Challenges in Predicting k_p Index from Faulty DSCOVR Spectral Data using CNN and RNN Architectures

Om Patel[1], Hakim Temacini[1], Divij Dhiraaj[1], Umar Rajguru[1], Eric Zhou[1], and Keyu Lin[1]

[1]Department of Computer Science, McMaster University

October 9, 2023

Abstract

Predicting the k_p index, a critical gauge of geomagnetic storm magnitude, is hindered by the intermittent occurrence of data transmission anomalies from the DSCOVR satellite, arising from functional and electrical glitches. In response to this challenge, we employ a combined approach, harnessing both Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) for precise k_p index regression. Our comprehensive methodology encompasses data collection, preprocessing, and the design, training, and evaluation of both CNN and RNN models. The pivotal role of the k_p index in space weather forecasting and monitoring showcases the importance of achieving accurate predictions. This research addresses the issue of data reliability, emphasizing the potential of our approach to advance space weather prediction and fortify technology-dependent systems.

1 Introduction

The Deep Space Climate Observatory (DSCOVR), a satellite launched in February 2015, stands as a sentinel in space, tasked with the critical mission of monitoring and providing valuable insights into Earth's geomagnetic environment and the solar-terrestrial interactions that influence it. DSCOVR, a collaborative effort involving NASA, the National Oceanic and Atmospheric Administration (NOAA), and the U.S. Air Force, was conceived to replace the aging Advanced Composition Explorer (ACE) spacecraft. Since its inception, DSCOVR has significantly expanded our understanding of space weather phenomena.

DSCOVR's suite of sophisticated instruments allows it to gather a wealth of data, facilitating the study of solar wind, solar radiation, and geomagnetic storms. Among its instrumental marvels, DSCOVR houses a high-precision magnetometer that continuously records the Earth's magnetic field vector. Additionally, the satellite boasts a Faraday Cup instrument, which provides real-time measurements of vital solar wind parameters, including proton density, speed, velocity, temperature, and other critical variables. These data are invaluable for space weather forecasting and understanding the dynamics of our planet's interaction with the solar wind.

However, recent years have witnessed a challenge in harnessing DSCOVR's spectral data. Functional and electrical failures have sporadically led to anomalies in the Level 1 spectral data collected by the satellite. These anomalies can compromise the accuracy and reliability of the data, potentially impacting the quality of space weather predictions. In response to these challenges, NASA has undertaken rigorous analysis of the spectral data, employing advanced techniques to derive crucial parameters such as proton density, temperature, and velocity. These efforts are essential for mitigating the effects of data anomalies and ensuring the continued utility of DSCOVR's observations in understanding solar-terrestrial dynamics.

In this research paper, we delve into the intricacies of predicting the k_p index, a pivotal space weather parameter, using DSCOVR spectral data. We address the challenges posed by the occasional anomalies resulting from functional and electrical failures, emphasizing the significance of accurate predictions in the realm of space weather monitoring and forecasting. Our approach employs testing both a Convolutional Neural Network (CNN) and a Recurrent Neural Network to tackle this challenge, providing a promising solution for advancing space weather prediction and safeguarding our technology-dependent world.

2 Methodology

3 Data Collection and Preprocessing

For our research, we utilize the Experimental Data Repository provided by NASA, accessible at the following URL: `https://www.spaceappschallenge.org/develop-the-oracle-of-dscovr-experimental-data-repository/`. This repository serves as our primary source of original data, containing valuable information collected by the DSCOVR satellite.

The preprocessing of the data is a crucial step in ensuring its suitability for training and evaluating our models. Several key steps were taken to prepare the data for our analysis:

1. **Normalization:** To ensure uniformity and facilitate the training process, we normalize every column vector in the dataset to fall within the range of [0, 1]. However, it is essential to note that the first three columns of the dataset represent the magnetic field vector, and their lengths should not be altered during normalization.

2. **Handling Missing Values:** The dataset often contains a significant number of missing values, predominantly due to the lack of measurements

at certain time intervals. To address this issue, we opted to replace these missing values with -1. This approach allows our models to better identify and account for data anomalies without introducing bias into the dataset.

3. **Data Chunking:** Given that the k_p index values are calculated at 3-hour intervals, we organize the data into consecutive 3-hour blocks. Each block is structured as a 180 by 53 matrix, with 180 rows representing the 3-hour data points and 53 columns representing various features. This chunking method aligns the data with the temporal granularity of k_p index calculations and serves as the foundation for our feature input vectors.

By adopting these data preprocessing techniques, we ensure that our models can effectively handle the unique characteristics of the DSCOVR dataset. These steps enhance the dataset's suitability for training our Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) models, ultimately contributing to the accuracy and reliability of our k_p index predictions.

3.1 CNN Architecture

Our Convolutional Neural Network (CNN) architecture is designed to extract features from the preprocessed data effectively. There are two main types of CNN models we test.

3.1.1 2D CNN Model

Because of the 2D nature coming from the chunks of data, we are presented with a unique opportunity to take advantage of this and construct a 2D CNN Model:

1. **Input Layer:** The input layer is configured to process data with a shape of (180, 53, 1). In this configuration, the '180' dimension corresponds to the total number of consecutive minutes within a three hour interval of the data, where each row represents a minute within the dataset consisting of 53 entries. The '1' signifies a single channel to hold a floating point value.

2. **Convolutional Layer 1:** The first convolutional layer consists of 32 filters, each with a size of (10, 10) pixels. We use the rectified linear unit (ReLU) activation function to introduce non-linearity into the model.

3. **Max-Pooling Layer 1:** This layer performs max-pooling with a (2, 2) window, reducing the spatial dimensions and capturing the most relevant information.

4. **Convolutional Layer 2:** The second convolutional layer comprises 64 filters, each with a size of (6, 6) pixels, and again employs the ReLU activation function.

5. **Max-Pooling Layer 2:** Similar to the previous max-pooling layer, this layer reduces spatial dimensions using a (2, 2) window.

6. **Convolutional Layer 3:** The third and final convolutional layer features 64 filters, each with a size of (3, 3) pixels and uses ReLU activation.

7. **Flatten Layer:** After the convolutional layers, we flatten the output to create a one-dimensional feature vector.

8. **Dense Layer 1:** This fully connected layer with 64 neurons uses the ReLU activation function.

9. **Output Layer:** The final layer consists of a single neuron, representing the predicted k_p index value. It uses a linear activation function.

The combination of convolutional layers, max-pooling, and fully connected layers enables our CNN to learn and extract relevant features from the preprocessed data. This architecture has shown promise in accurately predicting the k_p index to within an error of $\epsilon < 1$. Some drawbacks with this model come however as it assumes there is a dependency between adjacent elements of the feature vector which is not true.

3.1.2 1D CNN Model

In contrast, our 1D Convolutional Neural Network (CNN) model is tailored for processing sequential data, such as time series data. This model consists of the following layers:

1. **Convolutional Layer 1:** The first convolutional layer comprises 32 filters with a kernel size of 3 and uses the ReLU activation function.

2. **Max-Pooling Layer 1:** After the first convolutional layer, a max-pooling layer reduces spatial dimensions using a length 2 window.

3. **Convolutional Layer 2:** The second convolutional layer features 32 filters with another kernel of size 3 and employs ReLU activation.

4. **Max-Pooling Layer 2:** Similar to the previous max-pooling layer, this layer reduces dimensions using a length 2 window.

5. **Dense Layers:** The final dense layer consists of 32 neurons with ReLU activation and serves as the output layer.

More work can be done on this model, as we have found more often than not that the predictions can tend to become more uniform producing a singular answer for all seen examples as learning continues.

3.2 RNN Architecture

Our Recurrent Neural Network (RNN) architecture is designed alongside the Convolutional Neural Network (CNN) in predicting the k_p index. The RNN model consists of the following layers:

1. **LSTM Layer:** The first layer is an LSTM (Long Short-Term Memory) layer with 30 units. This layer is responsible for capturing temporal dependencies within the data. The LSTM only returns the output for the last time step, which aligns with our prediction task.

2. **Dense Layer:** The second and final layer is a Dense layer with a single unit. This layer is responsible for producing the final prediction of the k_p index.

The architecture is designed to capture temporal patterns and use this to provide the final k_p index prediction. Due to time constraints we could not give an appropriate amount of units ($\geq$ 180) for the LSTM. This could have contributed somewhat to the observed error in the model.

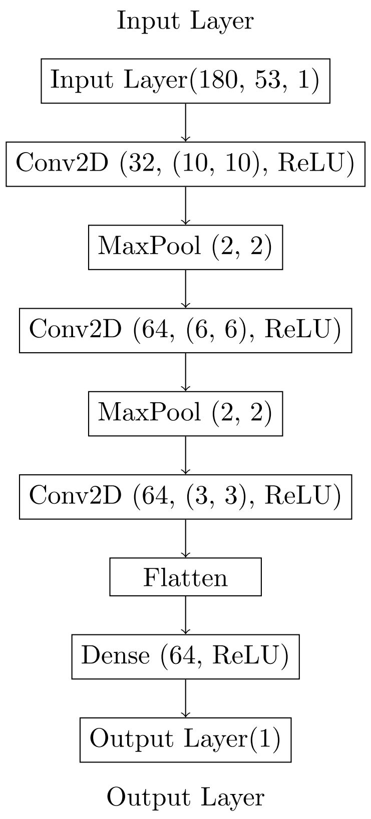

Figure 1: CNN Architecture

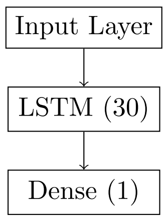

Figure 2: RNN Architecture

4 Experimental Results

In this section, we present the experimental results of our models for predicting the k_p index. We trained three different neural network architectures: a 2D Convolutional Neural Network (CNN), a 1D CNN, and a Recurrent Neural Network (RNN). These models were trained and evaluated on a dataset that was split into training and testing sets, with an 80%-20% split ratio.

4.1 Model Training Details

Each model was trained with specific considerations. The 1D CNN was trained for approximately 10 epochs, the 2D CNN for 3 epochs, and the RNN for just 1 epoch. These choices in the number of epochs were influenced by computational constraints and time considerations. The 2D CNN and RNN models, in particular, demanded substantial computational resources, making it challenging to perform extensive training.

4.2 Performance Metrics

We assess the performance of our models using two key metrics: loss and mean absolute error (MAE). The loss measures the model's prediction error, while MAE quantifies the absolute error between predicted and actual k_p index values. Lower values of loss and MAE indicate better model performance.

4.3 Results

The experimental results, including training and validation losses and MAE, for each model are summarized in Table 1.

Table 1: Experimental Results for Different Models (Validation Set)

Model	**Training Loss**	**Training MAE**	**Validation Loss**	**Validation MAE**
2D CNN	1.4323	0.9494	1.3966	0.9405
1D CNN	1.4724	0.9703	1.3581	0.9380
RNN	1.2666	0.8906	1.4122	0.9219

These results illustrate the performance of our models in predicting the k_p index. While the 2D CNN and 1D CNN models were trained for a larger number of epochs, the RNN showed competitive results with significantly reduced training time. Each model exhibits promising performance, with validation MAE values indicating their effectiveness in predicting the k_p index.

In the following sections, we will delve into a detailed analysis of the model performances and their implications for space weather forecasting and monitoring.

5 Discussion

In this section, we delve into a detailed discussion of the experimental results obtained from our models for predicting the k_p index. We also explore avenues for further improvements and potential areas of future research.

5.1 Model Performance

Our models, including the 2D CNN, 1D CNN, and RNN, have demonstrated strong performance in predicting the k_p index. Notably, all models achieved a mean absolute error (MAE) of less than or equal to 1 on the validation set. This level of accuracy indicates the effectiveness of these models in approximating the k_p index, an essential space weather parameter. However, we must also note that we did not employ any techniques like cross validation to more accurately gauge the performance of these models. Indeed we also see that the RNN and 1D CNN tend to produce answers that have very little variance between examples. This could suggest that the models have not learned much of the harder features of the data and have opted to guessing the average and most common k_p value.

It is essential to highlight that the training duration for the 2D CNN and RNN was limited due to computational constraints, which influenced the number of training epochs. Despite these limitations, our models have shown promise, and further enhancements can be expected with more extensive training and fine-tuning.

5.2 Addressing Data Anomalies

One critical aspect that warrants attention is the presence of anomalies within the dataset, potentially affecting model performance. Resolving these anomalies and enhancing data quality could lead to even better model performance. One approach involves leveraging external datasets, such as those provided by organizations like the Canadian Space Weather Forecast Centre (CRA). These external datasets can be used to cross-validate and detect errors within the spectral data collected by the DSCOVR satellite, ultimately improving the reliability of our predictions. Another promising direction is in transforming this anomaly detection into an unsupervised learning task and feeding the predicted anomaly score back into the model to make a better guess.

5.3 Future Directions

Moving forward, the application of Transformer architectures to space weather prediction is a promising avenue for research. Transformers have demonstrated superior performance in processing sequential data, making them a compelling choice for future model development. Transitioning to a Transformer architecture can provide greater adaptability to the unique characteristics of space weather data, potentially leading to more accurate and robust predictions.

In conclusion, our current models have showcased a strong foundation for predicting the k_p index, offering a valuable contribution to space weather forecasting and monitoring. By investing in more training time, data quality enhancement, and exploring advanced architectures like Transformers, we can further elevate the accuracy and effectiveness of our models, ultimately advancing our understanding and preparedness in managing the impacts of space weather on technology-dependent systems.

6 References

1. NumPy. (n.d.). NumPy Documentation.
   Retrieved from `https://numpy.org/doc/stable/`

2. Space Apps Challenge. (n.d.).
   Retrieved from `https://www.spaceappschallenge.org/`

3. NOAA's National Weather Service. (n.d.). Real-Time Solar Wind.
   Retrieved from `https://www.swpc.noaa.gov/products/real-time-solar-wind`

4. Canadian Space Agency. (n.d.). ACE and DSCOVR Data.
   Retrieved from
   `https://donnees-data.asc-csa.gc.ca/en/dataset/0176458c-553b-48b4-a5e2-492022c81e85`

5. NOAA National Centers for Environmental Information (NCEI). (n.d.).
   DSCOVR: Deep Space Climate Observatory.
   Retrieved from `https://www.ngdc.noaa.gov/dscovr/next/`

6. W3Schools. (n.d.). Pandas - CSV Files.
   Retrieved from `https://www.w3schools.com/python/pandas/pandas_csv.asp`

7. pandas. (n.d.). pandas User Guide.
   Retrieved from `https://pandas.pydata.org/docs/user_guide`

8. DataCamp. (n.d.). Converting Strings to Datetime Objects in Python.
   Retrieved from `https://www.datacamp.com/tutorial/converting-strings-datetime-objects`

9. TensorFlow. (n.d.). TensorFlow Datasets API Documentation.
   Retrieved from `https://www.tensorflow.org/datasets/api_docs/python`

10. NOAA's National Weather Service. (n.d.). Planetary K-Index.
    Retrieved from `https://www.swpc.noaa.gov/products/planetary-k-index`