputs by the JB2008 empirical thermosphere density model. The United States Air Force (USAF) operational High Accuracy Satellite Drag Model (HASDM) relies on JB2008, and forecasts of solar drivers made by a linear algorithm, to produce forecasts of density. Density forecasts are useful to the space traffic management community and can be used to determine orbital state and probability of collision for space objects. In this thesis, we aim to provide improved and probabilistic forecasting models for these solar drivers, with a focus on providing first time probabilistic models for $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$. We introduce auto-regressive methods to forecast solar drivers using neural network ensembles with multi-layer perceptron (MLP) and long-short term memory (LSTM) models in order to improve on the current operational forecasting methods. We investigate input data manipulation methods such as backwards averaging, varied lookback, and PCA rotation for multivariate prediction. We also investigate the differences associated with multi-step and dynamic prediction methods. A novel method for splitting data, referred to as striped sampling, is introduced to produce statistically consistent machine learning data sets. We also investigate the effects of loss function on forecasting performance and uncertainty estimates, as well as investigate novel ensemble weighting methods. We show the best models for univariate forecasting are ensemble approaches using multi step or a combination of multi step and dynamic predictions. Nearly all univariate approaches offer an improvement, with best models improving between 48 and 59% on relative mean squared error (MSE) with respect to persistence, which is used as the baseline model in this work. We show also that a stacked neural network ensemble approach significantly outperforms the operational linear method. When using MV-MLE (multivariate multi-lookback ensemble), we see improvements in performance error metrics over the operational method on all drivers. The multivariate approach also yields an improvement of root mean squared error (RMSE) for $F_{10.7}$, $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ of 17.7%, 12.3%, 13.8%, 13.7% respectively, over the current operational method. We additionally provide the first probabilistic forecasting models for $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$. Ensemble approaches are leveraged to provide a distribution of predicted values, allowing an investigation into robustness and reliability (R&R) of uncertainty estimates, using the calibration error score (CES) metric and calibration curves. Univariate models provided similar uncertainty estimates as other works, while improving on performance metrics. We also produce probabilistic forecasts using MV-MLE, which are well calibrated for all drivers, providing an average CES of 5.63%.

I dedicate this work to the late Dr. Paul Daniell; he always believed in me...

Acknowledgments

I want to start by thanking my advisor, Piyush Mehta, for welcoming me to do research with open arms. Starting as a GTA for your orbital mechanics course, through our in-depth weekly research presentations, you taught me invaluable skills for both presenting and listening. You pushed me out of my comfort zone consistently and taught me what it means to be a hard worker. You gave me the opportunity to travel and present at conferences, publish a paper for the first time, and do state-of-the-art research in an interesting field. I thank you for the tools you have given me to advance my career and the opportunity to spend a few more years in my home state.

Thank you to my committee members: Dr. Jason Gross and Dr. Hang Woon Lee. I appreciate the time you have given to review this thesis, provide feedback, and responding to my many questions. I also want to thank Dr. Andrew Rhodes, who always provided an ear when I felt the need to talk about my future, or share something new I was working on. I would also like to thank the other members of Dr. Mehta's research group, both past and present. They were always available to discuss problems and solutions, and provided great company; I will consider you all life-long friends. I would also like to thank Dr. Tzu-Wei Fang, Rob Steenburgh, and John Mayers for answering my countless questions and helping me to make comparisons to SWPC results.

I also want to thank several members of the Statler College community. Mike Brewster, thank you for giving me a guidance when I was a FEP tutor. It helped me learn so much about dealing with others and myself, and helped me meet a lot of the WVU community. Thank you to the dynamic duo, LaDawn Weaver and Diane Stewart, who always made me feel welcome. You both always listened when I needed to vent, supplied me with plenty of snacks, and made me laugh when I had rough days. You all also helped me navigate a particularly difficult time in my life.

Lastly, I want to thank my wife, Nicole. I could not have done this without you. You gave me constant encouragement, love, and companionship, sticking by my side while I pursued this degree. You always gave me the break I needed from my studies, you made me laugh and kept my head on straight. I will forever cherish the time that we got to spend together in Morgantown.

The results presented in this document rely partially on data collected by the Solar Radio Monitoring Program (https://www.spaceweather.gc.ca/forecast-prevision/solar-solaire/solarflux/sx-en.php) operated by the National Research Council and Natural Resources Canada. These data are available at https://www.spaceweather.gc.ca/forecast-prevision/solar-solaire/solarflux/sx-5-en.php. These data were accessed via the LASP Interactive Solar Irradiance Datacenter (LISIRD).

The JB2008 solar and geomagnetic indices are provided for scientific use courtesy of Space Environment Technologies and are available at https://spacewx.com/jb2008/. Figures were made with Matplotlib version 3.5.2, available under the Matplotlib license at https://matplotlib.org/.

NOAA SWPC's *Weekly* publication archive is stored on FTP servers and can be accessed readily. Processing archived data is non-trivial due to the nature of the PDF publication. Processing of the archived PDFs was accomplished with assistance from NOAA personnel. Current NOAA SWPC products are available at https://www.swpc.noaa.gov/products-and-data and archived products (such as the *Weekly* publication) can be accessed via FTP server at ftp://ftp.swpc.noaa.gov/pub/

Contents

List of Figures

List of Tables

Acronyms

ADAPT	Air Force Data Assimilative Photospheric Flux Transport
AR	Auto-Regressive
BGS	British Geological Survey
CES	Calibration Error Score
CHAMP	CHAllenging Minisatellite Payload
CI	Confidence Interval
CLS	Collecte Localisation Satellites
CNES	Centre National D'études Spatiales (French Space Agency)
CNN	Convolutional Neural Network
DCA	Dynamic Calibration of the Atmosphere
DRAO	Dominion Radio Observatory
DTM	Drag Temperature Model
DTW	Dynamic Time Warping
EMOS	Ensemble Model Output Statistics
ESA	European Space Agency
EUV	Extreme Ultraviolet
FUV	Far Ultraviolet
GOES	Geostationary Operational Environment Satellite
GRACE	Gravitational Recovery and Climate Experiment
HASDM	High Accuracy Satellite Drag Model
IAGA	International Association of Geomagnetism and Aeronomy
IDL	Interactive Data Language
ISS	International Space Station
JB2008	Jacchia-Bowman 2008 Empirical Thermospheric Density Model
LASP	Laboratory for Atmospheric and Space Physics
LEO	Low-Earth Orbit
LISIRD	LASP Solar Irradiance Datacenter
LR	Linear Regression
LSTM	Long-Short Term Memory Neural Network
MAE	Mean Absolute Error

Acronyms

ME	Mean Error
MGII cwr	Magnesium II Core to Wing Ratio
ML	Machine Learning
MLE	Multi-Lookback Ensemble
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
MSIS	Mass Spectrometer Incoherent Scatter Radar
MV-MLE	Multivariate Multi-Lookback Ensemble
NASA	National Aeronautics and Space Administration
N-BEATS	Neural Basis Expansion Analysis for Interpretable Time Series Forecasting
NN	Neural Network
NOAA	National Atmospheric and Oceanic Administration
ODPO	Orbital Debris Program Office
OSC	Office of Space Commerce
PCA	Principal Component Analysis
PoC	Probability of Collision
RELU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SBUV	Solar Backscatter Ultraviolet
SDA	Space Domain Awareness
SEE	Solar Extreme-Ultraviolet Experiment
SEM	Solar Extreme-Ultraviolet Monitor
SET	Space Environment Technologies
SFU	Solar Flux Units
SGD	Stochastic Gradient Descent
SIFT	Solar Indices Forecasting Tool
SOHO	Solar and Heliospheric Observatory
SOLSTICE	Solar/Stellar Irradiance Comparison Experiment
SORCE	Solar Radiation and Climate Experiment

Acronyms

SSA	Space Situational Awareness
STEREO	Solar Terrestrial Relations Observatory
STM	Space Traffic Management
SVR	Support Vector Regression
SWPC	Space Weather Prediction Center
TIE-GCM	Thermosphere-Ionosphere-Electrodynamics General Circulatory Model
TIMED	Thermosphere Ionosphere Magnetosphere Energetics & Dynamics
UARS	Upper Atmosphere Research Satellite
UQ	Uncertainty Quantification
USAF	United States Air Force
USSC	United States Space Command
UV-MLE	Univariate Multi-Lookback Ensemble
XRS	X-Ray Spectrometer

Nomenclature

Symbol	Units	Description
A	m^2	cross-sectional area
a_{drag}	m/s^2	acceleration due to drag
ap	$2nT$	geomagnetic 3-hourly planetary equivalent amplitude index
B	m^2/kg	ballistic coefficient
B_a	days	backwards average window size
C_D	-	drag coefficient
$F_{10.7}$	SFU	solar radio flux proxy index measured at 10.7 cm wavelength
f_s		flux tube expansion factor
H	days	horizon
Hz		hertz
L	days	lookback
m	kg	mass
M	-	number of ensemble members used
$M_{10.7}$	SFU	proxy index for FUV photospheric 160 nm Schumann-Runge Continuum emissions
N	-	number of samples in set
n_{inp}	-	number of inputs
n_{LS}	-	number of lags steps
n_{out}	-	number of outputs
R	-	Pearson correlation coefficient
$S_{10.7}$	SFU	index for integrated 26-34 nm bands solar chromospheric EUV emission
S	-	scaling factor
SFU	-	solar flux units
v_{rel}	m/s	relative velocity of object or spacecraft
w_t	-	error term associated with linear regression model at a given time, t
W	-	watt
$Y_{10.7}$	SFU	index combining X-ray and Lyman-α observations
z	-	z-score associated with given confidence interval
$\sim$	-	when placed above a term, it represents standardization scaling

Greek Letters

Symbol	Units	Description
μ	-	mean
ρ	kg/m^3	mass density
σ	-	standard deviation
σ^2	-	variance
ϕ_i	-	linear regression weighting coefficients

Chapter 1

Motivation and Contributions

1.1 Space Domain in LEO

Just a few decades ago, the number of objects in Low-Earth orbit (LEO) was small; the detection, tracking, and identification of artificial objects, known as catalog maintenance, was relatively easy. However, in the last two decades there has been an exponential growth in total space objects including spacecraft and orbital debris, which poses immediate danger to multi-billion dollar space assets and human spaceflight missions. It is evident from Figure 1.1 that the rate at which the number of space objects is growing is unlikely to slow down. As the total number of artificial objects in LEO grows, catalog maintenance has become non-trivial. A need for more real-time knowledge of the space environment has caused a shift to space domain awareness (SDA), which stresses the ability to accurately predict an object's orbital state.

LEO is currently the most densely populated region in space, especially due to satellite constellations introduced by private organizations such as SpaceX and OneWeb in recent years. As of August 2023, 5,000 mass-produced satellites have been launched into LEO by SpaceX; with a planned satellite total of 12,000 and potential extension to 42,000 [2]. A dramatic increase in the number of objects, especially from large constellations, has resulted in a shift from SDA to space traffic management (STM). Although catalog maintenance still occurs under SDA, STM highlights conjunction assessment and collision avoidance in the context of space safety and sustainability [3], which is performed by many different operators and organizations. The United States Office of Space Commerce (OSC) serves as the primary entity responsible for space commerce policy activities within the Department of Commerce, and seeks to foster the conditions necessary for economic growth and

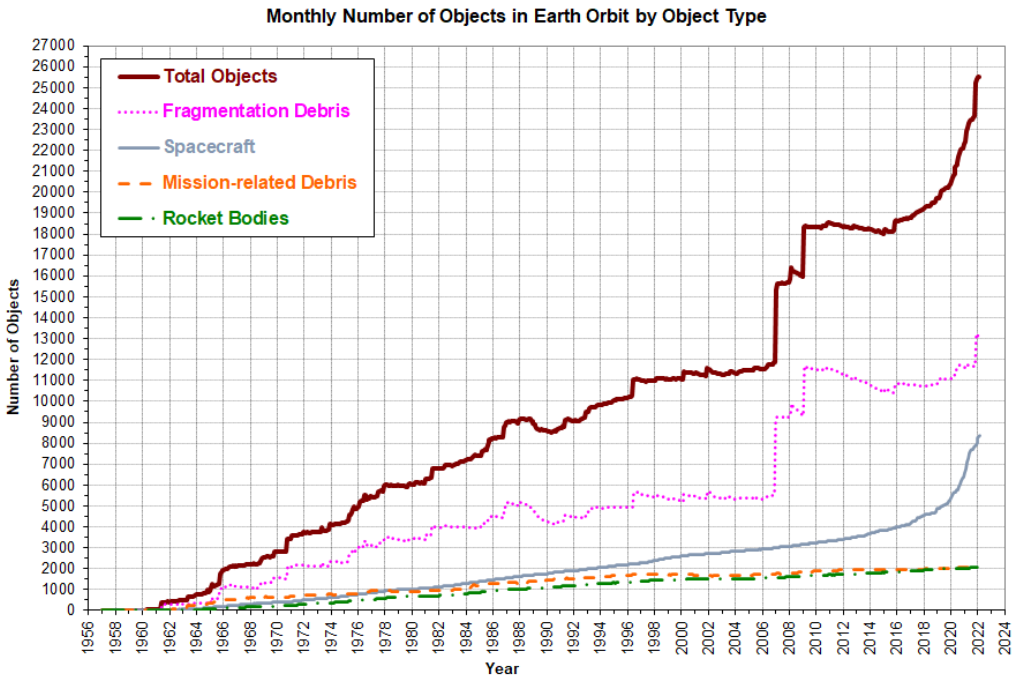


Figure 1.1: Since the space race in the 1950's and 1960's; the amount of debris has ballooned; steep spikes can be attributed to destructive events such as the destruction of Fengyun-1C in 2007 according to the NASA Orbital Debris Program Office (ODPO)'s LEGEND model [1].

technological advancement in the U.S. commercial space industry. The OSC plays a crucial role in coordinating with various stakeholders and operators to ensure safe and efficient operations in space. The OSC collects and shares data and cooperates with operators to manage space traffic and reduce the risk of collisions and space debris proliferation. In addition, the United States Space Command (USSC) currently maintains a catalog of space objects including debris, and provides probability of collision (PoC) and estimated orbital state and covariance to most operators. The operators can use these products for further assessment, risk analysis, and maneuvering, which can cost precious fuel.

1.1.1 Challenge of Atmospheric Drag

To provide improved methods for the growing STM efforts, especially in LEO, we consider one of the largest sources of uncertainty for objects in LEO, atmospheric drag. Atmospheric drag is not uniform, and greatly depends on current space weather conditions. Uncertainty and modeling errors affect drag calculations through the neutral density term, ρ, and the drag coefficient, C_D, which can be seen below.

$$a_{drag} = -\frac{1}{2}\frac{\rho\, C_D\, A}{m} v_{rel}^2 \quad with \quad B = \frac{C_D A}{m} \tag{1.1}$$

Other parameters such as cross sectional area and mass, will differ from object to object but are well known. The velocity of the orbiting object, v_{rel}, is also well known but can introduce uncertainty in the cases of strong neutral winds [4]. The typically used ballistic coefficient, B, allows for satellite specific parameters to be grouped into a single term.

In the fields of space operations and space weather, it is widely recognized that changes in the Earth's thermosphere induce variations in satellite drag. Over the past two decades, both empirical and physics-based density models have been developed, which have significantly enhanced our drag modeling capabilities. These models have demonstrated an average global error of less than 10% during solar maximum conditions [5]. Physics-based models, which utilize fundamental governing equations, offer the potential for greater accuracy but may exhibit biases from sources such as physical assumptions and numerical approximations. Physics-based models can also demand substantial computational resources due to the complexity of the physical processes which are modeled and the numerical methods used, such as finite difference.

Operators who rely on thermosphere forecasts for collision avoidance are challenged by uncertainty in forecasts. Uncertainty in thermosphere density forecasts can be attributed to either modeling error (uncertainty and/or error in the model itself) and driver error (uncertainty or errors in the inputs used by the model). The USSC and commercial or other operators usually perform 3-day forecasts during conjunction event assessment; such 3-day LEO forecasts assume that driver uncertainty is dominant. Recent work has showed that the impact of model uncertainty is on the same order as driver uncertainty [6].

Most methods for forecasting drivers rely on models which produce deterministic predictions. Operators who rely on short-term driver forecasts receive linearly forecasted values, with no associated uncertainty estimates. Typically, uncertainty in both model and drivers are simplified or overlooked. In order to make more informed decisions on conjunction risks, operators should be supplied with accurate forecasts which also provide driver uncertainty estimates. By including probabilistic density and driver models, associated uncertainty can be coupled with orbital state and covariance, as seen in Figure 1.2.

Recent work in probabilistic thermosphere density models, has allowed for investigation into the coupling of model uncertainty to the state and covariance of objects in LEO [7]. A similar probabilistic forecast and associated coupling must be established for the model drivers to be combined with model uncertainties. Robust and reliable uncertainty estimates for orbital state and covariance must be provided so that operators can make more informed decisions, especially when considering costly space assets with limited available fuel.

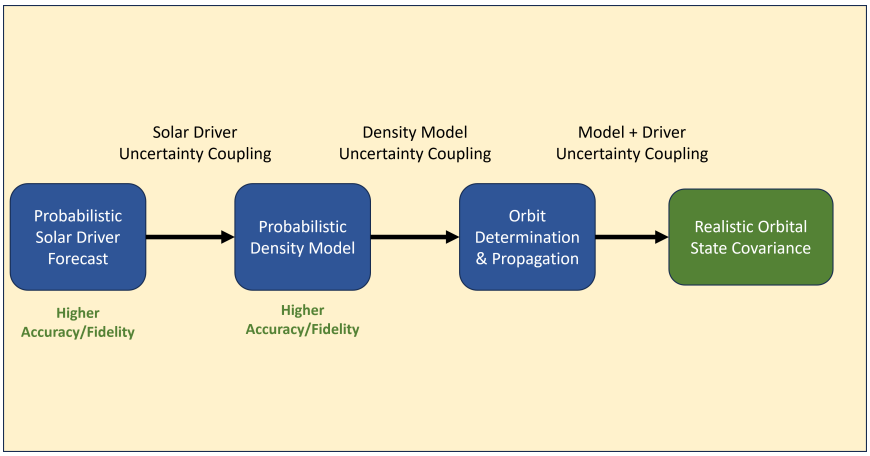


Figure 1.2: Accurate driver forecasting and uncertainty estimates are the first step in an overall framework which can be used to help STM efforts.

1.2 Contributions

Recent work using machine learning techniques have allowed for the enhancement of thermosphere density models, by providing probabilistic forecasts. Models such as HASDM-ML[8], CHAMP-ML, MSIS-UQ[9], and TIE-GCM ROPE[10], have allowed for the effects of density uncertainty on orbit uncertainty estimation to be investigated [11]. It has also been seen that sources of uncertainty from drivers and models significantly affect orbital state uncertainty estimates, with driver uncertainties dominating after several days [6]. This thesis aims to investigate and provide contributions to address solar driver uncertainty, which can be used stand-alone or incorporated with probabilistic density models such as HASDM-ML to address model and driver uncertainty simultaneously. This work provides higher accuracy and probabilistic modeling of solar drivers, which can be sampled for operations. This thesis aims to provide a significant step in the overall planned framework (seen in Figure 1.2), to provide robust and reliable orbit state uncertainty estimates to STM stakeholders.

1.2.1 Probabilistic Forecasting of Drivers

If density models were perfected, errors would still exist due to errors in forecasted drivers; inability to accurately forecast density prevents drag from being modeled accurately. Neural network machine learned models have been used for the historic $F_{10.7}$, with only one approach considering a probabilistic forecast [12]. This work provides machine learned models which improve over that work and current state-of-the-art forecasting methods for all solar drivers, and are capable of providing probabilistic forecasts. This work marks the first time probabilistic forecasting methods have been successfully applied to the $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ solar drivers.

1.2.2 Striped Sampling Technique

Several of the drivers used by the state-of-the-art JB2008 density model have only been recorded daily since 1997. Due to the lack of a substantial dataset, previous machine learning (ML) approaches, which followed traditional data splitting methods, were biased and performed poorly on new data. In Chapter 3 (Section 3.3.2), we identify the issues associated with poorly sampled data and provide a novel method to prepare statistically consistent data for use in machine learning training schemes. By utilizing the new striped sampling technique, limited datasets can be used effectively for future ML approaches in all time series forecasting fields.

1.2.3 Input Data Manipulation

Auto regressive (AR) models use only previous data to predict future values. Current operational methods for forecasting drivers rely on AR methods, which can be limited in their predictive ability. We consider several methods for manipulating input data to improve AR predictions for solar drivers. We present a new method, which uses a statistical technique known as principal component analysis (PCA) to transform multivariate model input data. This work marks the first time that PCA has been applied to solar drivers, which is capable of improving forecasts and uncertainty estimates. In Chapter 4 (Section 4.3), we introduce a new method to improve univariate forecasting by supplying a backwards averaged value to current model inputs. This work represents the first instance of manipulating solar driver data in this manner, and its applications extend beyond space weather, it can enhance AR forecasting in any domain where forecasting is essential.

1.3 Outline

There currently exists a gap in the overall modeling of uncertainty for orbital state because there lacks adequate uncertainty estimates for model drivers. This thesis focuses on providing more accurate driver forecasts which are probabilistic and can be used to provide more robust and reliable uncertainty estimates to the STM community. This work begins with essential background information on the interaction between the Sun and Earth, modeling approaches for thermosphere density and solar drivers, and ML techniques, which have been applied successfully to the field of space weather. The background of ML techniques include a discussion of ensemble approaches which have shown success in probabilistic forecasting and novel techniques to combine individual model predictions, providing an improved overall forecast. We then provide an in depth discussion of methods

for probabilistic solar driver forecasting using neural network ensembles, followed by a comparison with state-of-the-art methods. Lastly, we present the conclusions of this work and provide recommendations for future work for solar driver forecasting as well as a discussion of the work necessary to couple driver uncertainty to uncertainty estimates in orbital state.

Chapter 2

Background

This chapter's goal is to establish the essential background information required for comprehending the influence of the Sun on the thermosphere, explaining the connection between the thermosphere and STM, and examining the current methods and challenges associated with forecasting of thermosphere conditions, specifically density.

2.1 Space Weather and EUV

When we use the phrase "space weather", we are referring to the processes of the Sun that affect the space environment, as well as the Earth. The Sun's inner core is a nuclear reactor, which produces massive amounts of energy that radiate to its surface [13]. The energy which is released from the Sun, in the form of plasma, follows the complex and ever-changing magnetic field lines of the surface. As a result, these highly coupled energetic processes can be difficult to predict.

The Sun directly affects all of us, it is complex and not well understood, it is also constantly changing. Since the 17th century, astronomers have observed and meticulously recorded sunspots. A sunspot is a temporary, dark area on the Sun's surface caused by intense magnetic activity. These spots appear darker than their surroundings because they are cooler regions. Historical records reveal that, on average, the Sun follows an eleven year cycle of activity. The dramatic change in solar activity can be seen by historical observations of sunspot number, seen in Figure 2.1. The number of observed sunspots correlate well to solar activity, but they do not tell the whole story. Instead we must rely on scientific measurements of the solar-based electromagnetic radiation.

Although the Sun is on average 150 million km from Earth; the solar wind constantly bombards the Earth with high energy radiation. We focus on specific wavelength bands of this solar elec-

tromagnetic radiation to aid us in determining solar processes and potential effects on the Earth environment. Extreme-ultraviolet (EUV) radiation is electromagnetic radiation that lies on the electromagnetic spectrum with wavelengths between 121 nm and 10 nm. EUV radiation emitted by the Sun is absorbed by the thermosphere and is unable to pass all the way through to the Earth's surface. Consequently, Earth's atmosphere is considered opaque to EUV radiation, making ground-based measurements of EUV impossible. Since we cannot directly measure solar EUV, we must rely on measurements of other terms, solar indices and proxies. These indices and proxies are used to determine current solar activity levels, represent heating for a variety of thermosphere layers, and are inputs to a state-of-the-art model for density in the thermosphere.

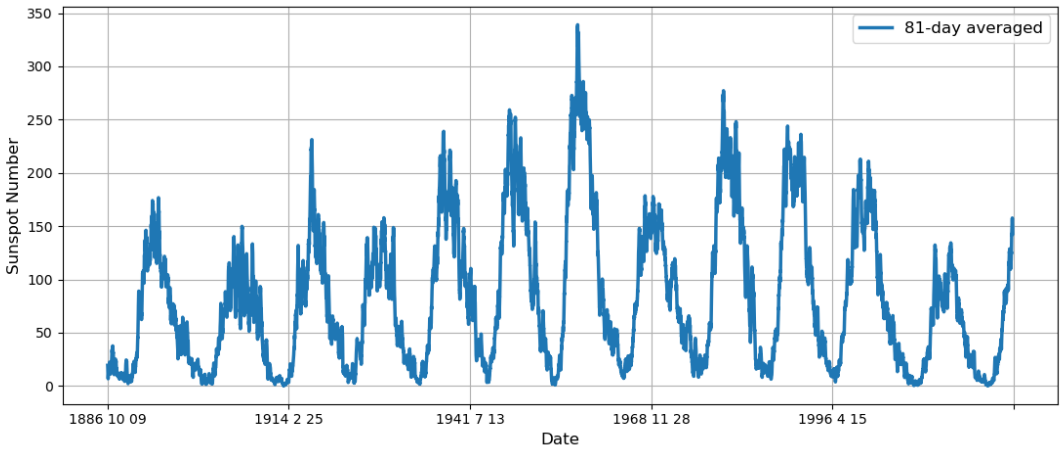


Figure 2.1: On average, the Sun follows an 11-year cycle, but it doesn't consistently reach the same level of solar maximum, and the duration it takes to reach that maximum point can vary.

2.2 Understanding the Thermosphere

The thermosphere is one of the upper-most neutral layers of the Earth's atmosphere; spanning roughly 90 km to 1000 km above Earth's surface, depending on current space weather conditions. The thermosphere maintains a much lower density than lower altitudes of the atmosphere; but solar EUV absorption can cause the thermosphere to heat and expand. Similarly, when solar EUV is reduced, the thermosphere cools and contracts [14]. During phases of high solar activity, the thermosphere is subjected to increased solar EUV irradiance. The high energy EUV radiation is absorbed predominately by the upper atmosphere, which causes heating and bulk movement (due to buoyancy) of air, expanding the thermosphere. The bulk movement effect creates a property change and density in the thermosphere can increase or decrease significantly.

Due to the chaotic nature of the Sun, such density changes can be unexpected, leading to challenges in density modeling. It is important to provide forecasts of expected density changes; but current models can provide drastically differing density values during storms and high solar activity levels [15]; seen in Figure 2.2. The STM community lacks a unified modeling approach, therefore many models have been developed (and are operational) for providing thermosphere density forecasts.

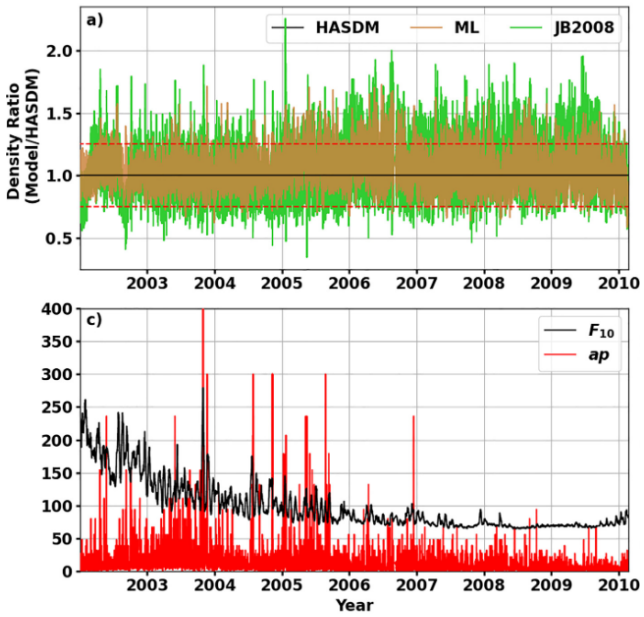


Figure 2.2: Top: The modeled density ratio between HASDM, a machine learning approach, and JB2008; indicate how much density can vary depending on level of solar activity!
Bottom: $F_{10.7}$ and ap represent the solar and geomagnetic activity levels for a given date. Licata et al. (2022) [8]

2.3 Historical Thermosphere Modeling

Much of our understanding of physical systems rely on observations, many thermosphere models are no different. Empirical thermosphere models rely on observation and data to make generalizations about physical processes. We have seen advancement in empirical modeling by the scientific community, especially with empirical models [14], such as MSIS [16], DTM [17], and Jacchia [5] groups.

Mass Spectrometer Incoherent Scatter Radar (MSIS) models use a mixture of mass spectrometer, incoherent scatter radar, and accelerometer-derived density estimates. The Drag Temperature Model (DTM) group relies on orbit-derived density data and accelerometer-derived density data. The

Jacchia group, such as JB2008, use orbit and accelerometer derived density estimates. More recent advances in modeling involve real-time data assimilation; an assimilative model integrates empirical data and observations to create a robust representation of a phenomenon or system. One such example of an assimilative model, the High Accuracy Satellite Drag Model (HASDM) [18] uses Dynamic Calibration of the Atmosphere (DCA) to adjust the density nowcast given by JB2008 from a set of calibration satellites.

Recent advances with accelerometers have enabled acceleration-derived density estimates to improve models. Using accelerometer data throughout the operational period of a spacecraft, e.g. CHAllenging Minisatellite Payload (CHAMP) and Gravity Recovery and Climate Experiment (GRACE), data is gathered across many space weather conditions and altitudes, and can be used for model development. By removing acceleration contributions from other sources, researchers have been able to isolate drag based acceleration and derive density [19], [17], [20].

Physics-based models are also effective for the upper atmosphere, such as the model for thermosphere density Thermosphere-Ionosphere-Electrodynamics General Circulatory Model (TIE-GCM) [21]. Some physics-based models, including TIE-GCM, use a finite difference method to solve the governing equations which couple the thermosphere and ionosphere, and are used to generate self-consistent electric fields at lower latitudes [14]. When short-term forecasts are needed, empirical and assimilative models may be more useful, as physics-based models are computationally expensive and challenging to develop; especially when considering the sheer number of LEO objects.

2.3.1 JB2008

The most recent Jacchia type empirical model, JB2008, improved over previous Jacchia family models by incorporating new solar and geomagnetic indices and proxies as inputs. JB2008 uses four solar indices and proxies, $F_{10.7}$, $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$, which are referred to as drivers. The drivers are used to represent variations caused by solar heating, specifically in different layers of the thermosphere. In addition to the four solar drivers, JB2008 utilizes two geomagnetic indices, Dst and ap to improve density modeling during high geomagnetic activity. JB2008 reduces non-storm density errors by over 5% and reduced storm-time errors from Jacchia-70 by over 60% from NRLMSISE-00 by more than 35% and from JB2008 with only ap by 16% [5].

2.3.2 High Accuracy Satellite Drag Model

Used solely by the Department of Defense (DoD), HASDM relies on an adjustment of the nowcast made by the density prediction of the JB2008 model. HASDM modifies 13 global temperature correction coefficients using the DCA algorithm; improving on the work done by [22] and [23]. HASDM relies on observations made on 70 calibration satellites to estimate local density values. The calibration satellites range from altitudes of 190-900 km. The DCA algorithm used by HASDM relies on a prediction filter that uses wavelet and Fourier analysis for calculating correction coefficients [18]. HASDM is a state-of-the-art model for thermosphere density, but is limited by its deterministic forecast and accuracy of the solar drivers input to JB2008. Recent enhancements to HASDM and other density models, have been introduced to provide probabilistic modeling of density, which would overcome the limitations of a deterministic forecast and provide operators with uncertainty estimates.

2.3.3 Recent Advancements

Recent work has demonstrated enhanced density modeling capabilities through the introduction of machine learning techniques that offer probabilistic density models, as exemplified by HASDM-ML[8], CHAMP-ML, MSIS-UQ[9], and TIE-GCM ROPE[10], which, rather than explicitly improving existing models, primarily focus on providing uncertainty estimates for density modeling, thus facilitating the study of density uncertainty's effects on orbit uncertainty propagation [11]; these advancements also pave the way for similar modeling of solar drivers to explore their impacts on uncertainty in modeled density and orbit propagation.

2.4 Solar Drivers

JB2008 and HASDM rely on the four solar drivers, $F_{10.7}$, $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ as inputs (driver collection sources can be seen in Figure 2.3.) Without accurate forecasts of these drivers, HASDM and JB2008 cannot produce accurate density forecasts, even with perfect density modeling (the connection between JB2008 drivers and HASDM can be seen in Figure 2.4.) The United States Air Force (USAF) currently contracts Space Environment Technologies (SET) to provide the operational forecasts of solar drivers for use with HASDM. SET uses a linear algorithm to provide deterministic forecasts for all drivers over a period of 6 days.

All solar drivers are scaled using linear regression, to units consistent with the historical $F_{10.7}$,

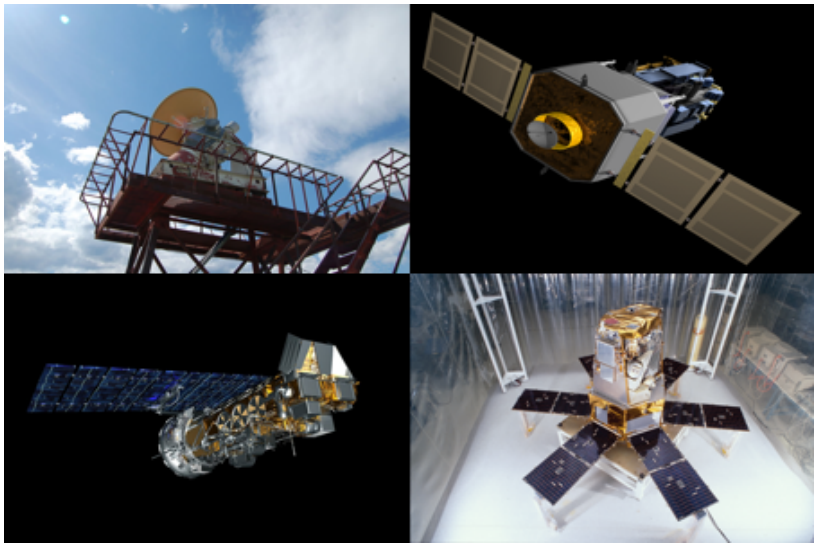


Figure 2.3: **Top Left:** Secondary DRAO telescope provides solar flux data for $F_{10.7}$. **Top Right:** The SOHO spacecraft carries the SEM instrument which is used to produce $S_{10.7}$. **Bottom Left:** The NOAA-18 spacecraft carries the SBUV instrument, which produces MG II cwr data that is transformed into $M_{10.7}$. **Bottom Right:** The SORCE spacecraft provides X-ray observations and Lyman-α measurements which are used to create $Y_{10.7}$.

which are known as solar flux units (SFU).

$$1\,SFU = 10^{-22}\frac{W}{Hz\ m^2} \tag{2.1}$$

Scaling of solar drivers to a consistent unit is done to more easily quantitatively compare the differing drivers. Additionally, drivers can be reported in either an observed or an adjusted form, which scales the driver value to be as though the Earth was located 1 AU ($\approx$ 150 million km) from the Sun. This work deals with the observed form of the solar drivers.

The $F_{10.7cm}$ solar radio flux proxy, denoted as $F_{10.7}$ for the remainder of the work, is one of the most widely used proxies for solar activity. $F_{10.7}$ is described by Tapping [24] as a "determination of the strength of solar radio emissions in a 100 MHz-wide band centered on 2800 MHz (a wavelength of 10.7 cm) averaged over an hour". Three additional solar indices and proxies were introduced by the Jacchia-Bowman 2008 thermosphere density model [5]. $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ are used to map energy from specific solar irradiances to major thermosphere layers [25]; their relationship to one another (and $F_{10.7}$) can be seen in Figure 2.5.

The $S_{10.7}$ index [5] is the integrated 26-34 nm irradiance measured by the Solar Extreme-ultraviolet Monitor (SEM) instrument on the NASA/ESA Solar and Heliospheric Observatory (SOHO), and is used to represent heating in regions near 180 or 200 km. SET provides an op-

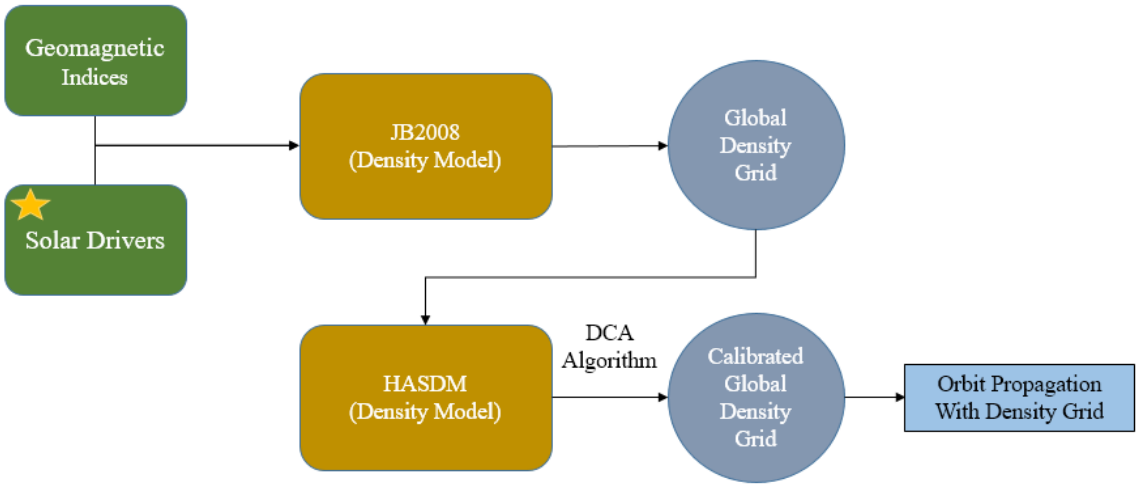


Figure 2.4: The solar drivers (starred) are inputs used in the framework to perform orbit propagation. JB2008 uses these drivers to produce a global density grid; whose nowcast is then corrected using dynamic calibration of the Atmosphere (DCA) by HASDM.

erational backup for SEM data processing as well as provides values of $S_{10.7}$. SEM has been making measurements since December 1995. A more specific description of the process for scaling the data and converting to solar flux units (SFU), units consistent with other drivers, are discussed by Tobiska et al. [25]. Daily values for index are archived and available since January 1, 1997;

The $M_{10.7}$ proxy [5] is created from the Magnesium II (Mg II) core to wing ratio, which originates from the NOAA (National Oceanic and Atmospheric Administration) satellites (NOAA -16,-17,-18). The satellites host the Solar Backscatter Ultraviolet (SBUV) spectrometer, which can make solar

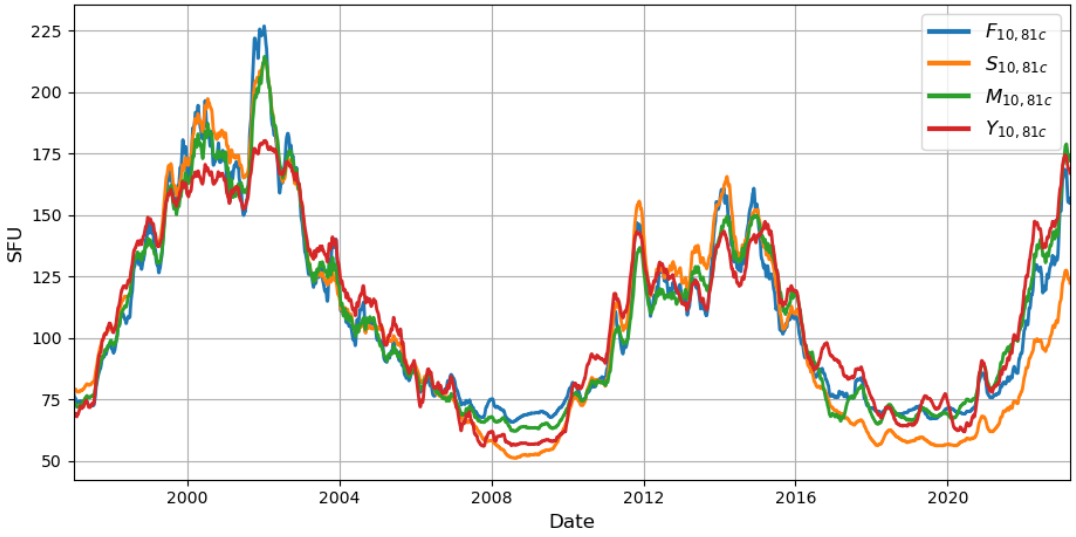


Figure 2.5: 81-day center average of indices & proxies show high correlation and long term 11-year solar cycle trend; yet differences are seen within the 11-year cycles.

UV measurements. MG II is a proxy for solar FUV and EUV emissions that is mapped into the lower thermosphere and represents heating in the thermosphere regions between 95-110 km. MG II is translated into SFU as discussed in the work by Tobiska et al. [25]. Daily values for $M_{10.7}$ are archived and available since January 1, 1997.

The $Y_{10.7}$ index [5] is created from the combination of both GOES/XRS 0.1-0.8 nm x-ray observations and Lyman-α, which is measured by the SOLSTICE instrument on the UARS and SORCE satellites as well as the SEE instrument on TIMED. The observations made are represented by an $L_{10.7}$ and $X_{10.7}$ index, which are combined by Tobiska et al. [25] to form the $Y_{10.7}$ index, representing heating in regions between 85-100 km. Data for $Y_{10.7}$ has been reported daily and archived since January 1, 1997.

2.4.1 History and Prediction of $F_{10.7}$

The $F_{10.7}$ proxy has been recorded consistently since 1947 by sites in Canada. Originally, these observations were performed by a site south of Ottawa, Ontario. Due to expansion of city infrastructure, large amounts of disturbances were noted in observations and, in 1962, a new observation site was constructed at the Algonquin Radio Observatory. A secondary flux measurement station was established at the Dominion Radio Astrophysical Observatory (DRAO), seen in Figure 2.6. Since the establishment of DRAO, both the Algonquin and Ottawa sites have closed, and measurements are made solely at DRAO [24]. Daily observed values of the Penticton $F_{10.7}$ proxy have been recorded and archived since 1947. The data used in this work contains many missing values during the initial year of operation, and these gaps have been excluded from analysis in this work. The historical data for $F_{10.7}$ used in this study spans from 1/1/1948 to 12/10/2021, as depicted in Figure 3.8. This time frame aligns with previous studies, such as [26] and [12]. To address any missing daily flux values, they have been substituted with the most recent observed value for the corresponding dates. As an input to the JB2008 model, future values of $F_{10.7}$ are necessary to provide density forecasts. Forecasting of $F_{10.7}$ varies in skill with solar activity level as cycles over an 11 year period and various activity levels are encountered during forecasting periods. Due to the chaotic nature of the Sun and importance of $F_{10.7}$, many forecasting methods exist, but can be weak in some areas.

2.4.2 Currently Used Models

Commonly used prediction methods for the $F_{10.7}$ proxy using historical values include a mixture of statistical, auto-regressive, linear, and deep learning methods. These varied methods have produced

Figure 2.6: The primary station where the $F_{10.7}$ proxy index is measured. DRAO uses a 26-meter single antenna telescope for observations. Photo Credit: Mark Klotz [27]

a variety of conclusions, some models claim that linear methods outperform neural networks seen in [28] and [29]. Other authors have claimed that they can use machine learning methods to outperform linear methods like [12] and [30]. Forecasting methods, such as these, must be carefully compared in order to reduce contradictory conclusions.

The National Oceanic and Atmospheric Administration (NOAA) operates the National Space Weather Prediction Center (SWPC) which provides many space weather related products and is a crucial source of data and forecasts for space operations. SWPC offers in-depth space weather products and includes an enthusiast's dashboard, an example of dashboard products is seen in Figure 2.7. The dashboard offers, forecasts, solar imaging, ionosphere, solar wind, and many more data products. SWPC also issues a publication called *The Weekly*, providing a 27-day forecast for $F_{10.7}$, created by SWPC personnel. Additionally, SWPC retransmits a 45-day $F_{10.7}$ forecast made by the USAF. Both forecasts use a combination of observations and recurrence. Recurrence is captured by monitoring H-α imagery, STEREO satellite observations, and monitoring of solar active regions. [31] The *Weekly* publication's user guide further explains forecasting methods but cautions against using these forecast results for research purposes. By providing improved $F_{10.7}$ forecasting with neural network ensembles, forecasts distributed by NOAA could include a probabilistic driver forecast created by this work.

To provide forecasts of the solar drivers used by the operational HASDM, SET uses a linear prediction algorithm that captures recurrence and persistence. The algorithm used by SET is the

"TS_FCAST" subroutine in the Interactive Data Language (IDL). This linear prediction algorithm fits and uses a *pth* order auto regressive model on p days lookback. The model is re-fit on data every time a prediction is to be made.

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + ... + \phi_p x_{t-p} + w_t \tag{2.2}$$

The short term (6-day) predictions made by this method have been benchmarked and analyzed thoroughly by [26] and will be used for comparison with the solar driver predictive models in this work. This algorithm uses auto regression and relies on a predictive method called "dynamic prediction" for which one-step predictions are calculated recursively to reach the desired forecast horizon ahead of the day the forecast is made, known as the *forecast epoch*.

2.4.3 Linear Methods

The persistence model is usually considered a naïve model and is a common first step used to compare the performance of various time series forecasting methods. Persistence is simple, easy to implement, and is used often as a baseline model to compare results; we choose the persistence model as the baseline model in this work. To model persistence, the previous time step value is persisted up to the desired forecasting horizon, H, where t is the day a forecast is made.

$$F_{10.7_t} = F_{10.7_{t+1}} = F_{10.7_{t+2}} = ... = F_{10.7_{t+H}} \tag{2.3}$$

A similar method to the SET algorithm was introduced by [28] for multi-step linear prediction of $F_{10.7}$. The authors used a linear regression method for predicting values at multiple time steps, rather than dynamic prediction, using a set of linear regression coefficients calculated for each horizon day. The authors also compared a machine learning approach using artificial neural networks, for prediction of $F_{10.7}$ with that of the linear model. The authors concluded that "forecasting via sophisticated artificial neural networks is not any better than a simple linear forecasting approach". An additional method by [32], performs multi step prediction using linear methods which considers both the correlation between predicted time steps as well as the heteroscedastic properties. While linear methods serve as a valuable initial approach, it's worth considering non-linear methods, which may yield more accurate results for modeling the complex and chaotic nature of solar irradiance.

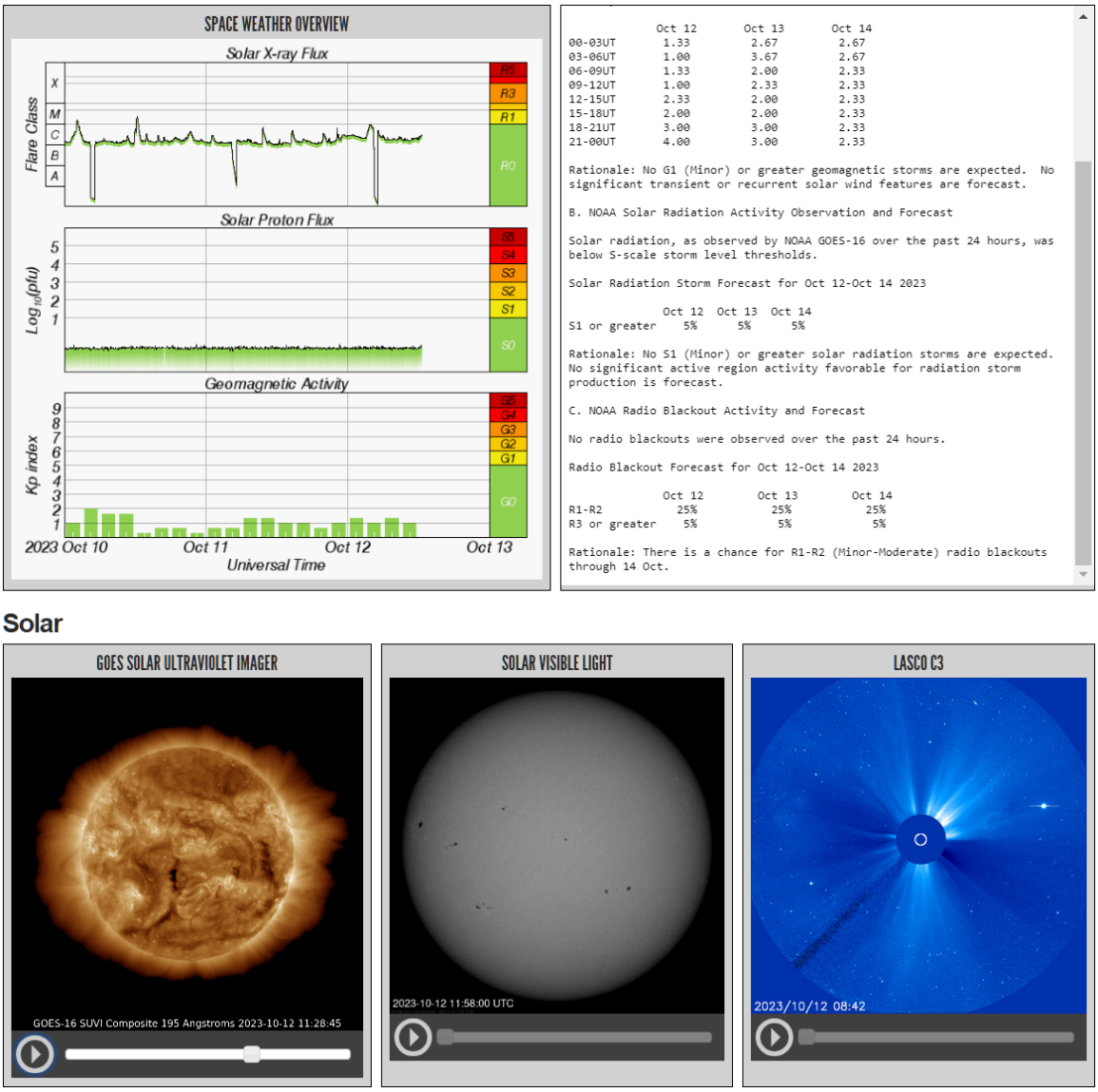


Figure 2.7: The NOAA SWPC enthusiast dashboard contains daily space weather products which are readily available for interested groups or individuals. https://www.swpc.noaa.gov/content/space-weather-enthusiasts-dashboard.

2.4.4 Non-Linear Methods

Huang et al. [29] investigated the usage of a Support Vector Regression (SVR) model to perform short term (3 day) predictions of $F_{10.7}$. An SVR approach uses a non-linear mapping of input data into a higher dimensional feature space, which is then fitted via linear regression. This linear regression on the higher dimensional space is optimized via minimization of a cost function. The authors determined that "our approach can perform well by using fewer training data points than the traditional neural network."

Stevenson et al. [12] discussed an approach for $F_{10.7}$ forecasting using neural networks. The authors implemented the network architecture created by [33], Neural Basis Expansion Analysis for Interpretable Time Series Forecast (N-BEATS), for multi step forecasting proxy values up to a horizon of 27 days. N-BEATS uses fully connected layers linked to non-linear basis functions to make both a forecast and back cast while training.

N-BEATS was proposed for usage without the need for knowledge of the domain. A neural network ensemble approach suggested by [33] for the direct predictions of $F_{10.7}$ and associated uncertainty was considered by [12]. N-BEATS was also used due to the computational efficiency over typical recurrent neural networks. It was determined that the NBEATS ensemble approach "systematically outperformed" both the statistical British Geological Survey (BGS) approach and the persistence model. The N-BEATS ensemble demonstrated either improved or comparable performance when contrasted with the CNES (French Space Agency) CLS model, which is a shallow neural network based on 4 different radio flux wavelengths.

Luo et al. [30] acknowledged previous methods for forecasting $F_{10.7}$ rely heavily on linear methods and propose the usage of Convolutional Neural Networks (CNNs) and a recurrent neural network (RNN) method known as long-short term memory (LSTM). The usage of linear methods for forecasting $F_{10.7}$ results in relatively stable for mid to long term predictions but fall short when high quality predictions on a smaller horizon are required. The author also proposes that a CNN will be of use in extracting features of the $F_{10.7}$ time series, and an LSTM will be useful in prediction of future values, the authors found an improvement over the linear methods by using such ML methods.

Some models for forecasting $F_{10.7}$ rely on inputs other than previous observations such as [29], who proposed the addition of a flux tube expansion factor, f_s, to the input of the SVR prediction method. The authors proposed that by including this as an input, the prediction of $F_{10.7}$ may be improved during large spikes in solar activity. The authors acknowledge that the SVR method is general and could use improvement, and that the addition of an input that represents general solar magnetic activity may help $F_{10.7}$ prediction, due to subtle changes in the solar magnetic field.

Benson [34] used an LSTM model to forecast various solar proxy and geomagnetic indices simultaneously. The authors considered the interaction between $F_{10.7}$, F_{30}, F_{15}, geomagnetic indices, and solar imaging in forecasting of proxy values one solar rotation (27 days) in advance. The authors concluded that machine learning methods outperform linear regression, persistence, and statistical mean value models. It was also determined that the addition of solar imaging data improves predictions in comparison to using proxy values alone.

Henney et al.[35] introduce an approach for forecasting of $F_{10.7}$ using solar magnetic flux trans-

port modeling. This physics based approach by the authors is incorporated into the Solar Indices Forecasting Tool (SIFT), which utilizes solar magnetic field distribution estimated with the ADAPT flux transport model to empirically predict space weather parameters. The authors discuss that the ADAPT driven model seems to be practical for estimating $F_{10.7}$ over short time periods, and further improvements on solar far-side magnetic activity estimates are key for improvements on this method's forecasting efforts over longer time periods.

It should be noted, that apart from the ML approach using N-BEATS [12], none of these models provide uncertainty estimates with their forecasts. A key issue that needs to be addressed in the field of forecasting $F_{10.7}$ is uncertainty in the predicted value. By providing a probabilistic forecast, steps may be taken in operations with the knowledge of forecast uncertainty, such as best or worst case situations. This work extends prior machine learning techniques and incorporates neural network ensembles to enhance predictive capabilities beyond current operational linear methods. This work also offers probabilistic forecasts that can be sampled for operational purposes by the STM community.

2.4.5 The $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ drivers

The dataset for the $F_{10.7}$ proxy is the largest, with observed values being recorded since 1947. The other 3 drivers are newer, and thus limited by data being produced by spacecraft. For example, $S_{10.7}$ data prior to the launch of the SOHO spacecraft would not be possible as no measurements could have been made. Due to this limitation, available data for all four drivers (at a one-day cadence) exist between January 1, 1997 and the present day. Missing values are noticed in the $S_{10.7}$ driver between 6/25/1998 and 10/24/1998 and linear interpolation was used to fill in missing data, accounting for about 1% of the $S_{10.7}$ driver data. An 81-day averaged value of the drivers, illustrating the high correlation, can be seen in Figure 2.5.

Currently, SET provides the only forecasts for non-$F_{10.7}$ drivers. SET relies on the same univariate forecasting methods used for $F_{10.7}$, providing independent forecasts of all drivers using the *pth* order auto-regressive linear model (Equation 2.2). There are no current methods for providing probabilistic forecasts for these drivers. This work deals with providing methods which utilize machine learning to provide for the first time ever, probabilistic driver forecasts for $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$. This work also aims on utilizing machine learning approaches and novel combination methods to improve driver single point forecasting and provide uncertainty estimates.

Chapter 3

Machine Learning

The chapter introduces the critical concepts for machine learning techniques and application. This work utilizes ML models to effectively capture trends based on historical driver observations and produce probabilistic forecasts for each of the solar drivers. In order to discuss the ML methods applied in this work (Chapter: 4 Methodology), it is first necessary to introduce the key concepts of machine learning.

3.1 Background

Data is the foundational building block of machine-learned models. To effectively discuss how machine learning occurs, we must begin with a discussion on the importance of the data that is used to drive ML models. Data is the fundamental source of information that enables ML models to learn from patterns, make generalizations to new situations, adapt to changing conditions, and aid in decision making. The quality and representativeness of the data is crucial, as biased or incomplete data can lead to biased or inaccurate predictions. Data is not only essential for initial model development but also for continious improvement in models, making it critical for applying ML in the real world. The importance of good data for ML cannot be understated, poorly sampled data does not represent the overall data well and leads to poorly performing models, which is discussed further in Chapter 4.

Machine learning is a branch of computer science which aims to allow computers to "learn" without being directly programmed [36]. Originating in the 1950's, machine learning emphasizes practical objectives and applications; such as prediction and classification. Learning occurs when a computer "experiences"; occurring when a computer is exposed (fit) to data. An exact definition of

machine learning can be a bit unclear. Bi et al. [37] argues a divide between machine learning and other statistical methods; "machine learning emphasizes prediction accuracy over hypothesis-driven inference, usually focused on large, high-dimensional data sets." A growing focus in the machine learning field, is to create models for which common statistical methods fail. Machine learning models can range from very simple to incredibly complicated.

A fundamental example of a data-driven machine learning technique which most people know, is linear regression. In linear regression, data is needed to fit a linear curve. It may seem, at first, that linear regression is too simple to be called machine learning. However, according to the general description of machine learning, linear regression is indeed an example of machine learning! In order to fit a linear regression model; data is needed to "teach" the model what trends occur and predictions on new data can be made without the need for explicit programming. With simple linear regression, the standard equation for a line is used for modeling,

$$\hat{y} = mx + b + \epsilon \tag{3.1}$$

where $\hat{y}$ is the output of the linear regression model, m is the slope, x is the input, ϵ is the random error, and b is the intercept. A linear regression model is fit to data by first assuming an initial value for m and b. As input data is passed into the model, the output is compared to the truth and an associated error is calculated. The goal of linear regression, is to minimize the error by varying the parameters, m and b. An example of a fit linear regression model and associated errors can be seen in Figure 3.1.

Linear regression works reasonably well in the case where data is close to linear. In the case of non-linear data, linear modeling approaches will inevitably fail and non-linear methods can be utilized, which can be seen in Figure 3.2. Once data is highly non-linear and contains more than a few dimensions, typical regression methods become difficult.

To tackle difficult regression tasks where data is non-linear, such as time series forecasting of solar drivers, state-of-the-art machine learning techniques can be applied. One of the newest models for sequential data tasks, *transformers*, have shown success in capturing long range dependencies in data [38]. Transformers rely, in part, on attention mechanisms; an identification of importance of a previous value. As a state-of-the-art model, initial investigation into application of transformers for solar driver forecasting was performed but did not show an improvement over baseline methods. Transformer application is discussed further in Section 6.1. Due to the challenge in usage of transformers, other well researched methods such as neural networks are investigated. Although prevalent

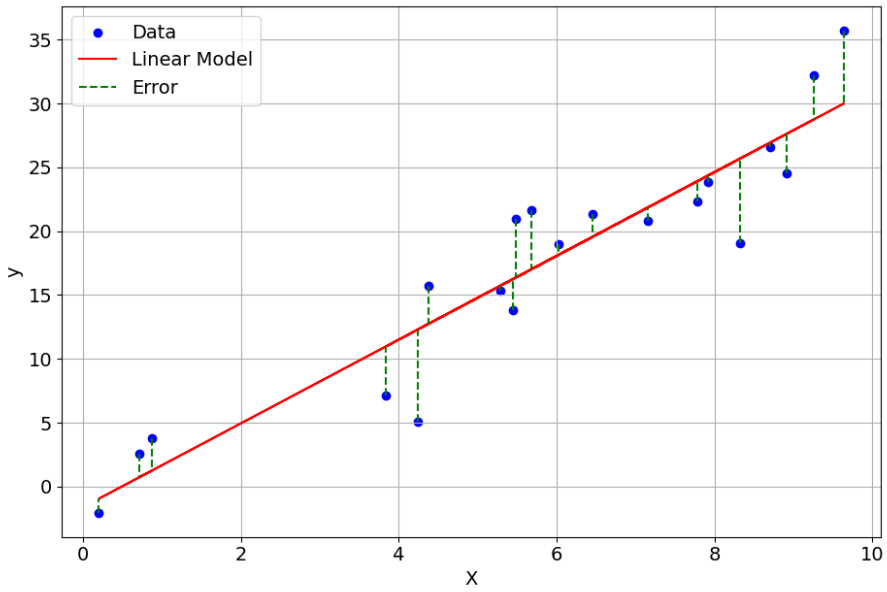


Figure 3.1: A linear regression model seeks to minimize the error between the regression model (red line) and the observed data (blue dots). The dashed lines indicate the magnitude of error, $|\hat{y} - y|$ associated with the model for a given input, x.

in science and technology today, many treat neural networks as "black boxes"; using them without understanding of how they function. To explain the approaches used in this work, it is critical to explain several of the key concepts used by neural networks.

3.2 Neural Networks

Over the past few decades, neural networks have emerged and become quite popular for both classification and regression tasks. Originally theorized by McCulloch and Pitts [39], application of models known as Artificial Neural Networks (ANNs) for tasks has shown a great deal of promise. With a similarity to the way neurons function in the human brain, an ANN is used as a universal function approximation which is considered a supervised learning approach using nonlinear functions. A typical usage for ANN regression models is to predict future time series values, which is the primary goal of this work. A "neural network" a subset of general machine learning algorithms.

3.2.1 Network Architectures

Neural networks can refer to many types of machine learning algorithms such as; convolutional neural networks, recurrent neural networks, multi-layer perceptron, and long short term memory models. The various neural networks can be used for many different tasks. The various NNs are

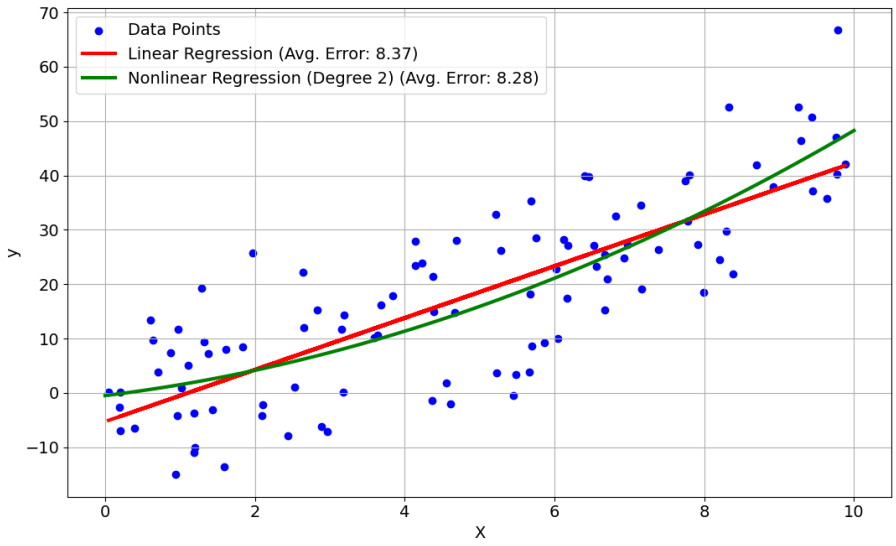


Figure 3.2: Even mildly non-linear data can be better suited by modeling with a non-linear approach, like a quadratic model.

skilled at specific tasks; CNNs are effective at processing grid like data, such as pictures and videos [40]; and recurrent neural networks are skilled at sequential data like natural language processing (NLP). For this work, it is necessary to investigate methods which have shown promise in time series forecasting. We choose to investigate two main neural network types. MLPs have shown to be effective in atmospheric sciences [41], geomagnetic index forecasting [42], and solar flare forecasting [43]. Long-short term memory models are skilled at time series tasks and have also had success in space weather such as the works by [12], [44], and [45].

3.2.1.1 Multi Layer Perceptron (MLP)

One of the most common types of neural network is the Multi-Layer Perceptron (MLP). Introduced in 1958 by Rosenblatt [46]; an input (or set of inputs) are introduced into a layer of neurons. A general example of the structure of an MLP can be seen in Figure 3.3. A chosen activation function is applied to inputs (or weights in further) layers. For the case of regression, the penultimate layer outputs pass through a linear activation function and are output from the model. These outputs can be taken, post processed if necessary, and are considered a prediction made by the model.

An MLP is a fundamental type of neural network which can solve non-linear tasks. By using architectures which contain sequential hidden layers; an abstract representation of data can provide better results than a single hidden layer. MLPs can have a single hidden layer or many hidden layers; with few or many neurons each. Given a single hidden layer with a finite number of neurons and

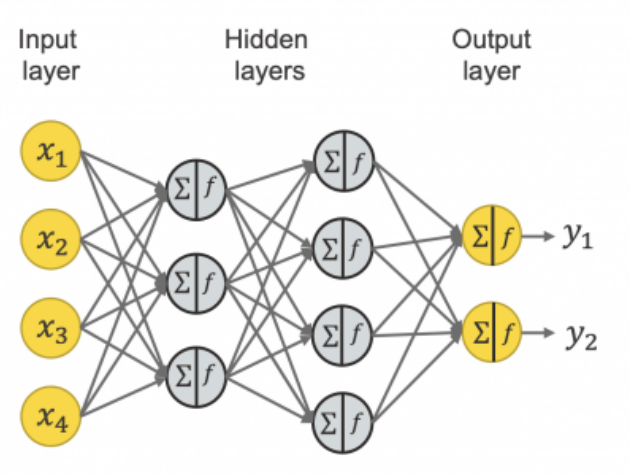


Figure 3.3: An ANN has an input layer, one (or more) hidden layers, and an output layer. Weights are updated via a backpropagation algorithm to minimize a chosen loss function [47].

weights, any continuous function can be approximated arbitrarily well [48]. In practice, there exists computational limitations for very large models. RAM (random access memory) for a CPU (central processing unit) and VRAM (video random access memory) for a GPU (graphics processing unit) are potential bottlenecks. Available memory limits the ability to store all necessary weights and perform calculations with those weights. A near infinitely large NN may be great but is impractical. Network complexity must be balanced; too many weights can overload the available memory, while too few weights limit the model performance. Too many weights may also cause difficulty for training and overfitting when limited data is available. MLPs lack features which identify trends outside of the explicitly provided inputs. It may be beneficial for sequential data models to have features which consider previously seen inputs and outputs; an important property seen in long-short term memory models.

3.2.1.2 Long-Short Term Memory (LSTM)

Specifically created for sequential tasks; Long-Short Term Memory (LSTM), was introduced by [49]. LSTM is a modification of the traditional Recurrent Neural Network (RNN), which prevents the vanishing/exploding gradient problems that were encountered during training of the original RNN. LSTM leverages previously seen time series values stored in the LSTM cell's hidden state "short-term memory", seen in Figure 3.4, to make a skilled prediction. This makes an LSTM model suitable for space weather applications where it is important to capture the hysteresis of the system.

During training, the LSTM learns not only the short term trends and how to predict data, but also the relevance of previous information, even with large time lags. This feature makes LSTMs

versatile and capable of picking up general trends. In the context of solar driver forecasting, the LSTM can learn underlying patterns in the data, allowing it to potentially outperform baseline models.

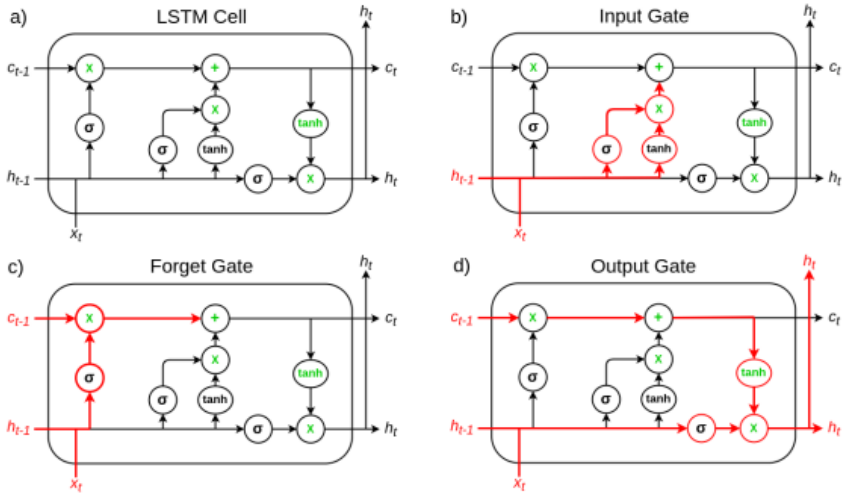


Figure 3.4: Overall construction of the LSTM cell (a) with the input gate (b), forget gate (c), and output gate (d) highlighted in red. Green is used to denote point-wise operations. [10]

In Figure 3.4, x refers to the input to the cell, c refers to the cell's internal state/memory, and h is the output. t and $t-1$ denote the current and previous step, respectively. The three σ nodes refer to internal layers with the sigmoid activation function, and tanh refers to either a layer with the tanh activation function or a point-wise operation – dependent on the color. The LSTM cell (a) is shown with each aforementioned gate highlighted: input gate (b), forget gate (c), and output gate (d). The internal sigmoid layers function similar to typical binary gates. If a gate is open (1), information passes through. Conversely if a gate is closed (0), no information gets through. As sigmoid has a continuous range between zero and one, it is ideal to function as a gate while keeping the LSTM cell differentiable. For the input gate, the input and previous output information get passed both the the sigmoid and tanh layers. The tanh layer acts as a normal neural network layer, while the sigmoid layer determines how much out the tanh output passes through.

3.2.2 Training Neural Networks

The desire, when using a neural network, is to create a model such that predictions and/or decisions can be made without the need for explicit programming. Supervised training is the process of using known data (separated into input/output pairs) to minimize the difference between model predictions and the ground truth. Training is typically done using optimization algorithms such as

gradient descent, which adjusts model parameters in the direction of the largest decrease of model loss function to minimize the error. The training process of a neural network is similar to the "fitting" of a linear regression model.

Training of a neural network model is the step where changes are made to weights and biases within the model in order to improve the output (similar to changing of the slope and intercept in Equation 3.1). Wasserman and Schwartz [50] provide a metaphor for the training step, claiming that training can be thought as "teaching by example". The model makes an educated guess based on the inputs, compares with the expected output, changes weights/biases, and makes another educated guess and compares once more. Training allows for the weights and biases to be adjusted to minimize a given optimization loss function. The weights and biases are internal to the model and act like "dials" that can be turned to improve results.

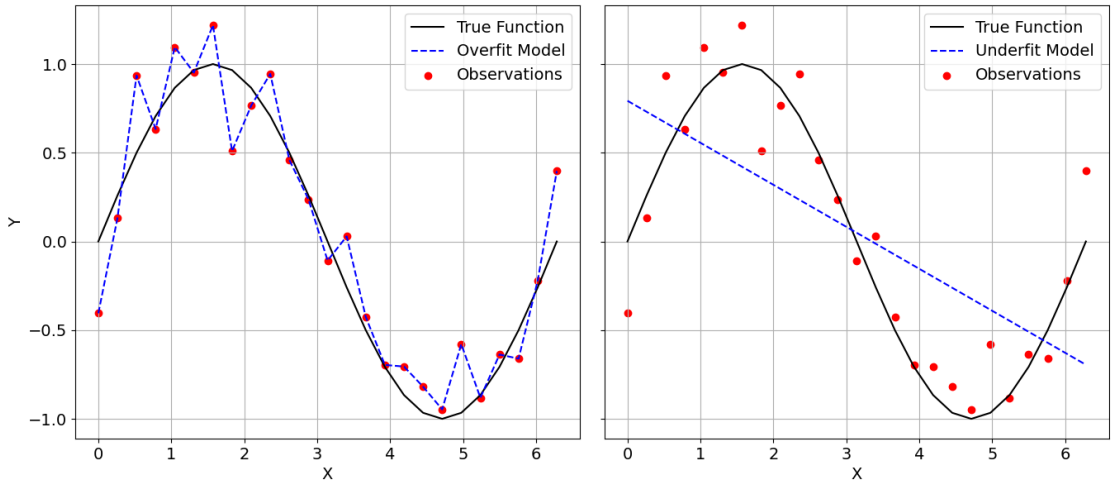


Figure 3.5: Left: Although an over fit model captures all of the data, it is expected to perform poorly on unseen data, becoming too specialized on the training data set. Right: An under fit model does not have much skill and may either need further training or a different architecture altogether.

A loss function is used to "score" the output of the model against the expected result and is necessary for training. Many regression tasks use Mean Squared Error (MSE) or Mean Absolute Error (MAE) as a loss function. Quantile loss and Huber Loss can also be used, but are generally less common. In this work, we use the two most common regression loss functions,

$$MSE = \frac{1}{N}\sum_{i}^{N}(y_i - \hat{y}_i)^2 \tag{3.2}$$

$$MAE = \frac{1}{N}\sum_{i}^{N}|y_i - \hat{y}_i| \tag{3.3}$$

where N is the number of predictions made for a given set, y_i is the expected value of a given sample, and $\hat{y}_i$ is the output of the model for a given sample. There are a few key challenges that arise when training a model. Two of these key challenges are overfitting and underfitting of a model.

In the case of overfitting; a model may learn the training data too well and essentially memorize rather than learn. A model which experiences overfitting, is expected to perform well on training data but poorly on unseen data. This lack of performance on new data is a critical issue, since the purpose of training a model is to use it on new data! Underfitting occurs when a model may not have received enough data to train correctly and is unable to identify underlying patterns during prediction. In both cases, poor generalization is expected on unseen data. To prevent overfitting of a model, regularization techniques such as the use of a validation set are often used.

As described by [51]; in the context of deep learning, regularization is a technology aimed at improving the generalization ability of a model. Regularization can be done a few ways, two of the most common techniques include the use of a validation set, and dropout [52]. A validation set is a set of data which has been hidden from a model during the training process. After a step of training is performed, the model makes predictions using the inputs of the validation set. The model predictions are compared to the validation set ground truths and an error term is calculated. Another training step is taken and validation predictions are evaluated again. If the error decreases, we say that the model has improved and is generalizing better. The validation set is not used to update model weights, but is instead used to "check" if a model is generalizing well. Regularization with a validation set is a popular approach, but can be difficult to implement in cases where data is not abundant [53].

Dropout is another powerful tool for regularization in deep learning. During training, a random subset of neurons are "dropped out" (output set to 0). When dropout occurs, the network output is perturbed significantly; many important connections between neurons are severed. In response to these severed connections, the network is encouraged to spread important connections across its architecture. By spreading the important connections across the network, regularization is greatly affected and has been studied extensively [54], [55], [56]. Multiple regularization techniques can be used. By combining dropout and validation data, regularization becomes more robust.

3.2.2.1 Hyperparameters

Hyperparameters are model parameters that are user specified and control the learning process. Examples of hyperparameters can include learning rate, batch size, number of layers, optimizers, and activation functions. When a set of hyperparameters are used to construct a model; it is referred to as a model's architecture. Models with different architectures can provide vastly different levels of performance. It should be noted that with number of model parameters is proportional to computational power. Choosing a suitable architecture is an incredibly important and challenging step in applying machine learning. A neural network with a given architecture may reach a point where it can no longer improve on the task. A model may result in a minimized loss function but does not perform as needed for the task. It may become necessary to change a model's hyperparameters in order to reduce the loss further.

Hyperparameter tuning, or *tuning*, is the process of experimenting with a variety of model hyperparameters to find optimal values. To perform tuning, hyperparameters are chosen, several training steps are taken, and an error value is calculated. The parameters which produce a minimum error are considered optimal and are used for further training. The initial ranges of hyperparameters to try are supplied by the user and tuners can follow schemes such as "Grid Search" or "Bayesian Optimization". Hyperparameter tuning can be performed via KerasTuner, a hyperparameter optimization framework in the Keras Machine Learning API. A range of hyperparameters (seen and discussed in Sections 4.4 and 5.2.4) are considered. KerasTuner is capable of outputting a list of models and hyperparameters which produced the lowest loss values for given inputs. A key hyperparameter in producing different outputs, an activation function changes all outputs in a given layer and may lead to more favorable results.

3.2.2.2 Activation Functions

An activation function is chosen by the user and assigned to each layer in a neural network. The activation function is used to determine the activation level of a specific neuron and can add nonlinear approximations to the model (if the activation function is non-linear). For each neuron, a weighted sum of inputs is passed through an activation function to produce an output, seen in Figure 3.3. In this work, we consider several activation functions for use in both the MLP and the LSTM models; whose functional forms can be seen in Table 3.1.

3.2.2.3 Optimizers

During the training process, a model's weights and biases need to be updated. If model weights and biases were never changed, a model would produce the same output and no skill would exist. Backpropagation is the fundamental algorithm which leads to this updating. The algorithm leads to "learning" and improved model performance. Backpropagation can be broken down into several main steps:

1. **Forward Pass:** A sample of data is input into the model and predictions are made on that sample.
2. **Error Calculation:** A comparison between prediction and expected output leads to an associated error.
3. **Backwards Pass:** Weights and biases of a model are adjusted starting near the output layer and working toward the input layer in order to change the error value.
4. **Weight Update:** Based on the amount that each weight or bias contributed to error, an update is performed using an optimization algorithm such as gradient descent.

Table 3.1: A range of activation functions which are used in this work.

Name	**Activation Function,** $f(X)$
Linear	$f(X) = X$
Tanh	$f(x) = \frac{e^X - e^{-X}}{e^X + e^{-X}}$
ReLU	$f(X) = max(0, X)$
ELU	$f(X) = X$ if $X > 0$ $f(X) = \alpha(e^X - 1)$ if $X \leq 0$
Softsign	$f(X) = \frac{X}{1+\|X\|}$
Sigmoid	$f(X) = \frac{1}{e^X + e^{-X}}$

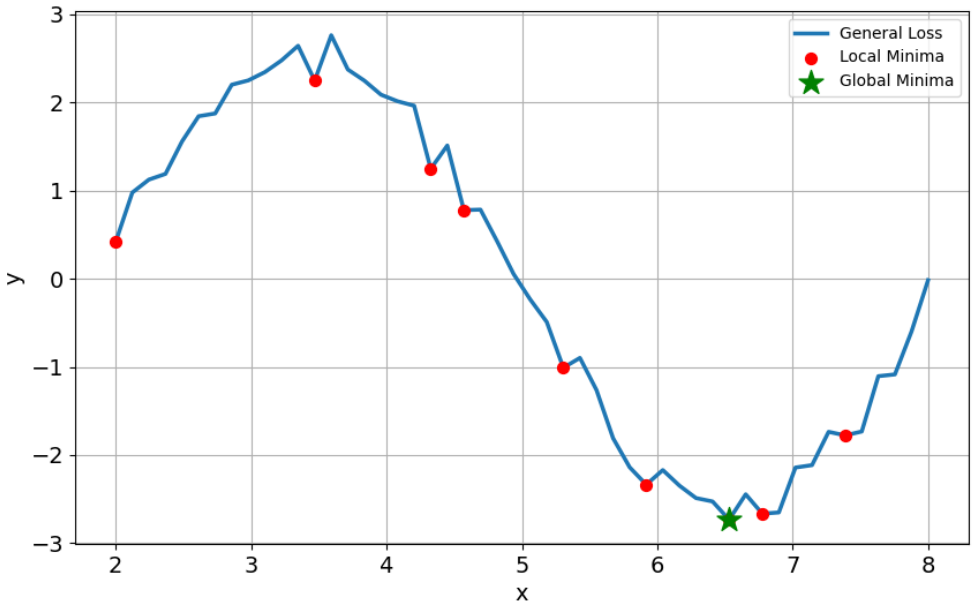


Figure 3.6: Depending on starting position (weight initialization), going in the direction of largest descent (gradient descent algorithms) will lead to a wide variety of minimum values. Such a loss architecture in high dimensionality becomes exponentially more difficult.

5. **Repeat:** This process is repeated many times through the entire data set (epochs), to continually improve performance.

After many training steps, one can expect the weights and biases to produce a minimal loss value. Ideally, a global minimum is achieved and the calculated weights and biases result in the best possible performance of a model. A global minimum is not guaranteed to be reached and a training process may lead to only a local minimum. The minimum loss reached heavily depends on the initial values of the weights and biases, as well as the overall architecture of the model. A visual representation of minimums in a loss landscape can be seen in Figure 3.6. Models which have the same architecture but different weight initialization may reach drastically different minimum loss values after training occurs [57] [58] . To encourage models to reach a more favorable minimum, a variety of weight initializations and architectures should be tested.

3.3 Data Preparation for Neural Networks

It is necessary to carefully preprocess data to effectively apply and train neural network models. It was shown that standardization of data is a critical step to both improve results and decrease

computational time. Data is standardized to improve convergence rates during training, as model parameters are encouraged to be lower and differentiation steps are more easily calculated [59]. To standardize the data, we use the standardization equation,

$$\widetilde{X} = \frac{X - mean(X)}{standard\,deviation(X)} = \frac{X - \bar{X}}{\sigma_X} \tag{3.4}$$

where X represents any arbitrary variable, $\bar{X}$ is the average of X, and σ is the standard deviation.

To transform the data set into a form that can be learned by a neural network, one must split the data into input and output (or target) pairs. Inputs and output pairs are needed for supervised learning. For time series forecasting, it is common to consider both lookback and forecast horizons to split the data. Lookback, L, represents the past observations to include into a sample for the model. An example of creating an array of lookback values from previous $F_{10.7}$ is given as,

$$\overrightarrow{L} = [F_{10.7_{t-1}}, F_{10.7_{t-2}}, F_{10.7_{t-3}}, ..., F_{10.7_{t-L}}] \tag{3.5}$$

where $t - i$ represents the day at which the $F_{10.7}$ value was recorded. The lookback can have a significant effect on the predicted value by a model. A larger lookback may allow a long-term trend to be identified by the model but may hinder the models' short-term performance due to lack of short-term trends that would have been seen directly in input data. It is necessary to investigate the effects of various lookbacks on the performance of a model as it may be beneficial to combine various lookbacks, gaining the benefits of both. For both MLP and LSTM models, we must specify both an amount of time steps to make the prediction for, the horizon (H) and the amount of previous days to use as an input, the lookback (L). An example of creation of a horizon array of $F_{10.7}$ is given as,

$$\overrightarrow{H} = [F_{10.7_t}, F_{10.7_{t+1}}, F_{10.7_{t+3}}, ..., F_{10.7_{t+H}}] \tag{3.6}$$

To prepare the data for both model types, the inputs and outputs are concatenated. The model inputs for a given time-step are a stacked input and output combination. Typically, this kind of data preparation can be accomplished using a *sliding window* approach. To be consistent in the analysis with the original benchmarking of driver forecasts done by Licata et al. [26] on the SET algorithm, we consider a short term prediction $H = 6$ Days, which will facilitate direct comparison.

The sliding window approach, seen in Figure 3.7, is used to separate the data into input/output pairs, called samples. The sliding windows create a set of samples. The samples created have input of size (Input Features x L) and output of size (Output Features x H). In the case of univariate

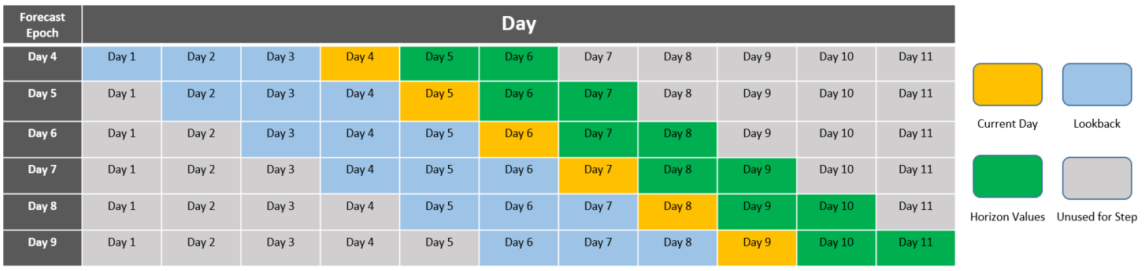


Figure 3.7: An example sliding window approach using a lookback of $L = 3$ days and a horizon of $H = 2$ days, with predictions starting on Day 4.

forecasting, we only consider the forecasting of a single feature. When performing multivariate forecasting, the number of features change to the number of drivers.

Licata et al.[8] discussed the necessary data preparation for LSTM model training. A standard feed forward NN requires the samples to have the same length about the first axis to achieve supervised training. "Consider the number of inputs (n_{inp}) and number of outputs (n_{out})) for a given data set with n samples. The concatenation will result in an array of shape n $\times$ ($n_{out} + n_{inp}$). The data must be stacked, so each row contains outputs and inputs for each lag-step (n_{LS}) and the current step. This orders in least-to-most recent from left-to-right. The data will now be of the shape n $\times$ ($n_{LS} + 1$)($n_{out} + n_{inp}$). The last n_{inp} columns are then dropped as they are not needed. The data can be split into training inputs and outputs where the first n $\times$ n_{LS}($n_{out} + n_{inp}$) columns are inputs and the last n_{out} columns are the associated output. The final step is to reshape the input data to the shape n $\times$ n_{LS} $\times$ ($n_{out} + n_{inp}$)."

Once input and output pairs have been created; it must be decided how to train a given model. Applying the training algorithm on all known data would provide a model that seems to work very well. If a model were created that way, it would be hard to tell if the model were skilled; there would be no way to test it! It is critical when training a model to maintain data that has not been seen during training, so that a model's performance can be truly evaluated. Without any testing data, the model would memorize everything and would lead to unrealistic performance. The question must be asked, "how can we effectively create such a testing set and still have enough data for our model to train?"

3.3.1 Data Splitting

For typical regression and model selection/training work flows, the data is split into 3 subsets; training data, validation data, and testing data (seen in Figure 3.8). The training data is what is

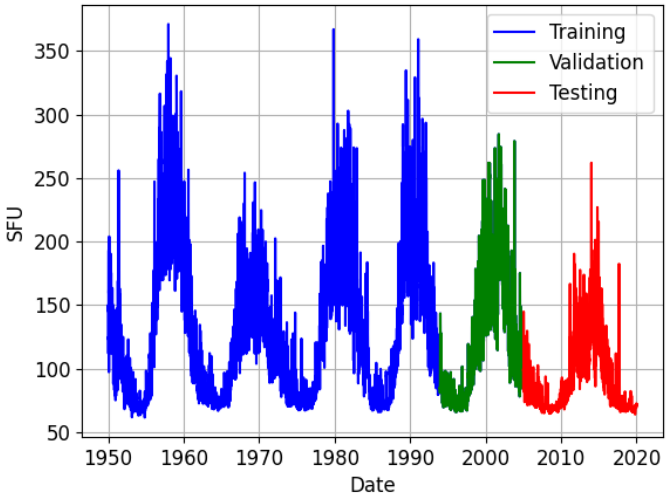


Figure 3.8: An example of a 70%-15%-15% holdout splitting technique used to train predictive models for $F_{10.7}$ [60].

used to teach the model by example; it is provided an input, makes a prediction, and compares against an expected output. This training set is used to adjust model weights and biases, which will change future outputs; A validation set is used as a method to mitigate model over-fitting, or a "remembering" of the training data, seen in Figure 3.5. The test set, is a set of data that has been entirely hidden from models during training steps. By keeping a hidden set, the model is "tested" on entirely new data and the model's ability to form generalizations can be evaluated. Regularization techniques must also be considered when determining the training data. Xu et al. [61] claim that the model validation step is the most important part of building a supervised learning model. The validation data is used after a training step and can provide insight into how well the model can generalize on data it has not seen yet.

In machine learning, the amount of data used to train a model greatly impacts the performance seen in final models. By providing larger number of samples to the model during training, better generalization can be expected. However, when training data is limited, worse generalization can be expected. It is important to ensure that the validation scheme chosen supplies the models with enough training data so that models can learn on a statistically similar subset to the full data set; this is done in this work to ensure that subsets capture similar levels of solar activity. Choosing what validation data to use and how much to use is non-trivial, especially when the amount of data is limited.

3.3.2 Validation Schemes

The holdout method is one of the most typical methods for splitting data into the three sets. A percentage for splitting, typically given in machine learning (ML) texts, is a 70%/15%/15% split for training/validation/testing sets. When considering time series data, such as the data set used in this work, typical holdout methods would preserve the temporal order of data by partitioning a percentage of data at the end of the full set, referred to as the test set. After the first partition is made, a second partition is made using a specified percentage of data at the end of the non-test set. This second partition will contain the data which can be used for training and validation steps, seen in Figure 3.8.

The holdout scheme has been used with great success in time series forecasting but other techniques, such as K-fold cross validation, have also shown promise [62]. However, when data lacks the sheer number of values typically seen for traditional validation schemes, holdout lacks the ability to create data sets which are statistically inconsistent and can lead to biased models, so we must choose a different validation method. A challenge arises due to the nature of time series and LSTM, data must maintain its time order, so we cannot randomly select samples throughout the dataset, as this would lead to out-of-order data. To combat this challenge, we introduce a novel method which samples data while maintaining its time-ordering and creates statistically consistent subsets, which is discussed in Section 4.1. With a robust scheme for splitting data into the three necessary sets, we can effectively train a set of neural network models.

3.4 Neural Network Ensembles

Typical probabilistic forecasting via machine learning methods involves generating a distribution of values for each time step, essentially predicting both a mean and variance of future time steps. With this method, inherent assumptions of the distribution of future values are made during training or sampling. Assumptions such as a specific distribution, like Gaussian or Poisson, are made. Extensive work in the field of machine learning has led to advancements in deterministic forecasting but due to limitations encountered in the learning process, a single machine learning architecture may not perform well in all areas of forecasting. For example, a model may be more skilled at predicting only low values, providing predictions with a constant bias. This lack of skill in certain areas can be addressed by considering a neural network model ensemble.

A neural network ensemble can be used to provide an improved forecast when compared to even

the best performing individual models [63]. Neural network ensembles are created using multiple models to provide outputs for a given set of inputs. Typical regression models provide a deterministic forecast, which only provides a single output for a given input. A neural network ensemble uses the concept of diversity [64], to allow for predictions to be spread across models with different strengths. An ensemble can be thought of as a divide and conquer approach. Neural network model ensemble diversity can be encouraged by considering varying architectures, model types, weight initialization, varying loss functions, and varied inputs. Variation of inputs have been used successfully in proxy forecasting [60] and [12].

A successful ensemble is one with accurate predictors that makes errors in different parts of the input space [65]. The authors further discuss the problem of combining ensemble predictions, or ensemble integration, and a process for removing models for which redundancy occurs. The ensemble process can be reduced to three main steps [66] and is referred to as the overproduce and choose method. The steps for creating an ensemble are as follows:

1. Generation
2. Pruning
3. Integration

Generation is the first step of this method and involves creating a set of models that attempt to fit the data. Pruning involves eliminating some of the models generated in the first step, usually models that provide redundant predictions. Integration involves determining the best way to combine these models. Using these 3 steps, a set of models can be transformed into an ensemble. To generate the models used in this work, we consider the nature of local minimums in the loss function landscape.

Due to the stochastic nature of the training process when weights are initialized randomly, different local minima may be encountered, seen in Figure 3.6. Models may reach different local minimums and can be combined to produce a favorable result. To select skilled architectures and to improve model performance, one can consider creating and training models several times per architecture, with varied initial weights. Since model hyperparameters (Section 3.2.2.1 greatly effect the individual performance of models; they can be leveraged to produce different models. By considering models created with a wide variety of hyperparameters, one can use them together as members of a neural network ensemble. Once the ensemble members are generated; it is a difficult task to determine which models should be used and how to combine them to create a probabilistic forecast.

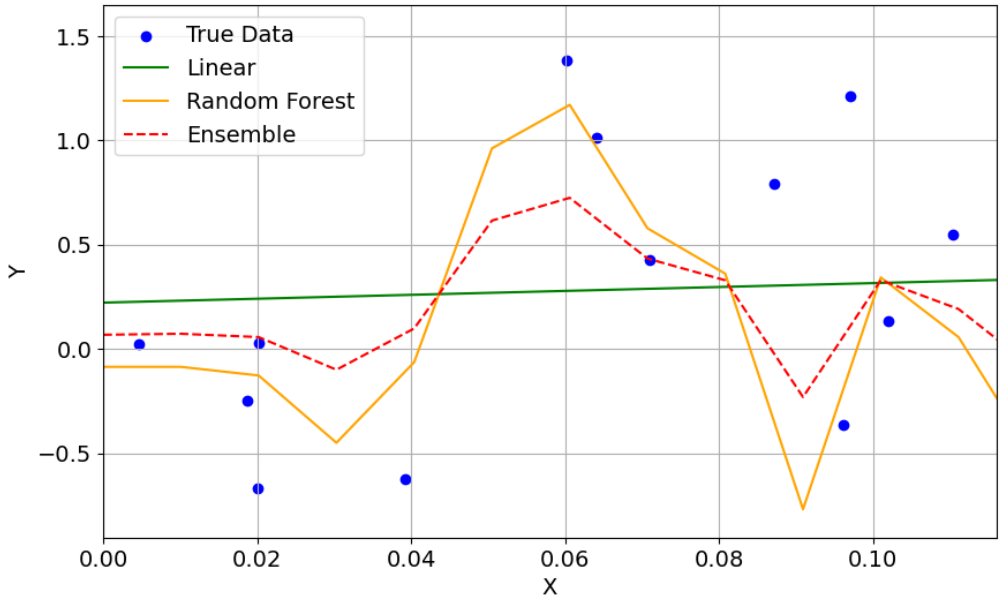


Figure 3.9: Toy Problem: We see that there are cases where a simple ensemble (average prediction) performs both better and worse than individual models. An ensemble's effectiveness depends greatly on its constituent models.

3.4.1 Diversity

By leveraging the concept of neural network ensembles, a combination of various models may improve prediction when compared to even the best performing single model [63]; a simple example is given in Figure 3.9. In addition to an improvement of direct value forecasting, an ensemble approach will create a distribution of predicted values, for which statistics can be evaluated and an uncertainty analysis can be performed. Even though each model is deterministic, a collection of models can be trained and combined so that a distribution of outputs is seen for any given sample. By evaluating the statistics associated with the ensemble prediction, uncertainty in prediction can be quantified which is referred to as uncertainty quantification (UQ). It is desired to achieve robust and reliable uncertainty estimates as well as accurate forecasting; to do this, various skills are needed across the ensemble members.

A difficult question is posed when generating a set of models for an ensemble, "which models should be used and which discarded?" This can be addressed by a key concept in model ensembles known as *diversity*. Model diversity is defined as the study of the degree of disagreement between models [65][64]. Another method of ensemble generation, discussed by [20], involves using interaction between ensemble members during training, in an evolutionary ensemble. To use neural networks

for regression tasks; we must decide what data to include; what base models to begin with; how these models perform; and whether using ensemble methods is advantageous.

A neural network model ensemble uses diversity to allow for predictions to be spread across models with different strengths. Model ensemble diversity can be encouraged by considering varying architectures, model types, weight initialization, varying loss functions, and varied inputs. Variation of inputs have been used successfully in deep learning approaches for proxy forecasting [60][12]. Several methods for promoting diversity in model generation are discussed by Sagi et al. [67]

- **Input Manipulation:** Base models are fitted using different input data.
- **Manipulated Learning Algorithm:** Manipulation of the way each base model traverses the hypothesis space. In the case of neural networks, this can be achieved by varying hyper-parameters, weights, optimizer, etc.
- **Partitioning:** Dividing the original training set into subsets then using each subset to train different models.
- **Ensemble Hybridization:** A combination of 2 or more of the other methods when generating an ensemble.

The authors also discuss methods for combining outputs of the ensemble members in the integration step. It is important to gain a single meaningful prediction from a set of individual predictions. Ensemble members can be combined using various weighting methods. In the most basic form, one could take a simple average of ensemble members,

$$\overline{\hat{y}} = \frac{1}{M} \sum_{m=1}^{M} \hat{y}_m \tag{3.7}$$

where M is the number of ensemble members used, $\hat{y}_m$ is the prediction made by the m^{th} model, and $\overline{\hat{y}}$ is the ensemble prediction. It is important to ensure that the ensemble members are skilled in different tasks.

To promote further diversity in ensemble members, we consider manipulation of the learning algorithm by varying the model hyperparameters, an idea discussed by [67]. To accomplish this task, we consider manipulation via two methods. First, we explore diversity through varied architecture using KerasTuner. Secondly, we explore diversity via random weight initialization. Due to varied performance, it may be beneficial to also consider using more than one model type at a time. For example, there may be periods of time where an MLP outperforms an LSTM (or other model type). Combination of models (or model types) is critical to the overall performance of an ensemble.

3.4.2 Model Combination

Once the ensemble members are selected, one must carefully consider combination methods. The simplest method of combination is equal weighting (EW) [68]. The EW method is identical to the mathematical mean or average of prediction. Equal weighting is a good initial choice for combination; it is computationally inexpensive and has shown to outperform constituent models [12], [68].

Another method of combination to consider, is the mathematical median,

$$Median(x) = \begin{cases} x[\frac{n+1}{2}] & if\ n\ is\ odd \\ \frac{x[\frac{n}{2}]+x[\frac{n}{2}+1]}{2} & if\ n\ is\ even \end{cases} \tag{3.8}$$

where x is an ordered list of values in the data set and n is the number of values in the data set. The median has outperformed the mean in certain machine learning model ensemble tasks [69] [70]. By using the median output value, we avoid issues associated with the mean, such as outliers.

One must also consider that certain models are more skilled than others; that a model's output should be weighted more heavily than a model with worse performance. Weighted ensembles performed better than unweighted for upper atmosphere models [68]. We consider a *stacked* ensemble approach [71], which uses linear regression to optimally combine predictions. The stacking algorithm can be used to provides a set of weights, indicating which models have "more say" in the ensemble output.

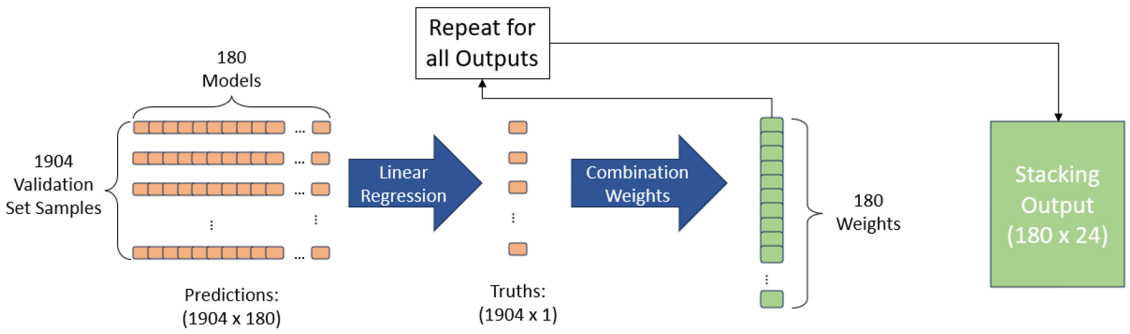


Figure 3.10: Twenty four linear regression models are fit using the validation data set, providing a 2-D array of weights. The weight array has 2 dimensions, 180: The number of models to combine and 24: The number of outputs per model (6-day prediction x 4 drivers).

Stacking, in practice, is the process of fitting a linear regression (Equation 3.4) of all model outputs and expected value over a set of samples. An ensemble made of 180 models will result in

180 coefficients (or weights), θ, associated with a single output. In order to implement stacking, a set of predictions and ground truths must be used, most often a hidden subset such as the validation set. The use of a validation set avoids any potential leakage into the test set, keeping it completely isolated from prior knowledge. A representation of the stacking process can be seen in Figure 3.10, for an ensemble with 180 models and 24 outputs. Once models are combined the result is deterministic, and with a deterministic forecast important information related to uncertainty is lost; so one must consider the distribution of predictions as well as the combined prediction.

Chapter 4

Methodology

This chapter discusses the methods applied for probabilistic forecasting of solar drivers using neural network ensembles. We introduce several new methods for manipulating solar driver data to enhance ensemble diversity and improve forecasting. We also provide methods for making univariate and multivariate probabilistic forecasts, investigate different forecasting methods, and introduce the performance metrics necessary to compare predictions.

4.1 Data Sampling

The data available for $F_{10.7}$ has been available since 1947, shown by the black curve in Figure 4.1. Previous works for $F_{10.7}$ forecasting relied on this substantial dataset to create ML models which performed well, including the probabilistic forecasting method by [12]. Previous works were able to apply the holdout validation scheme directly to this dataset, as there have been many solar cycles between since 1947. The holdout method applied to $F_{10.7}$, seen in Figure 3.8, allowed for four solar cycles of training data, and one solar cycle each for validation and testing data. The various solar cycles seen in the training set allowed for model to be trained which had seen a good mixture of high and low solar activity levels. Additionally, by validating and testing on full solar cycles, models would be less biased and a better measure of performance could be seen. However, applying a similar method for the other drivers did not yield well trained models.

For direct comparison against other $F_{10.7}$ forecasting methods, we carefully consider our training, testing and validation subsets, seen in Table 4.1. The testing chosen for our original work of $F_{10.7}$ contained the test set used by N-BEATS [12] and the data set used for the benchmarking of the SET model discussed by [26].

Table 4.1: Splitting of the original $F_{10.7}$ dataset using holdout techniques.

Set	Dates (YYYY/MM/DD)	Ratio of Total Data [%]
Training	1950/01/01 - 1993/12/12	63
Validation	1993/12/13 - 2004/12/31	16
[a]Testing	2005/01/01 - 2020/01/01	21

[a]Contains sets used by both [12] and [26] for consistency in comparison.

The historical solar driver data for S_{10}, M_{10}, and Y_{10}, existing only from 1997-present, have fifty less years of observations than $F_{10.7}$, seen in Figure 4.1.As seen in Figure 4.1, approaches for forecasting $F_{10.7}$ by [60] and [12] contained data across approximately seven solar cycles, the other drivers (available since 1997) have only about two and a half solar cycles of data. Due to the number of solar cycles between 1997 and present, there is a lack of statistical similarity between the three datasets when using traditional holdout methods, seen in Figure 4.2a. It was deemed necessary to introduce a novel method for sampling the driver data such that data maintains its sequential nature and is statistically consistent across the three sets. This novel method, referred to as *striped sampling*, involves sampling data for week long chunks. A statistical analysis is performed to compare the traditional holdout validation scheme and the new *striped sampling* scheme. By using striped sampling, we effectively capture consistent statistics between our training, validation, and testing sets; seen in Figure 4.2b. This result indicates that with limited data, striped sampling

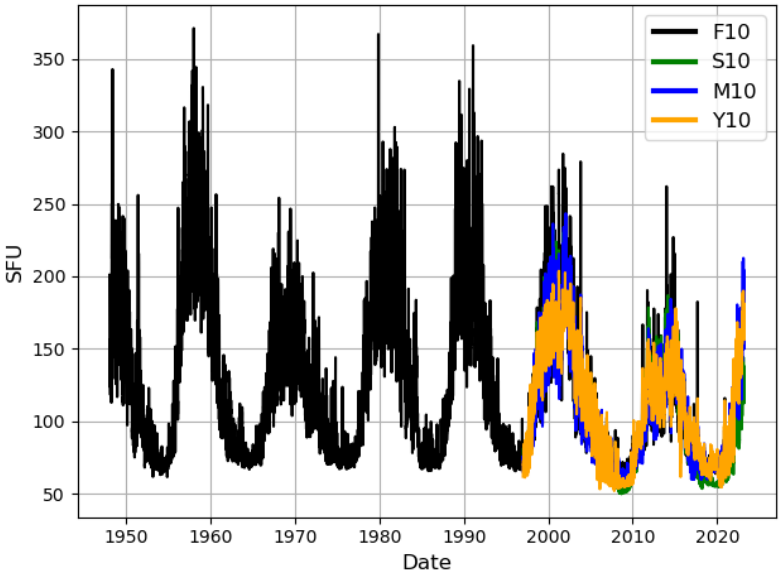
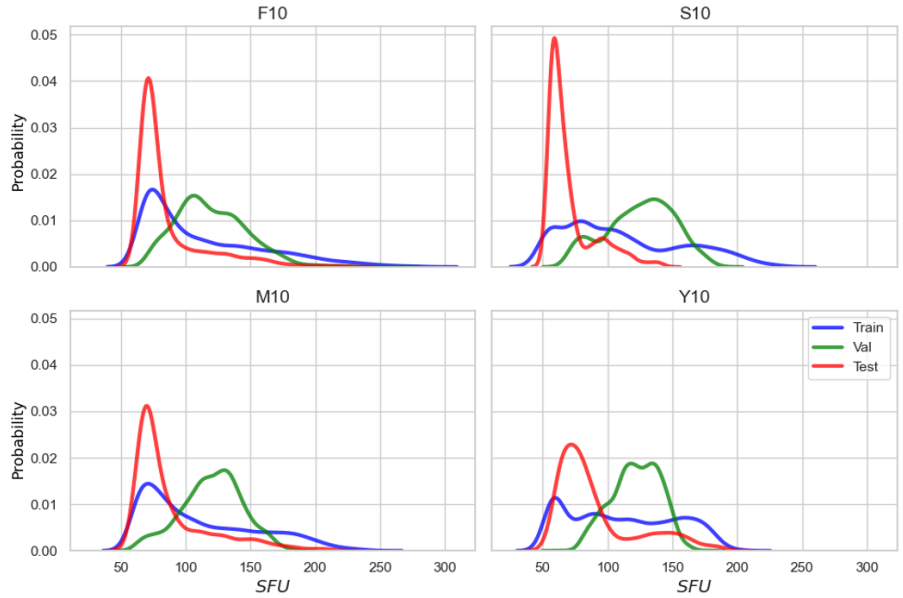
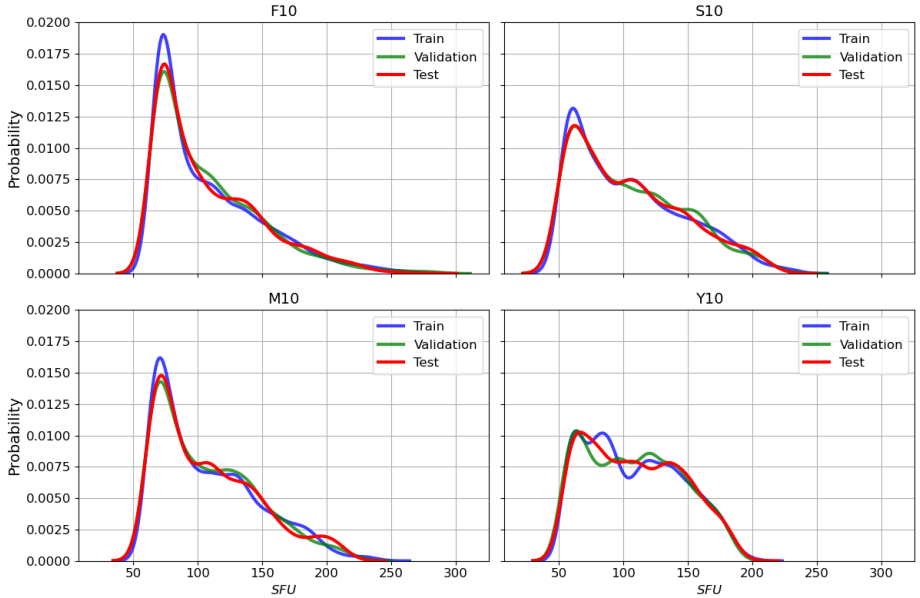


Figure 4.1: It is desired for a neural network model to see many repeated patterns during training. In the case of the newer drivers, parts of the solar cycle have only been seen a few times.

can create more useful datasets for ML methods. Once the data has been prepared and separated into sets using this new sampling, it is possible to train forecasting models effectively.



(a) Approximate PDFs indicate that the average solar activity level of the sets are drastically different, leading to difficulty with traditional machine learning sampling methods.



(b) By capturing similarly approximated PDFs between datasets using striped validation, we give machine learning models the best chance to effectively generalize and reduce potential bias.

Figure 4.2: **Top**: Holdout methods are used to split the data into the three ML subsets, which produces inconsistent statistics. **Bottom**: Striped sampling allows for consistent statistics between subsets.

The process of machine learning heavily relies on data, providing improperly sampled data pre-

vents models from learning effectively. The newly introduced approach to sampling data results in sets that are similar, but still have minor differences, as seen in Figure 4.2b. As solar activity level increases, notable differences are seen in $Y_{10.7}$, especially in the range of 70-130 SFU. Small variations can be seen in the other drivers, especially in the right tails of Figure 4.2b.

4.2 Data Preparation

Before employing training algorithms to apply machine learning techniques and create a neural network ensemble, it is necessary to complete data preparation steps, providing models with input/output pairs to learn from. To prepare our data, we apply the sliding window approach (Section 3.3). Additionally, to help models train more quickly, data is standardized (Equation 3.4), based on the statistics of the training data set. The statistics are used to standardize the data for the training, validation, and testing data sets. An additional step for processing data for use in LSTM models is discussed in Section 3.3. Once these preparation steps have been performed data can be fed used to create neural network models, and we must choose what length of forecast is desired.

4.2.1 Multi-Step and Dynamic Predictions

One notable difference between N-BEATS and the SET model, are their prediction types. The SET model relies on iterative forecasting, which we refer to as "dynamic forecasting" or "dynamic prediction". Dynamic forecasting relies on forecasts to be "chained" together to reach a desired forecast horizon. This method is inherent when using the linear approach seen in Equation 3.1. The N-BEATS ensemble approach relies on a different forecasting method; where multiple time steps are forecasted at once for a given input. We refer to forecasting where multiple days are predicted simultaneously as "multi-step forecasting". These very different types of forecasting may provide ensemble diversity and may improve overall performance when combined. An investigation into the benefits and shortcomings associated with both prediction types is necessary.

When splitting the data set using the sliding window approach, the outputs are of size H, and a model is directly trained to predict values for a horizon of H days. [72] concluded that dynamic forecasting typically outperform direct multi-step forecasts. To explore dynamic predictive methods, consider setting the training targets to a size of one, teaching the model to predict the next day. The model training is limited to prediction of only the next day and may perform very well at predicting one day rather than attempting to generalize multiple days at once. To create a dynamic forecast, we perform a single step prediction, shift the input by a time step, and append the predicted value

to the new input. This recursive method is done H times to generate our forecast over the desired horizon, illustrated in Figure 4.3. A benefit of this dynamic forecasting approach is the ability to actively change the prediction length. The model is trained for one day at a time but operationally could be used for any desired horizon, without the need for further training.

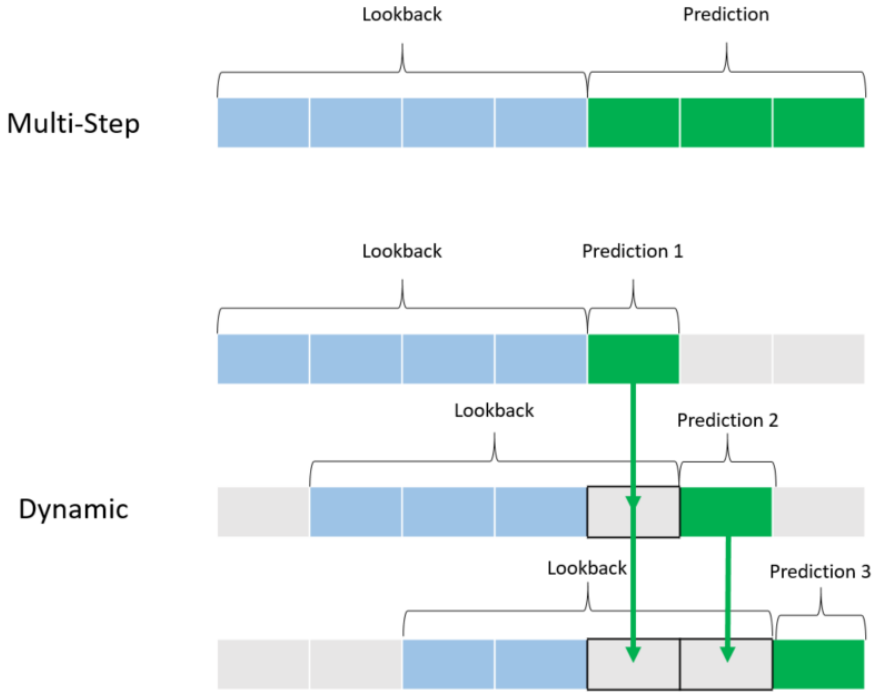


Figure 4.3: Consider a lookback of $L = 4$ days, we can predict a horizon of $H = 3$ days either by predicting all days at once (Multi-Step) or by recursively predicting single steps 3 times.

One potential problem with dynamic forecasting occurs when one considers a large forecast horizon. Every time a model makes a prediction, there will be an associated error with the output. As a model uses dynamic prediction, error terms can accumulate and led to instability and large errors. This instability occurs due to compounding errors [73], a prediction uses already predicted values as inputs, which each contain some predictive error. Since this work deals with short term forecasting, compounding errors from dynamic forecasting may not contribute as much as when long term forecasting occurs. It is important to investigate whether dynamic predictions have instability, if they can be used to enhance driver forecasts, and what inputs provide the best modeling capability.

4.3 Data Manipulation

There have been several investigations that have performed space weather proxy forecasting using both single-input and multi-input input models for $F_{10.7}$. These inputs usually contain autoregressive (AR) information, represented as previous observations of the proxy called the lookback window. In [12] the authors were able to improve upon the persistence baseline and other operational models, using only previous $F_{10.7}$ values by leveraging models created with different lookback windows. We build off of the previous work, considering AR forecasting methods which use a range of lookback values and additional novel data manipulation methods. We choose to limit our work to AR methods to enable a direct comparison between novel neural network ensembles and the operational baseline methods. In the original probabilistic $F_{10.7}$ work, there was little justification presented for choice of lookback window. To address this, we apply a range of lookback values and discuss affects on model performance in Section 5.1.1.

4.3.1 Backwards Averaging

Since an AR model only has previous values to use for forecasting, there may be difficulty in prediction using only those previous values, especially if a drastic change (a storm) occurs in the lookback window. We introduce an additional input known as a *backwards average* to extract more information from the previous values and provide a source for potential forecast stability,

$$\overline{F}_{10.7_B} = \frac{1}{B}\sum_{j=1}^{B}(F_{10.7_{T-j}}) \tag{4.1}$$

where B is the window of values to use for the backwards average. By providing AR models with backwards average, a short-term trend may be identified and may guide model towards more accurate predictions. The sensitivity of model performance to this backwards average value is evaluated and further discussed in Section 5.1.1.

4.3.2 PCA Rotation

Principal component analysis (PCA) is considered the most popular multivariate statistical technique and likely to be the oldest multivariate technique. PCA is typically used on large dimension data; compressing the size of the set while keeping the more important information. PCA involves a transformation from the original data into linear combinations of the original variables known as principal components (PCs). The PCs are calculated to maximize variance between the PCs and

constrains the components to be orthogonal [74].

Typical methods using PCA would truncate PCs to reduce the dimension of the dataset, allowing for easier applications of machine learning techniques; especially those involving very large or high dimension data. Since the PCs are a linear combination of the initial space, the dimensions are not interpretable; these new components no longer represent driver values explicitly, the steps to apply PCA algorithm are as follows:

1. Starting with a time series of four drivers (which are separated into the training, validation, and test sets), standardize the data based on statistics if the training set.

2. Calculate the covariance matrix based on the training set; a 4x4 symmetric matrix that contains the covariances associated with all pairs of variables, which is performed via *Numpy.cov()* in Python.

3. Compute eigenvectors and eigenvalues of covariance matrix $\mathbf{C}$ to identify PCs.

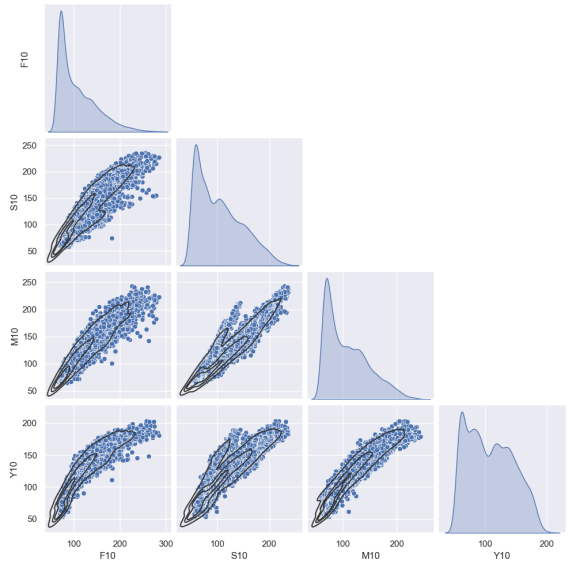
4. Sort eigenvalues and associated eigenvectors based on scale of eigenvalue (maximizing variance) and construct a feature vector matrix.

5. Perform ML methods (training, validation, and prediction) in the PCA rotated space using the feature vector matrix.

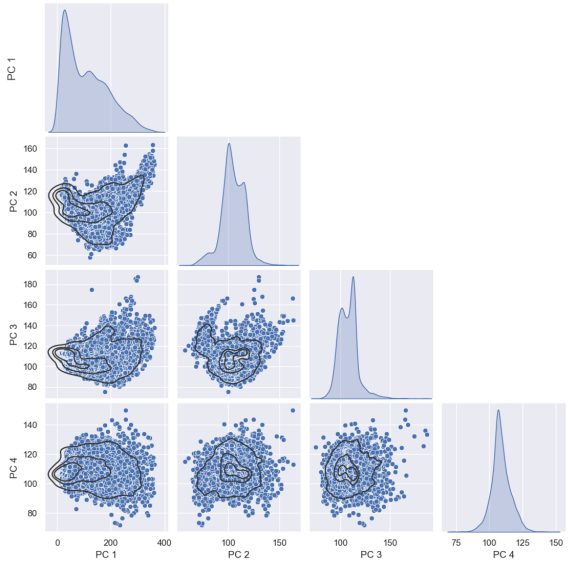
6. Transform predictions back into the original space and reverse standardize the outputs.

Redundant information is contained within the solar drivers and it may be considered less important to forecast them all at once. Applying ML techniques to make multivariate forecasts on highly correlated variables could be less useful. By applying a technique like PCA, we "untangle" our data, and force our dimensions to be orthogonal and have a maximized variance (less correlated than in Figure 4.4a). We have not seen a PCA rotation method used in the forecasting of density model drivers and introduce a rotation similar to the one discussed by [74]. Most ML applications for machine learning truncate PCs due to the small amount of variance that they capture [75]. We consider the typical PCA algorithm with no truncation, simply a "rotation" to maximize variance and create orthogonal PCs.

As a model attempts to learn from multiple variables, one must consider how similar the variables are. We aim to de-correlate the model inputs, to investigate the affect on multivariate model performance. We show the ability for PCA rotation to provide variables which have reduced correlation.



(a) The correlation matrix suggests that the $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ drivers are great candidates for transfer learning (Section 4.4.2) due to their high correlation with the $F_{10.7}$ driver.
Main Diagonal: Approximate distributions for individual drivers. **Lower Entries:** Correlation between pairs of drivers.



(b) **Main Diagonal:** Approximate distributions for individual PCs. **Lower Entries:** Correlation between pairs of PCs.

Figure 4.4: **Top**: Raw driver data is highly correlated and may hold promise for an ML approach known as transfer learning. **Bottom**: PCA rotation yields PCs which are significantly less correlated and should be investigated as ML model inputs.

Applying PCA rotation transforms the original drivers into principal components (PCs) with markedly distinct distributions. While the initial driver distributions were similar (Figure 4.4a), PCA rotation yields more unique distributions for the PCs, which are also less correlated, seen in Figure 4.4b. The rotated data can be used by neural networks in nearly the same way as unrotated

data. The only difference in the process of evaluating models with PC inputs, is that PCA based models will require data to be rotated back to the original driver space after predictions are made. After a reversal of the PCA transform is applied, predictions can be directly compared to other models. To our knowledge, this is the first investigation of applying a PCA rotation technique in the field of solar driver forecasting.

4.4 Neural Network Training

To promote further diversification in ensemble members, we consider manipulation of the learning algorithm by varying the model hyperparameters, an idea discussed by [67]. To accomplish this task, we consider manipulation via two methods. First, we explore diversity through varied architecture using a hyperparameter tuner. To select skilled architectures, hyperparameter tuning is performed via KerasTuner. KerasTuner is capable of outputting a list of models and hyperparameters which achieve optimal loss function values during the tuning process. We consider these results to be the best performing architectures for a given lookback and these architectures can be used to generate the individual ensemble members. Next, we explore diversity via random weight initialization, which was successfully used in ensemble models for forecasting of $F_{10.7}$ [12]. We also consider diversity which comes from the use of various types of models and predictions; MLP, LSTM, multi-step prediction, and dynamic prediction may all provide substantial diversity for neural network ensembles.

Tuner results from each architecture are used to create 10 random weight initialized copies and further training of models occur. The prediction of test set data is then made by each individual model and is saved for combination later, which is shown in Figure 4.5. Due to varied model performance, it may be beneficial to consider using more than one model type at a time. For example, there may be periods of time where an MLP outperforms an LSTM (or other models), so a hybrid ensemble approach like [76], will be considered. For univariate methods, we will evaluate the performance of combining the ensemble prediction across multiple model types, leveraging their differing skill. In the case of univariate predictions work, we first consider an unweighted average of predictions made by both MLP and LSTM ensembles.

4.4.1 Hyperparameters and Best Model Selection

One can create a large amount of models easily, but care should be taken to have model diversity. It is important to generate models with diversity in mind, and a scheme for generating and selecting models is needed, such as hyperparameter tuning. During hyperparameter tuning, it is necessary to

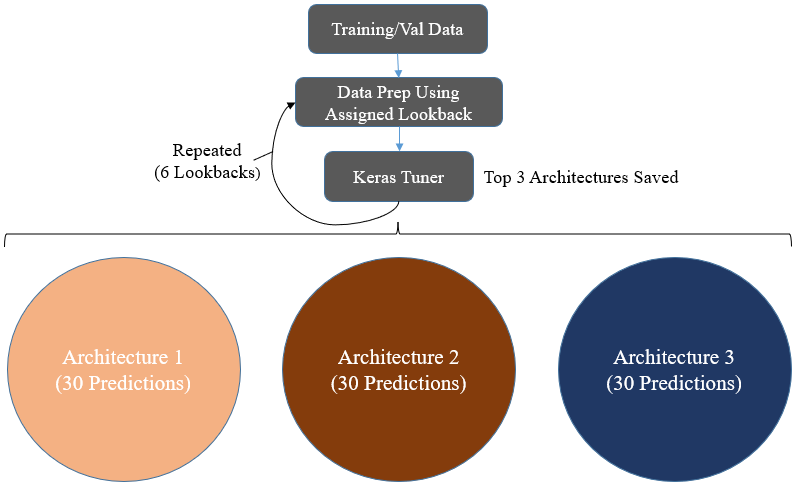


Figure 4.5: Predictions are made by 30 models of each architecture for each lookback step and are saved for combination later. A total of 180 models are used to generate a probabilistic forecast and associated uncertainty estimates.

specify ranges to perform a search over, which are seen in both Table 4.2 and Table 4.3. A Bayesian optimization scheme was used for selecting the best hyperparameters, as Bayesian optimization has been shown to optimize model hyperparameters, saving time and improves performance [77]. Based on the hyperparameter tuning results, the architectures which produce a minimal loss value on the validation set are selected as best performers. After tuning, the top 3 architectures are selected and saved to generate the base ensemble members. These architectures are used to generate individual models whose weights are initially randomized using a normal distribution, which is the default when creating a model in the ML API, Keras. These randomly initialized models are then trained further with an early stopping criterion, which is used to end training if performance is not improved. After training, predictions on the test set occur and are saved to files individually. It should be noted that ensemble integration has not occurred yet and that the predictions are independent at this point.

Table 4.2: Tuning configurations used to generate ensemble members at each lookback for MLP models.

MLP (Multi Layer Perceptron)

Tuner Option	Choice	Parameter	Value/Range
Scheme	Bayesian Optimization	Number of Dense Layers	[1-8]
Total Trials	100	Dense Neurons	[4-256]
Initial Points	5	Activation	[relu,tanh,sigmoid]
Repeats per Trial	3	Optimizer	[adam, SGD]
Minimization Parameter	MSE	Learning Rate	[.01, .001, .0001]
Epochs	100	Early Stopping Criteria	Validation Loss (MSE)
Early Stopping Patience	30		

Table 4.3: Tuning configurations used to generate ensemble members at each lookback for LSTM models.

LSTM (Long-Short Term Memory)

Tuner Option	Choice	Parameter	Value/Range
Scheme	Bayesian Optimization	Number of LSTM Layers	[1-3]
Total Trials	75	LSTM Neurons	[4-128]
Initial Points	5	LSTM Activation	tanh
Repeats per Trial	3	Number of Dense Layers	[1-4]
Minimization Parameter	MSE	Dense Neurons	[4-256]
Epochs	50	Dense Activations	[relu, tanh, sigmoid]
Early Stopping Criteria	Validation Loss (MSE)	Learning Rate	[.01, .001, .0001]
Early Stopping Patience	10	Optimizer	[adam, SGD]

In previous probabilistic forecasting work for $F_{10.7}$ [60], the author developed a neural network multi-lookback ensemble, which had better performance metrics than the prior NBEATS approach, but lacked a significant improvement in uncertainty estimates. It is believed that the variation in loss function considered by [12], led to more favorable uncertainty estimates and an overall better calibrated probabilistic forecast. Based on the results of [60], it is necessary to determine if the training loss function has a substantial impact on probabilistic model performance. The affects of optimization loss on the performance of models and ensemble diversity are covered in Section 5.1.2.

4.4.2 Transfer Learning

A common practice in the field of machine learning involves using a previously trained model (or set of models) as a starting point, known as *transfer learning.* Transfer learning is a powerful tool that allows the use of an already made model without the need for excessive training or hyperparameter selection. Due to the lack of a large available data set for $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$, seen in Figure 4.1, a transfer learning approach should be investigated, specifically using univariate models which were trained on the data abundant $F_{10.7}$ driver. It can be seen in Figure 4.4a, the drivers with limited data correlate well with the $F_{10.7}$ proxy. Transfer learning may provide reasonable forecasts for tasks that are related; such as highly correlated variables [78]. Transfer learning can also significantly improve the efficiency in learning by exploiting the relatedness between a data-scarce target task and a data-abundant source task [79].

To accomplish transfer learning, NN models which have been created for forecasting $F_{10.7}$ can be considered for the other drivers. To prepare for transfer learning, data for the newer drivers must be formatted identically to the data used by the original models. Models loaded for transfer learning may be starting "ahead" of newly created models and may be able to provide good performance

without the need for excessive training, only needing a small amount of training to "fine-tune" model weights and biases [80]. The models can then be evaluated on the training, validation, and test data.

Based on the work done by [60], we select the univariate MLE created from individual LSTM models due to its good performance metrics and well behaving uncertainty estimates. The model ensemble showed reasonable performance improvements over the SET algorithm when forecasts of $F_{10.7}$ were made and provided robust uncertainty estimates.

4.5 Neural Network Ensembles

With the basics of model ensembles discussed in Section 3.4 and currently used methods discussed in Section 2.4.2; it must be decided how best to approach forecasting solar drivers and whether ensembles will be useful. To use neural networks for prediction of the drivers; it must be decided what data to include, what base models to begin with, how these models perform, and whether using ensemble methods is advantageous. An overview of the methods used by our work is covered in Figure 4.6.

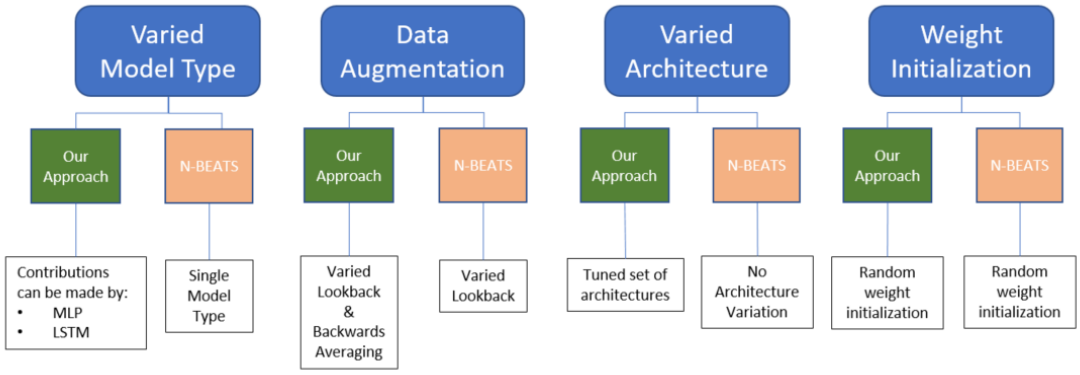


Figure 4.6: Example of how various ensemble diversity methods can be implemented. In this case between this work and the N-BEATS approach used by Stevenson et al. [12]

To create a neural network ensemble, one must consider models which have skill in various areas. The 27-day forecasting work done by [12], used a few methods, seen in Figure 4.6, to encourage model ensemble diversity. One successful source of diversity seen by the authors, involved combining models with various lookbacks. We refer to an ensemble which has been created using a mixture of lookbacks as a multi-lookback ensemble (MLE). The authors were able to improve over several forecasting methods and provide a probabilistic forecast for $F_{10.7}$. Short term prediction may function differently to the 27-day predictions; the sensitivity of model performance on lookback is

investigated to determine the best lookback values to use during training. Lookback is not the sole source of diversity in model ensembles and it is important to investigate other potential sources.

4.5.1 Univariate Approach (UV-MLE)

A logical step for forecasting $F_{10.7}$, $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ is to use the method described by [60] in Section 4.4 to create neural network ensembles which are tuned and trained, specifically for a single driver. The same steps are used to construct a set of models to forecast the drivers, using the training, validation, and test data splitting methods discussed in Section 3.3.1. Initial investigation into univariate approaches for $F_{10.7}$ made comparisons with prior work using the holdout validation scheme. However, when comparisons for forecasting all drivers are considered, models are created using the striped sampling scheme. A hyperparameter tuner is constructed for each driver and a set of lookbacks and backwards averaged values are considered for generating neural network ensemble members. A set of models are created for each driver and are trained and used to predict separately. After individual driver predictions are made, predictions are concatenated in a post-processing step to provide outputs for all drivers. We refer to models created this way as univariate multi-lookback ensembles (UV-MLE).

4.5.2 Multivariate Approach (MV-MLE)

Due to the high correlation between drivers, seen in Figure 4.4a, trends seen in one driver are most likely to be seen in the other drivers. Rather than limit a model to a single stream of data, we can consider a model which is input four sets of previous driver values and provides a forecast for all four simultaneously. Such a model would increase the dimensions of considered data by a factor of four; all inputs and outputs would involve all four drivers as opposed to just one.

By considering all drivers simultaneously, patterns seen in one driver may be useful for forecasting of another driver. For example, a short term decrease in $S_{10.7}$ may indicate a similar drop would occur in $M_{10.7}$, even if a pattern is not necessarily seen in previous $M_{10.7}$ data. In this work, simultaneous multivariate forecasting is performed using similar ensemble methods to the univariate models discussed in (Section 4.4) but using all drivers. This work is done to determine if such multivariate methods are more beneficial than univariate methods, and we refer to models created this way is multivariate multi-lookback ensembles (MV-MLE). We additionally choose to use LSTM models due to their skill and stable prediction type [60].

After creating and assessing the individual models for an ensemble, it is essential to establish

overall performance metrics based on a combined forecast and uncertainty estimates. We aim to use performance metrics to determine whether multivariate forecasting is better than univariate, and what data manipulation methods are beneficial.

4.6 Metrics

With regards to machine learning models, [81] poses a set of questions for the general ML community;

1. Are the developed machine learning models accurate?
2. Are the developed machine learning models useful to our task or field?
3. Are we properly validating the developed ML models?
4. How can we confidently answer "yes" to questions 1, 2, and 3?

Machine learning models need to be rigorously assessed. An assessment is required to ensure the validity of the developed model for understanding complex phenomenon, such as space weather; and to validate proper extension of a model toward a new/future data set. Traditionally, ML models use error metrics to assess their capability. Metrics are mathematical constructs which allow a measure of the closeness between the truth and prediction. A single metric may be useful, but is incapable of "telling the whole story." Similar to the ensemble diversity methods, error metrics should be diverse; allowing insight into the overall performance of a model [81].

Metrics used in the field of solar proxy forecasting vary from work to work, but [12] argues that the addition of many metrics allows for robust model evaluation. It is important to provide a diverse set of metrics for future analysis, as a large set of metrics can be used to compare future models more easily. When performing univariate forecasting, metrics apply to a single variable and performance is easily captured by error metrics. Although it may seem easiest to evaluate metrics for multivariate methods across the entire output, it is important to investigate performance on each individual driver. When we evaluate combining models with unequal weighting, it is inevitable to find models that may perform better on certain drivers. We consider using common metrics like root mean squared error (RMSE), mean absolute percentage error (MAPE), and the Pearson Correlation Coefficient (R); Equations 4.2, 4.3, and 4.4 respectively. We select these metrics to show general model performance, yet metrics should be generated for all drivers to better quantify the actual model skill for forecasting multiple drivers.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}^{N}(y_i - \hat{y}_i)^2} \tag{4.2}$$

$$MAPE = \frac{100\%}{N}\sum_{i}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{4.3}$$

$$R = \frac{cov(\hat{y}, y)}{\sigma_{\hat{y}}, \sigma_y} \tag{4.4}$$

The MAPE error metric is linear and, unlike MSE, does not give as much weighting to large errors [12]. The MAPE error metric is a good indicator of model performance but is aimed at single point prediction. Error in forecast is proportional to the number of days a forecast is made, for example, an error in a 1-day prediction is more than likely to be smaller than the errors associated with a 6-day prediction. The discussed metrics are no different and errors associated with 6-day prediction would dominate, and errors associated with 1 or 2-day predictions would not be apparent. To remedy the issue with dominant terms, a *relative metric* is considered. Relative metrics are an extension of the work in [82], which allows for a single metric over all days forecasted. Relative metrics distill error metrics down to a single easy to compare value for a given metric. The relative metric can be defined as a metric value which has been averaged over all forecast days, and is scaled against the baseline persistence model.

$$Relative\ X = \frac{1}{H_{max}}\sum_{h=1}^{H_{max}}\frac{X_{model,h}}{X_{persistence,h}} \tag{4.5}$$

where X can be any metric, and H_{max} is the maximum forecast length, which in this work is 6 days. Once a relative metric is calculated, a general percentage improvement over the baseline model can easily be seen. A relative metric greater than 1 would indicate that the metric is worse than the persistence baseline and less than 1 would indicate improvement. For example, a relative MSE of 0.85 (85%), would indicate a 15% improvement over the persistence model MSE; and a relative MSE of 1.03 would indicate that a model is 3% worse than persistence.

To identify potential bias in predicted values, we evaluate the univariate error statistics and compare using the binned results method presented in [26]. By creating a set of bins of predicted values, the model uncertainty and biases at various solar activity levels can also be highlighted. It is necessary to create models that perform well during all parts of the solar cycle, and not just minimum or maximum. The binning method involves placing a prediction into a bin depending on

the first predicted value. The binning values (bounds) used in [26] are seen in Table 4.4.

Table 4.4: $F_{10.7}$ Binning ranges used by Licata et al. [26]

Solar Activity Level	**Lower Bound [SFU]**	**Upper Bound [SFU]**
Low	<75	75
Moderate	75	150
Elevated	150	190
High	190	>190

4.6.1 Uncertainty Quantification (UQ)

Deterministic approaches such as the operational SET method, lack the ability to provide an associated uncertainty with their predictions. By providing both a predicted value (ensemble combination) and an associated uncertainty, proper response can be made by agencies that rely on a predicted solar driver, such as $F_{10.7}$. In addition to performance error metrics, it is important to provide UQ for our probabilistic forecasting methods to better understand uncertainty estimates, which can be used evaluate the robustness and reliability of the probabilistic forecast, as can be seen in the previous work by [83]. This can be performed by calculating the confidence interval using the true data,

$$CI = \bar{x} \pm z * \frac{\sigma}{\sqrt{n}} \tag{4.6}$$

where $\bar{x}$ is the sample mean, z is the z-score of an assumed distribution, σ is the sample standard deviation, and n is the number of samples. An example of a confidence interval from associated probabilistic models can be seen in Figure 4.7.

A set of deterministic forecasts can be used together to form a probabilistic forecast, and evaluating statistics across the forecast distribution allows the uncertainty of forecast to be quantified. A neural network model ensemble approach can generate a combined (single point) forecast (red-line in Figure 4.7) as well as a distribution for each output. The CES metric, originally by [84], is modified by [8] for use in uncertainty quantification. CES quantifies the average deviation from perfect calibration in percentage, averaged across each output and is given as,

$$CES = \frac{100\%}{r * m} \sum_{i=1}^{r} \sum_{j=1}^{m} \mid p(\alpha_{i,j}) - p(\hat{a}_{i.j}) \mid \tag{4.7}$$

where r is the number of model outputs, m is the number of prediction intervals investigated, p is the expected cumulative probability, and $\hat{p}$ is the observed cumulative probability. By calculating

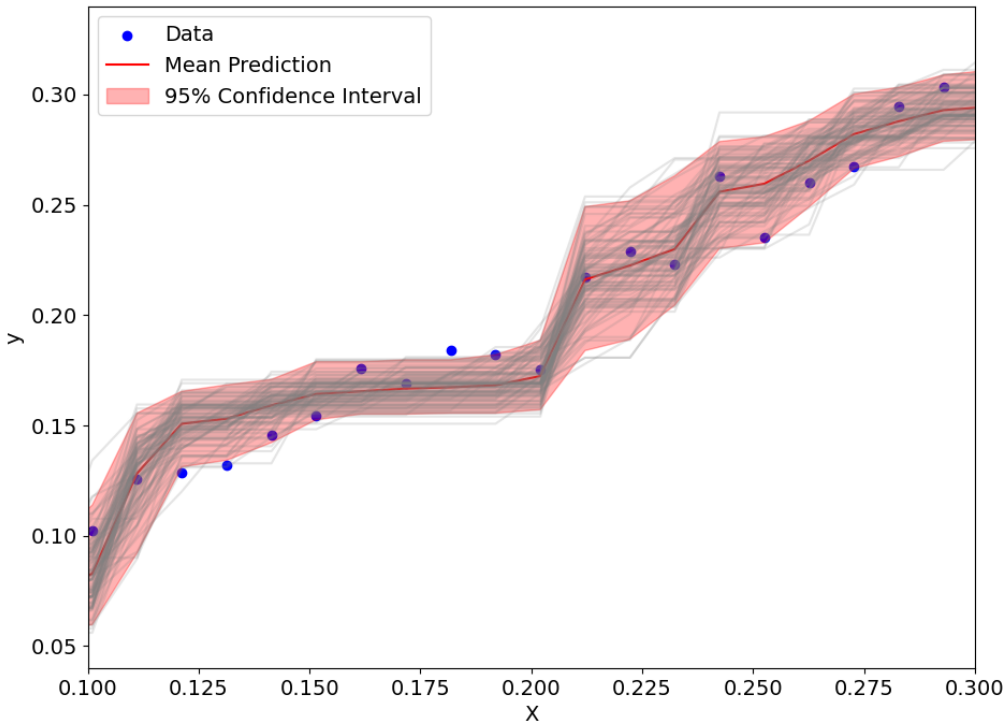


Figure 4.7: Toy Problem: Using 100 random forest regressors (grey lines) to construct an ensemble. In this case, the model ensemble outputs a distribution of predictions. Predictions can be averaged (red line) and statistics evaluated (pink CI).

the calibration error score (CES), a quantitative measure of a model's ability to provide reliable uncertainty estimates can be provided.

It is also desired to generate a qualitative measure of uncertainty, which is referred to as a *calibration curve.* A calibration curve is a plot that shows the expected and observed cumulative probability, plotting of p vs $\hat{p}$ from Equation 4.7. Calibration curves show how well calibrated the uncertainty estimates are at capturing the expected percentage of true samples, in the distribution. A model that is perfectly calibrated has a calibration curve with a slope of one. Models which under fit or over fit the uncertainty have calibration curves with slopes of less than one or greater than one respectively. In the event that a model is not calibrated well, steps such as applying a scaling factor can be taken.

A scaling factor can be applied to the uncertainty estimates, which is referred to as σ scaling. Sigma scaling, introduced by [85], uses the validation set to "check" the validity of uncertainty estimates. This check provides a scaling factor based on the results and adjusts the uncertainty estimates based on over or under prediction. For example, if a calibration curve shows a tendency to over predict on validation data, then a scaling factor can be generated to "correct" the uncertainty estimates. Applying σ scaling, better CES metrics and a generally more reliable uncertainty estimate may be found. The scaling factor is generated as follows,

$$\sigma_S = \sqrt{S} * \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \sigma_i^{-2} * (y - \hat{y})^2} * \sigma \tag{4.8}$$

where S is the scaling factor , N is the number of samples in the validation set, σ_i is the sample standard deviation at step i, σ_S is the scaled standard deviation and $(y - \hat{y})^2$ is the squared error of prediction. Sigma scaling is not guaranteed to improve results.

Another method for combining models, ensemble model output statistics (EMOS) was introduced by [86]. EMOS is a post-processing technique which addresses forecast bias, underdispersion, and spread-skill relationship. EMOS relies on linear regression, to yield a probabilistic forecast; formed by a Gaussian predictive probability density function (PDF). The general form of the Gaussian predictive distribution,

$$\mathcal{N}(a + b_1 X_1 + ... + b_m X_m, c + dS^2) \tag{4.9}$$

where a, b_i, c, and d are regression coefficients, X_i are individual model forecasts, and S^2 is the ensemble variance. EMOS uses either the continuous ranked probability score (CRPS) or

ignorance (IGN) scoring, to determine the linear regression coefficients. This distribution provides a probabilistic forecast which may outperform both the raw output and σ-scaling methods. EMOS techniques are investigated for their potential well calibrated probabilistic forecasts. With a thorough explanation for how performance is measured and how uncertainty is quantified, we are able to effectively discuss the results provided by ensemble methods for probabilistic solar driver forecasting.

Chapter 5

Results

To make meaningful contributions to the field of space weather forecasting, we must provide an approach that is both realistic and has been supported by necessary testing. In order to support the methods used, it is important to provide results which show a) new models outperform current operational methods and b) probabilistic forecasts can be relied upon. This chapter discusses the performance of UV-MLE and MV-MLE against the baseline persistence model, the linear operational method known as TS_FCAST, and the NBEATS neural network ensemble model. We begin with a discussion of results for univariate forecasting using the methods from Section 4.4, with a focus on results regarding the $F_{10.7}$ driver. After the comparisons between UV-MLE and current $F_{10.7}$ forecasting methods are made, we turn our attention to the results of MV-MLE, specifically comparing the multivariate approach to univariate methods and the operational linear model for $F_{10.7}$, $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$.

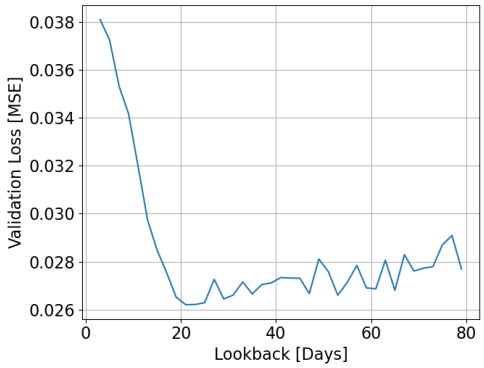
5.1 Univariate Forecasting of $F_{10.7}$ with UV-MLE

We choose to first examine the UV-MLE performance on $F_{10.7}$ because there exist many methods for $F_{10.7}$ forecasting (Section 2.4.2) which we seek to outperform. $F_{10.7}$ has been extensively studied and has a significant dataset to apply ML approaches. Additionally, there exists a univariate probabilistic model, N-BEATS, which we seek to outperform using the methods from this work. We begin our analysis with results related to input data manipulation.

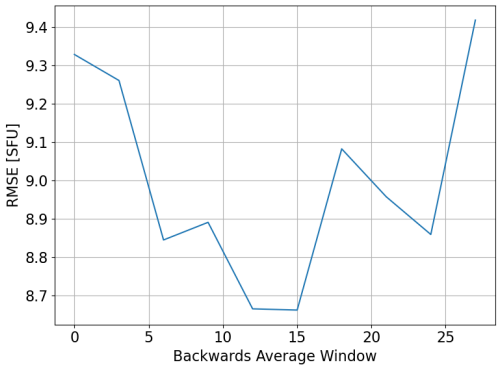
5.1.1 Input Sensitivity

In Figure 5.1a, a range of lookback values were considered over 3 solar rotations (81 days). Random architectures were trained and evaluated on a validation set for each lookback. This study resulted in a varied performance over different lookback windows and can be evaluated to identify important inputs for this work. Based on the results from this analysis, we select a range of lookbacks $L = [7, 10, 13, 16, 19, 22]$ days. We spread the lookbacks out by differences of 3 days, to prevent redundancy within models.

In Figure 5.1b, we clearly see an increase in performance by including back average values during tuning/training. Based on the performance metrics, we select the best value from this sweep to be used as an additional input to the neural network models. Based on the results of the backwards averaging study, we select a value for backwards average, B= 12 days.



(a) Averaged validation loss for a variety of architectures indicated a short term lookback between 2 and 3 weeks is better than very short or very long lookbacks.



(b) Training of models with different B values indicates that short term backwards averaging helps but longer term averaging performed the same or worse than B=0.

Figure 5.1: By manipulating input information, models with a wide range of performance can be produced.

Once we determined that varying model inputs can have an effect on performance, it is necessary to investigate the effects of including more models, so a set of models are created with the methods described in Section 4.4. With the created models, we can determine if combining models can offer an improvement over a single model approach (i.e. are ensemble approaches even useful?)

Using the optimal lookback and backwards average values, the performance of including additional models can be seen in Figure 5.2. This ensemble is created using MLP models with various lookbacks, random weighting, and architectures. The MAE metric is calculated based on the combined forecast, which is the average predicted value. In Figure 5.2, it can be seen that inclusion of

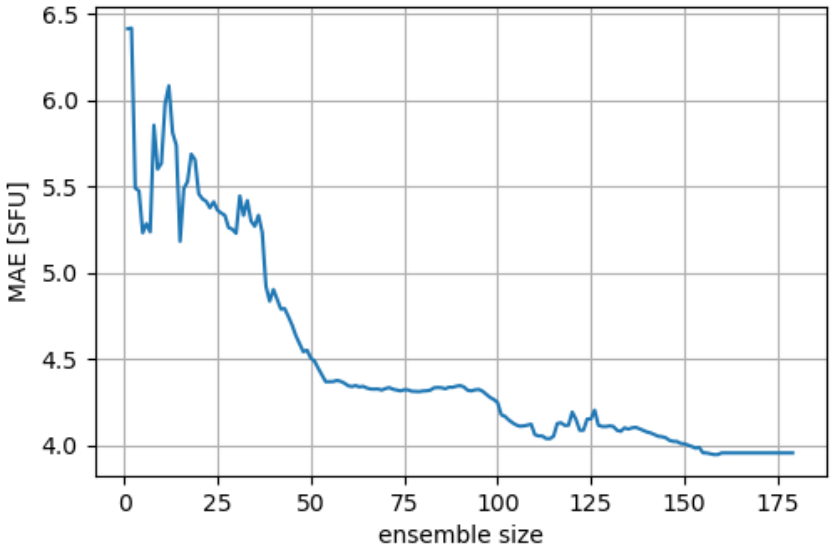


Figure 5.2: By incorporating more predictions, we are able to improve performance when compared to a single model.

more ensemble members, lowers the error substantially. The MAE decreases from 6.4 to 3.9 (approximately 48%) from 1 member to 180 members. The fluctuation in error, seen when including a small number of models, can be attributed to the stochastic training process and individual model performance. Model outputs are averaged for this error and a limited number of models contribute more to the average error, when using more models error is decreased on average. We also see a plateau and a lack of improvement at the end of the study, so 180 ensemble members are chosen as the total number of models to be used in this work. Since it is clear ensembles are useful for this work, it is necessary to determine the impact of changing the training loss used to create the ensemble members.

5.1.2 Diversity Through Loss Function

As a potential source of ensemble diversity, variation of the loss function is investigated. By evaluating models with different loss functions, we can identify if the introduced diversity provides enhanced probabilistic forecasts. We perform a sweep analysis by creating an ensemble composed of models with different loss functions. For this investigation, a model ensemble of size 100 was constructed using the 10 best architectures from KerasTuner, which were trained 10 times, with random weight initialization.

It is clear from Figure 5.3, seeing a clear minimum appear in the CES metric, that introducing diversity to an ensemble by augmenting the learning process indeed changes performance, especially

uncertainty estimates. Using an equal contribution from MSE and MAE models, we see an improvement in the calibration of the ensemble approach. To produce reliable probabilistic forecasts, we seek to minimize CES, and by using a neural network model ensemble with varied loss functions, we decrease the CES metric by about 1% in the training and validation sets, as well as close to 2% in the testing set. Performance metrics seen in Figure 5.3 indicates that diversity, by way of loss function variation, does not contribute much to the ensemble's performance metrics. Due to the improvement seen in CES and insignificant changes in performance error metrics; we opt for a neural network model ensemble which is constructed using an equal split of models trained with MSE and MAE loss functions.

5.1.3 Results of Different Univariate Models

To compare model performances directly we must ensure that metrics are reported for the same test set data. We use the original $F_{10.7}$ data sets presented in Figure 3.8 to make predictions using the N-BEATS ensemble, the SET model, and our novel UV-MLE created with a variety of model types. When directly compared over the short term, the proposed ensemble method outperforms persistence and the SET method in all cases, as seen in Figure 5.4, this indicates that ensemble methods are better suited for capturing non-linearity in the $F_{10.7}$ proxy. It can be concluded from Table 5.1, that the ensemble methods outperform persistence, the SET method, and N-BEATS for

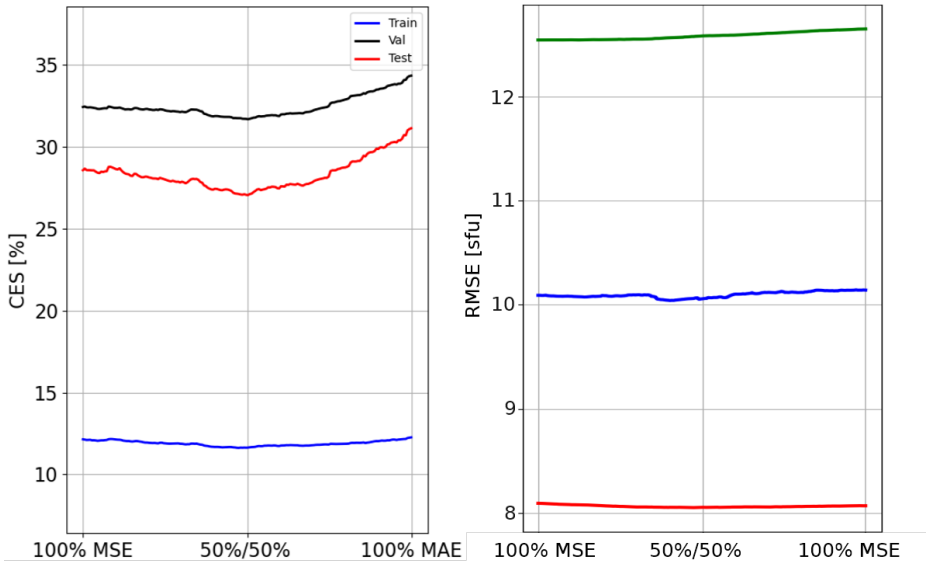


Figure 5.3: **Left:** A mixture of loss functions provide better calibrated uncertainty estimates. **Right:** The RMSE metric is not nearly as sensitive to loss function but may benefit from other sources of diversity.

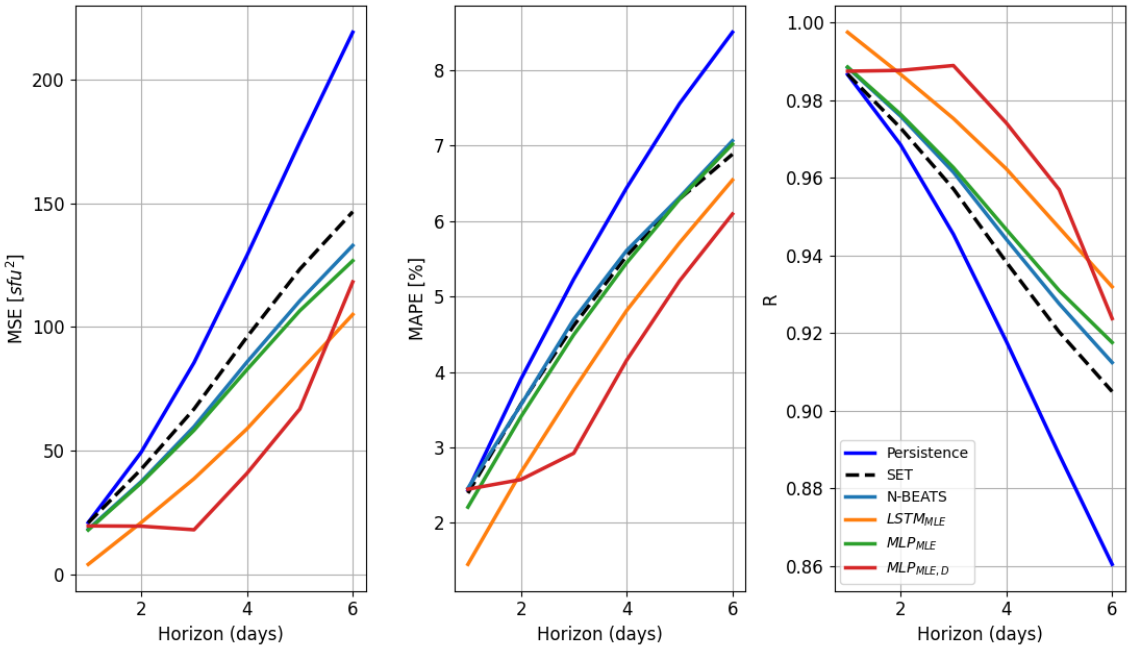


Figure 5.4: The subscripts MLE, D and MLE denote a dynamic prediction and multi-step prediction respectively. These metrics are non-relative and are not scaled to the baseline performance. Evolution of metrics over forecast horizon indicate short term prediction is improved by UV-MLE ensemble methods.

short term $F_{10.7}$ forecasting. It should be noted that dynamic prediction methods were found to improve performances in MLP models but hindered the predictions of the LSTM models. LSTMs did not seem to not benefit from the recursive prediction the same way MLPs did, especially in the test case. It is also important to note that various models perform differently across the various metrics and sets. This result indicates the importance of examining more than one metric at a time, providing a "fuller picture" of model performance. This may be attributed to instabilities in predicted values, where large error predictions continued to be fed into the model, magnifying small errors. It should also be noted that the validation set had a higher solar maximum than the test set, and may have contributed to a few metrics being worse on the validation set. An investigation and discussion into the statistical consistency of validation sets is performed in Section 4.1.

It was also found that mixing prediction type helped significantly, using a mixed prediction (average prediction of $\text{MLP}_{MLE,D}$ and LSTM_{MLE}), we see an approximate 14% MSE improvement over the single best non ensemble model on the validation set; this may be attributed to the greater number of models combined (360 models). These results indicate that when using an ensemble, we improve on single model approaches. The relative Pearson correlation coefficient across various approaches reinforces that deep-learning approaches all out perform the baseline model. Based on

Table 5.1: Relative metric comparison of Multi Lookback Ensemble (MLE) and Dynamic (D) methods with other forecasting methods. Metrics are scaled against the persistence baseline, and averaged over forecast horizons. Lower error metrics and higher correlation metrics are preferred, with a value of 1 exhibiting the same performance as persistence. The best performing values in each metric are highlighted in bold.

Model	Relative Metric		
Training Set Sample (1964-1974)	RMSE	MAPE	R (Corr. Coefficient)
SET	0.819	0.816	1.034
N-BEATS	0.799	0.922	1.011
Single MLP	0.747	0.910	1.031
MLP_{MLE}	0.800	0.878	1.038
Single LSTM	0.762	**0.769**	1.066
LSTM_{MLE}	**0.723**	0.877	**1.071**
MLP_{MLE} + LSTM_{MLE}	0.766	0.878	1.024
$\text{MLP}_{MLE,D}$	0.818	0.989	1.009
$\text{LSTM}_{MLE,D}$	0.898	0.975	1.008
$\text{MLP}_{MLE,D}$ + $\text{LSTM}_{MLE,D}$	0.814	0.881	1.067
$\text{MLP}_{MLE,D}$ + LSTM_{MLE}	0.731	0.838	1.065
MLP_{MLE} + $\text{LSTM}_{MLE,D}$	0.764	0.830	1.022
Validation Set (1994-2004)			
SET	0.854	0.850	1.020
N-BEATS	0.843	0.835	1.022
Single MLP	0.820	0.822	1.024
MLP_{MLE}	0.813	0.814	1.024
Single LSTM	0.827	0.832	1.023
LSTM_{MLE}	0.815	0.822	1.024
MLP_{MLE} + LSTM_{MLE}	0.813	0.816	1.024
$\text{MLP}_{MLE,D}$	0.921	0.967	1.034
$\text{LSTM}_{MLE,D}$	0.927	0.981	1.034
$\text{MLP}_{MLE,D}$ + $\text{LSTM}_{MLE,D}$	0.918	0.936	1.033
$\text{MLP}_{MLE,D}$ + LSTM_{MLE}	**0.692**	**0.703**	**1.042**
Test Set (2006-2020)			
SET	0.888	0.881	1.021
N-BEATS	0.839	0.893	1.027
Single MLP	0.840	0.875	1.028
MLP_{MLE}	0.826	0.857	1.029
Single LSTM	0.680	0.773	1.041
LSTM_{MLE}	**0.637**	0.722	1.043
MLP_{MLE} + LSTM_{MLE}	0.644	0.718	1.042
$\text{MLP}_{MLE,D}$	0.662	**0.713**	1.046
$\text{LSTM}_{MLE,D}$	1.070	1.194	1.027
$\text{MLP}_{MLE,D}$ + $\text{LSTM}_{MLE,D}$	0.807	0.904	1.046
$\text{MLP}_{MLE,D}$ + LSTM_{MLE}	0.713	0.803	**1.049**

the current results, the ensemble approach should be compared to the NOAA SWPC forecast, which is issued electronically to stakeholders.

5.1.4 Comparison with NOAA SWPC (2015-2019)

The NOAA SWPC *Weekly* publication is distributed every week, providing an $F_{10.7}$ forecast for a 27-day period. We choose a range of dates, for which *Weekly* forecast data is archived, which also align with our testing data set. The data selected for comparison also incorporates a mixture of high, moderate, and low activity levels over 5 years. We truncate *Weekly* publication forecasts from 27 days, to 6 days, to align with prediction horizons in this work.

In Figure 5.5, we see that the SWPC forecast has larger errors in both RMSE and MAE metrics

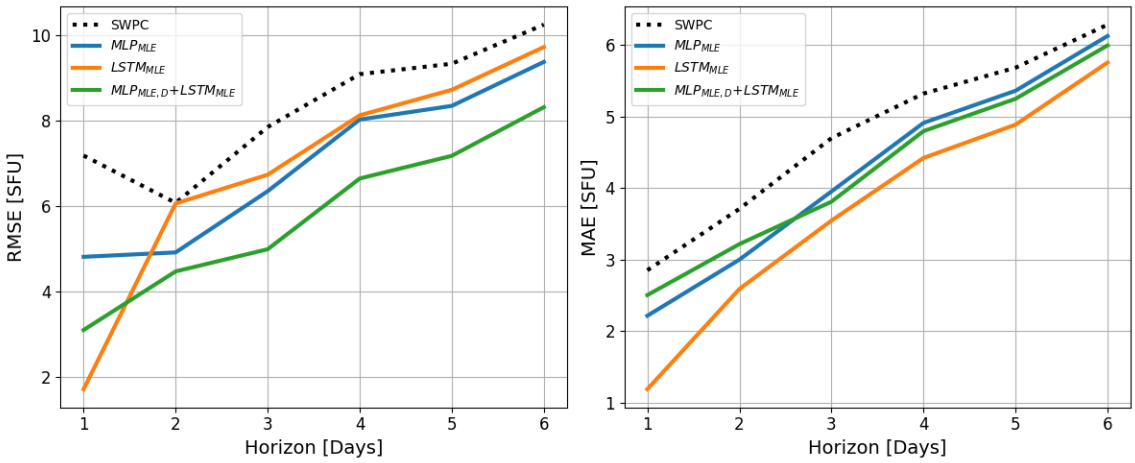


Figure 5.5: The non-relative RMSE and MAE plotted over forecast horizon indicate a substantial decrease in error when using neural network methods.

for all horizons than the neural network model ensembles. It is seen that a forecast horizon of one day is substantially improved by using the $LSTM_{MLE}$. The $LSTM_{MLE}$ provides a 58% and a 76% drop in RMSE and MAE over SWPC for a one day forecast horizon. As the horizon increases, the forecasting methods seem to approach similar errors. When average metrics are analyzed, as seen in Table 5.2, the ensemble approaches outperform the SWPC model.

Table 5.2: Comparison of metrics between the SWPC forecast and the neural network methods, averaged across the entire forecast horizon. Relative metric column indicates % change of metric compared to the SWPC method, negative (bold) indicates lower error than the NOAA SWPC method.

Model	RMSE [SFU]	Relative RMSE [%]	MAE [SFU]	Relative MAE [%]
SWPC	8.41	0	4.76	0
MLP_{MLE}	7.18	**-14.6**	4.26	**-10.4**
$LSTM_{MLE}$	7.31	**-13.0**	3.73	**-21.6**
$MLP_{MLE,D}$+$LSTM_{MLE}$	6.04	**-28.2**	4.22	**-10.4**

5.1.5 Solar Activity Level and Error Statistics

To identify uncertainty in prediction over forecast horizon we plot mean error (ME) vs the horizon; the ME indicate the average over or under predictions made by a model. This analysis uses the same dates for predictions as [26] which is October 1^{st} 2012 through December 31^{st} 2018 and uses the binning criteria in Table 4.4.

It can be seen in Figure 5.6, that predictions are less biased than persistence and the SET model at high solar activity levels. At low activity, the biases are worse than those for the more simple methods, indicating a tendency to over predict by 1.5 SFU on average by the 6th day. At elevated

and high activity levels, the dynamic prediction tended to under predict in some cases. We see that the ensemble that incorporates both multi step prediction and dynamic, made less biased predictions at both elevated and high activity levels but made more biased predictions at low activity levels. This indicates it may be useful to combine the predictions from the members of an ensemble in a way other than averaging. The bias at low activity levels may be due to the training scheme; during training, the MSE loss function was minimized, penalizing larger errors greater than smaller errors. With large errors, training would skew the model to make better predictions at higher activity levels, to minimize the loss function. The error statistics at the high activity level are only analyzed for a small number of predictions (about 20 forecasts) and may not be statistically significant. The low, moderate, and elevated activity levels have approximately 500, 1400, and 200 forecasts respectively.

Errors in prediction are suspected to be larger during periods of high solar activity. We break down the performance of the best ensemble approach and the SET method, along with the $F_{10.7}$ value from the maximum of solar cycle 24 to the minimum of solar cycle 25. The trends seen in Figure 5.7 highlight the high correlation between activity level and error. Our work is in agreement with the results seen in [26] and [12]; forecasting of $F_{10.7}$ is more challenging at higher solar activity

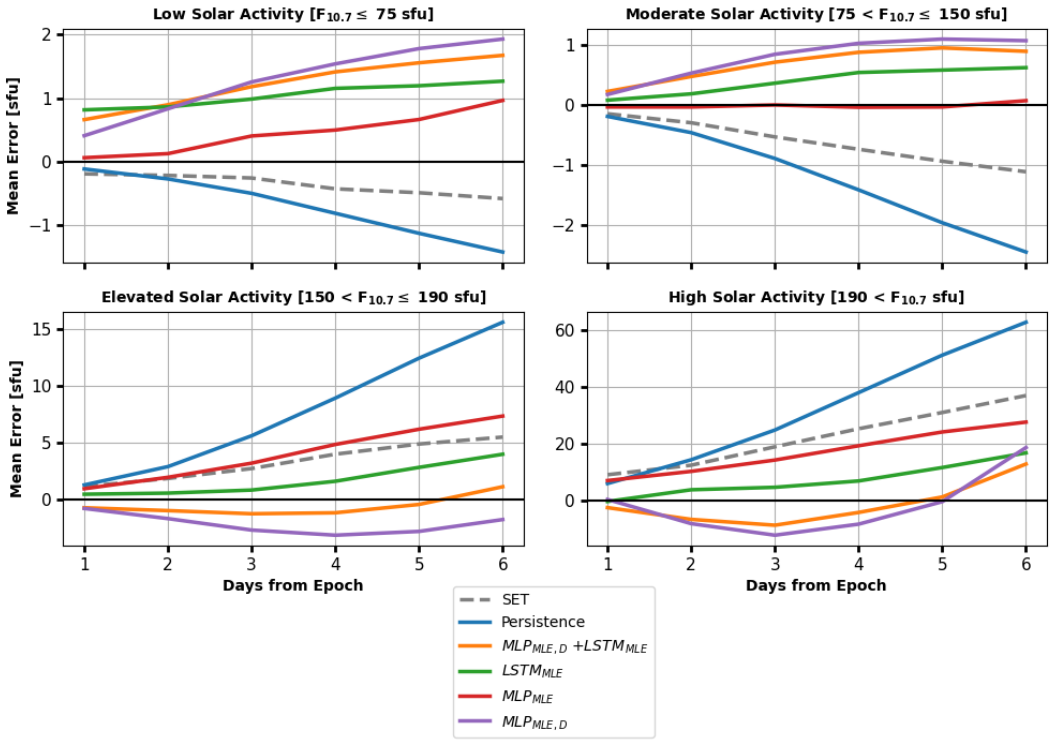


Figure 5.6: Bias of well performing models on test data indicates that ML methods provide much less biased predictions when solar activity is increased. The black line at zero indicates no bias.

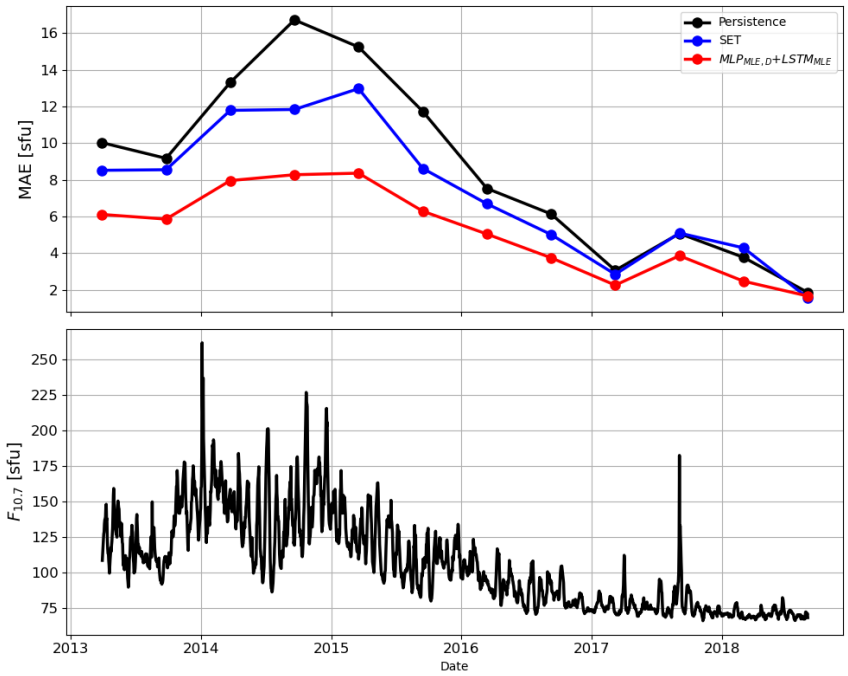


Figure 5.7: **Top:** 180 day averaged mean absolute error (MAE) for a forecast horizon of 6 days. **Bottom:** The $F_{10.7}$ driver changes with solar activity level over part of the test set, as solar cycle 24 turns into solar cycle 25.

levels. We have identified performance metrics in general, and at various solar activity levels, it is now crucial to compare the probabilistic forecast uncertainty estimates with the other probabilistic method.

5.1.6 Quantified Uncertainty

The UV-MLE ensemble approach would allow a user to sample from a probabilistic range of values, rather than use a single deterministic value, like the SET method. By outputting a range of $F_{10.7}$ values, a prediction contains both an average and associated uncertainty bounds, which creates a more robust and operationally useful forecast. By considering a large number of models and combining their output, the ensemble prediction should produce robust and reliable uncertainty estimates.

We clearly see in Figure 5.8, that models that have good metric scores are not necessarily the best when predicted uncertainties are evaluated. Based on work done, multi-step prediction models tend to be under confident in the predicted uncertainty bounds and dynamic prediction causes the prediction to be over confident. The use of σ-scaling (Section 4.6.1) is indicated by the dotted line in Figure 5.8. This scaling provides a better grouping of models along the 45^o line, indicating a

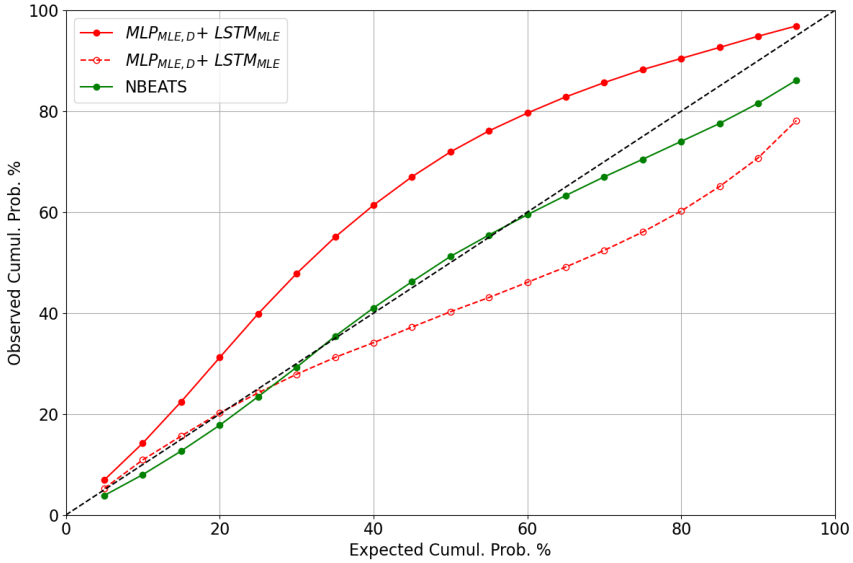


Figure 5.8: Test set calibration curves for ensemble predictions. The dashed red line indicates σ-scaling has been applied to the probabilistic forecast.

general improvement in uncertainty estimates after adjustments are made.

Table 5.3: Calibration Error Score (CES) of an ensemble with good performance metrics ($MLP_{MLE,D} + LSTM_{MLE}$) and another ensemble approach by (Stevenson et al., 2022). A bold values indicates the best metric.

Model	Test (Raw Output)	Scaled Test (σ-scaled)	Val. (Raw Output)	Scaled Val. (σ-scaled)	Train (Raw Output)	Scaled Train (σ-scaled)
UV-MLE	13.4	9.8	14.34	**5.22**	18.20	14.51
NBEATS	**2.87**	**2.38**	**12.52**	8.65	**14.79**	**11.8**

After σ-scaling was applied based on validation performance, we improve the uncertainty estimation significantly for our ensemble methods. The CES was improved by 4% and 9% on the test and validation sets respectively, while using the $MLP_{MLE,D} + LSTM_{MLE}$ ensemble. This indicates that σ-scaling can be useful for making more robust and reliable uncertainty estimates when validation data is available. It should also be noticed that uncertainty estimation on the test set is better than both validation and training sets, which may be attributed to the difference in solar activity levels (higher maximums), seen in Figure 3.8.

It is interesting that a model with worse performance metrics (error) has better uncertainty estimates, as seen in Table 5.3. One possibility for the NBEATS performance may be due to the diversity through varied loss function. The initial implementation of UV-MLE (Figure 5.8) did not consider a varied loss function, as there was not enough evidence from previous works to support its consideration. By varying loss function, NBEATS may be more suited for uncertainty estimation

than direct value prediction and indicates that without scaling, the NBEATS approach is well calibrated. Even with a single loss function in UV-MLE, once σ-scaling was applied, the approaches were similarly calibrated on the validation and training sets. Further investigations into improving the uncertainty estimates by improving ensemble diversity is discussed in the next section.

5.2 Multivariate Forecasting (MV-MLE)

Univariate methods showed substantial improvement over the operational SET method when error metrics were compared. We saw a similar uncertainty estimates between UV-MLE and the NBEATS method; it is now important to discuss the results for approaches of forecasting the other drivers.

5.2.1 Ensemble Member Combination Methods

Although a probabilistic forecast provides a distribution, it is critical to provide single point, or a combined forecast value. We believe it necessary to investigate methods for a combined forecast other than the historically used mean. Using both a stacked ensemble and by combining predictions using the mathematical median, we determine more sophisticated methods which outperform the previously used mean predicted value.

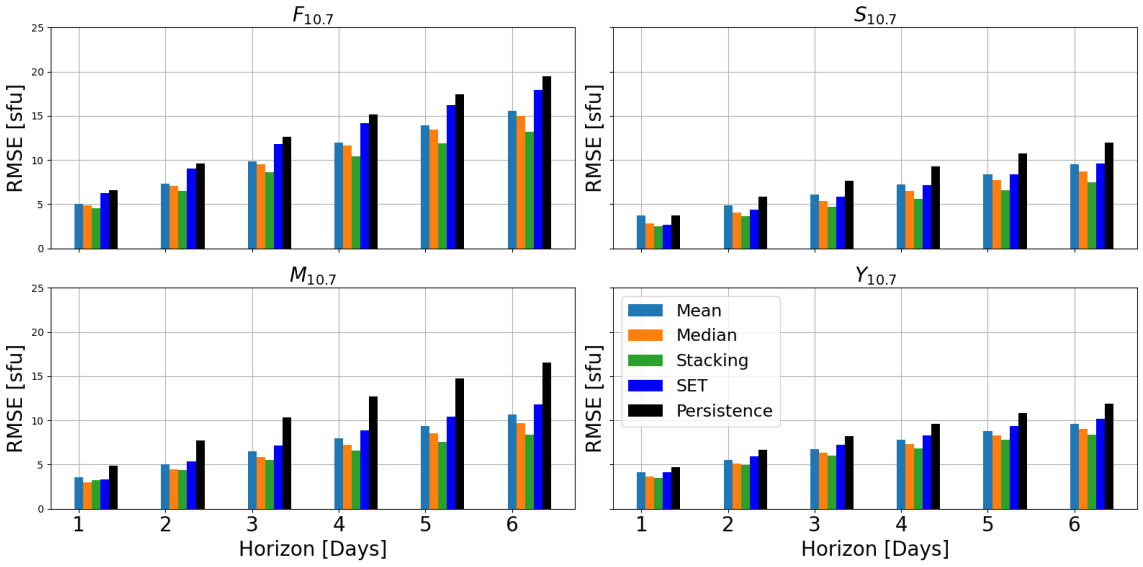


Figure 5.9: Ensemble combination methods using stacking or median (green & orange) show improved errors over all horizons when compared to persistence and the SET method.

In Figure 5.9 it is clear that combining an ensemble using the mean value leads to improved predictions across all forecasting horizons, outperforming both persistence and the linear SET al-

gorithm. We observe that the predicted mean performs effectively, but additional enhancements in performance metrics are achieved when employing methods like median prediction or stacking. We believe that by using the median of predicted values, we eliminate the outliers that may occur when considering the mean.

We perform a stacking approach using the validation data set, providing a weight associated with each model. Once the models have been combined using these weights, we see a dramatic increase in performance for nearly all drivers and horizons. We see that stacking increases performance at larger horizons more so than smaller horizons. This result may indicate that some models are better learners at larger horizons and have an associated larger weight. We see that predictions of $S_{10.7}$ using the mean of prediction provided similar errors to the SET method, it is not until stacking is performed do we see noticeable improvements.

When using median and stacking, we see a considerable improvement over the mean approach, seen in Table 5.4. When RMSE is averaged over the 6-day horizon, we see improvements in all drivers. Stacking outperforms median combination on the training, validation, and test sets. It should be noted that the significant improvement seen in the validation set is expected due to the procedure used to create model weights; stacking weights were created to fit the validation data and therefore are expected to perform well. Instead, we consider the training and test sets to be better measures of true stacking performance. Comparing stacking to mean combination, the smallest improvement observed is 0.57 SFU, while the most significant improvements reach up to 2.19 SFU.

Table 5.4: The RMSE metric was averaged over a 6-day horizon for each combination method. The difference between the mean prediction and both median and stacking approaches are reported. Negative values indicated an improvement over the mean combination method, with bold indicating favorable values.

Combination Method	**Difference vs. Mean (RMSE)**			
	$F_{10.7}$	$S_{10.7}$	$M_{10.7}$	$Y_{10.7}$
Training Set				
Median	-0.28	-0.80	-0.64	-0.40
Stacking	**-0.65**	**-1.33**	**-1.12**	**-0.57**
Validation Set				
Median	-0.29	-0.79	-0.63	-0.37
Stacking	**-2.19**	**-1.86**	**-1.87**	**-1.24**
Test Set				
Median	-0.36	-0.79	-0.70	-0.47
Stacking	**-1.40**	**-1.57**	**-1.23**	**-0.86**

Due to the decrease in error, we consider the stacking approach to be the most useful method for

combining our ensemble members. Since a validation or training set has already been "seen" by the models, we believe that the stacking approach can utilize such sets for another step in the machine learning process. In stacking, a linear regression model is used to "learn" the best combination method for the neural network models, which is referred to as a *meta learner*. We show that with stacking, we can effectively use a meta learner to enhance predictions. Based on the improvements in performance error metrics, we select stacking as the preferred method for combining models.

5.2.2 Comparison with Operational Models

In our analysis, we evaluate the performance of neural network models, which have undergone training using striped validation data. These models utilize both PCA and non-PCA input data, employ MSE and MAE training loss functions, and are combined through stacking. Given the statistical consistency between the split datasets, we consider the test set as the primary indicator for evaluating model performance. The testing set remains entirely hidden during training and stacking, making it an effective tool for measuring performance error metrics. Our goal is to identify the most effective forecasting methods among various ensemble approaches. We compare relative metrics for test set predictions made by various approaches, and the results are presented in Table 5.5.

Table 5.5: Relative metric comparison of the SET linear method and stacked ensemble approaches on the test set. Metrics are scaled against persistence and averaged over forecast horizons. Lower error metrics and higher correlation metrics are preferred, with a value of one exhibiting the same performance as persistence. The best performing values in each metric are highlighted in bold.

Driver	Relative Metric	SET	Transfer Learning	UV-MLE	MV-MLE	MV-MLE (PCA)
$F_{10.7}$	RMSE	0.927	0.799	0.911	**0.75**	0.773
	MAPE	0.939	0.823	0.904	**0.771**	0.805
	R	1.005	1.024	1.013	**1.029**	1.028
$S_{10.7}$	RMSE	0.854	0.735	0.738	0.731	**0.703**
	MAPE	0.835	0.758	0.755	0.803	**0.736**
	R	1.005	1.008	1.008	**1.01**	1.009
$M_{10.7}$	RMSE	0.761	0.646	0.751	0.623	**0.596**
	MAPE	0.771	0.687	0.764	0.658	**0.651**
	R	1.019	1.026	1.021	**1.029**	**1.029**
$Y_{10.7}$	RMSE	0.971	0.836	0.999	0.834	**0.832**
	MAPE	0.996	0.865	1.136	**0.863**	0.87
	R	1.003	1.009	1.002	**1.01**	1.009

We clearly see in Table 5.5, that ensemble approaches, specifically MV-MLE, outperform linear methods. The MV-MLE approach, with or without PCA inputs, provides significant improvement

over the SET method. When using MV-MLE with non-PCA inputs, we see an improvement of RMSE for $F_{10.7}$, $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ of 17.7%, 12.3%, 13.8%, 13.7% respectively, over the SET method. It is clear that the SET method outperforms persistence and ensemble methods further improve on the SET method.

Transfer learning methods also perform well, an improvement is seen over the SET method in all cases. This improvement indicates that architectures and models developed specifically for the $F_{10.7}$ driver can be applied to the other drivers (with adequate training and fine tuning of weights.) Interestingly, we see a dramatic difference between the transfer learning and UV-MLE methods. We believe that the performance difference between these methods can be attributed to the amount of data available for training. The models developed originally for univariate forecasting of $F_{10.7}$ had substantially more data to train from before being applied to the new drivers, while the UV-MLE models were limited to a much smaller historic dataset. It should be noted that comparison with univariate methods also provide an indirect comparison to SWPC methods for $F_{10.7}$.

The ensemble requiring PCA rotated inputs seems to outperform the standard MV-MLE ensemble on both $S_{10.7}$ and $M_{10.7}$, while the standard MV-MLE performs better on $F_{10.7}$. The methods perform similarly on $Y_{10.7}$, with a difference of only 0.2% RMSE and 0.7% MAPE. We show an additional comparison between the best performing, MV-MLE stacked ensemble approaches, seen in Table 5.6. Neither method stands out, when considering relative error metrics alone. We cannot definitively say whether PCA or non-PCA ensemble is preferred for probabilistic forecasting. We must instead look to uncertainty quantification to determine if one method is the better.

5.2.3 Quantified Uncertainties

The probabilistic forecasts made by MV-MLE and MV-MLE (PCA) are evaluated on the three data sets, seen in Table 5.7. We see that the CES varies across drivers; the calibration error scores associated with $F_{10.7}$ and $Y_{10.7}$ are smaller, while $S_{10.7}$ and $M_{10.7}$ have slightly larger CES values. In general, the raw outputs from both non-PCA and PCA inputs produce reasonable CES values. Regarding the effectiveness of σ-scaling, we notice that cases where CES is relatively large to begin with are improved more by scaling approaches. This can be seen most prominently in the $S_{10.7}$ driver, indicating that $S_{10.7}$ may benefit more from σ-scaling than other drivers. When probabilistic models are well calibrated to begin with, σ-scaling seems detrimental and leads to worse CES values.

We found that the EMOS method for probabilistic forecasting yielded unfavorable CES metrics for the multivariate ensemble methods. Due to the already reasonable calibration of the MV-MLE

Table 5.6: Relative metric comparison of the SET linear method and MV-MLE approaches on the training and validation sets. Metrics are scaled against the persistence baseline, and averaged over forecast horizons. Lower error metrics and higher correlation metrics are preferred, with a value of one exhibiting the same performance as persistence. The best metric values are highlighted in bold.

Driver	Relative Metric	SET	MV-MLE	MV-MLE (PCA)
Training Set				
$F_{10.7}$	RMSE	0.964	**0.632**	0.642
	MAPE	0.975	0.685	**0.677**
	R	1.008	1.043	**1.044**
$S_{10.7}$	RMSE	0.79	0.621	**0.585**
	MAPE	0.806	0.705	**0.633**
	R	1.007	1.012	1.011
$M_{10.7}$	RMSE	0.701	0.53	**0.504**
	MAPE	0.725	0.569	**0.555**
	R	1.024	**1.035**	1.034
$Y_{10.7}$	RMSE	0.983	**0.723**	0.73
	MAPE	0.985	**0.762**	0.771
	R	1.003	**1.013**	**1.1013**
Validation Set				
$F_{10.7}$	RMSE	0.96	**0.730**	0.759
	MAPE	0.982	**0.767**	0.794
	R	1.01	1.03	**1.028**
$S_{10.7}$	RMSE	0.811	0.745	**0.715**
	MAPE	0.783	0.814	**0.779**
	R	1.006	**1.009**	1.008
$M_{10.7}$	RMSE	0.704	0.64	**0.609**
	MAPE	0.715	0.681	**0.652**
	R	1.023	**1.029**	**1.029**
$Y_{10.7}$	RMSE	0.964	**0.835**	0.851
	MAPE	0.966	**0.863**	0.887
	R	1.003	**1.009**	1.008

and MV-MLE (PCA) ensembles, the application of EMOS did not help. EMOS generally worsened CES metrics, ranging from 6-18%. We believe that potentially, the number of coefficients necessary for EMOS may have caused the poor performance. To apply EMOS, regression for every model and output is needed (180 models and 24 outputs). This high number of terms may have caused typical linear regression to fail, as EMOS has been typically used for a small number of models with single outputs.

We choose to not apply σ-scaling or EMOS methods to the probabilistic MV-MLE forecasts. The application of σ-scaling helped marginally for some drivers but significantly worsened CES values for other drivers, and EMOS provided poor CES values overall. Direct use of the ensemble member outputs is a more intuitive method, and provides reasonable calibration. Additionally, no uncertainty

Table 5.7: Calibration error score (CES) for ensemble methods when evaluated on all datasets. Lower values are better and bold terms indicate the best method for a given driver.

Train Set Driver	**MV-MLE** (Raw Output)	**MV-MLE (PCA)** (Raw Output)	**MV-MLE** (σ-scaled)	**MV-MLE (PCA)** (σ-scaled)
$F_{10.7}$	**1.92**	3.00	8.76	8.39
$S_{10.7}$	9.94	11.26	9.22	**8.15**
$M_{10.7}$	8.55	**5.59**	7.02	8.17
$Y_{10.7}$	3.93	**2.34**	3.89	5.13
Validation Set Driver	**MV-MLE** (Raw Output)	**MV-MLE (PCA)** (Raw Output)	**MV-MLE** (σ-scaled)	**MV-MLE (PCA)** (σ-scaled)
$F_{10.7}$	1.9	**1.87**	8.58	8.61
$S_{10.7}$	9.21	10.67	8.68	**8.41**
$M_{10.7}$	7.53	**5.50**	6.73	7.78
$Y_{10.7}$	3.70	**2.07**	3.64	4.78
Test Set Driver	**MV-MLE** (Raw Output)	**MV-MLE (PCA)** (Raw Output)	**MV-MLE** (σ-scaled)	**MV-MLE (PCA)** (σ-scaled)
$F_{10.7}$	3.08	**2.62**	8.48	8.57
$S_{10.7}$	8.64	10.63	8.36	**8.62**
$M_{10.7}$	7.74	**5.63**	6.66	8.33
$Y_{10.7}$	**3.18**	6.27	4.04	6.54

scaling or regression is necessary after the prediction step. With no need for extra processing, using the direct predictions (raw output) directly is less computationally expensive and quicker.

We choose the test set for evaluation since it has been unseen during training and is statistically consistent with both the training and validation sets. The direct model outputs led to the calibration curves seen in 5.10. The MV-MLE (green) and UV-MLE (orange) models follow the same trends; methods are well calibrated for smaller confidence intervals, with a tendency to over predict uncertainty when the confidence interval grows for $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$. The $F_{10.7}$ driver is very well calibrated, with only a minor tendency to under predict uncertainty at very large confidence intervals. Neither MV-MLE or MV-MLE (PCA) stand out when examining the calibration curves. When averaged across all four drivers; the MV-MLE (PCA) ensemble yields an improvement in CES of only 0.16%. Based on the marginal differences, both methods can be considered useful, with a slight edge to MV-MLE (PCA) for uncertainty estimates.

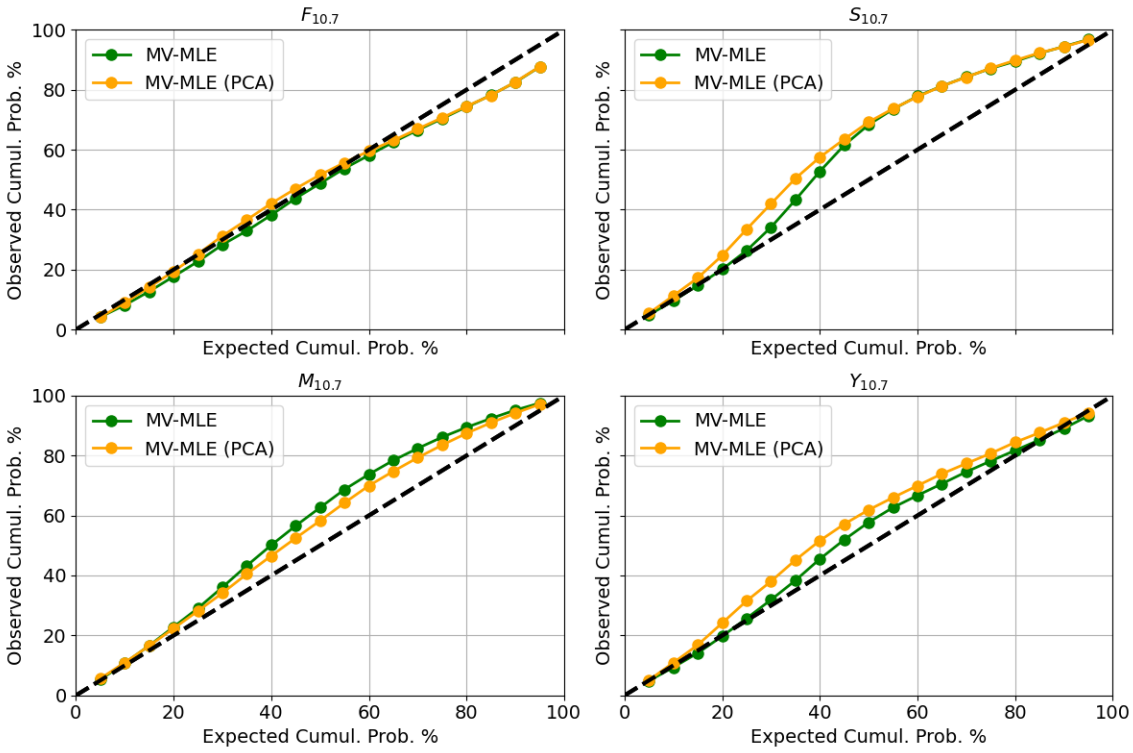


Figure 5.10: Evaluation of the multivariate ensemble methods on the test set. Curves above or below the 45^o line indicate over predicted uncertainty and under predicted uncertainty respectively. Curves closer to the 45^o line are desired.

Chapter 6

Summary and Conclusions

Ambitious plans by the commercial space industry are introducing large satellite constellations into LEO. These constellations are a considerable financial investment; the design, building, and deployment cost of the SpaceX constellation was estimated to be at least $10 Billion in 2018. The proliferation of LEO has led to a recent focus on space traffic management, which is necessary to protect these costly investments. Short-term forecasting in the LEO region is crucial for preventing collisions between critical assets and for averting collisions with space debris. Collisions in the space environment can be catastrophic. Approximately 2,300 observable debris objects were created when the Iridium and Cosmos satellites collided in 2009, 65% of which remained in LEO for seven years [87]. Small space debris is difficult to track and poses serious danger due to its high relative velocity and dangerous elliptical orbits, which may introduce debris into other orbital regimes.

We currently do not have the capability to predict thermosphere density perfectly for a set of space weather conditions. Satellite drag modeling relies on predicted density, which itself relies on predicted solar drivers. The presence of uncertainty from predicted drivers can result in uncertainty in forecasted orbital states, which, in turn, undermines confidence in making decisions regarding potential collisions. The space traffic management community needs to have access to robust and reliable uncertainty estimates for a satellite orbital state, to determine if a costly maneuver is necessary.

This thesis focused on the novel development of short-term probabilistic solar driver models, which can be used in the overall drag modeling framework to provide operators with better orbital state uncertainty estimates. **The development of UV-MLE and MV-MLE marks the first time probabilistic solar driver models were developed for $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ with**

demonstrated uncertainty estimation capability and is a major contribution of this work.

An investigation for probabilistic forecasting of the $F_{10.7}$ driver using neural network ensembles provided a model which outperformed the operational linear method and provided well calibrated uncertainty estimates. When ensemble methods were first explored, it became evident that an ensemble comprised of 180 models outperformed any single model, showing a substantial 48% reduction in MAE. We named this approach univariate multi-lookback ensemble (UV-MLE), and considered it a critical first step for probabilistic solar driver forecasting.

UV-MLE provided significant improvements over the currently used forecasting method for the operational HASDM. We initially investigated UV-MLE constructed with a variety of model types such as MLP and LSTM. Using similar training, validation, and testing sets as prior works for $F_{10.7}$, improvements were seen in the relative RMSE over the SET method of 9.6%, 3.9%, and 25% respectively. UV-MLE also improved upon the $F_{10.7}$ forecast distributed by SWPC and was less biased than persistence or the SET method at high solar activity levels. The high skill on the test set is thought to be attributed to the holdout scheme used to split the data; solar activity was lower on the test set and led to better performance by UV-MLE.

UV-MLE is also able to provide probabilistic forecasts, which are unavailable with the currently used SET method. The uncertainty estimates provided by UV-MLE were able to achieve similar calibration as the other available probabilistic method, while improving on performance metrics. We considered these initial results a success, but felt as though the uncertainty estimates could be better calibrated if more sources of ensemble diversity were considered.

With the capability of the initial implementation of UV-MLE seen on $F_{10.7}$, we aimed to provide similar forecasts for the other solar drivers used by JB2008, $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$. No probabilistic forecasting method for these drivers existed, so it was critical to provide such a model for use in the STM community. However, the same methods used for UV-MLE struggled with the small amount of historical data, and there was the possibility that considering more than one driver at once would be beneficial.

A novel probabilistic forecasting method was implemented for all four solar drivers simultaneously. This method is referred to as multivariate multi-lookback ensemble (MV-MLE), and is a neural network ensemble created with stacked LSTM models. The MV-MLE considered several sources of ensemble diversity, such as varied loss function, random weight initialization, varied lookback, and tuned architectures. The MV-MLE approach provided improvements in relative RMSE for $F_{10.7}$, $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ by 17.7%, 12.3%, 13.8%, 13.7% respectively, when compared to the

currently used method. Additionally, MV-MLE provides the first method for probabilistic forecasting of the $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ solar drivers. It was seen that MV-MLE provided well calibrated uncertainty estimates for all solar drivers, resulting in an average CES of approximately 5.8%.

It was also necessary to consider the best way to combine individual models. We discovered that employing a linear regression technique, known as stacking, to derive model weights based on the validation set resulted in combined model forecasts that were significantly superior to the conventional mean approach. With the traditional mean, $F_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ ensemble predictions outperformed SET and persistence, but $S_{10.7}$ predictions did not. A combined forecast of $S_{10.7}$ outperformed both persistence and the SET method when a median or stacking combination approach was used. Stacking outperformed all other combination methods and resulted in an approximate 1.3 SFU improvement in RMSE, when compared to the mean. Methods used for uncertainty estimates, such as σ-scaling, showed good performance only for poorly calibrated models. Models which were well calibrated initially did not benefit much, and raw output values were preferred.

The second contribution is a novel striped sampling approach for machine learning methods. Due to the limited historical data available for the $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ solar drivers, we saw poor performance when applying holdout methods for splitting data into training, validation, and testing sets. We determined that holdout becomes quite limiting for the new solar drivers, as there are not enough solar cycles in their history to properly train models using holdout. The statistics of the training, validation, and test sets were analyzed. We determined that applying traditional holdout led to data sets which had drastically different solar activity levels. Training of neural networks using data with inconsistent statistics created models with bias and led to poor test set performance. A novel method to create data sets with consistent statistics was necessary.

Striped sampling was introduced in this work to create more statistically consistent data sets than the traditional holdout method. The data was broken down into week long segments, which were sequentially combined to create the training, validation, and test sets. The sequential nature of this sampling method allows for LSTM models to be effective, as ordered data is necessary to leverage the LSTM architecture. In addition, one week sampling allowed for the three data sets to be spread out more evenly across the historical data, capturing similar solar activity levels and could address bias.

The third and final contribution of this thesis is input data manipulation for machine learning methods. It can be challenging to model physical systems using only historic values. Initial auto regressive models which only used daily driver values were promising, but did not show significant improvement over the operational method. We built on the work done in [12] by creating

neural networks which used inputs with different lookback periods, leading to improved ensemble performance and ensemble diversity. Models trained with only historic values were sensitive to short-term variations and had difficulty capturing longer term trends. If a storm appeared in the lookback period, models would be thrown off and would predict associated unrealistic values. To address this, we investigated the use of a backwards average input term, and determined that including a fourteen day backwards average input term improved the RMSE by approximately 7%, when compared to a model with no backwards average input.

Prior models for $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ were auto-regressive, and did not consider the correlation between drivers. Previous work in $F_{10.7}$ forecasting considered other terms as model inputs, but no such methods existed for the other solar drivers. This work considered the simultaneous forecasting of all solar drivers, and found that multivariate methods outperformed univariate methods. It is clear that by including multiple solar drivers as inputs, models are able to learn more effectively, most likely identifying connections between drivers which may not be apparent.

Initial work for forecasting the highly correlated drivers led us to consider that high correlation forecasting may be redundant. To investigate the connection between driver correlation and modeling, we considered the use of PCA rotation of the MV-MLE input data. Although PCA rotation effectively de-correlated the inputs, significant improvements were not seen, performance metrics of MV-MLE and MV-MLE(PCA) were similar. The MV-MLE(PCA) showed minor improvement when uncertainty estimates were investigated, providing a 0.16% improvement in CES when averaged across all solar drivers. Calibration curves were created to compare MV-MLE and MV-MLE(PCA), which showed that the PCA approach helped $F_{10.7}$ and $M_{10.7}$, but were not beneficial for $S_{10.7}$, and $Y_{10.7}$.

6.1 Future Work and Recommendations

This work has shed light on the importance of probabilistic modeling of solar drivers and the challenges associated with density modeling and uncertainties in conjunction assessment. However, considerable work is still necessary to address the challenges associated with the further use of the space environment. The next steps and recommendations for the continuation of this work are as follows:

Probabilistic Driver-Density Coupling: There has been successful coupling between probabilistic density models such as HASDM-ML, and orbital state uncertainty estimates. Building off the probabilistic driver models presented in this work, there should be a future focus into coupling driver

uncertainty with probabilistic density models and orbit propagation. This would require development of a framework which couples driver uncertainty and density model uncertainty with orbit propagation. Typical Monte Carlo approaches for driver uncertainty in orbit prediction are inefficient, so methods need to be explored to more efficiently sample drivers from probabilistic forecasts. If accomplished, density models would be robust and reliable, considering the two major sources of uncertainty simultaneously, which would be critical for risk assessment in the STM community.

Neural Networks and Ensemble Approaches: To improve forecasting of drivers using only historical values and neural network ensembles, investigations into more advanced ensemble approaches and NNs could be beneficial. Further investigation into state-of-the-art transformer models may prove beneficial, as new methods are consistently being introduced and could allow for better performing probabilistic forecasting. Creating NN ensembles with a method such as boosting [88] could also help, developing ensemble members that are skilled in different areas. Boosting is an ensemble generation scheme which sequentially creates simple NN model. In addition, the work by [20] discussed training of ensemble members simultaneously (evolutionary ensembles). These methods may allow for more diverse ensembles since these schemes enforce negative error correlation during training, considered a *divide and conquer* approach. If accomplished, probabilistic forecasting using ensembles will be more efficient, needing less overall models to provide the same or better performance. Greater ensemble diversity may also help, as models will be skilled in different areas and could provide better calibrated uncertainty estimates. In addition Bayesian Neural Networks (BNNs) can be used to provide direct prediction of statistics without the need for an ensemble approach but may prove challenging to scale due to their complex nature and computational expense

Data Splitting: To further enhance the forecasting of all four drivers, it may be necessary to consider adjustments in sampling schemes that promote greater consistency in statistics across the different sets. It may be necessary to reduce the striped sampling segments to a resolution of 1-day; using a sequence of 6 days, 2 days, and 2 days, in the training, validation, and test sets respectively. A finer time resolution could allow the sets to have even better statistical consistency. It is important to determine at what point the striped sampling resolution impacts LSTM results, however it should be noted that a key component of a LSTM model, is sequential inputs, leveraging the model's ability to "remember." By changing the striped sampling resolution to one day, the LSTM's internal states must be reset every 6 days for training, and every 2 days for validation and testing. Resetting these internal states "clears" the LSTM memory and may lead to worse performance. If better sampling methods are created, training of probabilistic solar driver models may be easier and lead to improved machine learned models.

Additional Solar Driver Data and New Inputs: The available data for $S_{10.7}$, $M_{10.7}$, and $Y_{10.7}$ is quite limited. Daily observations have only been archived since January 1, 1997, leading to a relatively small overall data set for ML applications. If we continue using daily values, it will take a considerable amount of time to build up enough data to apply more traditional ML validation schemes. Machine learning approaches rely on large amounts of data to make effective generalizations. It is important to build a data set that can easily be used by new and existing ML methods. We believe the STM community should seek to provide solar driver observations at a cadence the same as the geomagnetic indices used by JB2008, which are provided every three hours. By providing solar driver data at the same cadence as the geomagnetic indices, we will generate data eight times quicker than before. The data presented in this work for all drivers has observations from 1997-2023, resulting in approximately 9,500 daily values. If solar driver observations were done with a 3-hour cadence, the same size data set could be archived in a little over three years. Additionally, finer time-resolution data may help provide meaningful solar driver patterns which are not apparent when using daily values.

There is only so much information that can be gathered from previous values alone, even using neural networks. The addition of solar disk images, suggested by [34] may be critical in improving forecasts, as long as the time lag between visual solar activity and thermosphere response is considered. In future works, it may be critical to evaluate also the "time-lag" of predicted values. If a solar storm were to occur and a prediction of the driver is made only after the event, then the prediction is meaningless for risk assessment in the STM community. We recommend that future forecasting evaluations consider in addition to metrics, quantifying the delayed response of a model. Similarity between signals at different times can be accomplished with an algorithm such as Dynamic Time Warping (DTW) and may be useful in quantifying temporal delay. It is also noted, when using AR models, time lag response is typical and may not be completely avoided. To address time-lag issues, it may be critical to consider models which incorporate physical processes and consider solar regions that are not yet Earth facing. A comparison of time-lag between physics based models such as the proxy work done by [35] and auto-regressive methods should be performed. It is important to continue planning space weather missions which are capable of providing important solar images and scientific observations, capturing crucial data for new model development.

Bibliography

[1] NASA ODPO, "LEGEND : 3D/OD Evolutionary Model," *NASA ODPO*, 2019.

[2] McDowell, J. C., "The Low Earth Orbit Satellite Population and Impacts of the SpaceX Starlink Constellation," *The Astrophysical Journal Letters*, Vol. 892, No. 2, apr 2020, pp. L36.

[3] Muelhaupt, T. J., Sorge, M. E., Morin, J., and Wilson, R. S., "Space traffic management in the new space era," *Journal of Space Safety Engineering*, Vol. 6, No. 2, jun 2019, pp. 80–87.

[4] Doornbos, E., "Producing Density and Crosswind Data from Satellite Dynamics Observations," *Thermospheric Density and Wind Determination from Satellite Dynamics*, Springer Berlin Heidelberg, 2012, pp. 91–126.

[5] Bowman, B., Tobiska, W. K., Marcos, F., Huang, C., Lin, C., and Burke, W., "A New Empirical Thermospheric Density Model JB2008 Using New Solar and Geomagnetic Indices," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, American Institute of Aeronautics and Astronautics, jun 2008, pp. –.

[6] Licata, R., Mehta, P., and Tobiska, W. K., "Impact of driver and model uncertainty on drag and orbit prediction," *Proceedings of the 31st AAS/AIAA space flight mechanics meeting*, 2021.

[7] Licata, R. J., *Probabilistic Space Weather Modeling and Forecasting for the Challenge of Orbital Drag in Space Traffic Management*, Ph.D. thesis, West Virginia University, 2022.

[8] Licata, R. J., Mehta, P. M., Tobiska, W. K., and Huzurbazar, S., "Machine-Learned HASDM Thermospheric Mass Density Model With Uncertainty Quantification," *Space Weather*, Vol. 20, No. 4, apr 2022.

[9] Licata, R. J., Mehta, P. M., Weimer, D. R., Tobiska, W. K., and Yoshii, J., "MSIS-UQ: Calibrated and Enhanced NRLMSIS 2.0 Model With Uncertainty Quantification," *Space Weather*, Vol. 20, No. 11, nov 2022.

[10] Licata, R. J. and Mehta, P. M., "Reduced Order Probabilistic Emulation for Physics-Based Thermosphere Models," *Space Weather*, Vol. 21, No. 5, may 2023.

[11] Paul, S. N., Licata, R. J., and Mehta, P. M., "Advanced ensemble modeling method for space object state prediction accounting for uncertainty in atmospheric density," *Advances in Space Research*, Vol. 71, No. 6, mar 2023, pp. 2535–2549.

[12] Stevenson, E., Rodriguez-Fernandez, V., Minisci, E., and Camacho, D., "A deep learning approach to solar radio flux forecasting," *Acta Astronautica*, Vol. 193, 2022, pp. 595–606.

[13] Tobias, S. M., "The solar dynamo," *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, Vol. 360, No. 1801, oct 2002, pp. 2741–2756.

[14] Emmert, J., "Thermospheric mass density: A review," *Advances in Space Research*, Vol. 56, No. 5, sep 2015, pp. 773–824.

[15] Ridley, A., Deng, Y., and Tóth, G., "The global ionosphere–thermosphere model," *Journal of Atmospheric and Solar-Terrestrial Physics*, Vol. 68, No. 8, 2006, pp. 839–864.

[16] Picone, J. M., Hedin, A. E., Drob, D. P., and Aikin, A. C., "NRLMSISE-00 empirical model of the atmosphere: Statistical comparisons and scientific issues," *Journal of Geophysical Research: Space Physics*, Vol. 107, No. A12, dec 2002, pp. SIA 15–1–SIA 15–16.

[17] Bruinsma, S., "The DTM-2013 thermosphere model," *Journal of Space Weather and Space Climate*, Vol. 5, 2015, pp. A1.

[18] Storz, M., Bowman, B., and Branson, J., "High Accuracy Satellite Drag Model (HASDM)," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, American Institute of Aeronautics and Astronautics, jun 2002.

[19] Mehta, P. M., Walker, A. C., Sutton, E. K., and Godinez, H. C., "New density estimates derived using accelerometers on board the CHAMP and GRACE satellites," *Space Weather*, Vol. 15, No. 4, apr 2017, pp. 558–576.

[20] Liu, Y., Yao, X., and Higuchi, T., "Evolutionary ensembles with negative correlation learning," *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 4, 2000, pp. 380–387.

[21] Qian Liying, e. a., "The NCAR TIE-GCM: A community model of the coupled thermosphere/ionosphere system," *Geophysical Monograph Series 201*, 2013.

[22] Marcos F., e. a., "Precision Low Earth Orbit Determination using Atmospheric Density Calibration," *Journal of the Astrnautical Sciences*, 1998.

[23] A.I. Nazarenko, P.J. Cefola, V. Y., "Estimating atmospheric density variations toimprove LEO orbit prediction accuracy," *AIAA/AAS Space Flight Mechanics Meeting.AAS 98-190*, 1998.

[24] Tapping, K. F., "The 10.7 cm solar radio flux (F10.7)," *Space Weather*, Vol. 11, No. 7, 2013, pp. 394–406.

[25] Tobiska, K. W., Bowman, B., and Bouwer, D. S., "Solar and Geomagnetic Indices for the JB2008 Thermosphere Density Model," *Space Environment Technologies Publication*, 2009.

[26] Licata, R. J., Tobiska, W. K., and Mehta, P. M., "Benchmarking Forecasting Models for Space Weather Drivers," *Space Weather*, Vol. 18, No. 10, 2020, pp. e2020SW002496, e2020SW002496 10.1029/2020SW002496.

[27] Klotz, M., "Dominion Radio Astrophysical Observatory (DRAO)," Published in *flickr*, 2011.

[28] Warren, H. P., Emmert, J. T., and Crump, N. A., "Linear forecasting of the F10.7 proxy for solar activity," *Space Weather*, Vol. 15, No. 8, 2017, pp. 1039–1051.

[29] Huang, C., Liu, D.-D., and Wang, J.-S., "Forecast daily indices of solar activity, F10.7, using support vector regression method," *Research in Astronomy and Astrophysics*, Vol. 9, No. 6, jun 2009, pp. 694.

[30] Luo, J., Zhu, L., Zhang, K., Zhao, C., and Liu, Z., "Forecasting the 10.7-cm Solar Radio Flux Using Deep CNN-LSTM Neural Networks," *Processes*, Vol. 10, No. 2, 2022.

[31] NOAA SWPC, "Users Guide to The Preliminary Report and Forecast of Solar Geophysical Data," August 2012.

[32] Wang, Z., Hu, Q., Zhong, Q., and Wang, Y., "Linear Multistep F10.7 Forecasting Based on Task Correlation and Heteroscedasticity," *Earth and Space Science*, Vol. 5, No. 12, 2018, pp. 863–874.

[33] Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y., "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," *CoRR*, Vol. abs/1905.10437, 2019.

[34] Benson, B., Brown, E., Bonasera, S., Acciarini, G., Pérez-Hernández, J. A., Sutton, E., Jah, M. K., Bridges, C., Jin, M., and Baydin, A. G., "Simultaneous Multivariate Forecast of Space Weather Indices using Deep Neural Network Ensembles," *n/a*, Dec. 2021.

[35] Henney, C. J., Toussaint, W. A., White, S. M., and Arge, C. N., "Forecasting $F_{10.7}$ with solar magnetic flux transport modeling," *Space Weather*, Vol. 10, No. 2, feb 2012, pp. n/a–n/a.

[36] Samuel, A. L., "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, Vol. 3, No. 3, 1959, pp. 210–229.

[37] Bi, Q., Goodman, K. E., Kaminsky, J., and Lessler, J., "What is Machine Learning? A Primer for the Epidemiologist," *American Journal of Epidemiology*, Vol. 188, No. 12, 10 2019, pp. 2222–2239.

[38] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I., "Attention is all you need," *Advances in neural information processing systems*, Vol. 30, 2017.

[39] McCulloch, W. S. and Pitts, W., "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, Vol. 5, No. 4, dec 1943, pp. 115–133.

[40] Li, Z., Liu, F., Yang, W., Peng, S., and Zhou, J., "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, 2021.

[41] Gardner, M. and Dorling, S., "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric Environment*, Vol. 32, No. 14, 1998, pp. 2627–2636.

[42] Muralikrishna, A., Lago, A. D., and da Silva, J. D. S., "Geomagnetic Dst index forecast using a multilayer perceptrons artificial neural network," *International Association of Geomagnetism and Aeronomy IAGA*, International Association of Geomagnetism and Aeronomy IAGA, 2009, pp. 1212–1213.

[43] Florios, K., Kontogiannis, I., Park, S.-H., Guerra, J. A., Benvenuto, F., Bloomfield, D. S., and Georgoulis, M. K., "Forecasting Solar Flares Using Magnetogram-based Predictors and Machine Learning," *Solar Physics*, Vol. 293, No. 2, jan 2018.

[44] Larsen, E. E., "Predicting Solar Flares with Machine Learning," 2021.

[45] Wang, P., Chen, Z., Deng, X., Wang, J., Tang, R., Li, H., Hong, S., and Wu, Z., "The prediction of storm-time thermospheric mass density by LSTM-based ensemble learning," *Space Weather*, Vol. 20, No. 3, 2022, pp. e2021SW002950.

[46] Rosenblatt, F., "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review*, Vol. 65, No. 6, 1958, pp. 386–408.

[47] Melcher, K., "A Friendly Introduction to [Deep] Neural Networks," *KNIME Blog*, 2021.

[48] Cybenko, G., "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, Vol. 2, No. 4, 1989, pp. 303–314.

[49] Hochreiter, S. and Schmidhuber, J., "Long Short-Term Memory," *Neural Computation*, Vol. 9, No. 8, 1997, pp. 1735–1780.

[50] Wasserman, P. and Schwartz, T., "Neural networks. II. What are they and why is everybody so interested in them now?" *IEEE Expert*, Vol. 3, No. 1, 1988, pp. 10–15.

[51] Tian, Y. and Zhang, Y., "A comprehensive survey on regularization strategies in machine learning," *Information Fusion*, Vol. 80, 2022, pp. 146–166.

[52] Wei, C., Kakade, S., and Ma, T., "The Implicit and Explicit Regularization Effects of Dropout," *Proceedings of the 37th International Conference on Machine Learning*, edited by H. D. III and A. Singh, Vol. 119 of *Proceedings of Machine Learning Research*, PMLR, 13–18 Jul 2020, pp. 10181–10192.

[53] Xu, Y. and Goodacre, R., "On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning," *Journal of Analysis and Testing*, Vol. 2, No. 3, jul 2018, pp. 249–262.

[54] Wager, S., Fithian, W., Wang, S., and Liang, P. S., "Altitude Training: Strong Bounds for Single-Layer Dropout," *Advances in Neural Information Processing Systems*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Vol. 27, Curran Associates, Inc., 2014, p. n/a.

[55] Helmbold, D. P. and Long, P. M., "On the inductive bias of dropout," *The Journal of Machine Learning Research*, Vol. 16, No. 1, 2015, pp. 3403–3454.

[56] Arora, S., Cohen, N., Hu, W., and Luo, Y., "Implicit regularization in deep matrix factorization," *Advances in Neural Information Processing Systems*, Vol. 32, 2019.

[57] Swirszcz, G., Czarnecki, W. M., and Pascanu, R., "Local minima in training of neural networks," 2016.

[58] Al-Shareef, A. and Abbod, M., "Neural Networks Initial Weights Optimisation," *2010 12th International Conference on Computer Modelling and Simulation*, 2010, pp. 57–61.

[59] Sola, J. and Sevilla, J., "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Transactions on Nuclear Science*, Vol. 44, No. 3, jun 1997, pp. 1464–1468.

[60] Daniell, J. D. and Mehta, P. M., "Probabilistic Solar Proxy Forecasting With Neural Network Ensembles," *Space Weather*, Vol. 21, No. 9, sep 2023.

[61] Xu, C., Lu, C., Liang, X., Gao, J., Zheng, W., Wang, T., and Yan, S., "Multi-loss Regularized Deep Neural Network," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 26, No. 12, dec 2016, pp. 2273–2283.

[62] Blum, A., Kalai, A., and Langford, J., "Beating the hold-out," *Proceedings of the twelfth annual conference on Computational learning theory*, ACM, jul 1999, p. n/a.

[63] Hansen, L. and Salamon, P., "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 10, 1990, pp. 993–1001.

[64] Brown, G., Wyatt, J. L., Tino, P., and Bengio, Y., "Managing diversity in regression ensembles." *Journal of machine learning research*, Vol. 6, No. 9, 2005.

[65] Mendes-Moreira, J. a., Soares, C., Jorge, A. M., and Sousa, J. F. D., "Ensemble Approaches for Regression: A Survey," *ACM Comput. Surv.*, Vol. 45, No. 1, dec 2012.

[66] Giacinto, G. and Roli, F., "Design of effective neural network ensembles for image classification purposes," *Image and Vision Computing*, Vol. 19, No. 9, 2001, pp. 699–707.

[67] Sagi, O. and Rokach, L., "Ensemble learning: A survey," *WIREs Data Mining and Knowledge Discovery*, Vol. 8, No. 4, 2018, pp. e1249.

[68] Elvidge, S., Granados, S. R., Angling, M. J., Brown, M. K., Themens, D. R., and Wood, A. G., "Multi-Model Ensembles for Upper Atmosphere Models," *Space Weather*, Vol. 21, No. 3, mar 2023.

[69] Mishra, U., Gautam, S., Riley, W. J., and Hoffman, F. M., "Ensemble machine learning approach improves predicted spatial variation of surface soil organic carbon stocks in data-limited northern circumpolar region," *Frontiers in big Data*, Vol. 3, 2020, pp. 528441.

[70] Ahmad, W., Aamir, M., Khalil, U., Ishaq, M., Iqbal, N., and Khan, M., "A new approach for forecasting crude oil prices using median ensemble empirical mode decomposition and group method of data handling," *Mathematical Problems in Engineering*, Vol. 2021, 2021, pp. 1–12.

[71] Sridhar, D. V., Seagrave, R. C., and Bartlett, E. B., "Process modeling using stacked neural networks," *AIChE Journal*, Vol. 42, No. 9, sep 1996, pp. 2529–2539.

[72] Marcellino, M., Stock, J. H., and Watson, M. W., "A comparison of direct and iterated multistep AR methods for forecasting macroeconomic time series," *Journal of Econometrics*, Vol. 135, No. 1, 2006, pp. 499–526.

[73] Lambert, N., Pister, K., and Calandra, R., "Investigating compounding prediction errors in learned dynamics models," *arXiv preprint arXiv:2203.09637*, 2022.

[74] Abdi, H. and Williams, L. J., "Principal component analysis," *WIREs Computational Statistics*, Vol. 2, No. 4, jul 2010, pp. 433–459.

[75] Hu, Z., Pan, G., Wang, Y., and Wu, Z., "Sparse Principal Component Analysis via Rotation and Truncation," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 27, No. 4, 2016, pp. 875–890.

[76] Phyo, P. P. and Byun, Y.-C., "Hybrid Ensemble Deep Learning-Based Approach for Time Series Energy Prediction," *Symmetry*, Vol. 13, No. 10, oct 2021, pp. 1942.

[77] Victoria, A. H. and Maragatham, G., "Automatic tuning of hyperparameters using Bayesian optimization," *Evolving Systems*, Vol. 12, No. 1, may 2020, pp. 217–223.

[78] Qureshi, A. S., Khan, A., Zameer, A., and Usman, A., "Wind power prediction using deep neural network based meta regression and transfer learning," *Applied Soft Computing*, Vol. 58, sep 2017, pp. 742–755.

[79] Gerace, F., Saglietti, L., Mannelli, S. S., Saxe, A., and Zdeborová, L., "Probing transfer learning with a model of synthetic correlated datasets," *Machine Learning: Science and Technology*, Vol. 3, No. 1, feb 2022, pp. 015030.

[80] Weiss, K., Khoshgoftaar, T. M., and Wang, D., "A survey of transfer learning," *Journal of Big Data*, Vol. 3, No. 1, may 2016.

[81] Naser, M. Z. and Alavi, A. H., "Error Metrics and Performance Fitness Indicators for Artificial Intelligence and Machine Learning in Engineering and Sciences," *Architecture, Structures and Construction*, nov 2021.

[82] Yaya, P., Hecker, L., de Wit, T. D., Fèvre, C. L., and Bruinsma, S., "Solar radio proxies for improved satellite orbit prediction," *Journal of Space Weather and Space Climate*, Vol. 7, 2017, pp. A35.

[83] Licata, R. J., Mehta, P. M., Tobiska, W. K., and Huzurbazar, S., "Machine-Learned HASDM Thermospheric Mass Density Model With Uncertainty Quantification," *Space Weather*, Vol. 20, No. 4, 2022b, pp. e2021SW002915, e2021SW002915 2021SW002915.

[84] Anderson, G. J., Gaffney, J. A., Spears, B. K., Bremer, P.-T., Anirudh, R., and Thiagarajan, J. J., "Meaningful uncertainties from deep neural network surrogates of large-scale numerical simulations," *arXiv preprint arXiv:2010.13749*, 2020.

[85] Laves, M.-H., Ihler, S., Fast, J. F., Kahrs, L. A., and Ortmaier, T., "Recalibration of Aleatoric and Epistemic Regression Uncertainty in Medical Imaging," *n/a*, April 2021.

[86] Gneiting, T., Raftery, A. E., Westveld, A. H., and Goldman, T., "Calibrated Probabilistic Forecasting Using Ensemble Model Output Statistics and Minimum CRPS Estimation," *Monthly Weather Review*, Vol. 133, No. 5, may 2005, pp. 1098–1118.

[87] Pardini, C. and Anselmo, L., "Revisiting the collision risk with cataloged objects for the Iridium and COSMO-SkyMed satellite constellations," *Acta Astronautica*, Vol. 134, may 2017, pp. 23–32.

[88] González, S., García, S., Ser, J. D., Rokach, L., and Herrera, F., "A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities," *Information Fusion*, Vol. 64, dec 2020, pp. 205–237.