

Faculté des sciences

Prediction of electron fluxes in the outer layers of the Earth's atmosphere

In relation with the royal belgian institute for space
aeronomy

Author: **Mathieu GRAF**

Supervisors: **Robin VAN OIRBEEK, Viviane PIERRARD**

Readers: **Edith BOTEK, Johan SEGERS**

Academic year 2023–2024

Master [120] en science des données, orientation statistique

Contents

1	Introduction	7
1.1	Motivation	7
1.2	Physics context	8
1.3	Outline	9
2	Physics Team Achievements	10
2.1	Methodologies Employed	10
2.2	Feature Selection and Model Training	10
2.3	Performance Evaluation and Outcomes	11
2.4	Future Directions	11
3	Data Exploration	11
3.1	The data of our study	11
3.2	Variable Exploration	13
3.3	Correlation Matrix Analysis	15
3.4	Principal Component Analysis (PCA)	16
4	Importance of Stationarity in Time Series Analysis	19
4.1	Understanding Trend and Seasonality	19
4.1.1	Trend	19
4.1.2	Seasonality	19
4.2	Addressing Trend and Seasonality	19
4.3	Analyzing and Removing Trend and Seasonality	19
4.3.1	Analyzing Trend and Seasonality	19
4.4	Mathematical Explanation for Removing Trend and Seasonality	20
4.4.1	Differencing	20
4.4.2	Detrending	21
4.4.3	Log Transformation	21
4.4.4	Moving Average	21
4.5	Seasonal Decomposition of Time Series (STL)	22
4.6	Practical Implications for Our Study	23
5	Features selection	24
5.1	Introduction	24
5.2	Filter Method	24
5.2.1	Correlation Plot	24
5.2.2	Mutual Information	25
5.3	Wrappers Method	26
5.4	Embedded Method	28
5.4.1	Ridge Regression (L2 Regularization)	28
5.4.2	Lasso Regression (L1 Regularization)	29

6	Models	32
6.1	LSTM	32
6.1.1	Origins and Development of LSTM Models	32
6.1.2	LSTM Architecture and Mathematical Formulations	32
6.2	CNN	34
6.2.1	Origins and Development of Convolutional Neural Network Models	34
6.2.2	CNN architecture and Mathematical Formulation	36
7	Training of the Models	40
7.1	Hyperparameter tuning	40
7.1.1	Tuning space	40
7.1.2	Method	41
7.1.3	Analysis of Hyperparameter Tuning Process	42
7.2	Final Models	46
7.2.1	Fine tuning	46
7.2.2	Features selected	46
7.2.3	Models Structure	47
8	Comparison Of the model	49
8.1	Metrics	49
8.1.1	RMSE	49
8.1.2	Correlation	49
8.1.3	Percentage Explained	50
8.1.4	Median Symmetric Accuracy (MSA)	50
8.1.5	Symmetric Signed Percentage Bias (SSPB)	50
8.2	Results	51
8.2.1	log e5	51
8.2.2	log e1	52
8.2.3	Comparison with the physics team results	52
9	Further Amelioration	53
10	Conclusion	54
11	ANNEX	57
11.1	Batchsize - learning rate optimization search with CNN 1D	68

List of Figures

1	Van Allen Radiation Belt	8
2	Solar Flares of May 10, 2024 [1]	9
3	Kernel density plots of e5 L3-8 variables	14
4	Pearson Correlation matrix of e5 flux in L3-8 region	15
5	Variable representation	18
6	Variance explained	18
7	data of eletron flux for e5 channel in L3-8 region	23
8	Correlation Plots of All Available Variables	26
9	Mutual Information of All Available Variables	26
10	Forward Selection Performance	28
11	LASSO Regression	31
12	Ridge Regression	31
13	LSTM Cell Diagram	33
14	1D Convolution	38
15	2D Convolution	39
16	(a) The heatmap shows the validation error over a two-dimensional search space with red corresponding to areas with lower validation error. Configuration selection methods adaptively choose new configurations to train, proceeding in a sequential manner as indicated by the numbers. (b) The plot shows the validation error as a function of the resources allocated to each configuration (i.e. each line in the plot). Configuration evaluation methods allocate more resources to promising configurations.	41
17	Parallel Coordinates Plot CNN1d e1 L3-8	42
18	Correlation Plot CNN1d e1 L3-8	43
19	Variance explained	44
20	variable representation	44
21	parallel plot of hyperparameter tuning CNN1d e1 L3-8	58
22	Correlation plot CNN1d e1 L3-8	58
23	parallel plot of hyperparameter tuning CNN1d e5 L3-8	60
24	Correlation plot CNN1d e5 L3-8	60
25	parallel plot of hyperparameter tuning CNN2d e1 L3-8	62
26	Correlation plot CNN2d e1 L3-8	62
27	parallel plot of hyperparameter tuning CNN2d e5 L3-8	64
28	Correlation plot CNN2d e5 L3-8	64

List of Tables

1	Comparison of Different Feature Selection Methods	46
2	E5 L3-8 prediction	51
3	E1 L3-8 prediction	52
4	Performance metrics for e1 and e5 with their LSTM models	52
5	Summary Statistics of Variables	57
6	E1 L1-8 prediction	66
7	E5 L1-8 prediction	66

8	E1 L2-8 prediction	67
9	E5 L2-8 prediction	67
10	e1 L3-8 - batchsize = 100 ; learning rate = 1e-4	68
11	e1 L3-8 - batchsize = 10 ; learning rate = 1e-4	68
12	e1 L3-8 - batchsize = 18 ; learning rate = 1e-4	69
13	e1 L3-8 - batchsize = 20 ; learning rate = 1e-4	69
14	e1 L3-8 - batchsize = 25 ; learning rate = 1e-4	69
15	e1 L3-8 - batchsize = 50 ; learning rate = 1e-4	70
16	e1 L3-8 - batchsize = 10 ; learning rate = 1e-3	70
17	e1 L3-8 - batchsize = 25 ; learning rate = 1e-3	70

Acknowledgments

I would like to express my deepest gratitude to Mr. Robin Van Oirbeek for being incredibly encouraging and helpful. His discussions about the future ahead of me as a data scientist were tremendously interesting and insightful.

My heartfelt thanks go to Edith Botek and Viviane Pierrard for their unwavering support and for always answering my questions with patience and dedication. To my dear friends, your encouragement has been a constant source of strength and motivation. My gratitude extends to the team at the Royal Belgian Institute for Space Aeronomy for their invaluable insights and guidance.

A special thanks to Josh Starmer for his enlightening videos on machine learning, which have been a true friend through these two years.

I am also deeply grateful to Michaël de Marliave, whose work inspired my journey into data science. Additionally, I am profoundly thankful to Mr. Falomir for challenging me to reach new heights, I won't forget your sayings. Your collective contributions have been instrumental in my growth and success.

Abstract

Space weather, encompassing various dynamic phenomena originating from the Sun and its interactions with the Earth's magnetosphere, ionosphere, and thermosphere, poses significant challenges to modern technological systems and infrastructure. The impacts of space weather events range from disruptions in satellite communications and GPS navigation to potential hazards for astronauts and damage to critical infrastructure such as power grids and pipelines. Consequently, accurate forecasting of space weather phenomena is crucial for mitigating these impacts and ensuring the resilience of technological systems and societal infrastructure.

The aim of this master thesis is to establish a milestone for teams engaged in time series forecasting tasks by creating guidelines for good practices. Specifically tailored for space weather prediction, this endeavor involves exploring and developing techniques for time series forecasting. By integrating insights from the field of time series analysis and incorporating domain-specific knowledge of space weather dynamics, this research aims to improve the accuracy and reliability of space weather forecasts.

1 Introduction

1.1 Motivation

The motivation behind this research stems from the critical need to address the increasing vulnerability of modern technological infrastructure to space weather events. Over recent decades, society has become increasingly reliant on satellite-based services, wireless communications, and interconnected power grids. These technologies are integral to various aspects of everyday life, including telecommunications, navigation, weather forecasting, and emergency response systems.

However, the dependence on such technologies also exposes them to the disruptive effects of space weather phenomena. These phenomena, driven by solar activity and the interactions between the solar wind and Earth's magnetosphere, ionosphere, and thermosphere, can lead to a range of adverse impacts. For instance, geomagnetic storms resulting from solar flares and coronal mass ejections can induce currents in power transmission lines, potentially leading to blackouts and damage to electrical infrastructure. Similarly, ionospheric disturbances can disrupt satellite communications, GPS navigation, and high-frequency radio transmissions, affecting both civilian and military operations.

The economic and societal consequences of space weather events can be substantial. Studies estimate that the cost of space weather-related disruptions to the U.S. economy alone could range from millions to billions of dollars annually [2], depending on the severity of the event. Moreover, the potential risks to public safety, national security, and critical infrastructure underscore the urgency of improving our ability to predict and mitigate the impacts of space weather.

By advancing the state-of-the-art in space weather forecasting [3], this research aims to address these challenges and contribute to the development of more resilient and adaptive technological systems. Accurate and timely forecasts of space weather phenomena can enable proactive measures to mitigate their impacts, such as adjusting satellite operations, rerouting flights, and implementing protective measures for electrical grids and communication networks.

Furthermore, by enhancing our understanding of the underlying mechanisms driving space weather phenomena, this research has broader implications for space exploration, planetary science, and fundamental physics. Insights gained from studying space weather dynamics can inform our understanding of similar processes occurring in other planetary atmospheres and astrophysical environments.

1.2 Physics context

Van Allen Radiation Belts: Importance and Definition

The Van Allen radiation belts are zones of charged particles, primarily electrons and protons, trapped by Earth's magnetic field. They consist of two main belts: the inner belt, dominated by high-energy protons, and the outer belt, predominantly composed of high-energy electrons. These belts are named after James Van Allen, who discovered them in 1958.

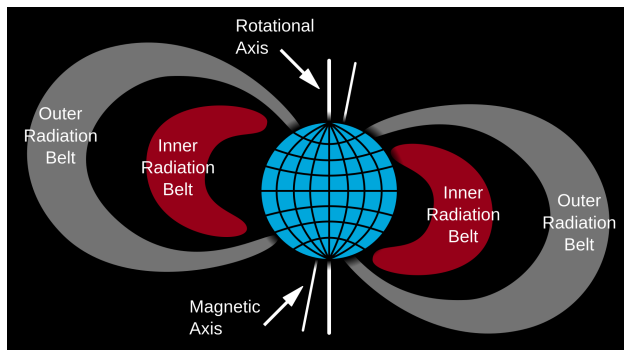


Figure 1: Van Allen Radiation Belt

The Van Allen belts play a critical role in space weather phenomena. They affect satellite operations, as the radiation found in these belts can damage electronic components and pose risks to satellite communications and lifespan. Understanding the dynamics of these belts is crucial for designing spacecraft that can withstand these harsh conditions. Additionally, the belts are a key feature in studying Earth's magnetospheric system and its interactions with solar wind and magnetic storms.

Electron Fluxes: Nature, Energy, and Risks

Electron fluxes in the context of space weather refer to the flow rate of electrons passing through a unit area in the radiation belts. These electrons are part of the particle population in space that varies in energy from a few keV (kiloelectron volts) to several MeV (megaelectron volts).

The energy of these electrons determines their penetration ability and the kind of risk they pose. Lower-energy electrons contribute to surface charging of spacecraft, while higher-energy electrons can penetrate deeper and contribute to internal charging or "deep dielectric charging," which can disrupt spacecraft operations and damage internal electronics.

Solar Activities: Definition and Impact on Earth

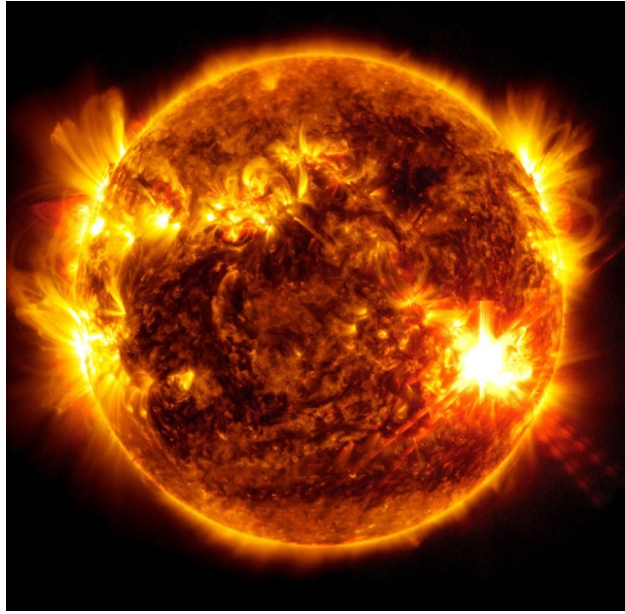


Figure 2: Solar Flares of May 10, 2024 [1]

Solar activities encompass various phenomena including solar flares, solar wind, coronal mass ejections (CMEs), and solar energetic particles. These phenomena stem from the Sun's magnetic field and its dynamic processes, which can eject massive quantities of particles and energy into space [4]. Interactions between solar wind and Earth's magnetic field can cause geomagnetic storms, which may disrupt communication and navigation systems on Earth. Another more appealing aspects of these events are auroras. These are visual manifestations of solar particles interacting with Earth's atmosphere, commonly seen as auroras (Northern and Southern Lights). On the other hand of the spectrum, cases where we do not want the sun to interfere is aviation. Increased solar activity can enhance radiation belt dynamics, leading to higher risks for satellites and potentially hazardous radiation exposure for aviators on polar flight routes.

1.3 Outline

The goal of this master thesis is to provide a comprehensive step-by-step guide to time series forecasting on space weather data [5], emphasizing best practices for machine learning. The primary objective of this master thesis is to detail the techniques involved in time series forecasting to ensure good practice. We will examine two prominent models: Long Short-Term Memory (LSTM) used by the physics team and the potential of Convolutional Neural Networks (CNN) for forecasting tasks.

Space weather data is often characterized by its complexity and irregularity, necessitating thorough data preprocessing to enhance usability. This includes data cleaning, normalization, and handling missing values. Additionally, time series decomposition techniques may be applied to extract meaningful patterns and trends. We will investigate detrending and deseasonalizing the data using differencing.

Feature selection is crucial in improving the performance of forecasting models. Given the multidimensional nature of space weather data, identifying the most informative features through techniques like correlation analysis, principal component analysis (PCA), and domain knowledge is essential. The selected

features serve as input variables for the forecasting models, aiding in capturing the underlying dynamics of space weather phenomena.

LSTM networks are effective for modeling sequential data with long-range dependencies. We will discuss the architecture and implementation details of the LSTM model for space weather forecasting. The LSTM network is designed to learn and exploit the temporal relationships present in the data, enabling accurate predictions of future space weather conditions. Various configurations of LSTM layers, activation functions, and regularization techniques will be explored to optimize model performance.

CNNs are gaining prominence in time series analysis due to their ability to capture spatial and temporal patterns simultaneously. We will outline the utilization of CNNs for space weather forecasting, leveraging their capability to extract hierarchical features from multidimensional data. The CNN architecture is tailored to accommodate the unique characteristics of space weather time series, such as spatial correlations and non-linear dependencies. Different convolutional layer configurations, kernel sizes, and pooling strategies will be investigated to construct an effective CNN-based forecasting model.

By meticulously addressing each of these topics, we aim to establish a detailed and practical guide for time series forecasting on space weather data. Through comprehensive data preprocessing, feature selection, and the implementation of LSTM and CNN models, this thesis aspires to provide accurate predictions of space weather phenomena, contributing to advancements in space weather forecasting and mitigation strategies.

In this master's thesis, we will focus exclusively on the L3-8 work related to the E5 electron flux. To avoid overwhelming the readers with repetitive information, we will not include similar analyses for other datasets, as the same principles apply to all, the results for all datasets will be put in annex.

2 Physics Team Achievements

The ultimate goal of this research is to enhance the accuracy and reliability of space weather forecasting by building on the work of the physics team. Their methodologies and achievements form the foundation of this study. This thesis aims to refine their methodologies and incorporate new techniques to improve these forecasting models.

2.1 Methodologies Employed

The physics team, led by Edith Botek, Viviane Pierrard, and Alexandre Winant, has focused on predicting the electron fluxes in the Van Allen radiation belts using data-driven models. They have employed Long Short-Term Memory (LSTM) neural networks, trained with electron flux measurements from the Energetic Particle Telescope (EPT) onboard the PROBA-V satellite. Their methodology involves meticulous data collection and preprocessing. They utilize high-energy electron flux data from the PROBA-V/EPT, covering measurements from a low Earth orbit at various energy levels (0.5-20 MeV for electrons). Additional input data includes solar wind parameters and geomagnetic indices retrieved from the OMNIWEB database. The data undergoes rigorous preprocessing steps, such as cleaning, normalization, and handling of missing values, ensuring its suitability for training the neural network models.

2.2 Feature Selection and Model Training

Feature selection is crucial in their approach, with key features selected based on their correlation with the electron fluxes. These features include satellite coordinates (L, magnetic local time, latitude), solar wind speed and pressure, and geomagnetic indices (SYM/H, AL). Correlation analysis helps identify the most informative features, which are then used as inputs for the LSTM models.

The team trains and validates their models with these selected features. The LSTM model architecture is chosen for its ability to capture long-term dependencies in sequential data. The model is trained using a combination of these features, with different configurations tested to optimize performance. The training data set is divided into training, validation, and test sets to ensure robust model evaluation and prevent overfitting.

2.3 Performance Evaluation and Outcomes

Performance evaluation of the models is a key aspect of their methodology. Metrics such as Root Mean Square Error (RMSE), Spearman correlation coefficient, Prediction Efficiency (PE), and Median Symmetric Accuracy (MSA) are used to assess the model’s performance. The results show a high correlation between predicted and observed fluxes, demonstrating the model’s effectiveness in capturing space weather dynamics.

The physics team’s research has achieved several notable outcomes. The LSTM model demonstrates a correlation of over 0.9 between predicted and observed electron fluxes, RMSE and MAE corroborate to indicate high prediction accuracy. The model effectively predicts significant space weather events such as flux dropouts, enhancements, and recovery features. By incorporating data from the PROBA-V/EPT and additional geomagnetic and solar wind parameters, the team constructs a comprehensive data set that significantly improves model performance. The use of both non-relativistic and relativistic electron flux data allows for a more detailed analysis of the Van Allen belts’ dynamics. Their models have practical applications in predicting space weather impacts on near-Earth space operations, such as satellite performance and communication systems. Their work also contributes to the broader field of space weather research by providing valuable insights into the mechanisms driving radiation belt dynamics.

2.4 Future Directions

Building on the physics team’s methodologies and achievements, this thesis seeks to further enhance space weather forecasting accuracy. By exploring advanced machine learning techniques, and improving model robustness and generalization, this research aims to develop more accurate and reliable space weather forecasting models. Ultimately, these efforts will contribute to the resilience of technological systems and infrastructure against space weather impacts.

3 Data Exploration

3.1 The data of our study

For this study, the Royal Belgian Institute for Space Aeronomy provided access to six different datasets: three concerning the E1 channel (0.5–0.6 MeV) of electrons and three concerning the E5 channel (1.0–2.4 MeV). The original time resolution of these datasets was up to 2 seconds, but we modified it to hourly intervals, resulting in approximately 33,800 observations. The study spans four years, from January 1, 2015, to December 31, 2018. Each dataset includes 18 variables:

- **Differential Electron flux:** Measurements of the electron channel, which refer to a specific energy range for electron measurements: e1 or e5, This unit describes the number of electrons per unit area (cm^{-2}), per unit time (s^{-1}), per unit solid angle (sr^{-1}), and per unit energy (MeV^{-1}).
- **lat (Latitude):** The geographic coordinate that specifies the north–south position of a point on the Earth’s surface.

- **rad (Radius)**: The distance from the center of the Earth to the satellite.
- **BU (Magnetic Field - B-Up)**: The intensity of the geomagnetic field directed upward from the Earth's surface at the satellite's location in the magnetic field's coordinate system.
- **LU (Magnetic Field - L-Up)**: Describes the location of a magnetic field line in the Earth's magnetosphere in terms of its distance from the Earth's center at the magnetic equator, the value is expressed in Earth radii with $1Rad = 6,378km$.
- **MLTU (Magnetic Local Time)**: A measure of the local time of day in the magnetic field's coordinate system.
- **cosMLTU (Cosine of Magnetic Local Time)**: The cosine of the magnetic local time, used to describe periodic variations in the geomagnetic field.
- **sinMLTU (Sine of Magnetic Local Time)**: The sine of the magnetic local time, used to describe periodic variations in the geomagnetic field.
- **Dst (Disturbance Storm Time Index)**: An index measuring the the intensity of the Earth's geomagnetic storms by tracking variations in the horizontal component of the Earth's magnetic field, particularly influenced by the ring current, which is affected by solar wind and geomagnetic activity.
- **kp_def (Kp Index - Definitive)**: A planetary geomagnetic activity index derived from measurements around the globe, indicating fluctuations in the Earth's magnetic field caused by solar wind, with values ranging from 0 (very quiet) to 9 (extremely disturbed).
- **WSpeed (Solar Wind Speed)**: The speed of the solar wind which impacts the Earth's magnetosphere.
- **ProtonDensity**: The density of protons in the solar wind.
- **ProtonTemperature**: The temperature of protons in the solar wind.
- **FlowPressure**: The dynamic pressure of the solar wind flow, which is a function of the solar wind density and speed.
- **AE-index (Auroral Electrojet Index)**: An index that measures the electrojet activity in the auroral zone, reflecting geomagnetic activity in high latitudes.
- **AL-index (Auroral Lower Index)**: An index similar to the AE-index but specifically measures the lower bound of auroral electrojet activity.
- **SYM/D (Symmetric Disturbance - perpendicular)**: A measure of the symetric disturbance, perpendicular to the dipole, part of the ring current.
- **SYM/H (Symmetric Disturbance - parallel)**: A measure of the symetric disturbance, parallel to the dipole, part of the ring current.

3.2 Variable Exploration

Our initial approach to analyzing the dataset involved a comprehensive exploration of each variable without applying transformations. By utilizing density plots, we uncovered notable variations across variables, prompting a deeper investigation through a suite of descriptive statistics. These included mean, variance, median, mode, standard deviation, minimum, maximum, range, first and third quartiles (Q1, Q3), interquartile range (IQR), suspected outlier count, skewness, and kurtosis. This analysis revealed deviations from normal distribution characteristics, suggesting the need for data transformation to improve model performance.

Explorative Study Using KDE Plots The explorative study focused on understanding variable distributions and characteristics via Kernel Density Estimate (KDE) plots, generated using the Seaborn library. These plots offered insights into distribution shapes, skewness, kurtosis, and potential outliers. Specifically, we observed:

- **Skewness:** Indications of asymmetry in distribution, with positive skewness suggesting right-tail elongation, and negative skewness indicating left-tail extension.
- **Kurtosis:** Observations of tail heaviness or lightness relative to a normal distribution. High kurtosis pointed to more pronounced peaks and heavier tails, whereas low kurtosis suggested flatter distributions.
- **Standard Deviation and Outliers:** The degree of data spread and the identification of outliers through deviation from the IQR were noted.

These observations underscored the benefits of applying transformations for normality, a common prerequisite for many statistical models. We also tried to preprocess these data applying Box-Cox transformation. The Box-Cox transformation is a technique used to make data more symmetrically distributed, and in some cases even normally distributed, which is a key requirement for many statistical models and analyses. This transformation helps stabilize the variance within the data, ensuring that the impact of large values is not disproportionately large compared to smaller values. By normalizing the data, the Box-Cox transformation allows statistical tests and models to perform more accurately and reliably. Additionally, this transformation addresses skewness in the data, correcting asymmetry and long tails, making the data easier to analyze. Overall, the Box-Cox transformation enhances the suitability of data for analysis by addressing issues related to non-normality and variance instability, ultimately leading to more robust and reliable results in statistical modeling. However, deep learning models, such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTMs), are often more flexible and capable of handling various data distributions without strict assumptions about normality. While these transformations often enhance model accuracy, this was not the case in our specific study. Despite our attempts to implement them, they proved ineffective and did not lead to any improvement in accuracy.

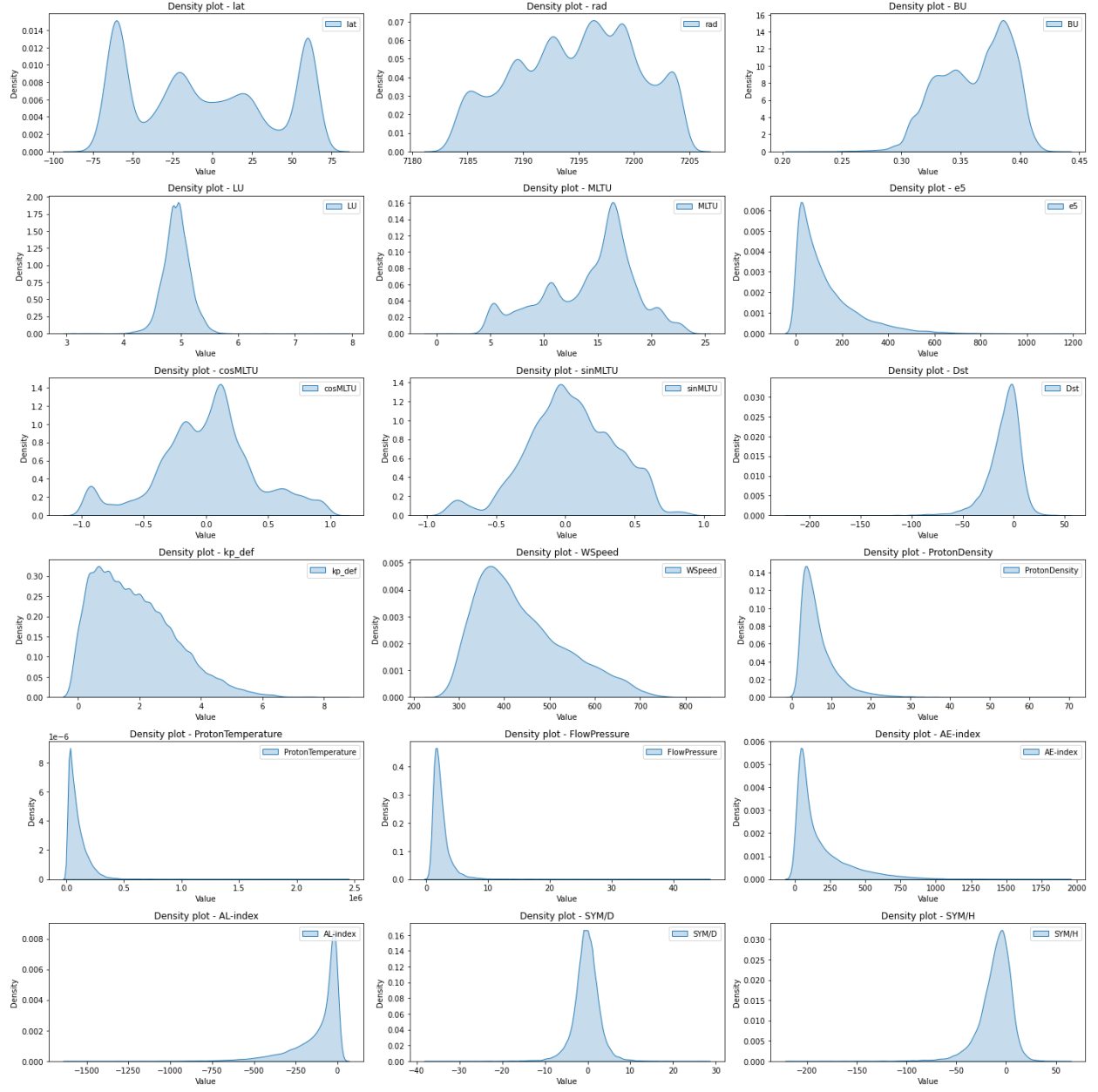


Figure 3: Kernel density plots of e5 L3-8 variables

3.3 Correlation Matrix Analysis

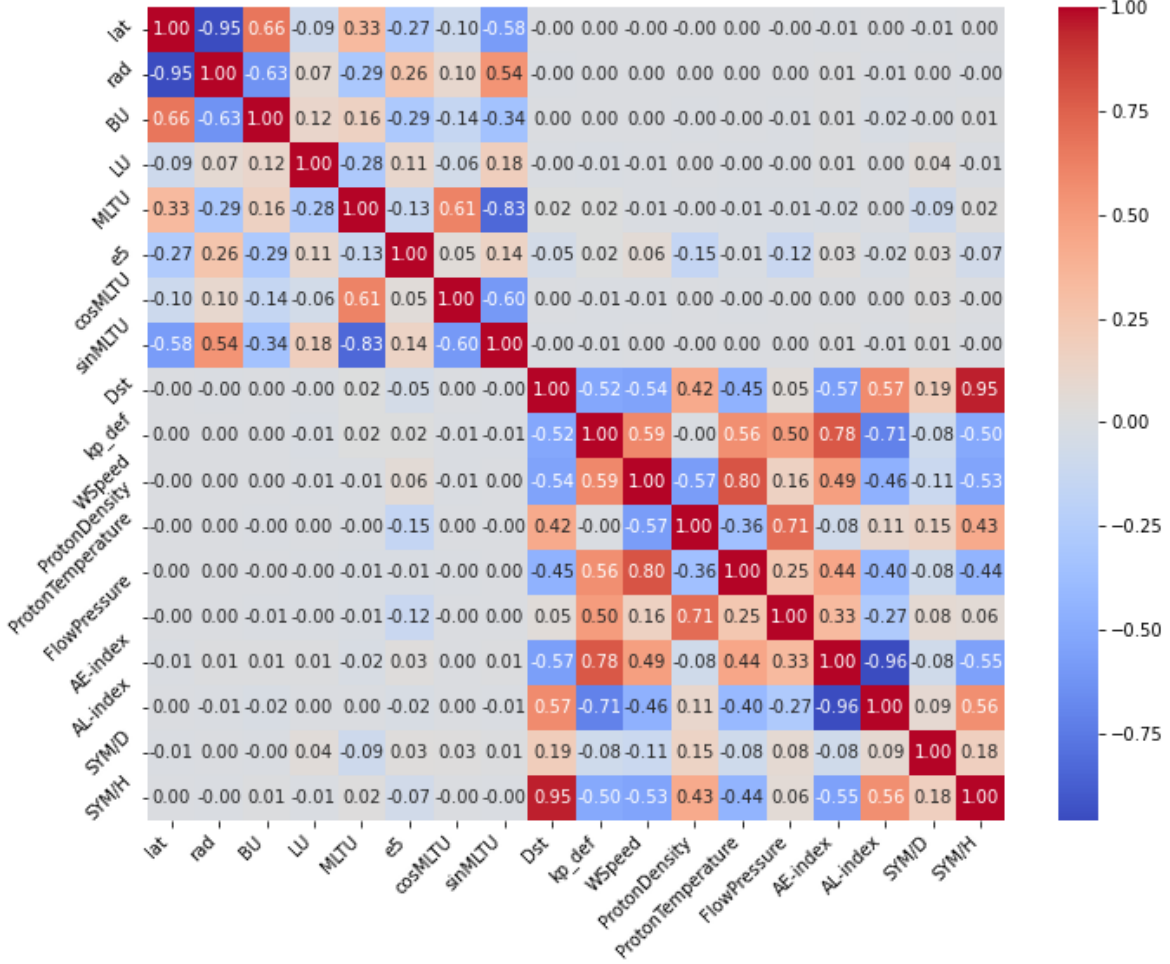


Figure 4: Pearson Correlation matrix of e5 flux in L3-8 region

The correlation matrix analysis illuminated linear relationships among variables, highlighting potential redundancies and unique contributions to the model's predictive power. Strong positive and negative correlations were identified, alongside variables exhibiting weak or negligible linear relationships.

Low or No Correlation (light gray cells Cells):

Many pairs of variables show very low to no correlation (values close to 0). For instance, solar wind parameters like ProtonDensity and ProtonTemperature have low correlations with geomagnetic indices like cosMLTU and sinMLTU.

Very strong Correlation ("Dark" Cells): We observe three pairs of intense relationships in our data. RAD and LAT are very strongly negatively correlated with a coefficient of -0.95 (Through this relation we could deduce the trajectory of the satellite as both express GPS coordinates), while the AE and AL indexes also show a strong negative correlation of -0.96 (This relation is expected since AE index is defined from AL index [6]. Conversely, there is a strong positive correlation of 0.95 between DST and Sym/H.

These strong correlations will play a crucial role in the feature selection phase, as they may indicate multicollinearity among the variables. Multicollinearity can impact the accuracy of regression methods; although its effects are less significant for neural networks[7], we will still consider it in our analysis.

It is intriguing to visually observe the aggregation of two distinct datasets: the first comprising solar wind parameters, and the second, geomagnetic data. Later, we will explore whether augmenting the solar wind dataset with geomagnetic data proves beneficial in feature selection. Correlation can be used for exploratory analysis as well as for feature selection.

3.4 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a powerful statistical technique used to reduce the dimensionality of a dataset while preserving as much variance as possible. This technique is widely used in data science for feature extraction, noise reduction, and visualization. Here's an extensive explanation of PCA, including both the mathematical foundations and the technical implementation.

Mathematical Foundations of PCA

Data Standardization:

PCA is sensitive to the scales of the features. Therefore, it's important to standardize the data, i.e., center the data around the mean and scale to unit variance.

$$X_{\text{standardized}} = \frac{X - \mu}{\sigma}$$

Where X is the data matrix, μ is the mean, and σ is the standard deviation.

Covariance Matrix Computation:

After standardizing the data, the next step is to compute the covariance matrix, which represents the pairwise covariance of features in the dataset.

$$\Sigma = \frac{1}{n-1}(X^T X)$$

Where X is the standardized data matrix and n is the number of samples.

Eigen Decomposition:

The covariance matrix Σ is then decomposed into its eigenvalues and eigenvectors. The eigenvectors (principal components) represent the directions of maximum variance, while the eigenvalues represent the magnitude of variance in these directions.

$$\Sigma V = V \Lambda$$

Where V is the matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues.

Sorting and Selecting Principal Components:

The eigenvalues and their corresponding eigenvectors are sorted in descending order. The top k eigenvectors corresponding to the k largest eigenvalues are selected as the principal components.

Transforming the Data:

The original data is projected onto the new space defined by the top k principal components.

$$X_{\text{PCA}} = X V_k$$

Where V_k is the matrix containing the top k eigenvectors.

Principal Component Analysis (PCA) is a crucial technique in data science, primarily used for reducing the dimensionality of datasets, which in turn enhances interpretability and improves the efficiency of models. The implementation of PCA involves several key steps, each playing a significant role in the overall process.

To begin with, data standardization is essential. This step ensures that each feature contributes equally to the analysis by normalizing the data, so that no single feature dominates due to its scale. Once the data is standardized, the next step is the calculation of the covariance matrix. This matrix captures the relationships between the features, providing insights into how changes in one feature may correspond with changes in another.

Following this, eigen decomposition is performed on the covariance matrix. This mathematical procedure identifies the directions, known as principal components, that maximize the variance in the data. These principal components are then sorted according to the magnitude of their corresponding eigenvalues, which indicates the importance of each component.

The data is then transformed by projecting it onto this new feature space defined by the principal components. This transformation is not merely a change of perspective; it serves as a powerful tool for various analytical purposes. For instance, it enables dimensionality reduction by allowing us to retain only the most significant components, thus simplifying the dataset while preserving the most critical information. Additionally, this transformation facilitates the visualization of high-dimensional data and aids in noise reduction, as the components with the highest variance typically capture the most meaningful patterns in the data.

When applying PCA in practice, a few considerations are paramount. One crucial decision is determining the number of principal components to retain. This choice is often guided by the explained variance ratio, where the goal is to retain a large percentage of the total variance, such as 95%, to ensure that the most important information is preserved. It is also important to standardize the data before applying PCA, especially if the features have different units or scales. Finally, one must be aware of the assumptions underlying PCA, such as the linearity of relationships among variables and the technique's sensitivity to outliers.

In conclusion, PCA stands as a foundational method in data science for effectively managing large datasets and complex feature spaces. By mastering the mathematical principles and technical steps involved in PCA, data scientists can leverage this technique to reduce dimensionality, enhance data interpretability, and ultimately improve the performance and efficiency of their models.

Detailed Analysis of PCA Plots

On our dataset the PCA biplot and scree plot offer significant insights, crucial for an extensive analysis.

Scree Plot Analysis

The scree plot helps to determine the number of principal components to retain. The first few components capture the most variance, with a slight drop in variance contribution after the second component. Retaining more than three or four components would add little explanatory power, as indicated by the "elbow" in the plot. This elbow can be very effective for choosing the number of dimension to take into account. However for data visualisation 2 components are still the best.

The cumulative variance explained by the first two components is not significant, slightly above 40-45%. This does not justifies focusing on these components too much for further analysis, as they does not capture most of the critical information without the complexity of additional components. Still we will present how we would analyse it as this technique can be usefull for other datasets.

PCA Biplot Analysis

The PCA biplot combines the projection of variables onto the first two principal components with a scatter plot of the data (We dropped the scatter plot for the ease of presentation). This visualization helps identify the relationships and relative importance of various variables in the dataset.

In the biplot, the first principal component (PCA 1) captures a significant portion of the variance. Variables such as ProtonDensity, ProtonTemperature, and FlowPressure are closely clustered, indicating they share similar

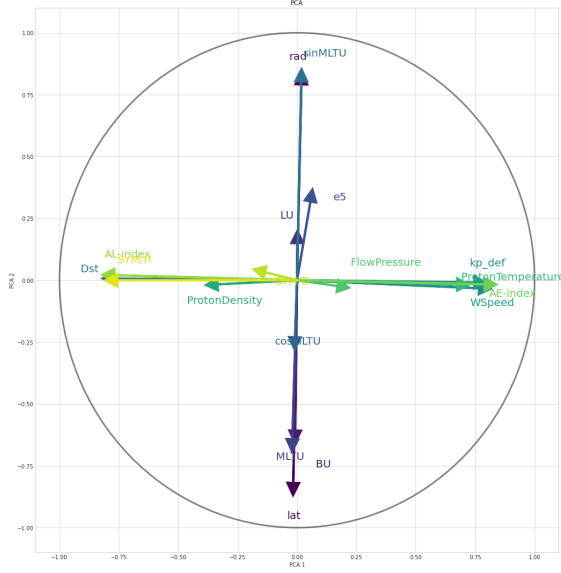


Figure 5: Variable representation

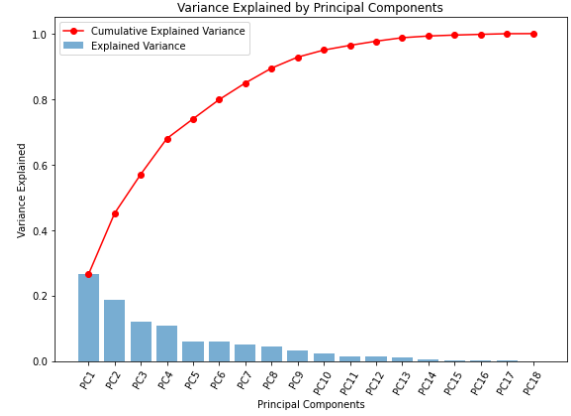


Figure 6: Variance explained

information or behaviors. These variables likely represent related physical processes in space weather phenomena.

The second principal component (PCA 2) seems to capture variance related to temporal and spatial patterns. Variables like $e5$, $\sin MLTU$, $\cos MLTU$, and L have strong projections along PCA 2, suggesting that magnetic local time and spatial factors are significant in the dataset. The clustering of many variables near the center indicates that they may not be as influential in explaining the variance or are possibly correlated with each other.

The biplot shows that $e5$ and L variables are close to $\sin MLTU$ and $\cos MLTU$ on PCA 2, suggesting these features are related to spatial or temporal patterns. The SYM/H index, a measure of global geomagnetic conditions, seems moderately correlated with PCA 1, highlighting its importance in capturing geomagnetic activity.

Conclusion

From the PCA analysis, it is evident that dimensionality reduction effectively simplifies the dataset while retaining essential information. The first two principal components capture the key variance, allowing for a more manageable number of features without losing too much significant information.

The importance of variables such as $e5$, $\sin MLTU$, $\cos MLTU$, and L is highlighted. These variables should be prioritized in feature selection for forecasting models due to their significant contribution to the variance. The clustering of variables like $ProtonDensity$, $ProtonTemperature$, and $FlowPressure$ indicates potential redundancy, suggesting the need for techniques like regularization to handle multicollinearity.

Temporal and spatial factors play a crucial role, as indicated by the strong projections of $\sin MLTU$, $\cos MLTU$, and L along PCA 2. These findings align with the physical understanding of space weather phenomena, where magnetic local time and spatial coordinates are significant. The role of geomagnetic indices and solar wind parameters, as shown by their moderate correlation with PCA 1, further emphasizes their importance in space weather dynamics.

Implication

The PCA findings suggest focusing on features that contribute significantly to the first two principal components for building predictive models. This approach reduces complexity and enhances model interpretability. Using PCA-transformed features for training models can efficiently capture the most critical variance, improving forecasting accuracy.

Models must account for both linear and non-linear relationships among variables. Advanced techniques like LSTM and CNN, which can capture temporal dependencies and spatial patterns, are particularly suitable for this task. Ensuring models address multicollinearity and redundancy through regularization techniques will enhance robustness and reliability.

4 Importance of Stationarity in Time Series Analysis

Stationarity is fundamental in time series analysis because it ensures that the statistical properties of the series, such as the mean and variance, remain constant over time. This stability is crucial for several reasons:

1. **Predictability:** A stationary series is easier to predict since its properties do not change over time. Non-stationary series, which may have trends or other patterns, are harder to model and forecast accurately.
2. **Model Validity:** Many time series models, such as ARIMA, rely on the assumption that the data is stationary. Applying these models to non-stationary data can result in poor performance and unreliable predictions.
3. **Statistical Inference:** Consistent statistical properties make it easier to draw meaningful conclusions. Estimating metrics like the mean and variance is more straightforward and accurate with stationary data.

4.1 Understanding Trend and Seasonality

4.1.1 Trend

This represents the long-term direction in the data. A series showing a general upward or downward movement over time indicates a trend. For example, a company's revenue increasing steadily over a decade shows an upward trend.

4.1.2 Seasonality

This refers to regular, repeating patterns within a specific period, such as daily, monthly, or yearly cycles. These patterns occur due to seasonal factors. For instance, retail sales often spike during the holiday season each year, showing a seasonal pattern.

4.2 Addressing Trend and Seasonality

To analyze and forecast time series data effectively, it is often necessary to make the series stationary by addressing trend and seasonality. Removing the trend involves eliminating the long-term movement, while removing seasonality means adjusting for regular, repeating patterns. Techniques to achieve this include differencing (subtracting previous values), detrending (removing the trend component), and seasonal adjustment (eliminating the seasonal effects). This ensures that the series' statistical properties remain consistent over time, making it suitable for various analytical methods and models.

4.3 Analyzing and Removing Trend and Seasonality

4.3.1 Analyzing Trend and Seasonality

To effectively analyze and model time series data, it is crucial to follow a structured approach that includes visualization, statistical testing, and the removal of trends and seasonality.

The first step in time series analysis is visualization. By plotting the data, you can visually inspect it for trends and seasonal patterns. An upward or downward slope in the plot typically indicates a trend, while recurring patterns suggest the presence of seasonality.

Once you’ve visually assessed the data, the next step is to perform statistical tests to determine its characteristics. One commonly used test is the Augmented Dickey-Fuller (ADF) test, which helps assess whether the time series is stationary. If the test result is significant, it may indicate non-stationarity in the data, often due to trends or seasonal patterns.

To further investigate the structure of the time series, it is important to examine autocorrelation. The Autocorrelation Function (ACF) analyzes the correlation between observations at different lags, revealing patterns that may indicate seasonality. Complementing this, the Partial Autocorrelation Function (PACF) identifies the specific lag relationships relevant for autoregressive models, showing how current values relate to past values independently of other lags.

When a time series exhibits a trend, it becomes necessary to remove this trend to make the data stationary. One common method is differencing, where each observation is subtracted from the previous one. First differencing, for instance, can effectively remove a linear trend, transforming a series into a stationary one. Alternatively, detrending can be achieved by fitting a linear regression line to the series and subtracting it, which is particularly useful for data with a linear trend. Additionally, applying a logarithmic transformation can stabilize variance and linearize trends, especially in cases where the series shows exponential growth.

Addressing seasonality in the data is another crucial step. Seasonal differencing involves subtracting the observation from the same period in the previous cycle, such as subtracting this month’s value from the same month’s value in the previous year. This method effectively removes seasonal effects while preserving the overall structure of the series. Another approach is the centered moving average, where the average over a specific window that covers a full seasonal cycle is calculated and subtracted from the series. This helps smooth out seasonal fluctuations. For more complex seasonal patterns, Seasonal-Trend decomposition using Loess (STL) can be employed. STL decomposes the series into its trend, seasonal, and residual components, allowing for a more detailed analysis and removal of the seasonal component.

Through these steps—visualization, statistical testing, and the removal of trends and seasonality—you can prepare time series data for more effective analysis and modeling, ensuring that any patterns and relationships within the data are clearly understood and appropriately handled.

4.4 Mathematical Explanation for Removing Trend and Seasonality

To effectively remove trend and seasonality from a time series, it is important to understand the mathematical basis for the techniques used. Here’s a breakdown of the key methods:

4.4.1 Differencing

First differencing involves subtracting the value of the series at time $t - 1$ from the value at time t .

$$y'_t = y_t - y_{t-1}$$

This technique helps to remove linear trends and stabilize the mean of the time series.

Example: If the original series y_t has a consistent increase of 2 units each period, differencing will result in a series where the trend is removed, showing a difference of 2 units.

Seasonal differencing subtracts the value from the same period in the previous season.

$$y''_t = y_t - y_{t-s}$$

Here, s is the seasonal period (e.g., 12 for monthly data with yearly seasonality).

Example: For monthly sales data, subtract the sales in January of the current year from the sales in January of the previous year.

4.4.2 Detrending

To remove a linear trend, you can fit a linear regression model to the time series and subtract the fitted values.

$$\text{Fit: } y_t = \beta_0 + \beta_1 t + \epsilon_t$$

$$\text{Detrended: } y'_t = y_t - (\beta_0 + \beta_1 t)$$

Where:

- β_0 is the intercept
- β_1 is the slope of the trend
- ϵ_t is the residual error

By subtracting the fitted linear trend from the original series, you obtain a detrended series y'_t with the linear trend removed.

4.4.3 Log Transformation

A log transformation stabilizes the variance and makes exponential trends linear.

$$y'_t = \log(y_t)$$

For data with exponential growth, applying the logarithm converts multiplicative relationships into additive ones, making the series easier to model and interpret.

4.4.4 Moving Average

To remove seasonality, compute the moving average over a window that spans a full seasonal cycle.

$$MA_t = \frac{1}{2k+1} \sum_{i=-k}^k y_{t+i}$$

Here, k is half the window size (for instance, for a yearly cycle in monthly data, $k = 6$).

Example: For monthly data, a 12-month centered moving average smooths out seasonal effects by averaging over a full year.

4.5 Seasonal Decomposition of Time Series (STL)

STL decomposes a time series into trend, seasonal, and residual components.

$$y_t = T_t + S_t + R_t$$

Where:

- T_t is the trend component
- S_t is the seasonal component
- R_t is the residual component

Decomposition Steps:

The first step in decomposition is to estimate the trend within the series. This can be achieved by applying a smoothing method, such as LOESS (Locally Estimated Scatterplot Smoothing). LOESS is particularly effective because it adapts to the local characteristics of the data, providing a flexible estimate of the trend without assuming a specific form.

Once the trend has been estimated, the next step is to remove it from the original time series. This is done by subtracting the estimated trend from the data, leaving a detrended series that no longer contains the long-term upward or downward movements that define the trend.

With the trend removed, attention shifts to estimating the seasonality. By analyzing the detrended series, you can compute seasonal averages, which capture the regular, repeating patterns that occur at specific intervals within the data. These seasonal averages represent the seasonal component of the series.

The final step is to remove the seasonality from the detrended series. This is accomplished by subtracting the seasonal component, leaving you with the residual series. The residual series should contain the random noise or irregular patterns that cannot be attributed to either trend or seasonality.

By carefully following these steps—estimating and removing the trend, then estimating and removing the seasonality—you can decompose a time series into its core components, making it easier to analyze, interpret, and model the data.

4.6 Practical Implications for Our Study

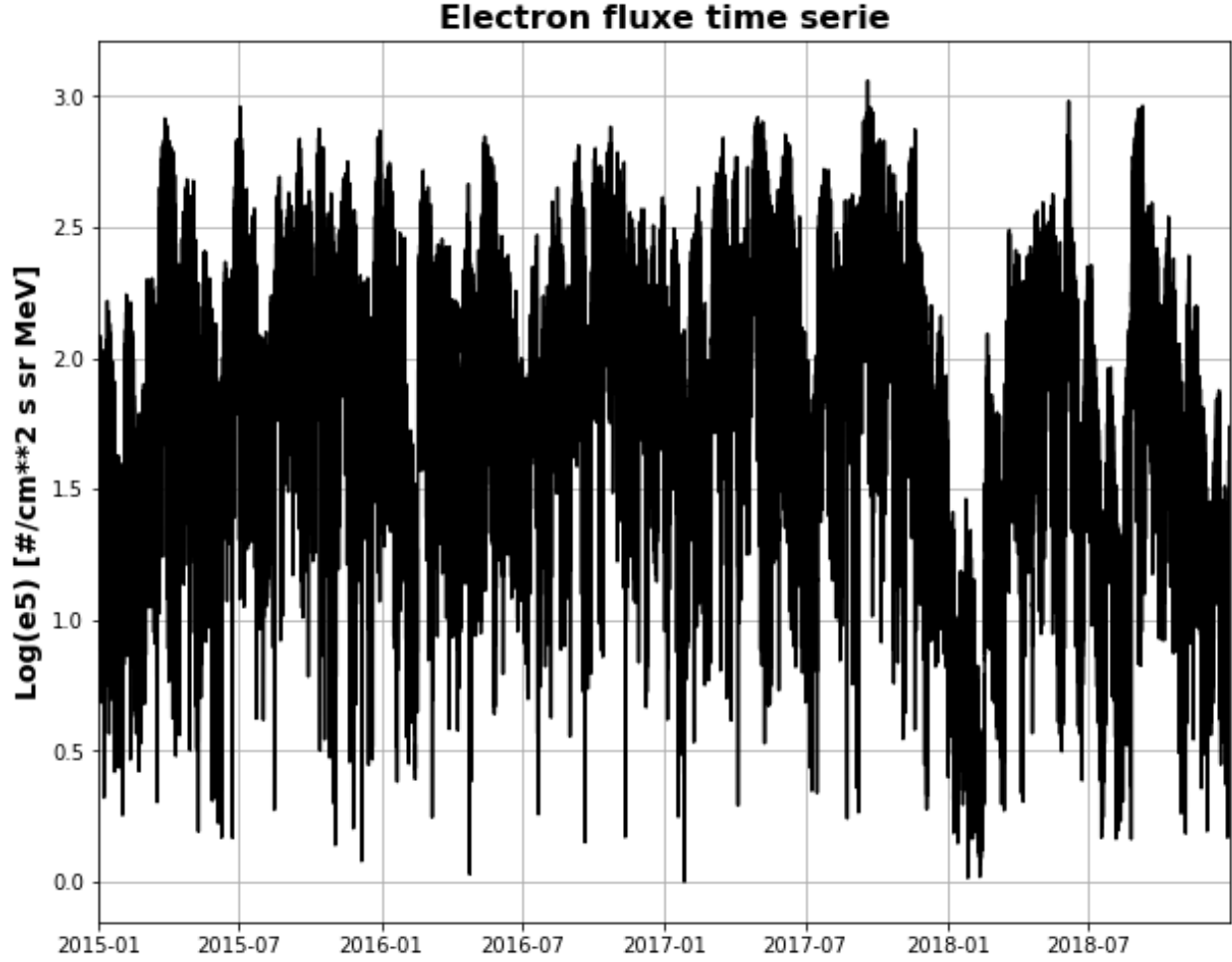


Figure 7: data of eletron flux for e5 channel in L3-8 region

The physics team had already chosen to use log transformation on the electron fluxes, and I decided to apply the same transformation to facilitate better comparison with their results. We have verified that the time series is stationnary with the ADF test mentionned earlier.

The results of the Augmented Dickey-Fuller (ADF) test are as follows:

- **ADF Statistic:** -7.6432
- **p-value:** 1.87×10^{-11}
- **Critical Values:**
 - 1%: -3.4305
 - 5%: -2.8616
 - 10%: -2.5668

Given that the ADF statistic (-7.6432) is much lower (more negative) than all the critical values (-3.4305 for 1%, -2.8616 for 5%, and -2.5668 for 10%), and the p-value is extremely small (1.87×10^{-11}), we can conclude that:

- The null hypothesis of a unit root (non-stationarity) is strongly rejected.

- The time series is stationary.

Even though our data are stationary, it is important to consider the sun’s 11-year cycle [8]. The sun is currently entering a more active phase, but our dataset is not extensive enough to capture the seasonality associated with this cycle. While it is possible to impute the missing data, we have chosen not to address this aspect in the present study. It is worth noting that our model could benefit from additional data, which would allow us to incorporate the sun’s seasonality in future analyses.

5 Features selection

5.1 Introduction

Feature selection is a critical component of the model building process, serving to identify the most impactful features while eliminating those that are redundant or irrelevant. This step not only enhances the accuracy of models but also reduces computational demands and simplifies the interpretation of the models. Feature selection improves model performance by reducing overfitting. This allows models to generalize better from training data to unseen data. Additionally, it reduces computational costs as fewer data points need to be processed, which is particularly beneficial when dealing with large datasets. By focusing on fewer, more relevant features, the model also becomes easier to understand and explain. There are several techniques for feature selection, each suited to different scenarios and objectives. Filter methods, for example, evaluate features based on their intrinsic properties, independent of any model. Techniques like the Correlation Matrix and Mutual Information are commonly used to assess the relevance of features. The Correlation Matrix helps identify redundant features through the Pearson correlation coefficients between pairs, while Mutual Information measures how much information the presence or absence of one variable informs about another, which is particularly useful for detecting informative features that are not linearly related to the target. Wrapper methods take a different approach by evaluating subsets of variables to determine their effectiveness in improving model performance. Forward selection and backward elimination are typical examples, where features are added or removed based on their contribution to model accuracy. Embedded methods integrate feature selection as part of the model training process and are specific to certain algorithms. Lasso Regression is one such method where feature selection is naturally incorporated through regularization. This technique adds a penalty equivalent to the absolute value of the coefficients, reducing some to zero, thereby selecting a simpler and more interpretable model. Determining the “best” feature selection technique depends on various factors including the characteristics of the dataset, the objectives of the modeling task, available computational resources, and the desired balance between performance and interpretability. Each method has its strengths and limitations, which makes it essential to consider the context in which they are applied. For instance, while filter methods are quick and efficient, particularly suitable for preliminary analyses, they may not capture interactions between features as effectively as wrapper methods, which directly optimize model performance but at a higher computational cost and potential risk of overfitting. Embedded methods like Lasso Regression, on the other hand, offer a balance by performing feature selection during the model training phase, making them suitable for scenarios where both feature relevance and model performance are equally important.

5.2 Filter Method

5.2.1 Correlation Plot

Correlation measures the linear relationship between two variables. It’s quantified by the Pearson correlation coefficient (r), which ranges from -1 to 1:

- $r = 1$: Indicates a perfect positive linear relationship.
- $r = -1$: Indicates a perfect negative linear relationship.
- $r = 0$: Indicates no linear relationship.

The formula for the Pearson correlation coefficient between two variables X and Y is:

$$r_{XY} = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}}$$

where:

X_i and Y_i are individual data points.

$\bar{X}$ and $\bar{Y}$ are the means of X and Y , respectively.

n is the number of data points.

In a correlation plot (also known as a correlation matrix), we calculate the correlation coefficients between pairs of features. This matrix helps identify pairs of features with strong linear relationships, which might be redundant for the model. Features with high absolute correlation values (close to 1 or -1) can be candidates for removal to avoid multicollinearity.

Another method is to use the Spearman Correlation, which measures the strength and direction of the association between two ranked variables. The Spearman correlation coefficient, denoted by ρ , is calculated using the formula:

$$\rho = 1 - \frac{6 \sum_{i=1}^n d_i^2}{n(n^2 - 1)}$$

where: d_i is the difference between the ranks of corresponding values of the two variables for the i th observation. n is the number of observations. Unlike Pearson's correlation, Spearman's correlation does not assume a linear relationship between the variables and is robust to outliers and non-normal data distributions, making it a useful measure for ordinal data or when the relationship is not linear. This method is preferred when we want to select variables to retain, as it helps identify strong associations. Conversely, Pearson's correlation is typically used when deciding which variables to exclude, as it is more sensitive to linear relationships and can highlight variables that may introduce multicollinearity or have weaker linear correlations with the target variable.

5.2.2 Mutual Information

Mutual Information (MI) measures the amount of information obtained about one variable through another. Unlike correlation, MI can capture non-linear relationships. It's based on the concept of entropy, which quantifies the uncertainty in a random variable.

The mutual information between two variables X and Y is defined as:

$$I(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right)$$

where:

$p(x, y)$ is the joint probability distribution of X and Y .

$p(x)$ and $p(y)$ are the marginal probability distributions of X and Y , respectively.

Mutual information is always non-negative and symmetric: $I(X; Y) = I(Y; X)$. A higher value indicates a higher dependency between variables.

Steps to Calculate Mutual Information:

1. Estimate Probability Distributions: Use methods like histograms, kernel density estimation, or other non-parametric techniques to estimate $p(x)$, $p(y)$, and $p(x, y)$.
2. Compute MI: Plug these estimates into the MI formula.

Analyse both techniques

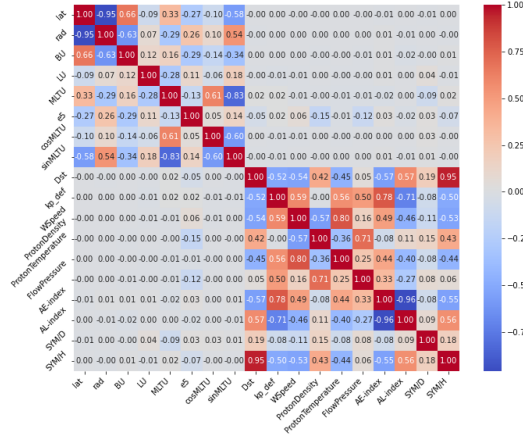


Figure 8: Correlation Plots of All Available Variables

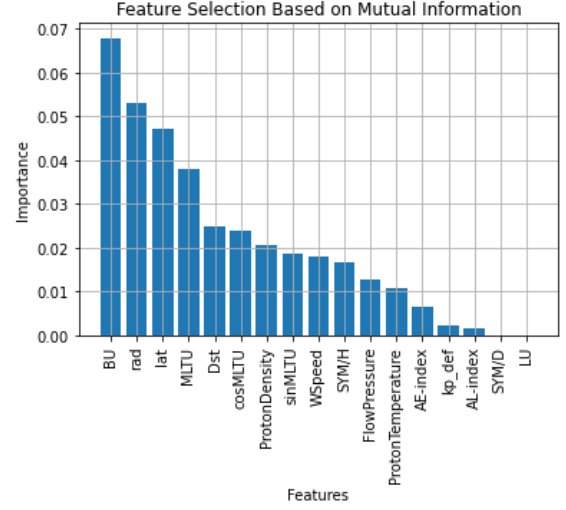


Figure 9: Mutual Information of All Available Variables

Utilizing correlation plots and mutual information bar charts, we can assess feature relevance in different contexts. The correlation plot highlights instances of multicollinearity, such as the strong negative correlation between 'rad' and 'lat', suggesting redundancy. By applying a correlation threshold, typically around 0.8, we can identify and exclude redundant features like (kp-def, LU) and (ProtonTemperature, FlowPressure), streamlining the feature set.

On the other hand, mutual information provides a lens into the nonlinear relationships between features and the target. For instance, despite 'lat' showing high negative correlation with 'rad', its mutual information indicates substantial unique predictive power. This highlights the necessity of considering both linear and nonlinear dependencies in feature selection, as mutual information reveals valuable features that correlation analysis might overlook.

The subset selected from the correlation was :

$$['log' + e, 'SYM/H', 'Dst', 'rad', 'BU', 'LU', 'MLTU', 'cosMLTU', 'sinMLTU', 'kp_def', 'SYM/D', 'WSpeed', 'ProtonDensity', 'ProtonTemperature', 'FlowPressure', 'AL-index'] \quad (1)$$

while the one selected by MI was :

$$['log' + e, 'SYM/H', 'Dst', 'rad', 'BU', 'MLTU', 'cosMLTU', 'sinMLTU', 'kp_def', 'WSpeed', 'ProtonDensity', 'ProtonTemperature', 'FlowPressure', 'AL-index', 'lat', 'AE-index'] \quad (2)$$

5.3 Wrappers Method

Forward Selection is a type of wrapper method used in feature selection. It involves iteratively adding features to a model based on their contribution to the model's performance. Here's a step-by-step explanation of the mathematics and process behind forward selection.

The process of building a model using a step-by-step approach begins with the initialization phase. Initially, the model starts with no features, essentially serving as an empty template ready for refinement.

From there, the process enters an iterative cycle. In each iteration, the model's performance is assessed by considering the potential impact of adding each remaining feature. This involves evaluating how the inclusion of a

new feature would enhance the model's overall performance. The feature that contributes the most to improving the model is then selected and added to the model. This cycle of evaluation and addition continues until no further significant improvements can be observed. Alternatively, the process may also stop when a predefined criterion is met, such as reaching a certain number of features or achieving a specific performance threshold.

The performance of the model throughout this process is evaluated using metrics that are appropriate for the type of problem being solved. For instance, in classification tasks, accuracy might be the chosen metric, while in regression problems, the coefficient of determination (R^2) could be used. The performance metric, denoted as $M(F)$, reflects how well the model performs with the current set of selected features F .

Algorithm 1 Forward Selection Algorithm

```

1: procedure FORWARDSELECTION
2:    $F_0 \leftarrow \emptyset$  ▷ Initial Step: Start with an empty set of selected features
3:   for  $k \leftarrow 1$  to  $n$  do ▷ Iteration  $k$ 
4:     for each feature  $f$  not in  $F_{k-1}$  do
5:       Calculate the model performance  $M(F_{k-1} \cup \{f\})$ 
6:        $f_k \leftarrow \arg \max_{f \notin F_{k-1}} M(F_{k-1} \cup \{f\})$ 
7:        $F_k \leftarrow F_{k-1} \cup \{f_k\}$  ▷ Update the set of selected features
8:       if stopping criterion is met then ▷ Stopping Criterion
9:         break ▷ Stop the process

```

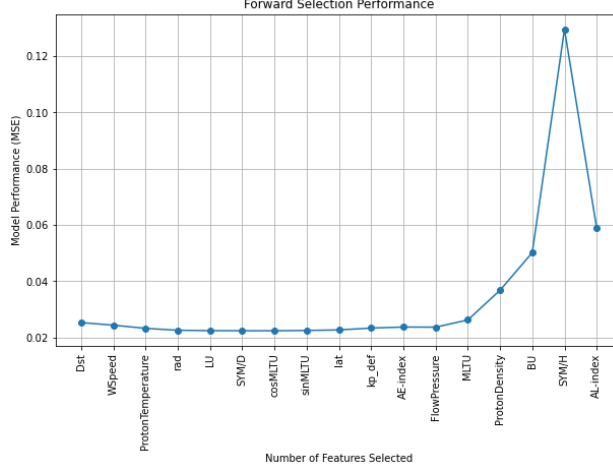


Figure 10: Forward Selection Performance

Through the application of forward selection, we observe the model's mean squared error (MSE) trends, which offer insights into feature importance over iterations. Initially, adding features significantly reduces MSE, showcasing their positive impact on model performance. However, the increase in MSE upon including features beyond a critical point suggests diminishing returns or potential overfitting. This is indicative of some features introducing noise rather than explanatory power, emphasizing the importance of stopping criteria in wrapper methods to avoid overfitting. For this method, our selected subset was:

$[log', SYM/H', SYM/D', rad', LU', cosMLTU', sinMLTU', BU', kp_def', AE-index', ProtonTemperature', lat']$

This subset was chosen after testing the differences between stopping at lat or at FlowPressure, which is at the elbow of the plot. At lat, we observe a slight uptick, and that was the deciding threshold.

5.4 Embedded Method

Lasso and Ridge Regression are embedded methods used for feature selection and regularization in linear regression. They add a penalty to the loss function to prevent overfitting and handle multicollinearity.

5.4.1 Ridge Regression (L2 Regularization)

Ridge regression adds a penalty based on the sum of the squares of the coefficients. This penalty also shrinks the coefficients, but it does so more uniformly, without driving them to zero. Ridge keeps all predictors in the model, making it a good choice when you believe that all variables have some influence on the outcome, even if their impact is small. Ridge is particularly useful when the predictors are highly correlated, as it stabilizes the model and makes it more robust. In order to do that Ridge Regression adds an L2 penalty (the sum of the squared coefficients) to the ordinary least squares (OLS) loss function. The Ridge loss function is:

$$J(\beta) = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2$$

where:

- y_i is the observed response.
- x_{ij} is the value of the j -th predictor for the i -th observation.

- β_0 is the intercept.
- β_j are the coefficients.
- λ is the regularization parameter controlling the penalty strength.

The primary objective in this process is to minimize the loss function, which is a measure of how well the model's predictions align with the actual outcomes. The loss function is defined as the sum of squared errors between the observed values (y_i) and the values predicted by the model. This prediction is determined by a linear combination of the features, weighted by their corresponding coefficients (β_j). Additionally, the loss function includes a regularization term, which penalizes the size of the coefficients, shrinking them towards zero to prevent overfitting. Mathematically, the goal is to find the coefficients β that minimize this combined function.

$$\min_{\beta} \left(\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 \right)$$

The impact of this regularization is twofold. Firstly, the shrinkage effect ensures that the coefficients are reduced towards zero, though they do not reach exactly zero. This means that all features remain in the model, but their influence is moderated, helping to prevent the model from becoming overly complex.

Secondly, the method addresses the issue of multicollinearity, where multiple features are highly correlated with each other. Ridge regression, the technique used here, mitigates this problem by retaining all the features but reducing the impact of those that are correlated. By doing so, it stabilizes the model and ensures that the presence of correlated features does not distort the results.

5.4.2 Lasso Regression (L1 Regularization)

On the other hand, Lasso regression adds a penalty based on the sum of the absolute values of the model's coefficients. This penalty has a unique effect: it can shrink some of the coefficients to exactly zero. As a result, Lasso not only helps in reducing overfitting but also performs feature selection by essentially removing irrelevant predictors from the model. This makes the resulting model simpler and easier to interpret, as it only includes the most important predictors. In order to do that, Lasso Regression adds an L1 penalty (the sum of the absolute values of the coefficients) to the OLS loss function. The Lasso loss function is:

$$J(\beta) = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

The terms have the same meaning as in Ridge regression, the goal is also the same mathematically, we want to minimize this function in order to find the optimal β .

Ridge Regression Solution

The solution to Ridge regression can be derived from the normal equations:

$$J(\beta) = \mathbf{y}^T \mathbf{y} - 2\mathbf{y}^T \mathbf{X}\beta + \beta^T \mathbf{X}^T \mathbf{X}\beta + \lambda \beta^T \beta$$

To minimize $J(\beta)$, we take the derivative with respect to β and set it equal to zero:

$$\frac{\partial J(\beta)}{\partial \beta} = -2\mathbf{X}^T \mathbf{y} + 2\mathbf{X}^T \mathbf{X}\beta + 2\lambda \beta = 0$$

Simplifying this, we get:

$$\mathbf{X}^T \mathbf{X}\beta + \lambda \beta = \mathbf{X}^T \mathbf{y}$$

Factoring out β gives:

$$(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})\beta = \mathbf{X}^T \mathbf{y}$$

Finally, we solve for β by multiplying both sides by the inverse of $(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})$:

$$\beta = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$$

In this solution:

- $\mathbf{X}$ is the design matrix, where each row corresponds to an observation and each column corresponds to a predictor.
- $\mathbf{I}$ is the identity matrix of the same size as $\mathbf{X}^T \mathbf{X}$, ensuring that the regularization term $\lambda \mathbf{I}$ has the appropriate dimensions.

The term $\lambda \mathbf{I}$ ensures that $\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}$ is invertible even when $\mathbf{X}^T \mathbf{X}$ is not, thus handling multicollinearity.

Lasso Regression Solution

Lasso does not have a closed-form solution and is typically solved using iterative algorithms such as coordinate descent or Least Angle Regression (LARS). The presence of the L1 penalty makes the optimization problem non-differentiable at zero, which encourages sparsity.

Analysis

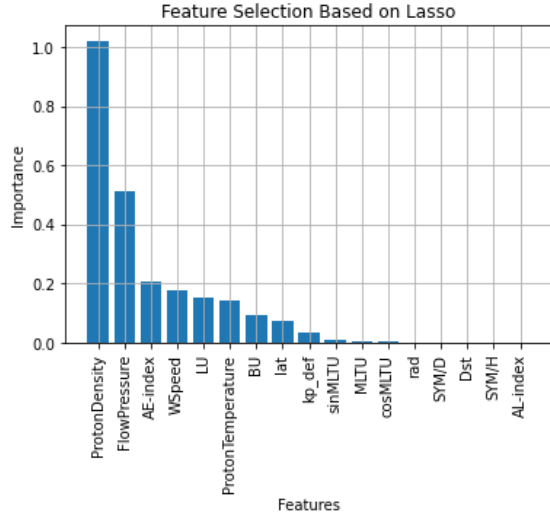


Figure 11: LASSO Regression

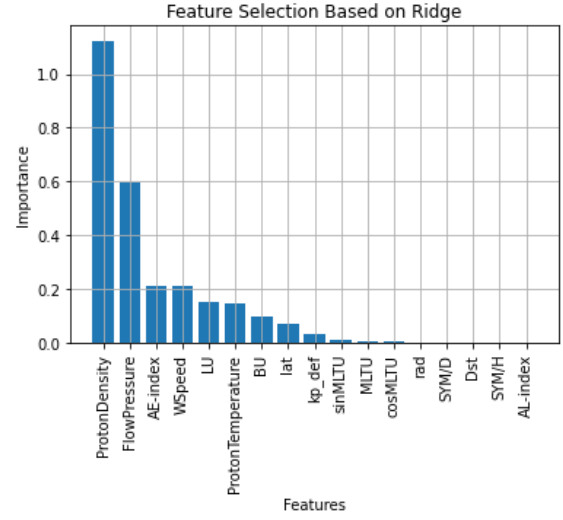


Figure 12: Ridge Regression

Lasso and Ridge regression analyses provide insights into how embedded methods inherently balance feature selection and regularization. Lasso's sparsity-promoting nature is evident as it zeroes out less relevant coefficients, simplifying the model by excluding features like SYM/D and AL-index, which show minimal importance. In contrast, Ridge regression, by maintaining all features with reduced coefficient magnitudes, allows for a nuanced understanding of feature contributions without excluding any outright. This method is valuable when model interpretability with all features retained is crucial. Both regularization techniques underscore the importance of features such as Proton-Density and FlowPressure, suggesting strong predictive relationships with the target. The consistent identification of these features across both methods reinforces their robustness as predictors.

Strategic Application:

Given the insights from these analyses, a dual approach using Lasso for initial coarse feature selection followed by Ridge refinement offers a strategic balance. This approach leverages Lasso's ability to reduce dimensionality effectively, while Ridge's capacity to handle multicollinearity among the retained features ensures a comprehensive model that minimizes overfitting.

The subset chosen for Lasso was :

$[log' + e, ProtonDensity', FlowPressure', WSpeed', AE-index', ProtonTemperature', lat', kp_{def}', sinMLTU']$

while the one selected for Ridge was :

$[log' + e, ProtonDensity', FlowPressure', WSpeed', AE-index', ProtonTemperature', lat', kp_{def}']$

This decision was made from the plot looking at the "elbow" of each barplots.

6 Models

6.1 LSTM

6.1.1 Origins and Development of LSTM Models

Long Short-Term Memory (LSTM) models are an advanced type of Recurrent Neural Network (RNN) engineered to address the problem of learning dependencies across long sequences of data. Traditional RNNs struggle with this due to the vanishing gradient problem—a phenomenon where the gradients of a loss function decay exponentially with the length of the input sequence during backpropagation. LSTMs overcome this by incorporating a complex system of gates and a cell state mechanism that regulates information flow, allowing the network to retain or discard information dynamically over time. LSTM models have a rich history that dates back to the 1990s, emerging from foundational research in neural networks and the desire to overcome the limitations faced by earlier models of recurrent neural networks (RNNs), LSTMs were first introduced in 1997 by Sepp Hochreiter and Jürgen Schmidhuber [9]. Their paper, "Long Short-Term Memory," was a response to the shortcomings of traditional RNNs.

Over the years, LSTMs have been refined and adapted for a wide range of applications. They have become a cornerstone technique in sequence modeling and prediction tasks, prominent in fields such as:

- Natural Language Processing (NLP): LSTMs have powered advancements in machine translation, speech recognition, and text generation.
- Time Series Prediction: They are extensively used in forecasting stock prices, energy demand, and weather conditions due to their ability to capture temporal dynamics.
- Music and Video Processing: LSTMs have been applied to generate music and analyze video content, where understanding the sequence is crucial.

But also other fields like OpenAI who beats the world champion on Dota 2, the precursor of League of Legends [10].

6.1.2 LSTM Architecture and Mathematical Formulations

The architecture of a Long Short-Term Memory (LSTM) network is designed to address some of the key challenges faced by traditional recurrent neural networks (RNNs), particularly the vanishing gradient problem. At the core of an LSTM unit are several components that work together to manage and retain information over time, ensuring that the network can learn and remember long-term dependencies in sequential data. The central component of an LSTM is the cell state (C_t), which serves as the unit's memory. This cell state carries relevant information throughout the sequence processing, acting as a conveyor belt that runs unchanged unless the LSTM decides to add or remove information. This capability makes the LSTM particularly effective at preserving long-term dependencies. To regulate the flow of information into and out of the cell state, LSTMs use gates, which are critical structures within each unit. These gates are controlled by sigmoid activation functions. These outputs determine the proportion of information that should be allowed through the gate. Each gate has a specific role:

- The forget gate (f_t) decides what information from the previous cell state (C_{t-1}) should be discarded. It evaluates the previous hidden state (h_{t-1}) and the current input (x_t), determining how much of the past information is relevant for the current step.
- The input gate (i_t) governs the addition of new information to the cell state. It generates new candidate values ($\tilde{C}_t$) by passing the input and hidden state through a tanh function, which is then filtered by the input gate to decide what should be added to the cell state.
- The output gate (o_t) determines which part of the updated cell state will influence the current hidden state (h_t) and, consequently, the output of the LSTM unit at the current time step.

Mathematically, these operations within the LSTM unit unfold in a structured sequence. The forget gate first computes how much of the previous cell state should be retained. The input gate then calculates the new information to be added to the cell state. The cell state is updated by a combination of these two processes: old information is partially retained (as determined by the forget gate), and new information is integrated (as guided by the input gate and candidate cell state). Finally, the output gate modulates the current cell state's information to produce the hidden state and the unit's output.

Mathematical Formulations

The operations within an LSTM cell during a single time step are defined by the following equations: The forget gate decides which parts of the cell state C_{t-1} are to be forgotten, using the previous hidden state h_{t-1} and the current input x_t .

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

The input gate controls the update values for the cell state by generating new candidate values $\tilde{C}_t$, which are filtered by the input gate's output.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

The cell state is updated by linearly interpolating between forgetting old information and adding new candidate values, allowing the LSTM to maintain a stable gradient over time.

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t$$

The output gate decides what part of the current cell state will contribute to the output at time t , modulated by the output of the sigmoid layer.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \times \tanh(C_t)$$

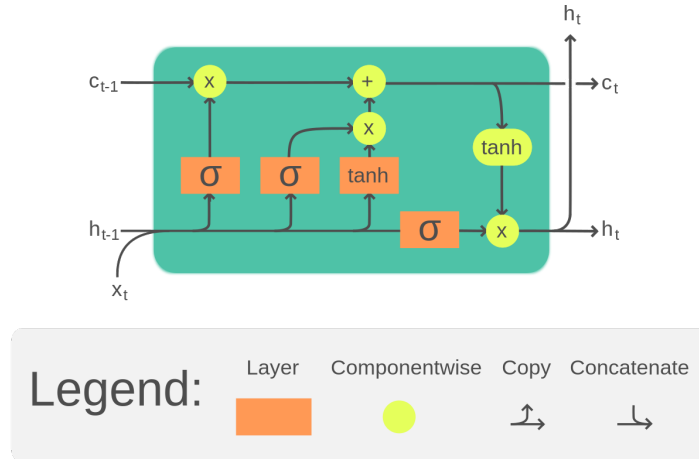


Figure 13: LSTM Cell Diagram

This diagram represents the internal structure and flow of information within an LSTM (Long Short-Term Memory) cell during one timestep.

6.2 CNN

6.2.1 Origins and Development of Convolutional Neural Network Models

The intellectual foundations of Convolutional Neural Networks (CNNs) can be traced back to the neocognitron, introduced by Kunihiko Fukushima in 1980 [11]. The neocognitron, a hierarchical, multilayered artificial neural network designed for handwritten character recognition, was the progenitor of the convolutional pooling structure. This early model drew inspiration from the structural organization of the visual cortex in biological systems, particularly how neurons respond hierarchically to visual stimuli.

In 1989, Yann LeCun [12] and collaborators significantly advanced these foundational ideas by integrating the backpropagation algorithm into such architectures for digit recognition tasks. This seminal work culminated in the development of LeNet-5 [13], a pioneering convolutional network that demonstrated effective application in visual data processing by optimizing the number of connections and parameters in comparison to contemporaneous fully connected networks.

Rapid Evolution and Technological Refinements

The adoption and evolution of CNNs accelerated with the advent of larger datasets and the availability of powerful GPU computing. A notable breakthrough occurred in 2012 with the introduction of AlexNet [14] by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by a substantial margin, AlexNet featured deeper layers and employed innovative techniques such as ReLU activations and dropout to address challenges like the vanishing gradient problem and overfitting.

Subsequent influential models include:

- **VGGNet (2014)**: Developed by Karen Simonyan and Andrew Zisserman[15], which deepened the architecture to 16-19 layers, simplifying it by employing uniformly small convolutional filters.
- **GoogLeNet/Inception (2014)**: Introduced by Christian Szegedy and colleagues from Google [16], featuring an inception module that dramatically reduced parameter counts through dimension reductions and sparse connections.
- **ResNet (2015)**: Developed by Kaiming He and others [17], this model introduced residual learning to facilitate training networks substantially deeper than previous architectures, demonstrating CNNs' capacity to efficiently train models with over 150 layers.

CNNs have achieved remarkable success, particularly in the field of computer vision, where they are foundational to nearly all recognition and detection systems, surpassing human-level performance in tasks such as image classification, object detection, and facial recognition. Their applications extend beyond vision, proving effective in domains such as audio processing and natural language processing (NLP), where they analyze sequential data and undertake complex tasks like sentiment analysis and machine translation.

In healthcare, CNNs have revolutionized diagnostic procedures, aiding in the identification of diseases from imaging data and interpreting radiological images, thereby enhancing the speed and accuracy of diagnostic tools.

Currently, CNNs are evolving to integrate into systems requiring real-time processing and decision-making, such as autonomous vehicles and real-time surveillance. Ongoing research aims to enhance CNN efficiency and reduce their dependency on large data sets through approaches like transfer learning, few-shot learning, and synthetic data generation.

Moreover, with the rise of explainable AI (XAI), there is an increasing necessity to understand and interpret how these models make decisions, especially in critical fields such as healthcare and autonomous driving, where safety and reliability are paramount.

Overall, CNNs represent a significant advancement in the ability of machines to understand and interpret the world visually. On another hand, Convolutional Neural Networks (CNNs), have also significantly impacted time series forecasting due to their ability to efficiently handle spatial data. This capability has been adapted for analyzing time-dependent data, leading to enhanced performance in various time series forecasting tasks.

Key Advantages in Time Series Forecasting

In the realm of time series forecasting, Convolutional Neural Networks (CNNs) offer several significant advantages that enhance their effectiveness. One of the primary benefits is automatic feature extraction. CNNs have the ability to autonomously identify and extract relevant features from raw time series data. This capability eliminates the need for manual feature engineering, which can be both labor-intensive and requires specialized domain knowledge. This is particularly advantageous in situations where the data displays complex or nonlinear patterns that traditional statistical methods struggle to capture.

Another notable advantage of CNNs is their ability to handle non-linear relationships. Many systems, especially in fields like economics, finance, and biology, exhibit non-linear dynamics. CNNs are inherently designed to model these non-linearities, making them more effective for forecasting tasks compared to traditional linear models, which often fall short in such scenarios.

Additionally, CNNs demonstrate efficiency when processing large datasets. Their architecture is optimized for handling substantial amounts of data, a common requirement in modern time series applications. This efficiency is further enhanced by advanced GPU computing, which allows for the rapid processing of extensive datasets within reasonable timeframes.

Application Areas

The versatility of CNNs extends across various domains, showcasing their applicability in numerous time series forecasting tasks. In the financial markets, CNNs are utilized for predicting stock prices, market indices, and economic indicators. The energy sector benefits from CNNs in forecasting demand and supply within power systems, as well as for predictive maintenance of energy infrastructure. In healthcare, these networks are employed to predict disease outbreaks or patient admissions based on historical healthcare data. Furthermore, in retail and supply chain management, CNNs play a crucial role in forecasting product demand, which aids in optimizing inventory management.

Future Prospects

The flexibility of CNNs suggests considerable potential for future applications in time series forecasting. Research continues to explore less data-intensive models, enhancing the ability of CNNs to perform with limited data through transfer learning and meta-learning approaches. Additionally, the integration of CNNs with other neural architectures, such as Recurrent Neural Networks (RNNs) and attention mechanisms, proposes promising improvements in forecasting accuracy and model interpretability.

In summary, the impact of CNN models on time series forecasting is profound, offering substantial improvements over traditional methods, particularly in handling complex and large-scale datasets. Their continued evolution and adaptation are likely to drive further advancements in this field, making them a cornerstone technology in predictive analytics.

6.2.2 CNN architecture and Mathematical Formulation

1. Convolutional Layers in Convolutional Neural Networks (CNNs):

Let's delve into the details, including the mathematical foundations and key parameters such as padding, stride, and other common options.

The convolution operation involves sliding a filter (also known as a kernel) across the input image to produce a feature map. Mathematically, for a 2D convolution, this can be expressed as:

$$(I * K)(i, j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(i+m, j+n) \cdot K(m, n)$$

where:

- I is the input image matrix.
- K is the kernel matrix of size $M \times N$.
- (i, j) represents the position in the output feature map.

Padding refers to the process of adding extra pixels around the border of the input image. It helps control the spatial size of the output feature map and retains information at the borders. Common types of padding include:

- (a) **Valid Padding (No Padding):** No extra pixels are added. The output size is smaller than the input size.

$$\text{Output size} = (W - K_w + 1, H - K_h + 1)$$

- (b) **Same Padding (Zero Padding):** Padding is added so that the output size is the same as the input size. For a stride of 1, the padding size P can be calculated as:

$$P = \lceil \frac{K_w - 1}{2} \rceil$$

$$\text{Output size} = (W - K_w + 2P + 1, H - K_h + 2P + 1)$$

Stride determines the number of pixels by which the filter moves at each step. A larger stride reduces the spatial dimensions of the output.

- **Stride of 1:** The filter moves one pixel at a time.
- **Stride of 2:** The filter moves two pixels at a time, effectively downsampling the input.

Mathematically, the output dimensions for stride S are given by:

$$\text{Output width} = \left\lfloor \frac{W - K_w + 2P}{S} + 1 \right\rfloor$$

$$\text{Output height} = \left\lfloor \frac{H - K_h + 2P}{S} + 1 \right\rfloor$$

The filter or kernel is a small matrix (e.g., 3x3, 5x5) that is convolved with the input. Multiple filters can be used to capture different features, resulting in multiple feature maps.

- **Number of filters:** The number of different filters applied in a convolutional layer determines the depth (number of channels) of the output feature map.

Mathematical Example

Consider an input image of size 6×6 , a 3×3 filter, padding of 1, and stride of 1:

- Input size: 6×6
- Kernel size: 3×3
- Padding: $P = 1$
- Stride: $S = 1$

The output dimensions are calculated as follows:

$$\text{Output width} = \left\lfloor \frac{6 - 3 + 2 \times 1}{1} + 1 \right\rfloor = 6$$

$$\text{Output height} = \left\lfloor \frac{6 - 3 + 2 \times 1}{1} + 1 \right\rfloor = 6$$

So, the output feature map will be 6×6 .

After the convolution operation, an activation function (like ReLU) is typically applied element-wise to introduce non-linearity:

$$\text{ReLU}(x) = \max(0, x)$$

2. ReLU Layer:

- **Purpose:** Introduces non-linearity to the network's function without affecting the scale of input values, essential for modeling complex patterns.
- **Function:** Defined as $f(x) = \max(0, x)$, it sets all negative values in the input to zero, maintaining positive values.

3. Pooling Layer:

- **Objective:** Reduces spatial dimensionality of the input feature maps, decreasing the computational load and model overfitting.
- **Max Pooling and Average Pooling:**

$$O_{ij} = \max_{(m,n) \in R} I_{mn} \quad \text{where } R \text{ is the region}$$

$$O_{ij} = \frac{1}{|R|} \sum_{(m,n) \in R} I_{mn} \quad \text{where } |R| \text{ is the count of elements in region } R.$$

4. Fully Connected Layer:

- **Role:** Synthesizes the learned features into predictions by connecting every neuron to all activations from the previous layer.
- **Mathematics:** Each neuron output is computed as $y = w^T x + b$, where x is the input vector, w is the weight matrix, and b is the bias.

5. Batch Normalization:

- **Process:** Standardizes the activations in a layer, normalizing the input x_i as:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sigma_B^2 + \epsilon}$$

Scaled and shifted by learned parameters γ and β , the output is $y_i = \gamma \hat{x}_i + \beta$.

6. Dropout:

- **Mechanism:** Randomly omits neurons during training with probability p , effectively reducing overfitting by promoting independent neuron activations.

7. Softmax/Sigmoid Layer:

- **Softmax Function:** Used in multi-class classifiers to convert logits z into probabilities:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_k e^{z_k}}$$

- **Sigmoid:** Applied in binary classification scenarios, transforming logits into a probability between 0 and 1.

8. Adapting CNNs for Time Series Forecasting:

- **One-Dimensional Convolution:** In 1D convolution, filters slide along the time axis, detecting time-dependent patterns within the data. This technique captures local temporal features effectively. The following graphic illustrates the application of a 1D convolutional filter (Kernel size = 2) over a time series data matrix (Shape = 4,4)

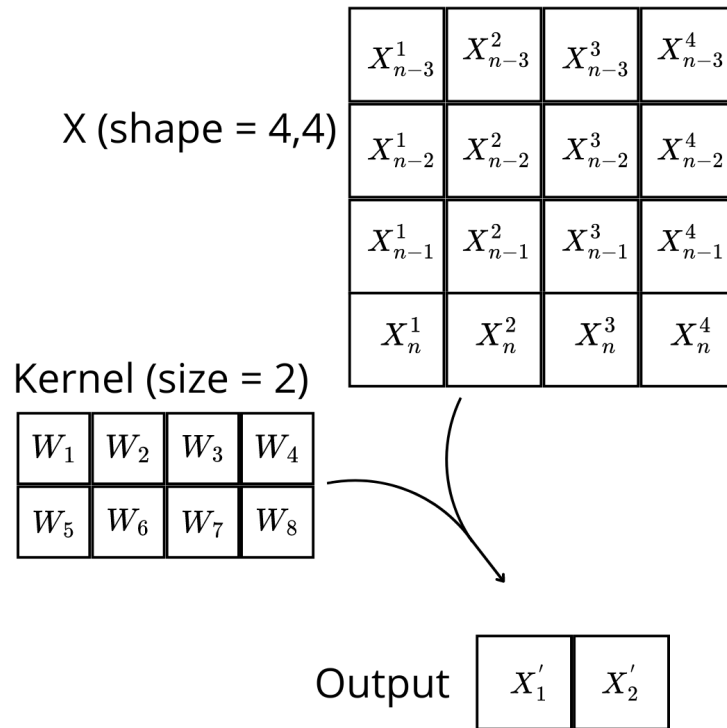


Figure 14: 1D Convolution

Key parameters are : **Kernel Size** = 2; **Stride** = 1; **Padding** = 0 (valid padding)

In the graphic, the kernel slides over the input time series data matrix with stride 1 and no padding, producing a smaller output by capturing the essential time-dependent features.

- **Two-Dimensional Convolution:** In 2D convolution, filters slide both horizontally and vertically over a two-dimensional input matrix. This method is typically used in image processing but can be adapted for time series with multiple features or lagged time steps. The output layers interpret and synthesize the extracted temporal features into outputs suitable for various tasks such as regression and classification. This ability allows CNNs to handle both local and global patterns in time series data effectively, making them versatile for a wide range of forecasting tasks.

The following graphic demonstrates the application of a 2D convolutional filter (Kernel shape = 2x2) over the same input matrix:

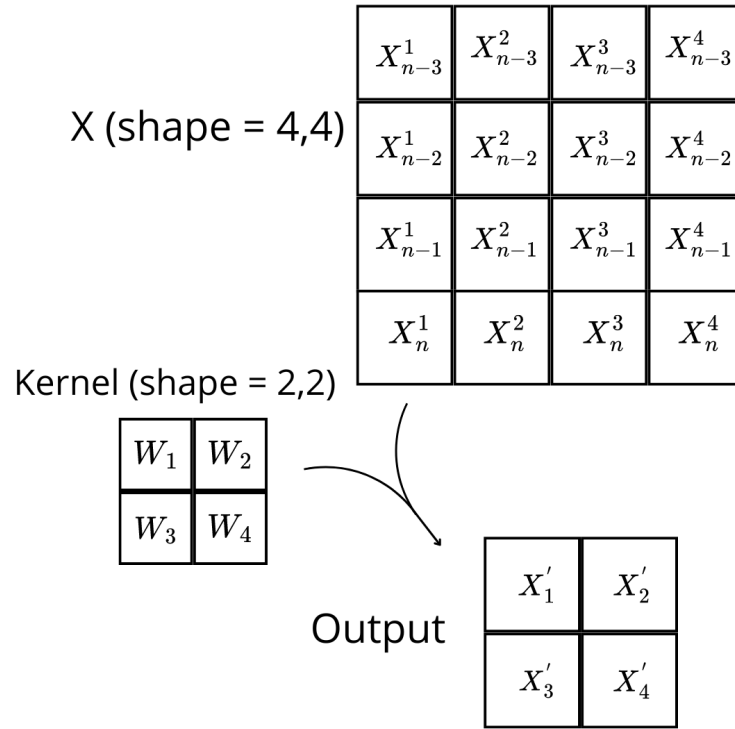


Figure 15: 2D Convolution

Key parameters are : **Kernel Shape** = 2x2; **Stride** = 1; **Padding** = 0 (valid padding)

In this graphic, the kernel processes the input matrix by moving across both dimensions with stride 1 and no padding, producing a smaller output matrix that encapsulates the spatial-temporal features of the input data.

7 Training of the Models

7.1 Hyperparameter tuning

Hyperparameter tuning is a pivotal step in the development of machine learning models, significantly impacting their performance and generalizability. In this part we explore the concepts of hyperparameter tuning space and methods, with a focus on Convolutional Neural Networks models. The discussion emphasizes the importance of defining an appropriate tuning space and the utilization of the Hyperband algorithm for efficient hyperparameter optimization.

In the realm of machine learning, hyperparameters are critical factors that influence the learning process and overall performance of models. Unlike model parameters, which are learned from data, hyperparameters are set prior to training and govern various aspects of the training process. Proper tuning of hyperparameters is essential for optimizing model performance and achieving robust predictions. Hyperparameter tuning involves searching for the optimal set of hyperparameters that result in the best model performance. The tuning process can be broadly categorized into two main components: the tuning space and the tuning method. Here we will only present the analysis of the hyperparameter tuning of the CNN in 1 dimension with the data of the e1 electron flux in L3-8 region. This choice was made for simplicity and avoid redundancy as the analysis follow the same principle for all three models and each datasets, the results will be put in annex.

7.1.1 Tuning space

The tuning space defines the range of possible values for each hyperparameter. An adequately defined tuning space is crucial as it impacts both the efficiency and effectiveness of the tuning process. Key hyperparameters for CNNs in time series forecasting include:

1. Number of Convolutional Layers: This refers to the depth of the CNN, which influences the model's ability to capture complex patterns in the data.
2. Number of Filters: This denotes the number of output channels in each convolutional layer, influencing the model's feature extraction capability.
3. Kernel Size: The size of the convolutional window, which affects the granularity of feature detection.
4. Number of Dense Layers: The count of fully connected layers, impacting the model's capacity to learn abstract representations.
5. Units in Dense Layers: This indicates the number of neurons in each dense layer, determining the model's learning capacity.
6. Learning Rate: The step size for the optimization algorithm, which affects the convergence speed and stability of the training process.

Defining an appropriate tuning space involves setting reasonable bounds and steps for each hyperparameter, ensuring a balance between exploration and computational feasibility.

7.1.2 Method

The tuning method refers to the algorithm used to explore the tuning space and identify the optimal hyperparameter configuration. Several strategies exist, including Grid Search, Random Search, Bayesian Optimization, and Hyperband. This document focuses on Hyperband due to its efficiency and effectiveness.[18]

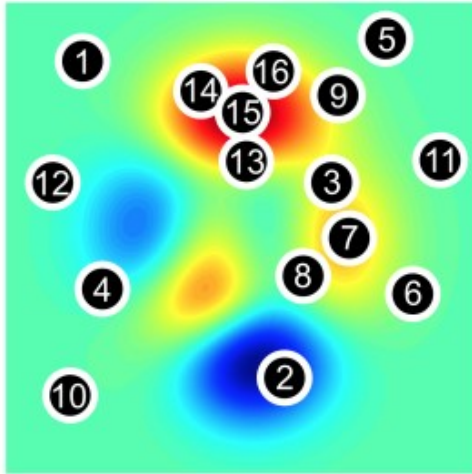
Hyperband Algorithm

Hyperband is an advanced hyperparameter optimization algorithm that combines the strengths of Random Search with adaptive resource allocation and early stopping. The key steps of the Hyperband algorithm are as follows:

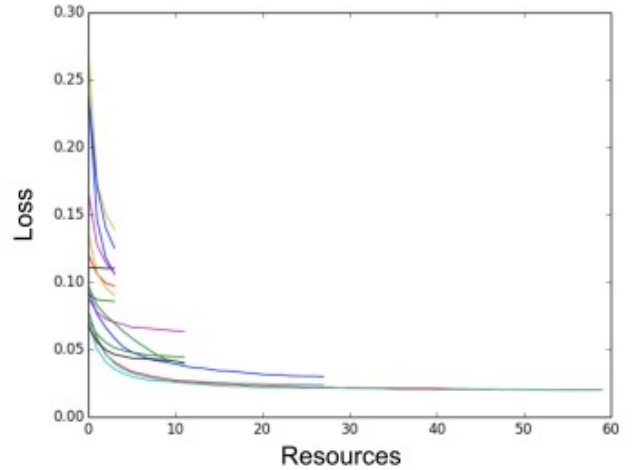
1. Initialization: Define the maximum resources (e.g., number of epochs) to allocate to each model configuration.
2. Bracketing: Divide the maximum resources into different brackets, each with varying configurations of hyperparameters.
3. Successive Halving: Iteratively train models, halving the number of configurations and doubling the resources allocated to the remaining configurations in each round.
4. Stopping: The process concludes when the maximum number of iterations is reached or when no further improvements are observed.

By dynamically allocating resources and focusing on promising configurations early in the training process, Hyperband achieves a balance between exploration of the hyperparameter space and exploitation of the best configurations.

The following plot[18], shows visually how hyperband acts :



(a) Configuration Selection



(b) Configuration Evaluation

Figure 16: (a) The heatmap shows the validation error over a two-dimensional search space with red corresponding to areas with lower validation error. Configuration selection methods adaptively choose new configurations to train, proceeding in a sequential manner as indicated by the numbers. (b) The plot shows the validation error as a function of the resources allocated to each configuration (i.e. each line in the plot). Configuration evaluation methods allocate more resources to promising configurations.

7.1.3 Analysis of Hyperparameter Tuning Process

After the computing time, we completed the hyperparameter tuning process to identify the most influential hyperparameters. In this process, we trained and tested 90 models to explore the hyperparameter search space defined by the following parameters:

- Convolutional Layers:
 - Number of layers: 1 to 3
 - Filters: 32 to 256, in steps of 32
 - Kernel size: 2 to 4
- Dense Hidden Layers:
 - Number of layers: 1 to 3
 - Neurons per layer: 64 to 512, in steps of 64
- Learning Rate: 1×10^{-3} to 1×10^{-4}

So, the total number of possible models is 16,844,032.

Our objective is to determine which hyperparameters have the most significant impact on model performance and use this information to create a balanced model.

To analyze the results of our hyperparameter search, we utilize three visualization techniques: parallel coordinates, a correlation matrix, and principal component analysis (PCA). These plots help us understand the relationships between hyperparameters and their effects on model performance.

Parallel Coordinates Plot

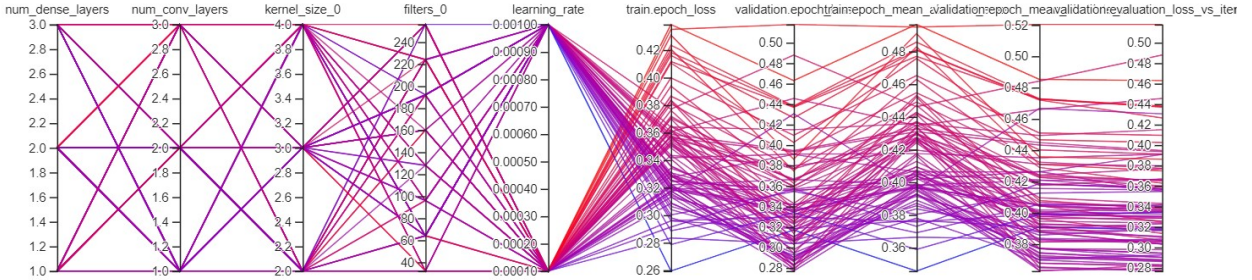


Figure 17: Parallel Coordinates Plot CNN1d e1 L3-8

The parallel coordinates plot provides a comprehensive view of hyperparameter configurations alongside their performance metrics. Each line represents a unique configuration, with color indicating metric performance—red indicating poorer performance and blue suggesting better results. After changing the hyperparameter order, we discerned that the learning rate notably influences model efficacy, favoring 1×10^{-3} over 1×10^{-4} . Despite its insights, the plot's limitations constrain further analysis.

Correlation Heatmap

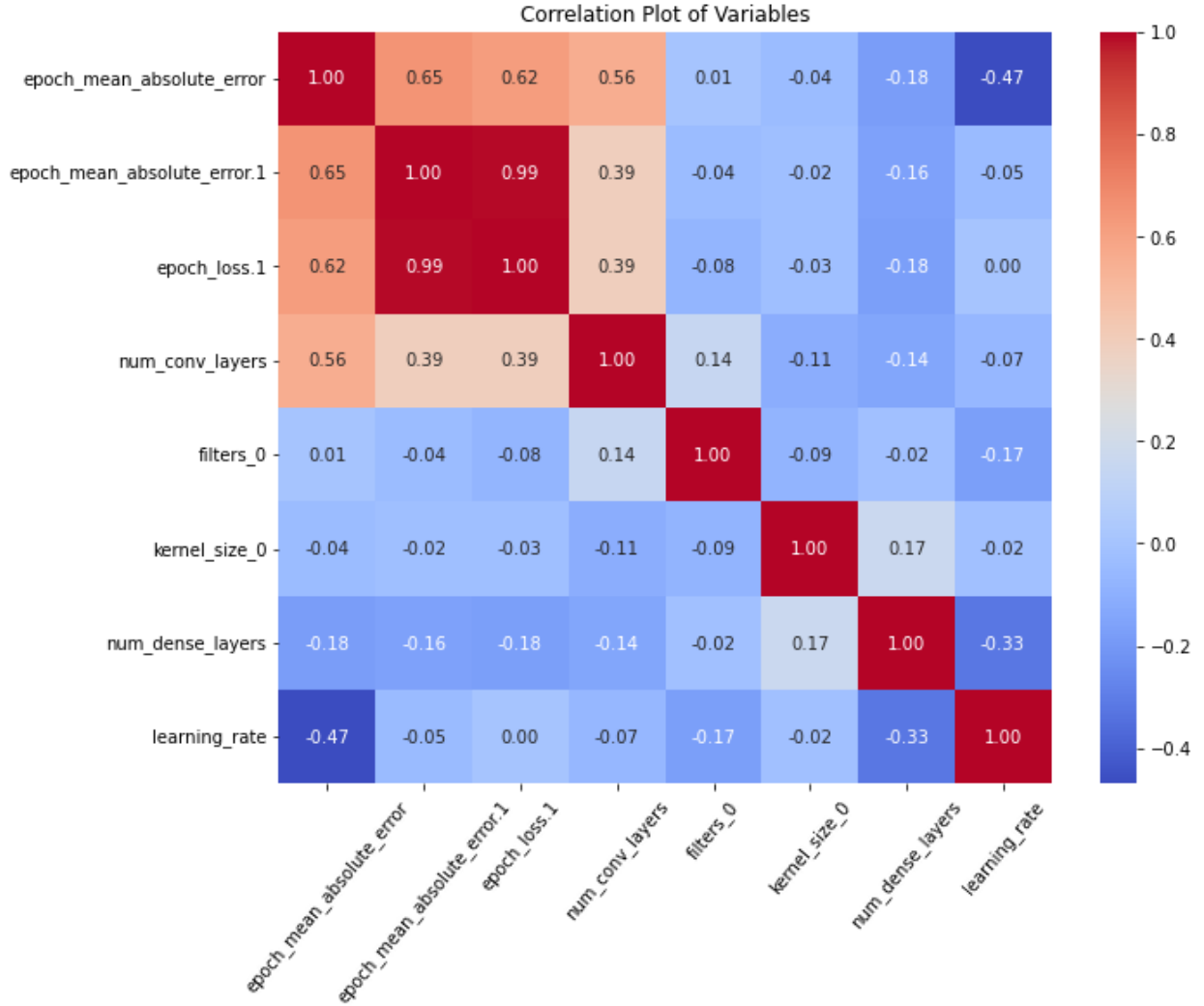


Figure 18: Correlation Plot CNN1d e1 L3-8

1. Performance Metrics:

The relationship between epoch metrics like mean absolute error (MAE) and epoch loss is extremely close, with a correlation of 0.99. This high correlation is expected since these metrics are calculated in a similar manner. The epoch loss is represented by the Mean Squared Error (MSE), which is calculated by averaging the squared differences between the predicted and actual values. On the other hand, the Mean Absolute Error (MAE) is computed by averaging the absolute differences between these values. Although MSE and MAE differ in how they treat the errors—MSE penalizes larger errors more severely—their fundamental calculations are similar enough to result in almost identical patterns over epochs.

When examining the relationship between the number of convolutional layers and the epoch mean absolute error, there is a moderate positive correlation of about 0.56. This suggests that as the number of convolutional layers increases, the model tends to produce higher errors. This could be due to the added complexity introduced by more convolutional layers, which might lead to overfitting or difficulties in optimizing the model effectively, especially if the model architecture becomes too deep relative to the available data.

The learning rate also plays a significant role in model performance, showing a moderate negative correlation

with the epoch mean absolute error, around -0.47. This indicates that higher learning rates are generally associated with better performance, likely because they allow the model to converge more quickly during training. However, it's important to balance this with the risk of overshooting the optimal minima, which can happen if the learning rate is too high. The negative correlation suggests that within a certain range, increasing the learning rate can help reduce errors more effectively, but careful tuning is required to avoid instability in the training process.

2. Hyperparameter Relationships:

The relationship between different hyperparameters in a machine learning model is crucial for fine-tuning and optimizing its performance. For instance, when examining the interaction between the number of convolutional layers and the number of filters in a neural network, there is a minimal correlation, specifically around 0.14. This low level of correlation suggests that these two parameters can be adjusted almost independently of each other. In other words, changing the number of convolutional layers is unlikely to significantly impact the optimal setting for the number of filters, allowing greater flexibility in tuning each parameter separately.

On the other hand, the relationship between the number of dense layers and the learning rate is characterized by a moderate negative correlation, approximately -0.33. This indicates that as the number of dense layers in the model increases, it may become necessary to reduce the learning rate to maintain optimal performance. The inverse relationship suggests that models with more complex architectures—those with additional dense layers—tend to benefit from smaller learning rates, likely due to the need for more precise adjustments during training to avoid overshooting the minima of the loss function.

PCA Biplot

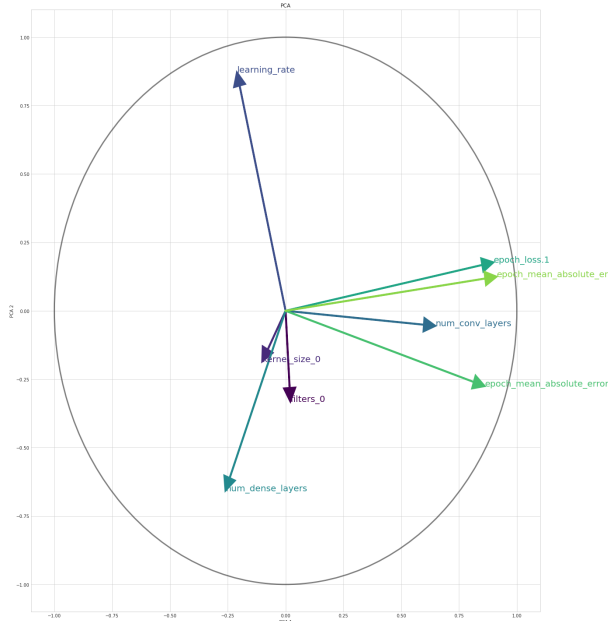


Figure 19: Variance explained

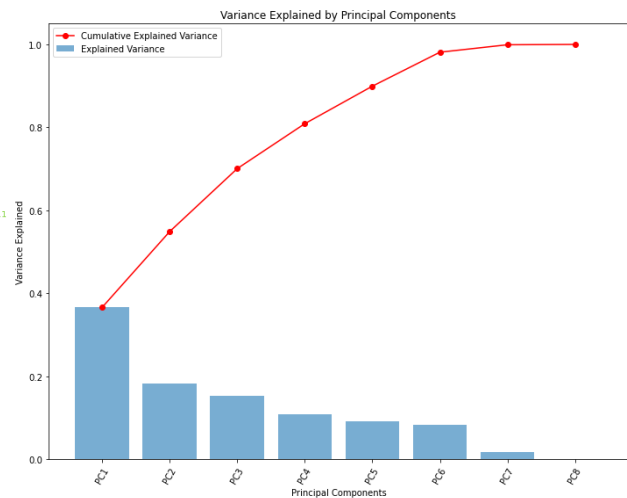


Figure 20: variable representation

The PCA biplot offers a visual representation of how hyperparameters and performance metrics project onto the first two principal components, revealing critical insights into their relationships and contributions.

In terms of hyperparameters, the biplot shows that the learning rate and the number of dense layers are oriented in different directions. This suggests that these two hyperparameters influence the principal components in distinct ways, likely contributing unique aspects to the model’s performance. On the other hand, the vectors representing the number of filters and kernel size are depicted as small arrows. The short length of these arrows indicates that these hyperparameters contribute minimally to the variance captured by the first two principal components. This minimal contribution implies that these hyperparameters may have a less significant impact on overall model performance compared to others.

Regarding performance metrics, the vectors for epoch loss and epoch mean absolute error are closely aligned. This alignment confirms the high correlation between these metrics, as previously observed in the heatmap analysis. The close alignment reflects that these metrics behave similarly and are likely influenced by similar factors during training.

The spread of vectors in the biplot also highlights the diverse influences of hyperparameters and performance metrics. The divergent directions of the vectors underscore the complexity of hyperparameter tuning, showing that different parameters and metrics impact the model in varied ways. This diversity makes it clear that a one-size-fits-all approach to tuning is insufficient, and instead, a nuanced, multi-faceted tuning strategy is required to optimize performance effectively.

In conclusion, this analysis reveals several critical aspects of hyperparameter tuning for Convolutional Neural Networks (CNNs). The complexity of relationships between hyperparameters and performance metrics suggests that careful attention is needed when tuning, particularly concerning the learning rate, which has a pronounced impact on performance. Additionally, while adding more convolutional and dense layers increases the model’s depth, this does not automatically result in better performance, indicating that model complexity should be balanced with the risk of overfitting.

Dimensionality reduction through PCA proves to be a valuable tool in understanding the variance within the hyperparameter space, allowing for more efficient tuning strategies. The use of visualizations, such as those generated through TensorBoard, is essential in tracking and optimizing the hyperparameter tuning process, providing a comprehensive view that guides more informed and effective decisions.

7.2 Final Models

7.2.1 Fine tuning

After conducting a thorough hyperparameter tuning analysis, I decided to investigate the learning rate further and fine-tune this hyperparameter in conjunction with the batch size. The learning rate and batch size are two critical hyperparameters that significantly impact the performance and training dynamics of neural networks.

The learning rate determines the size of the steps the optimizer takes during each iteration to minimize the loss function. If the learning rate is too high, the model might converge too quickly to a suboptimal solution or even diverge. Conversely, if the learning rate is too low, training can become excessively slow and may get stuck in local minima.

Batch size refers to the number of training samples used to calculate the gradient in each iteration. A smaller batch size provides a more noisy estimate of the gradient, which can help escape local minima and potentially lead to better generalization. However, smaller batches can slow down the training process and result in less stable updates. Larger batch sizes, on the other hand, offer more accurate gradient estimates and faster training but can require higher learning rates to avoid getting stuck in sharp minima.

There is a well-documented relationship between batch size and learning rate: as the batch size increases, the learning rate can typically be increased proportionally to maintain stable and efficient training. This is because larger batches reduce the variance of the gradient estimate, allowing for more aggressive updates.

To optimize these hyperparameters, I trained several models with different combinations of learning rates and batch sizes. By carefully analyzing their performance, I selected the model that demonstrated the best balance between convergence speed and generalization capability. The results, given in annex, shows that the best compromise was to choose a batchsize of 18 with a learning rate of 1e-4.

7.2.2 Features selected

We tested all subsets of selected features using various methods, and surprisingly, the differences were not large. The methods we employed focused solely on data science aspects without any prior assumptions, where feature selection is conducted without preconceived notions, allowing any feature to be potentially valuable. On the other hand, the physics team incorporated domain knowledges to do their subsets of features and was the best among the other showing the importance of collaborating work. However the results indicate that the features selection does not change a lot. This suggests that while collaboration with domain experts is beneficial, the improvements to the model are not substantial in this particular case. The CNN models are known to be very flexible and it is shown here. The best results were obtained using the work of the physics team, we will use the *PhysicsSubset* for all models :

$$[log' + e, LU', cosMLTU', sinMLTU', SYM/H', lat']$$

Method	RMSE	r
Physics Subset	0.2646	0.9114
Only Electron Flux	0.2660	0.9104
All Variables	0.2671	0.9094
Mutual Information	0.2677	0.9089
Ridge	0.2683	0.9085
Lasso	0.2683	0.9085
Forward	0.2676	0.9089
Differences Min/Max	1.3838%	0.3203%

Table 1: Comparison of Different Feature Selection Methods

7.2.3 Models Structure

CNN1d Best Model e5 L3-8

Layer (type)	Output Shape	Param #	Learning Rate: 1e-4
=====			
conv1d (Conv1D)	(None, 1, 46, 192)	3,648	Kernel Size: 3
batch_normalization (Batch Normalization)	(None, 1, 46, 192)	768	
max_pooling2d (MaxPooling2D)	(None, 1, 23, 192)	0	
dropout (Dropout)	(None, 1, 23, 192)	0	
flatten (Flatten)	(None, 4416)	0	
dense (Dense)	(None, 128)	565,376	
dense_1 (Dense)	(None, 192)	24,768	
dense_2 (Dense)	(None, 12)	2,316	
dropout_1 (Dropout)	(None, 12)	0	
=====			
Total Params: 596,876			
Trainable Params: 596,492			
Non-trainable Params: 384			

CNN1d Best Model e1 L3-8

Layer (type)	Output Shape	Param #	Learning Rate: 1e-4
=====			
conv1d (Conv1D)	(None, 1, 46, 96)	1,824	Kernel Size: 3
batch_normalization (Batch Normalization)	(None, 1, 46, 96)	384	
max_pooling2d (MaxPooling2D)	(None, 1, 23, 96)	0	
dropout (Dropout)	(None, 1, 23, 96)	0	
conv1d_1 (Conv1D)	(None, 1, 22, 224)	43,232	Kernel Size: 2
batch_normalization_1 (Batch Normalization)	(None, 1, 22, 224)	896	
max_pooling2d_1 (MaxPooling2D)	(None, 1, 11, 224)	0	
dropout_1 (Dropout)	(None, 1, 11, 224)	0	
flatten (Flatten)	(None, 2464)	0	
dense (Dense)	(None, 320)	788,800	
dense_1 (Dense)	(None, 384)	123,264	
dense_2 (Dense)	(None, 192)	73,920	
dense_3 (Dense)	(None, 12)	2,316	
dropout_2 (Dropout)	(None, 12)	0	
=====			
Total Params: 1,034,636			
Trainable Params: 1,033,996			
Non-trainable Params: 640			

We observe that the model for e1 is significantly more complex than the model for e5. This complexity is evident in the larger number of layers, higher parameter count, and additional operations involved in the e1 model. Specifically, the e1 model has almost double the number of parameters compared to the e5 model, indicating a substantial increase in its capacity to learn and represent data.

For instance, the e1 model includes multiple convolutional layers, batch normalization layers, and dense layers, contributing to its greater overall complexity. The increased number of parameters, from 596,876 in e5 to 1,034,636 in e1, highlights the significant expansion in the model's architecture.

Such behavior is not isolated to these two models alone; it is a recurring pattern observed across all models in

our study. Predicting e1 needs more complex architecture, those models tend to have a greater capacity for capturing intricate patterns in the data due to their deeper and more sophisticated architectures. This increased complexity, however, comes at the cost of higher computational requirements.

8 Comparison Of the model

8.1 Metrics

In the context of time series forecasting, evaluating the performance of predictive models is crucial. Different metrics offer various perspectives on the accuracy and reliability of the forecasts. Here, we delve into the mathematical and technical details of several commonly used metrics: RMSE, Correlation, Percentage Explained, Mean Signed Error (MSA), and Sum of Squared Prediction Bias (SSPB).

8.1.1 RMSE

Root Mean Squared Error (RMSE) is a widely used metric for measuring the accuracy of time series forecasts. It represents the square root of the average of the squared differences between the forecasted and actual values. RMSE is particularly useful because it penalizes larger errors more significantly, making it sensitive to outliers.

Mathematically, RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2}$$

where:

- y_t is the actual value at time t
- $\hat{y}_t$ is the forecasted value at time t
- n is the total number of observations

To understand RMSE further, consider the following breakdown:

1. **Squaring the Errors:** $(y_t - \hat{y}_t)^2$ ensures all differences are positive and emphasizes larger errors.
2. **Averaging the Squared Errors:** $\frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2$ provides a mean of these squared errors.
3. **Square Root:** Taking the square root of the average gives the RMSE its unit back, making it comparable to the original data.

8.1.2 Correlation

Correlation measures the strength and direction of the linear relationship between the actual and forecasted values. For time series forecasting, the Pearson correlation coefficient is commonly used. This coefficient ranges from -1 to 1, where values closer to 1 or -1 indicate a strong linear relationship, and values close to 0 indicate little to no linear relationship.

The Pearson correlation coefficient r is defined as:

$$r = \frac{\sum_{t=1}^n (y_t - \bar{y})(\hat{y}_t - \bar{\hat{y}})}{\sqrt{\sum_{t=1}^n (y_t - \bar{y})^2 \sum_{t=1}^n (\hat{y}_t - \bar{\hat{y}})^2}}$$

where:

- $\bar{y}$ is the mean of the actual values
- $\bar{\hat{y}}$ is the mean of the forecasted values

Steps to calculate Pearson correlation:

1. **Center the Data:** Subtract the mean from each actual and forecasted value.
2. **Product of Deviations:** Multiply the deviations of actual and forecasted values.
3. **Sum of Products:** Sum these products over all time points.
4. **Normalization:** Divide by the product of the standard deviations of actual and forecasted values.

8.1.3 Percentage Explained

Percentage Explained, often referred to as the coefficient of determination R^2 , represents the proportion of the variance in the actual values that is predictable from the forecasted values. It ranges from 0 to 1, where a value of 1 indicates a perfect fit, and a value of 0 indicates that the model does not explain any variability in the data.

$$R^2 = 1 - \frac{\sum_{t=1}^n (y_t - \hat{y}_t)^2}{\sum_{t=1}^n (y_t - \bar{y})^2}$$

Steps to calculate R^2 :

1. **Total Sum of Squares (TSS):** Measure the total variance in the actual data $\sum_{t=1}^n (y_t - \bar{y})^2$.
2. **Residual Sum of Squares (RSS):** Measure the variance that is not explained by the forecast $\sum_{t=1}^n (y_t - \hat{y}_t)^2$.
3. **Variance Explained:** Subtract the proportion of unexplained variance from 1.

8.1.4 Median Symmetric Accuracy (MSA)

Median Symmetric Accuracy (MSA) is a metric used to evaluate the accuracy of predictions relative to actual values, particularly useful for handling skewed data and outliers. The calculation involves first determining the log ratio between predicted ($\hat{y}$) and actual (y) values using the formula:

$$\text{log_ratio} = \log \left(\frac{\hat{y}}{y} \right)$$

Next, the absolute value of these log ratios is taken to focus on the magnitude of errors without considering their direction:

$$\text{abs_log_ratio} = \left| \log \left(\frac{\hat{y}}{y} \right) \right|$$

The median of these absolute log ratios is then computed to find a central tendency that is less affected by extreme values compared to the mean:

$$\text{median_abs_log_ratio} = \text{median} \left(\left| \log \left(\frac{\hat{y}}{y} \right) \right| \right)$$

Finally, this median log ratio is transformed back to a percentage scale:

$$\text{MSA} = 100 \times (\exp(\text{median_abs_log_ratio}) - 1)$$

The MSA provides a percentage that reflects the typical prediction error relative to actual values. An MSA of 0% indicates perfect predictions, where all predicted values exactly match the actual values. A lower MSA indicates better accuracy, with smaller typical errors. Since MSA uses the median of log ratios, it is robust to outliers, providing a more reliable measure of typical accuracy in datasets with extreme values.

8.1.5 Symmetric Signed Percentage Bias (SSPB)

The Symmetric Signed Percentage Bias (SSPB) indicates both the direction and magnitude of bias between two data sets. A positive SSPB means the predictions are generally higher than the actual values, while a negative SSPB means they are lower. The absolute value of SSPB shows the percentage difference; for example, +20% indicates predictions are 20% higher than actual values, and -20% indicates they are 20% lower. SSPB uses the median log

ratio, making it robust to outliers and symmetric, treating overestimations and underestimations equally. This metric effectively shows how much and in which direction predictions deviate from actual values in percentage terms.

$$\text{median_log_ratio} = \text{median} \left(\log \left(\frac{\hat{y}}{y} \right) \right)$$

$$\text{SSPB} = 100 \times \text{sign}(\text{median_log_ratio}) \times (\exp(|\text{median_log_ratio}|) - 1)$$

A lower SSPB indicates less cumulative bias in the forecasts, reflecting more accurate predictions.

By using these metrics collectively, we can gain a comprehensive understanding of the performance of time series forecasting models, addressing various aspects such as accuracy, linear relationship, variance explanation, bias, and cumulative error.

8.2 Results

8.2.1 log e5

	LSTM					CNN 2D					CNN 1D				
HS ahead	RMSE	r	PE	MSA	SSPB	RMSE	r	PE	MSA	SSPB	RMSE	r	PE	MSA	SSPB
1	0.192	0.942	0.882	28.863	1.675	0.181	0.946	0.895	25.335	-2.392	0.187	0.940	0.882	26.313	-2.156
2	0.205	0.931	0.865	30.422	-2.750	0.190	0.941	0.884	27.371	-2.740	0.197	0.933	0.870	28.391	-4.334
3	0.219	0.920	0.846	32.908	-2.222	0.205	0.930	0.865	30.562	-5.346	0.208	0.925	0.855	30.055	-5.514
4	0.227	0.914	0.835	34.575	-0.369	0.212	0.925	0.856	31.888	-5.166	0.214	0.920	0.846	31.753	-6.318
5	0.231	0.912	0.829	35.110	-0.883	0.223	0.919	0.840	33.994	-11.144	0.222	0.915	0.835	33.894	-9.470
6	0.241	0.904	0.814	36.579	0.916	0.223	0.917	0.840	33.794	-7.779	0.235	0.911	0.815	37.987	-18.842
7	0.245	0.900	0.808	37.137	0.208	0.229	0.913	0.832	35.462	-7.813	0.230	0.908	0.821	33.902	-4.535
8	0.248	0.899	0.803	39.039	-0.898	0.236	0.907	0.821	36.722	-9.189	0.233	0.905	0.818	35.311	-4.552
9	0.252	0.894	0.797	39.509	-2.925	0.239	0.904	0.817	36.934	-6.798	0.238	0.901	0.809	34.836	-0.497
10	0.257	0.889	0.788	40.214	0.272	0.244	0.900	0.809	37.710	-7.491	0.238	0.900	0.809	34.737	-4.267
11	0.259	0.887	0.784	40.175	0.824	0.243	0.900	0.810	38.902	-7.062	0.244	0.897	0.800	33.813	-1.049
12	0.255	0.890	0.792	38.457	-1.720	0.248	0.896	0.803	40.331	-8.688	0.242	0.898	0.802	34.758	-2.532
MEAN	0.236	0.907	0.820	36.082	-0.656	0.223	0.917	0.839	34.084	-6.801	0.224	0.913	0.830	32.979	-5.339

Table 2: E5 L3-8 prediction

In the results table here we observe that CNN 1D and CNN 2D models show slightly lower RMSE values compared to LSTM, indicating better performance in terms of prediction error. For the correlation r , all models show high correlation values, with CNN 1D having the highest mean correlation. CNN 1D shows the least variability in amplitude, followed by CNN 2D and LSTM. SSPB offer a surprising result with respect to the other, LSTM has the lowest bias, followed by CNN 2D and CNN 1D. In conclusion, while the CNN models generally perform better in terms of RMSE and correlation, the LSTM outperform both over the deviation with the prediction. CNN models tends to underestimate in their prediction.

8.2.2 log e1

	LSTM					CNN 2D					CNN 1D				
HS ahead	RMSE	r	PE	MSA	SSPB	RMSE	r	PE	MSA	SSPB	RMSE	r	PE	MSA	SSPB
1	0.235	0.930	0.863	29.752	1.528	0.215	0.941	0.885	25.791	-4.267	0.218	0.940	0.882	25.214	5.259
2	0.243	0.926	0.854	30.580	3.091	0.229	0.937	0.870	28.848	4.326	0.225	0.935	0.874	27.402	0.188
3	0.246	0.923	0.849	32.477	1.417	0.232	0.931	0.866	30.000	-4.866	0.238	0.931	0.860	29.776	9.076
4	0.257	0.916	0.836	34.852	3.153	0.241	0.928	0.856	31.564	-1.939	0.240	0.926	0.857	29.965	3.009
5	0.260	0.914	0.832	35.806	2.275	0.248	0.921	0.848	33.546	-6.522	0.245	0.923	0.851	31.735	3.574
6	0.267	0.910	0.823	36.705	4.021	0.252	0.919	0.843	34.706	-7.332	0.254	0.918	0.840	33.395	4.000
7	0.278	0.906	0.808	38.394	5.324	0.255	0.917	0.838	35.436	-4.925	0.259	0.914	0.834	33.606	3.832
8	0.275	0.904	0.812	39.658	2.658	0.261	0.913	0.831	36.618	-7.093	0.262	0.912	0.830	33.500	-3.170
9	0.278	0.900	0.808	40.336	-0.172	0.267	0.908	0.823	37.408	-5.587	0.265	0.909	0.825	34.522	0.321
10	0.290	0.894	0.791	41.503	3.759	0.269	0.906	0.820	36.803	-5.408	0.270	0.905	0.819	37.469	-1.577
11	0.297	0.891	0.781	40.453	8.618	0.279	0.899	0.807	39.157	-5.039	0.275	0.902	0.812	39.386	-8.554
12	0.292	0.892	0.788	41.392	6.945	0.281	0.898	0.804	39.984	-4.038	0.277	0.901	0.809	38.045	-8.240
MEAN	0.268	0.909	0.820	36.826	3.551	0.252	0.918	0.841	34.155	-4.391	0.252	0.918	0.841	32.835	0.643

Table 3: E1 L3-8 prediction

For the prediction over the e1 channel, we observe similar behavior: CNN models outperformed LSTM models in all metrics except SSPB, where there is a notable bias across all models. Specifically, LSTM models tend to overestimate, while 2D CNN models tend to underestimate, both with nearly the same degree of error. Interestingly, for the 1D CNN model, we notice distinctive patterns: the last values and the eighth value are particularly challenging to predict, resulting in a lower RMSE but also a negative bias. Additionally, the model tends to overestimate during the initial hours of prediction.

8.2.3 Comparison with the physics team results

Hs ahead	e1					e5				
	RMSE	r	PE	MSA	SSPB	RMSE	r	PE	MSA	SSPB
1	0.241	0.926	0.856	32.7	-3.1	0.223	0.917	0.840	34.8	-3.6
2	0.260	0.915	0.833	34.2	-2.0	0.235	0.907	0.823	38.1	-4.3
3	0.256	0.917	0.838	35.0	-3.0	0.238	0.905	0.818	37.1	-0.0
4	0.277	0.904	0.810	37.8	-0.8	0.237	0.906	0.820	37.0	-2.5
5	0.264	0.911	0.827	37.0	-0.4	0.247	0.899	0.805	39.1	-1.3
6	0.275	0.902	0.812	40.7	-5.7	0.259	0.892	0.796	44.3	1.1
7	0.287	0.893	0.796	41.6	-4.7	0.265	0.881	0.774	44.3	-0.8
8	0.286	0.893	0.797	42.8	-6.9	0.270	0.876	0.767	44.1	-3.7
9	0.300	0.883	0.776	43.4	-4.3	0.273	0.874	0.761	44.7	-0.0
10	0.289	0.890	0.792	41.2	-1.6	0.280	0.860	0.754	46.0	2.3
11	0.297	0.885	0.781	43.1	-4.7	0.279	0.865	0.767	46.2	1.1
12	0.299	0.882	0.778	43.4	-4.3	0.281	0.866	0.747	45.3	-0.8
MEAN	0.278	0.900	0.808	39.408	-3.458	0.257	0.887	0.789	41.75	-1.042

Table 4: Performance metrics for e1 and e5 with their LSTM models

Overall, this master’s thesis successfully achieved its goal of enhancing the work of the physics team, resulting in a slight but meaningful improvement in their model. During the process, we identified an opportunity to enhance the initial model by addressing data leakage during the normalization phase. While this leakage had a minor impact on the results, leading to a slight overestimation of the model’s predictive capabilities, we observed an improvement in the performance metrics in the new version we developed. The extent of the overestimation wasn’t precisely measurable due to different inputs, but the updated model reflects a more robust approach.

Our findings underscore the importance of rigorous data handling and validation techniques in scientific modeling. Addressing data leakage is crucial for ensuring the accuracy and reliability of predictive models. This experience highlights the value of interdisciplinary collaboration, as the combined expertise of physicists and statisticians can lead to more robust and effective solutions.

By integrating statistical rigor with physical modeling, both disciplines can benefit. The physics faculty gains improved model accuracy and reliability, while the statistics faculty gains practical applications for theoretical techniques. This synergy can foster innovative approaches and drive advancements in both fields.

9 Further Amelioration

For further enhancement in this subject, several avenues can be pursued. One critical improvement would be to incorporate more data, specifically extending the dataset to cover a much longer period. This extension would allow for a more comprehensive evaluation of the influence of the sun’s 11-year cycle on the observed phenomena. By encompassing multiple solar cycles, the model could better account for long-term periodicities and improve its predictive accuracy.

The Convolutional Neural Network (CNN) in two dimensions (CNN2D) presents a promising model for this task due to its ability to capture spatial hierarchies in data. However, the CNN2D model comes with a substantial increase in computational demands. Optimizing the numerous hyperparameters associated with CNN2D is resource-intensive, and in this study, it was not feasible to fully explore the potential of this model due to hardware limitations. Access to more powerful computational resources would enable the comprehensive tuning and training of the CNN2D model, potentially yielding significant performance improvements.

Another promising direction would be to experiment with hybrid models that combine the strengths of different neural network architectures. For example, integrating a Long Short-Term Memory (LSTM) network with a CNN, where the CNN serves as an encoder to extract features and the LSTM captures temporal dependencies, could leverage the strengths of both models. The CNN would efficiently handle spatial feature extraction, while the LSTM would excel in modeling the sequential nature of the data.

However, implementing such hybrid models would necessitate even greater computational resources. The complexity of training a CNN followed by an LSTM is substantial, and it would likely require access to high-performance computing facilities to achieve optimal results.

In summary, future research could focus on extending the temporal scope of the data, leveraging more advanced computational resources to fully exploit sophisticated models like CNN2D, and exploring hybrid architectures that combine CNNs and LSTMs. These enhancements hold the potential to significantly improve model performance and provide deeper insights into the influence of solar cycles on the data.

10 Conclusion

In this master thesis, I aim to delve deeper into the differences between Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTMs), with a particular focus on their impact on model fitting, computational efficiency, and broader implications such as usability and environmental sustainability. One of the most significant distinctions between CNNs and LSTMs lies in their model fitting times and computational resource requirements. To investigate these differences, I conducted experiments using my personal computer, which features an AMD Ryzen 7 processor optimized for enhanced performance, 8GB of RAM, and an 8-core CPU. Importantly, I opted not to use parallelized coding techniques or a GPU to ensure that the code remains accessible and user-friendly for a broader audience. During my experiments, I observed that fitting the LSTM model took approximately 6 minutes per run. LSTMs, designed for handling sequential data, involve complex computations and multiple gates (input, forget, and output gates) which contribute to their longer training times. In contrast, the 2-dimensional CNN model, commonly used for image data, took about 4 minutes per run. CNNs utilize convolutional layers to automatically and adaptively learn spatial hierarchies, which are computationally efficient for tasks involving grid-like topology data. Furthermore, the 1-dimensional CNN model, used for sequential data similar to LSTMs but with a simpler architecture, required only 3 minutes per run. This significant reduction in training time highlights the efficiency of CNNs in handling similar tasks more rapidly than LSTMs. The longer fitting times and higher computational demands of LSTM models pose significant challenges to their usability. During extended training sessions, the high computational load required by LSTMs caused my system to become unstable, leading to occasional crashes. This instability underscores the practical difficulties of using LSTMs on standard personal computers without advanced hardware. Moreover, the necessity for prolonged training times and the risk of system crashes make LSTMs less accessible for users who do not have access to high-performance computing resources. In contrast, CNNs, with their faster fitting times and lower computational demands, are more user-friendly and can be effectively utilized on more modest systems. As artificial intelligence becomes increasingly integral to various aspects of modern life, the environmental impact of different models gains importance. The computational intensity of LSTMs translates to higher energy consumption, which has several implications. The extended training times of LSTMs result in higher electricity usage, contributing to a larger carbon footprint. Given the growing concern about climate change and the push towards more sustainable technologies, this is a crucial factor. The need for more energy-efficient models is evident. CNNs, with their reduced training times and lower energy requirements, offer a more sustainable alternative. By minimizing energy consumption, CNNs help reduce the overall environmental impact of AI applications. The findings of this thesis highlight the clear advantages of CNN models over LSTMs in terms of performance, usability, and environmental sustainability. The efficiency of CNNs not only makes them more practical for everyday use but also aligns with the broader goal of reducing the environmental impact of technology. Given these insights, it is imperative to further develop and optimize CNN models for a wide range of applications. While LSTMs have their strengths, particularly in handling specific types of sequential data, the overall benefits of CNNs make them a superior choice for many tasks. Future research should continue to explore innovative ways to enhance the capabilities of CNNs, ensuring that they remain at the forefront of AI development while also addressing the critical need for sustainability in technology.

References

- [1] TERESA NIEVES-CHINCHILLA. How nasa tracked the most intense solar storm in decades, 2024.
- [2] Xin Huang, HuaNing Wang, and XingHua Dai. Influences of misprediction costs on solar flare prediction. *Science China Physics, Mechanics and Astronomy*, 55:1956–1962, 2012.
- [3] Edith Botek, Viviane Pierrard, and Alexandre Winant. Prediction of radiation belts electron fluxes at a low earth orbit using neural networks with proba-v/ept data. *Space Weather*, 21(7):e2023SW003466, 2023.
- [4] Viviane Pierrard, E Botek, J-F Ripoll, and G Cunningham. Electron dropout events and flux enhancements associated with geomagnetic storms observed by proba-v/energetic particle telescope from 2013 to 2019. *Journal of Geophysical Research: Space Physics*, 125(12):e2020JA028487, 2020.
- [5] Viviane Pierrard, G Lopez Rosson, K Borremans, Joseph Lemaire, J Maes, S Bonnewijn, E Van Ransbeeck, E Neefs, Mathias Cyamukungu, Sylvie Benck, et al. The energetic particle telescope: first results. *Space Science Reviews*, 184:87–106, 2014.
- [6] Bingxian Luo, Xinlin Li, M Temerin, and Siqing Liu. Prediction of the au, al, and ae indices using solar wind parameters. *Journal of Geophysical Research: Space Physics*, 118(12):7683–7694, 2013.
- [7] Jireh Yi-Le Chan, Steven Mun Hong Leow, Khean Thye Bea, Wai Khuen Cheng, Seuk Wai Phoong, Zeng-Wei Hong, and Yen-Lin Chen. Mitigating the multicollinearity problem and its machine learning approach: a review. *Mathematics*, 10(8):1283, 2022.
- [8] David H Hathaway. The solar cycle. *Living reviews in solar physics*, 12(1):4, 2015.
- [9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [10] Jose M Font and Tobias Mahlmann. Dota 2 bot competition. *IEEE Transactions on Games*, 11(3):285–289, 2018.
- [11] Kuniyiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.
- [12] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.
- [13] Guangfen Wei, Gang Li, Jie Zhao, and Aixiang He. Development of a lenet-5 gas identification cnn structure for electronic noses. *Sensors*, 19(1):217, 2019.
- [14] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.
- [15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [16] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [18] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.
- [19] Viviane Pierrard. *Effects of the Sun on the Space Environment of the Earth*. Presses Universitaires de Louvain, 2024.

- [20] D.N. Baker. What is space weather? *Advances in Space Research*, 22(1):7–16, 1998. Solar-Terrestrial Relations: Predicting the Effects on the Near-Earth Environment.
- [21] F Siciliano, Giuseppe Consolini, Roberta Tozzi, M Gentili, Fabio Giannattasio, and Paola De Michelis. Forecasting sym-h index: A comparison between long short-term memory and convolutional neural networks. *Space Weather*, 19(2):e2020SW002589, 2021.
- [22] Sidra Mehtab and Jaydip Sen. Analysis and forecasting of financial time series using cnn and lstm-based deep learning models. In *Advances in Distributed Computing and Machine Learning: Proceedings of ICADCML 2021*, pages 405–423. Springer, 2022.
- [23] Jiasheng Cao and Jinghan Wang. Stock price forecasting model based on modified convolution neural network and financial time series analysis. *International Journal of Communication Systems*, 32(12):e3987, 2019.
- [24] Alexiei Dingli and Karl Sant Fournier. Financial time series forecasting-a deep learning approach. *International Journal of Machine Learning and Computing*, 7(5):118–122, 2017.
- [25] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.
- [26] Ioannis E Livieris, Emmanuel Pintelas, and Panagiotis Pintelas. A cnn-lstm model for gold price time-series forecasting. *Neural computing and applications*, 32:17351–17360, 2020.
- [27] Jason Brownlee. How to develop convolutional neural network models for time series forecasting, 2020.
- [28] Wikipedia. Van allen radiation belt, 2024.
- [29] Graf Mathieu. Times-series-for-space-weather-forecasting-royal-belgian-institute-for-space-aeronomy; <https://github.com/mattmousse/times-series-for-space-weather-forecasting-royal-belgian-institute-for-space-aeronomy>, 2024.

11 ANNEX

Variable	Mean	Variance	Median	Mode	Std Dev	Min	Max	Range	Q1	Q3	IQR	Nb of Outliers	Skewness	Kurtosis
lat	-4.9	1395.9	-8.5	-77.4	37.4	-77.4	69.5	146.9	-37.7	23.9	61.6	0.0%	0.2	-1.2
rad	7195.1	22.0	7195.4	7189.2	4.7	7183.1	7204.6	21.5	7191.7	7198.7	7.0	0.0%	-0.2	-0.7
BU	0.3	0.0	0.3	0.4	0.0	0.2	0.4	0.2	0.3	0.4	0.1	0.0%	-0.4	-0.4
LU	4.0	0.1	3.9	2.0	0.3	2.0	7.6	5.6	3.8	4.1	0.3	2.9%	0.2	6.4
MLTU	14.9	13.8	15.5	10.8	3.7	0.6	23.8	23.2	12.1	17.4	5.3	0.0%	-0.2	-0.5
e5	79.2	6419.4	52.7	0.0	80.1	0.0	723.8	723.8	20.4	111.6	91.2	4.8%	1.7	3.5
cosMLTU	0.0	0.2	0.0	-0.9	0.4	-1.0	1.0	2.0	-0.3	0.3	0.6	0.0%	-0.1	-0.2
sinMLTU	0.0	0.1	0.0	0.3	0.3	-0.9	0.9	1.8	-0.2	0.2	0.4	0.3%	-0.1	-0.2
Dst	-9.6	290.1	-6.5	-4.0	17.0	-217.2	50.1	267.3	-16.9	1.0	17.9	3.9%	-2.1	11.5
kp_def	1.8	1.7	1.7	0.3	1.3	0.0	8.3	8.3	0.7	2.7	2.0	0.7%	0.8	0.4
WSpeed	436.9	9570.9	414.8	370.7	97.8	259.2	817.7	558.5	361.5	497.4	135.9	0.7%	0.8	-0.1
ProtonDensity	6.9	27.8	5.4	4.2	5.3	0.3	70.3	70.0	3.6	8.4	4.8	5.8%	3.2	17.7
ProtonTemperature	95700.7	8025240180.4	70566.7	14267.5	89583.7	5584.8	2504435.0	2498850.2	36392.4	127943.9	91551.5	4.2%	4.6	63.3
FlowPressure	2.4	2.5	2.0	1.3	1.6	0.2	47.3	47.1	1.4	2.8	1.4	5.5%	4.4	50.5
AE-index	184.6	39187.2	104.3	57.0	198.0	6.8	1842.6	1835.8	49.7	250.4	200.7	6.5%	2.0	4.9
AL-index	-117.7	22216.1	-54.1	-24.0	149.1	-1579.6	12.3	1591.9	-161.2	-20.9	140.3	7.2%	-2.3	7.3
SYM/D	-0.3	9.9	-0.2	-1.0	3.2	-36.7	30.2	66.9	-1.9	1.3	3.2	4.6%	-0.6	8.4
SYM/H	-11.1	290.2	-8.3	-5.0	17.0	-217.7	58.9	276.6	-18.4	-0.6	17.8	4.0%	-2.0	10.6

Table 5: Summary Statistics of Variables

Variables Metrics for observation

Observations and Surprising Aspects

Latitudinal Distribution and Variability Latitude shows a wide range of variability, indicating the extensive spatial coverage of the dataset. The distribution's skewness and kurtosis suggest a relatively flat distribution with a long tail towards higher latitudes.

Radiation Dose Rate (rad) The radiation dose rate exhibits minimal variance and skewness, indicating a remarkably stable measurement across the dataset. This stability is surprising given the dynamic nature of space weather phenomena.

Burst Units (BU) and Load Units (LU) Both BU and LU show very low variance and skewness, but the notable difference in their ranges and mean values indicates different scales of measurement or units of analysis.

MLTU and its Trigonometric Transformations The representation of magnetic local time using its cosine and sine forms is noted. Both transformations have similar ranges and IQRs, indicating a symmetrical distribution around their mean.

Solar Wind Speed (WSpeed) Shows a wide range and a high standard deviation, reflective of the dynamic and fluctuating nature of solar wind speeds. The slight positive skewness indicates more frequent high-speed outliers.

Proton Density and Proton Temperature Both variables display significant skewness and kurtosis, with proton temperature showing particularly high values. These metrics suggest a highly skewed distribution with a heavy tail.

Geomagnetic Indices These indices exhibit a range of skewness and kurtosis values, with some showing notable negative skewness, indicating a tendency towards extreme negative disturbances.

Conclusion The statistical analysis reveals a complex interplay of distributions, variances, and extremities in space weather variables. These insights are pivotal for enhancing the predictive models of space weather phenomena.

CNN 1D – E1

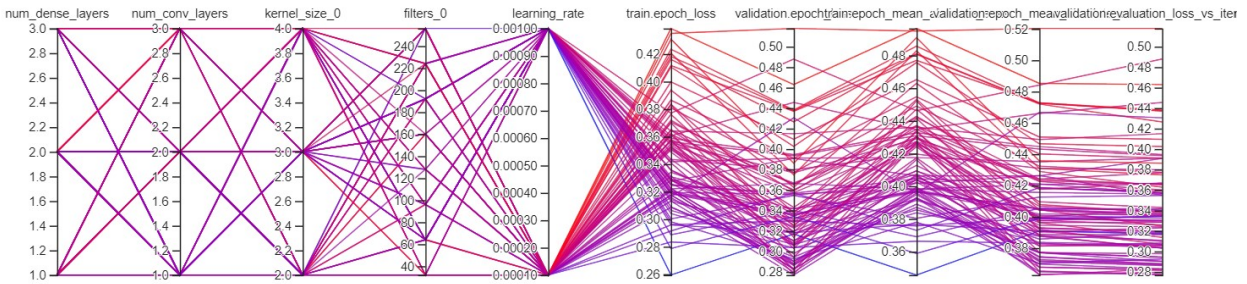


Figure 21: parallel plot of hyperparameter tuning CNN1d e1 L3-8

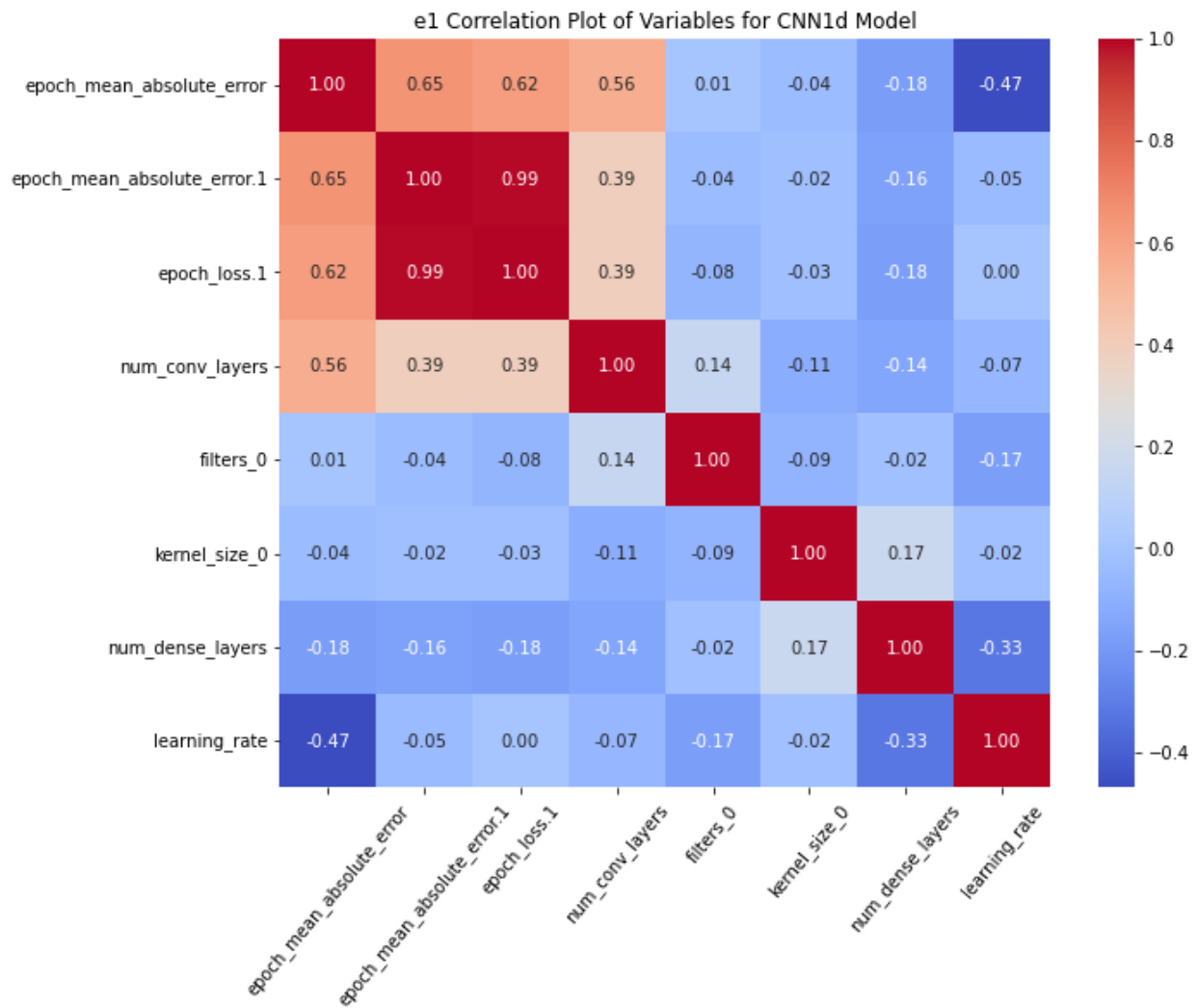


Figure 22: Correlation plot CNN1d e1 L3-8

Conclusion on Variables and Metrics

- Learning Rate: Should be minimized.
- Number of Filters: Has limited impact, can be reduced by half.
- Convolution Layer Placement: Prefer positioning in the upper third of the network.
- Kernel Size: Should be relatively large, though not the largest possible.
- Dense Layers: Multiple layers should be concentrated in the first quartile (Q1).

Conclusion on Inter-variable Relationships

- Attention is required to balance the number of dense layers and the learning rate.
- There is a tendency to increase the kernel size with an increasing number of dense layers.

CNN 1D – E5

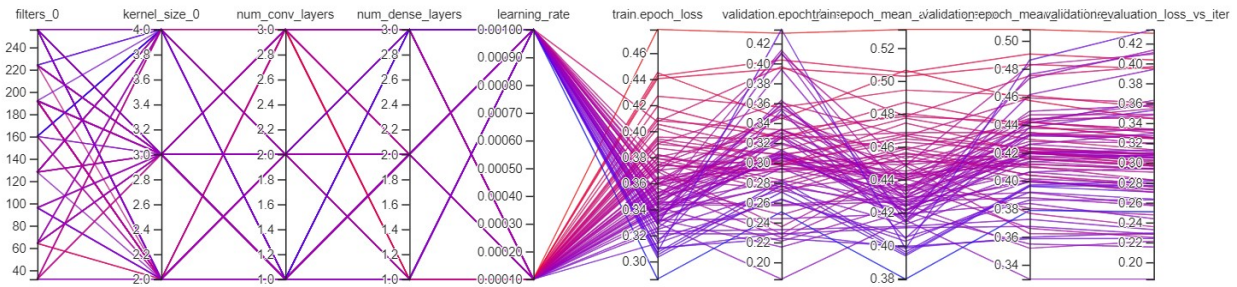


Figure 23: parallel plot of hyperparameter tuning CNN1d e5 L3-8

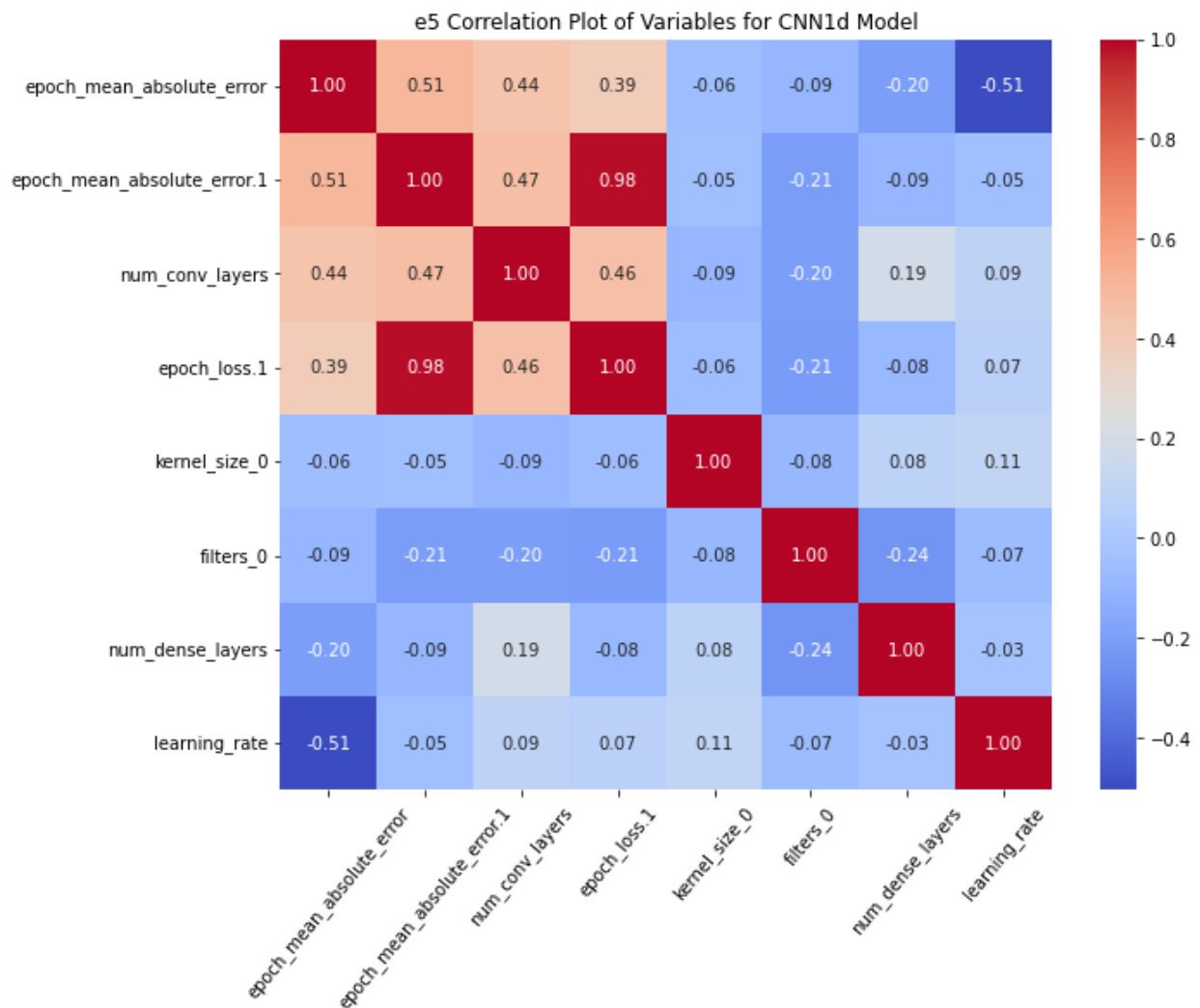


Figure 24: Correlation plot CNN1d e5 L3-8

Conclusion on Variables and Metrics

- Learning Rate: Should be minimized.
- Dense Layers: Should be concentrated in the first quartile (Q1).
- Convolution Layers: Multiple layers are recommended.
- Kernel Size: Preferably smaller.

Conclusion on Inter-variable Relationships

- The number of filters should be inversely proportional to the number of dense layers.

CNN 2D – E1

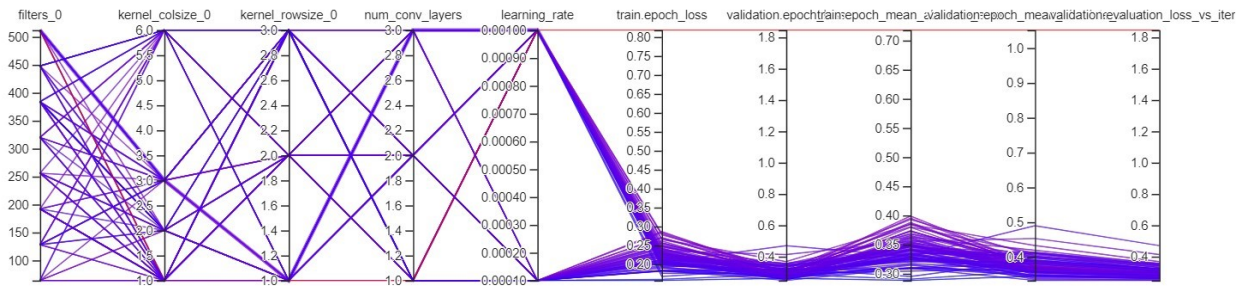


Figure 25: parallel plot of hyperparameter tuning CNN2d e1 L3-8

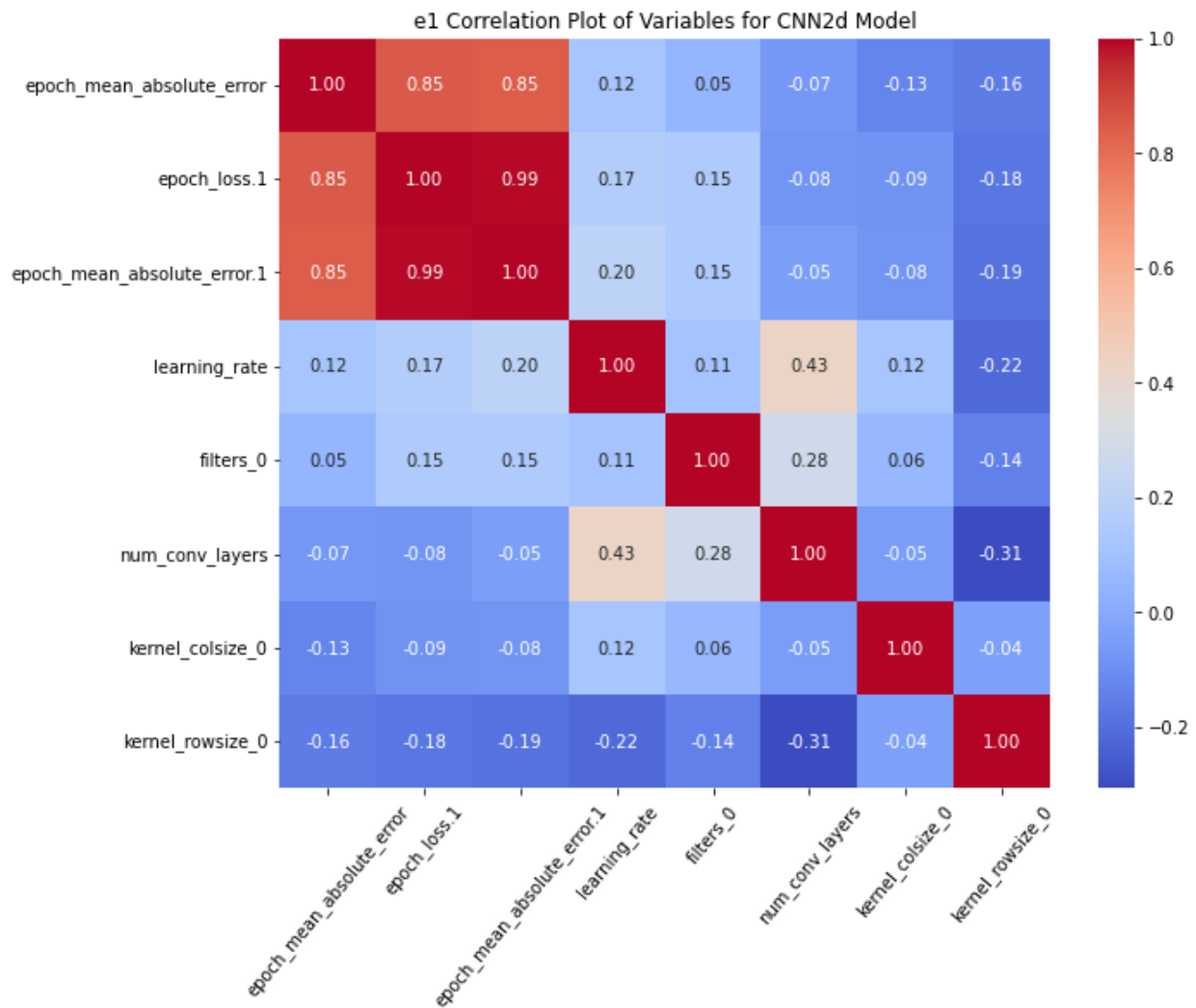


Figure 26: Correlation plot CNN2d e1 L3-8

Conclusion on Variables and Metrics

- Learning Rate: Should be increased.
- Number of Filters: Less significant in impact.
- Kernel Size: Preferably larger; the number of rows is more critical.

Conclusion on Inter-variable Relationships

- Increase the number of convolution layers in conjunction with the learning rate.
- If there are numerous convolution layers, increase the number of filters.
- If there are numerous convolution layers, use a smaller kernel row size.

CNN 2D – E5

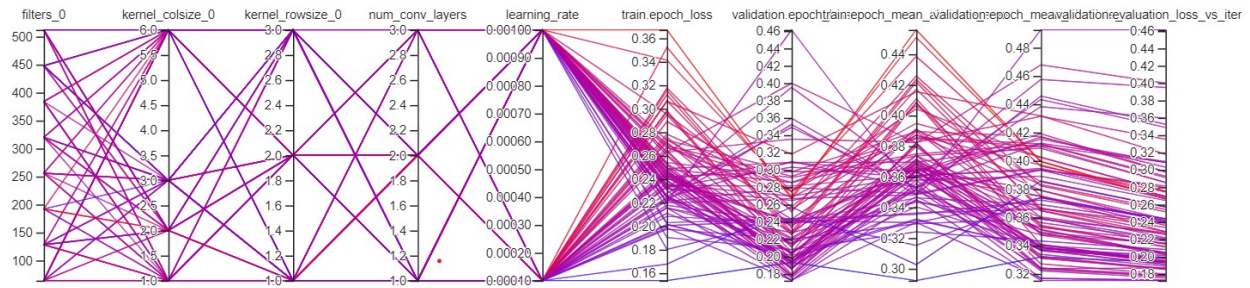


Figure 27: parallel plot of hyperparameter tuning CNN2d e5 L3-8

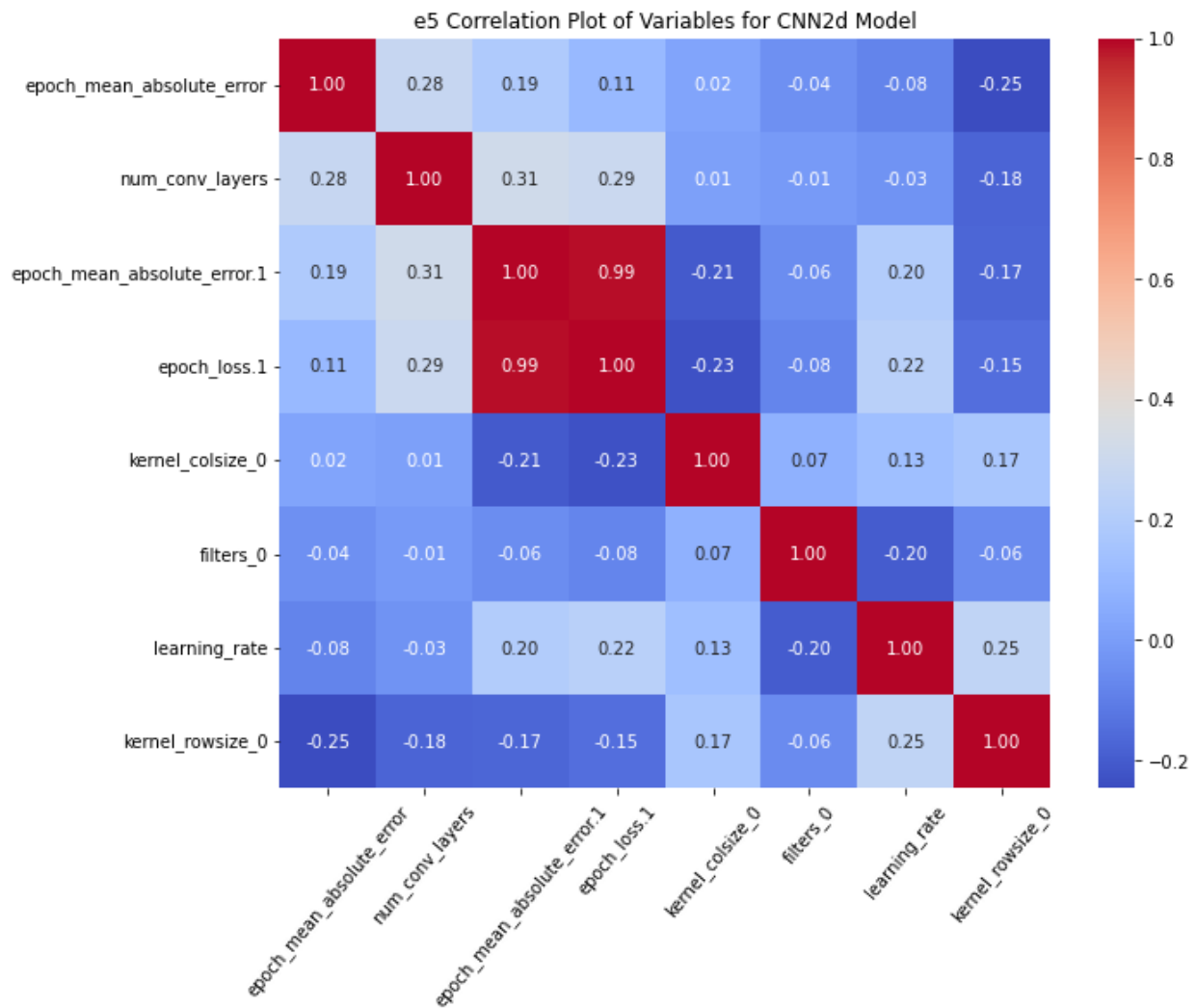


Figure 28: Correlation plot CNN2d e5 L3-8

Conclusion on Variables and Metrics

- Convolution Layers: Should be numerous.
- Kernel Row Size: Should be smaller.
- Kernel Column Size: Should be larger.

Conclusion on Inter-variable Relationships

- With a smaller learning rate, use a greater number of filters.
- With a larger learning rate, use a smaller kernel row size.

	LSTM					CNN 2D					CNN 1D				
HS ahead	RMSE	r	PE	MSA	SSPB	RMSE	r	PE	MSA	SSPB	RMSE	r	PE	MSA	SSPB
1	0.334	0.853	0.727	45.854	2.383	0.317	0.869	0.755	41.054	0.467	0.346	0.849	0.708	45.780	-10.727
2	0.347	0.842	0.706	48.406	4.982	0.320	0.866	0.750	40.340	-0.950	0.351	0.841	0.698	47.055	-6.740
3	0.350	0.840	0.701	46.884	5.339	0.328	0.860	0.737	41.215	-2.818	0.354	0.839	0.694	48.393	-10.851
4	0.351	0.839	0.698	47.854	6.089	0.331	0.857	0.732	43.217	-2.526	0.355	0.835	0.691	52.090	8.992
5	0.352	0.836	0.697	49.053	6.873	0.331	0.858	0.731	42.391	-5.696	0.354	0.835	0.694	48.926	3.144
6	0.360	0.834	0.681	48.806	1.963	0.334	0.854	0.727	44.977	-3.595	0.363	0.829	0.677	50.933	-8.198
7	0.359	0.832	0.684	48.360	4.115	0.335	0.852	0.725	45.216	-2.547	0.360	0.827	0.682	52.049	4.364
8	0.355	0.831	0.690	48.860	2.499	0.342	0.847	0.713	45.909	-5.625	0.384	0.820	0.638	55.930	-19.565
9	0.356	0.830	0.688	49.411	3.848	0.339	0.847	0.717	45.260	-2.359	0.372	0.819	0.661	53.497	-8.131
10	0.358	0.830	0.686	49.692	4.802	0.345	0.844	0.708	45.422	-5.687	0.369	0.817	0.666	54.541	-4.760
11	0.362	0.824	0.678	51.929	1.331	0.349	0.839	0.701	47.338	-5.901	0.385	0.810	0.636	57.956	-16.082
12	0.372	0.816	0.665	52.913	4.727	0.358	0.832	0.689	46.805	-5.321	0.383	0.805	0.644	53.113	-0.686
MEAN	0.355	0.834	0.692	49.002	4.079	0.336	0.852	0.724	44.095	-3.547	0.365	0.827	0.674	51.689	-5.770

Table 6: E1 L1-8 prediction

	LSTM					CNN 2D					CNN 1D				
HS ahead	RMSE	r	PE	MSA	SSPB	RMSE	r	PE	MSA	SSPB	RMSE	r	PE	MSA	SSPB
1	0.172	0.943	0.885	24.089	4.248	0.164	0.946	0.895	23.250	-4.957	0.162	0.948	0.897	21.785	2.752
2	0.185	0.934	0.867	26.500	4.417	0.173	0.940	0.883	25.181	-4.181	0.170	0.942	0.888	23.801	-0.632
3	0.201	0.922	0.842	29.724	7.022	0.181	0.935	0.873	26.706	-4.355	0.181	0.935	0.873	26.375	-6.324
4	0.206	0.916	0.834	30.756	4.648	0.185	0.931	0.866	27.507	-4.701	0.189	0.929	0.860	27.671	5.384
5	0.213	0.912	0.822	32.416	8.855	0.189	0.927	0.860	28.017	-4.036	0.194	0.924	0.853	28.849	-4.010
6	0.208	0.913	0.830	32.172	2.917	0.195	0.925	0.851	29.985	-7.083	0.202	0.918	0.841	29.992	2.093
7	0.216	0.907	0.817	32.510	5.815	0.197	0.921	0.849	29.729	-4.629	0.209	0.919	0.830	34.136	-16.543
8	0.214	0.907	0.822	33.474	1.245	0.201	0.918	0.842	30.365	-6.951	0.212	0.912	0.824	31.677	-8.617
9	0.217	0.904	0.817	32.947	1.391	0.203	0.917	0.839	31.189	-5.844	0.211	0.911	0.826	31.394	-5.133
10	0.218	0.904	0.815	33.319	1.133	0.206	0.914	0.834	32.031	-5.873	0.216	0.909	0.817	32.341	1.268
11	0.223	0.899	0.806	34.808	2.749	0.208	0.913	0.830	32.721	-9.277	0.222	0.904	0.807	32.776	-0.849
12	0.224	0.897	0.804	34.927	-0.321	0.212	0.909	0.824	33.120	-5.650	0.232	0.906	0.790	37.262	-21.165
MEAN	0.208	0.913	0.830	31.470	3.677	0.193	0.925	0.854	29.150	-5.628	0.200	0.922	0.842	29.838	-4.315

Table 7: E5 L1-8 prediction

	LSTM					CNN 2D					CNN 1D				
HS ahead	RMSE	r	PE	MSA	SSPB	RMSE	r	PE	MSA	SSPB	RMSE	r	PE	MSA	SSPB
1	0.355	0.844	0.709	44.823	-0.931	0.343	0.857	0.729	44.594	0.996	0.345	0.853	0.726	44.171	6.595
2	0.355	0.844	0.710	46.765	-1.728	0.346	0.855	0.724	45.978	-4.793	0.353	0.847	0.713	46.305	-2.355
3	0.357	0.841	0.706	48.389	-1.755	0.344	0.856	0.727	46.406	-5.186	0.350	0.848	0.718	47.723	1.460
4	0.363	0.835	0.696	50.474	-1.897	0.349	0.852	0.719	48.840	1.147	0.352	0.847	0.715	51.081	0.181
5	0.369	0.828	0.686	53.329	0.075	0.353	0.848	0.712	50.556	-5.266	0.358	0.840	0.705	49.456	-1.900
6	0.370	0.828	0.685	51.307	-2.486	0.358	0.841	0.704	49.781	-3.656	0.375	0.832	0.675	57.281	20.312
7	0.376	0.821	0.673	54.992	2.486	0.360	0.841	0.702	52.865	-6.431	0.368	0.830	0.688	54.209	2.151
8	0.376	0.823	0.675	55.927	-4.397	0.363	0.836	0.696	51.400	-4.727	0.376	0.823	0.673	52.185	-4.899
9	0.379	0.818	0.668	56.212	-2.785	0.366	0.835	0.690	53.366	-4.880	0.375	0.823	0.676	55.785	3.582
10	0.383	0.814	0.661	56.798	-2.830	0.368	0.832	0.688	51.256	-9.093	0.385	0.811	0.658	57.983	-0.355
11	0.393	0.803	0.643	60.111	-3.934	0.371	0.830	0.683	53.363	-6.430	0.407	0.815	0.617	65.507	31.257
12	0.397	0.798	0.636	61.198	-1.701	0.378	0.822	0.669	52.911	-7.436	0.390	0.807	0.649	58.990	4.136
MEAN	0.373	0.825	0.679	53.360	-1.824	0.358	0.842	0.704	50.110	-4.646	0.369	0.831	0.685	53.390	5.014

Table 8: E1 L2-8 prediction

	LSTM					CNN 2D					CNN 1D				
HS ahead	RMSE	r	PE	MSA	SSPB	RMSE	r	PE	MSA	SSPB	RMSE	r	PE	MSA	SSPB
1	0.199	0.932	0.867	27.820	3.013	0.205	0.933	0.859	33.610	-14.374	0.195	0.939	0.873	25.842	8.140
2	0.211	0.924	0.851	30.809	4.233	0.205	0.928	0.858	30.922	-9.132	0.199	0.932	0.868	28.020	-2.141
3	0.223	0.913	0.833	32.834	4.016	0.214	0.922	0.847	33.939	-10.051	0.219	0.922	0.839	30.373	8.791
4	0.230	0.907	0.822	34.563	1.486	0.222	0.916	0.835	36.087	-11.888	0.222	0.917	0.834	31.990	7.613
5	0.245	0.897	0.798	38.576	7.713	0.229	0.911	0.823	36.698	-14.232	0.226	0.912	0.829	32.985	-1.356
6	0.240	0.900	0.806	36.335	3.724	0.234	0.908	0.816	38.425	-13.191	0.232	0.907	0.820	34.211	0.603
7	0.250	0.894	0.791	37.529	7.835	0.234	0.906	0.815	39.614	-15.748	0.241	0.901	0.805	34.609	-2.674
8	0.249	0.892	0.791	38.649	4.114	0.241	0.903	0.806	40.266	-15.257	0.244	0.897	0.799	36.287	5.872
9	0.262	0.881	0.770	40.871	7.588	0.242	0.900	0.803	40.980	-16.641	0.249	0.895	0.792	35.590	6.774
10	0.260	0.882	0.772	40.704	5.075	0.245	0.897	0.798	41.153	-15.920	0.247	0.894	0.795	35.572	2.112
11	0.261	0.880	0.772	41.720	2.868	0.247	0.895	0.795	41.123	-14.737	0.252	0.891	0.786	35.240	3.329
12	0.262	0.877	0.768	40.694	2.902	0.247	0.894	0.794	42.507	-14.900	0.252	0.890	0.786	36.871	-8.298
MEAN	0.241	0.898	0.803	36.759	4.547	0.230	0.909	0.821	37.944	-13.839	0.231	0.908	0.819	33.132	2.397

Table 9: E5 L2-8 prediction

11.1 Batchsize - learning rate optimization search with CNN 1D

t	RMSE	r	PE	MSA	SSPB
1	0.218	0.940	0.882	25.193	-4.490
2	0.227	0.937	0.872	28.757	-12.488
3	0.248	0.932	0.848	36.549	-24.576
4	0.243	0.928	0.854	33.864	-14.495
5	0.267	0.923	0.824	40.863	-26.958
6	0.265	0.920	0.826	38.755	-22.770
7	0.269	0.916	0.820	38.881	-20.750
8	0.271	0.912	0.812	40.206	-23.832
9	0.277	0.909	0.809	40.194	-18.418
10	0.279	0.906	0.810	40.206	-13.814
11	0.276	0.903	0.810	40.652	-12.810
12	0.278	0.901	0.807	40.340	-12.435
MEAN	0.260	0.919	0.831	37.215	-17.461

Table 10: e1 L3-8 - batchsize = 100 ; learning rate = 1e-4

t	RMSE	r	PE	MSA	SSPB
1	0.2234	0.9366	0.8764	27.0476	-2.4541
2	0.2318	0.9312	0.8668	29.3188	1.1248
3	0.2451	0.9267	0.8511	32.1500	-11.7008
4	0.2471	0.9214	0.8485	32.2615	-2.1220
5	0.2589	0.9152	0.8337	33.8906	4.4415
6	0.2607	0.9140	0.8314	36.8695	-10.4862
7	0.2728	0.9068	0.8152	37.1976	6.6320
8	0.2732	0.9045	0.8146	40.4329	-10.6649
9	0.2782	0.9004	0.8077	40.8888	1.3423
10	0.2842	0.8967	0.7992	41.2363	5.1393
11	0.2879	0.8931	0.7940	44.0559	-10.2887
12	0.2999	0.8888	0.7763	43.3730	8.4508
MEAN	0.2636	0.9113	0.8263	36.5602	-1.7155

Table 11: e1 L3-8 - batchsize = 10 ; learning rate = 1e-4

t	RMSE	r	PE	MSA	SSPB
1	0.219	0.940	0.881	26.110	-6.407
2	0.226	0.935	0.873	27.637	-2.508
3	0.236	0.929	0.861	29.494	-8.045
4	0.243	0.926	0.854	32.418	-10.838
5	0.264	0.916	0.827	36.022	7.555
6	0.256	0.916	0.837	36.237	-6.612
7	0.274	0.908	0.814	37.271	8.668
8	0.271	0.905	0.818	39.920	-8.671
9	0.275	0.902	0.812	40.110	-1.961
10	0.283	0.899	0.801	39.889	6.589
11	0.282	0.896	0.802	42.224	-8.760
12	0.297	0.894	0.781	41.724	12.315
MEAN	0.251	0.914	0.830	35.755	-1.556

Table 12: e1 L3-8 - batchsize = 18 ; learning rate = 1e-4

t	RMSE	r	PE	MSA	SSPB
1	0.2192	0.9396	0.8811	26.1100	-6.4065
2	0.2262	0.9348	0.8732	27.6373	-2.5077
3	0.2364	0.9294	0.8614	29.4937	-8.0455
4	0.2429	0.9259	0.8537	32.4177	-10.8376
5	0.2640	0.9157	0.8271	36.0220	7.5550
6	0.2561	0.9158	0.8372	36.2375	-6.6121
7	0.2741	0.9080	0.8135	37.2706	8.6681
8	0.2709	0.9053	0.8178	39.9200	-8.6709
9	0.2752	0.9022	0.8118	40.1101	-1.9609
10	0.2831	0.8989	0.8008	39.8891	6.5886
11	0.2820	0.8964	0.8024	42.2236	-8.7598
12	0.2966	0.8937	0.7812	41.7237	12.3149
MEAN	0.2605	0.9138	0.8301	35.7546	-1.5562

Table 13: e1 L3-8 - batchsize = 20 ; learning rate = 1e-4

t	RMSE	r	PE	MSA	SSPB
1	0.218	0.940	0.882	25.214	5.259
2	0.225	0.935	0.874	27.402	0.188
3	0.238	0.931	0.860	29.776	9.076
4	0.240	0.926	0.857	29.965	3.009
5	0.245	0.923	0.851	31.735	3.574
6	0.254	0.918	0.840	33.395	4.000
7	0.259	0.914	0.834	33.606	3.832
8	0.262	0.912	0.830	33.500	-3.170
9	0.265	0.909	0.825	34.522	0.321
10	0.270	0.905	0.819	37.469	-1.577
11	0.275	0.902	0.812	39.386	-8.554
12	0.277	0.901	0.809	38.045	-8.240
MEAN	0.252	0.918	0.841	32.835	0.643

Table 14: e1 L3-8 - batchsize = 25 ; learning rate = 1e-4

t	RMSE	r	PE	MSA	SSPB
1	0.215	0.941	0.885	25.330	2.198
2	0.226	0.935	0.873	27.506	5.016
3	0.233	0.931	0.866	28.435	-1.425
4	0.240	0.927	0.857	31.067	-5.165
5	0.244	0.924	0.853	31.403	-2.011
6	0.253	0.919	0.841	33.576	-8.758
7	0.258	0.915	0.835	33.444	-4.902
8	0.263	0.912	0.828	36.040	-8.458
9	0.269	0.906	0.820	36.049	-6.045
10	0.271	0.905	0.818	36.390	-0.563
11	0.277	0.901	0.810	38.590	-7.383
12	0.281	0.897	0.804	36.983	2.659
MEAN	0.260	0.918	0.841	32.901	-2.903

Table 15: e1 L3-8 - batchsize = 50 ; learning rate = 1e-4

t	RMSE	r	PE	MSA	SSPB
1	0.269	0.921	0.820	35.986	8.757
2	0.249	0.923	0.847	35.219	-12.832
3	0.260	0.918	0.833	34.784	3.948
4	0.261	0.915	0.831	34.935	-1.941
5	0.275	0.910	0.812	37.087	6.118
6	0.275	0.906	0.813	37.363	3.798
7	0.279	0.905	0.807	39.779	1.547
8	0.274	0.905	0.814	36.890	3.404
9	0.289	0.899	0.792	42.511	2.555
10	0.293	0.893	0.787	41.409	4.836
11	0.294	0.889	0.785	42.783	0.873
12	0.292	0.889	0.788	42.809	-6.014
MEAN	0.261	0.906	0.811	38.463	1.254

Table 16: e1 L3-8 - batchsize = 10 ; learning rate = 1e-3

t	RMSE	r	PE	MSA	SSPB
1	0.236	0.938	0.863	29.813	13.176
2	0.228	0.935	0.872	28.386	0.977
3	0.234	0.930	0.864	29.660	-1.813
4	0.251	0.924	0.843	33.655	7.580
5	0.248	0.922	0.848	32.795	3.224
6	0.257	0.916	0.836	34.688	3.875
7	0.263	0.912	0.829	35.378	2.555
8	0.264	0.910	0.827	34.123	-0.278
9	0.271	0.905	0.818	37.326	-5.471
10	0.275	0.903	0.812	36.566	6.961
11	0.280	0.898	0.805	40.863	1.804
12	0.278	0.899	0.808	38.327	-2.426
MEAN	0.257	0.916	0.835	34.298	2.514

Table 17: e1 L3-8 - batchsize = 25 ; learning rate = 1e-3

