A Transformer-Based Framework for Geomagnetic Activity Prediction

Table of Contents

Author(s)

- Author1 = {"name": "Yasser Abduallah", "affiliation": "Department of Computer Science, New Jersey Institute of Technology", "email": "ya54@njit.edu (mailto:ya54@njit.edu)", "orcid": "https://orcid.org/0000-0003-0792-2270"} (https://orcid.org/0000-0003-0792-2270"%7D)
- Author2 = {"name": "Jason T. L. Wang", "affiliation": "Department of Computer Science, New Jersey Institute of Technology", "email": "wangj@njit.edu (mailto:wangj@njit.edu)", "orcid": "https://orcid.org/0000-0002-2486-1097"} (https://orcid.org/0000-0002-2486-1097"%7D)
- Author3 = {"name": "Chunhui Xu", "affiliation": "Department of Computer Science, New Jersey Institute of Technology", "email": "cx4@njit.edu (mailto:cx4@njit.edu)", "orcid": ""}
- Author = {"name": "Haimin Wang", "affiliation": "Institute for Space Weather Sciences, New Jersey Institute of Technology", "email": "haimin.wang@njit.edu (mailto:haimin.wang@njit.edu)", "orcid": "https://orcid.org/0000-0002-5233-565X"} (https://orcid.org/0000-0002-5233-565X"%7D)

Purpose

Geomagnetic activities have a crucial impact on Earth, which can affect spacecraft and electrical power grids. Geospace scientists use a geomagnetic index, called the Kp index, to describe the overall level of geomagnetic activity. This index is an important indicator of disturbances in the Earth's magnetic field and is used by the U.S. Space Weather Prediction Center as an alert and warning service for users who may be affected by the disturbances. Early and accurate prediction of the Kp index is essential for preparedness and disaster risk management. In this paper, we present a novel deep learning method, named KpNet, to perform short-term, 1-9 hour ahead, forecasting of the Kp index based on the solar wind parameters taken from the NASA Space Science Data Coordinated Archive. KpNet combines transformer encoder blocks with Bayesian inference, which is capable of quantifying both aleatoric uncertainty (data uncertainty) and epistemic uncertainty (model uncertainty) when making Kp predictions. Experimental results show that KpNet outperforms closely related machine learning methods in terms of the root mean square error and R-squared score. Furthermore, KpNet can provide both data and model uncertainty quantification results, which the existing methods cannot offer. To our knowledge, this is the first time that Bayesian transformers have been used for Kp prediction.

In this notebook we provide an overview of the KpNet model to demonstrate how to forecast Kp index using deep learning (DL) and solar wind parameters and provide uncertatinty quantification with Bayesian network.

Technical Contributions

- We provide the community with a new tool to forecast the occurrance of the planetary Kp index for the next 1, 2, 3, 4, 5, 6, 7, 8, or 9 hours ahead. The tools also provide data and model uncertainty quantification.

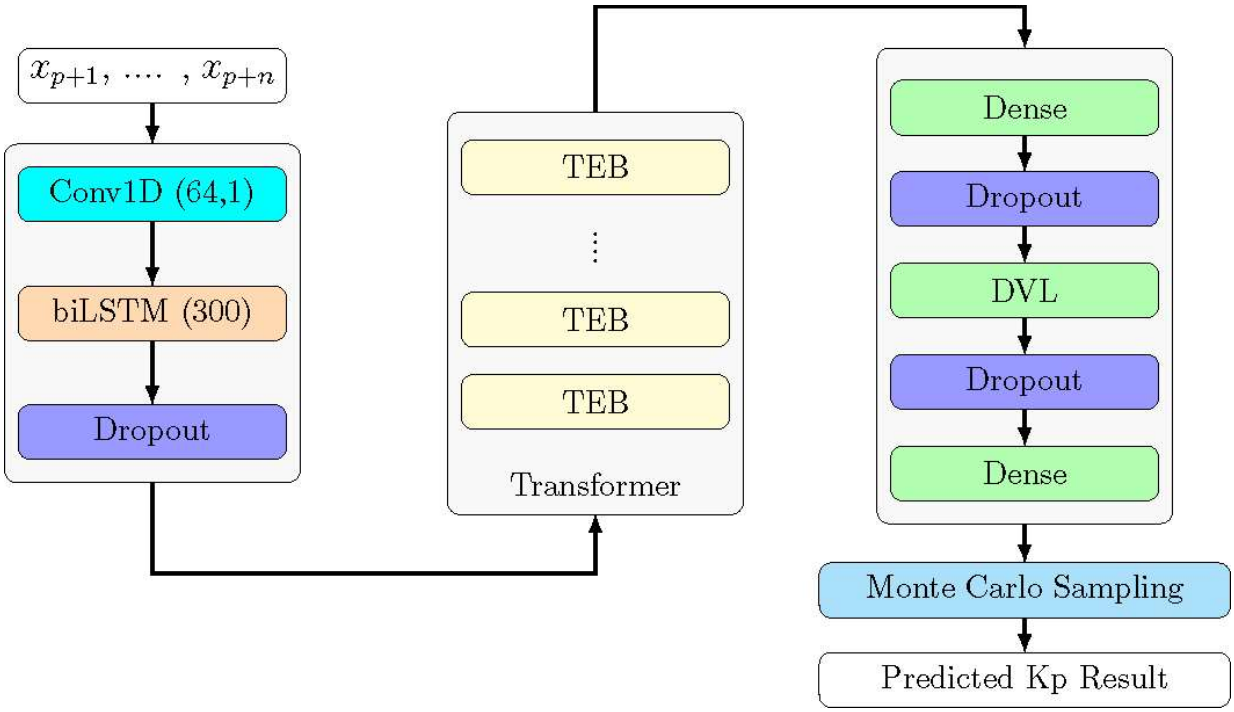
Methodology

The Figures below present the architecture of KpNet, which is created using the keras framework from tensorflow. To enhance the KpNet learning capability and its performance, we add multiple layers to the network. The input of KpNet consists of non-overlapping sequences of records x_{p+1}, x_{p+2}, ..., x_{p+n}, where n is set to 512 in our study. The sequences are passed to a one-dimensional convolution (Conv1D) layer with 64 kernels where the size of each kernel is 1. Conv1D was proven to be well suited for sequential data and was also previously used for geomagnetic index prediction; it learns internal patterns from the input data sequence and passes them to a bidirectional long short-term memory (biLSTM) layer that is configured with 300 neurons. Combining Conv1D and biLSTM layers has shown substantial improvement in performance when dealing with time series as our ablation studies show later.

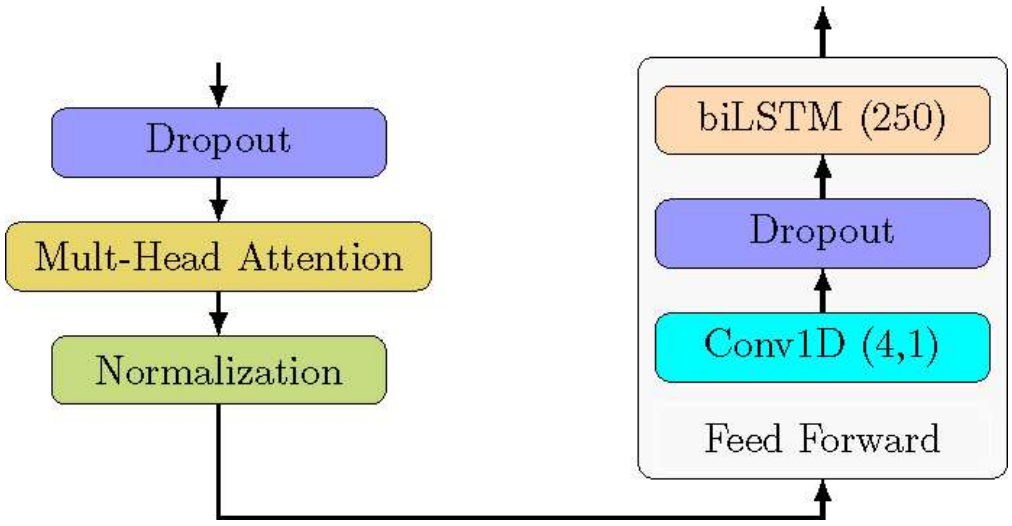
The biLSTM layer transfers the learned patterns down to a transformer network, which is composed of b transformer encoder blocks (TEBs) (we set b to 8 in this study). Each TEB consists of a multi-head attention layer, a batch normalization layer, and a feed-forward network. Generally, transformers for natural language processing (NLP) use layer normalization leading to significant performance gains over batch normalization. However, we use batch normalization here to avoid the effect of outliers in time series which do not exist in NLP word embedding. The multi-head attention layer provides transformation on the sequential input values to obtain distinct metrics of size h. Here, h is the number of attention heads that is set to 4 and the size of each attention head

is also set to 4; the other parameters are left with their default values. The feed-forward network, composed of a Conv1D layer, with 4 kernels where the number of kernels equals the number of attention heads and each kernel size is 1, followed by a biLSTM layer with 250 neurons, helps reduce over-fitting. Notice that each TEB uses a transformer encoder without the decoder because we are dealing with time series instead of language processing and therefore translation is not involved.

Furthermore, a dense variational layer (DVL) with 10 neurons is added that uses variational inference to approximate the posterior distribution over the model weights. DVL is similar to a regular dense layer but requires two input functions that define the prior and posterior distributions over the model weights. DVL allows KpNet to represent the weights by a distribution instead of estimated points. In addition, KpNet includes several dense and dropout layers. Each dense layer is strongly connected with its preceding layer where every neuron in the dense layer is connected with every neuron in the preceding layer. Each dropout layer provides a mechanism to randomly drop a percentage of hidden neurons to avoid over-fitting, as explained below.



(a)



(b)

Uncertainty Quantification

With the dropout layers, our model's internal structure is slightly different each time the neurons are dropped. This is an important behavior to the Monte Carlo (MC) class of algorithms that depends on random sampling and provides useful information. We use this technique to introduce a distribution interval of predicted values. More details about uncertainty quantification can be found in our full paper at: NOT SET YET ()

This notebook leverages python deep learning to describe the steps on how to use the KpNet tool to forecast the Kp index for 1 to 9 hours ahead.

Funding

This work was supported by U.S. NSF grants AGS-1927578 and AGS-1954737 and supported by NASA under grants 80NSSC18K1705, 80NSSC19K0068, and 80NSSC20K1282.

Keywords

keywords=["Kp","Index","Forecasting", "Prediction", "Machine", "Learning","Solar",'Wind"."Uncertainty","Quantification"]

Citation

To cite this notebook: Yasser Abduallah, Jason T. L. Wang, Chunhui Xu Genwei Zhang, Firas Gerges, and Haimin Wang. Forecasting the Disturbance Storm Time Index with Bayesian Deep Learning, available at: https://github.com/ccsc-tools/kp-prediction/YA_01_ATransformerBasedFrameworkForGeomagneticActivity.ipynb (https://github.com/ccsc-tools/kp-prediction/blob/main/YA_01_ATransformerBasedFrameworkForGeomagneticActivity.ipynb).

Acknowledgements

We acknowledge the use of NASA/GSFC's Space Physics Data Facility's OMNIWeb service and OMNI data

Setup

Installation on Local Machine

Running this notebook in a local machine requires Python version 3.9.x with the following packages and their version:

Library	Version	Description
keras	2.8.0	Deep learning API
numpy	1.22.4	Array manipulation
scikit-learn	1.0.2	Machine learning
sklearn	latest	Tools for predictive data analysis
matlabplot	3.5.1	Visutalization tool
pandas	1.4.1	Data loading and manipulation
seaborn	0.11.2	Visualization tool
scipy	1.8.1	Provides algorithms for optimization and statistics
tensorflow	2.8.0	Comprehensive, flexible ecosystem of tools and libraries for machine learning
tensorflow-gpu	2.8.0	Deep learning tool for high performance computation
tensorflow-probability	0.14.1	For probabilistic models

Library Import

The following libraries need to be imported.

In [1]:
```
#supress warning messages
import warnings
warnings.filterwarnings('ignore')
print('Importing packages..')
# Data manipulation
import pandas as pd
import numpy as np
import os
print('Packages imported')
#make sure the scripts are executed in the correct pacakage location.
import sys
sys.path.append('.')

```

```
Importing packages..
Packages imported
```

Data Processing and Analysis

The Kp measurements used in this study are provided by the GFZ German Research Centre for Geoscience available here: https://www.gfz-potsdam.de/en/kp-index/ (https://www.gfz-potsdam.de/en/kp-index/). The Kp values in the GFZ site are collected with a 3-hour cadence where the values range from 0 to 8.33. The solar wind parameters used in this study are taken from the NASA Space Science Data Coordinated Archive. We collect the data in the time period between January 1, 2010 and March 31, 2022. We select the time resolution of the hourly average for the solar wind parameters. We consider eight solar wind parameters, namely the interplanetary magnetic field (IMF) magnitude average, magnetic field Bx, By, and Bz components, plasma temperature, proton density, plasma speed, and flow pressure. Due to the difference in cadence, where Kp uses a 3-hour cadence whereas the solar wind parameters use a 1-hour cadence, we process the data by temporally matching the Kp measurements from the GFZ site with the solar wind parameters from the NASA site to create the final dataset.

Binder

This notebook is Binder enabled and can be run on mybinder.org (https://mybinder.org/) by using the image link below:

launch binder (https://mybinder.org/v2/gh/ccsc-tools/Kp-prediction/HEAD?labpath=YA_01_ATransformerBasedFrameworkForGeomagneticActivity.ipynb)

Please note that starting Binder might take some time to create and start the image.

KpNet Workflow and Results

Data Preparation and Loading

The data directoryr includes all training and test data sets required to run the run the notebook. The files are loaded and used during the testing and training process.

Predicting with Pretrained Models

There are default and pretrained models that can be used to predict without running your own trained model. The models_directory is set to default_models which uses all pretrained algorithms.

In [2]:
```
#Test default models for 4hour ahead.
from KpNet_test import test

models_directory='default_models'
print('Test default models for Kp forecasting for 1-9 hours ahead.')
start_hour=4
end_hour=4
test(start_hour,end_hour+1,models_directory=models_directory)

```

Time
(b)

```
show_figures: False
Plotting figures for default models results for Kp forecasting for 1-9 hours
ahead.
Graphing results for h = 1 hour ahead
Graphing results for h = 2 hour ahead
Graphing results for h = 3 hour ahead
Graphing results for h = 4 hour ahead
Graphing results for h = 5 hour ahead
Graphing results for h = 6 hour ahead
Graphing results for h = 7 hour ahead
Graphing results for h = 8 hour ahead
Graphing results for h = 9 hour ahead
Loading the train_model function...
Train custom model for h=4
Running training for h = 4 hour ahead
Epoch 1/100
```

Plotting the Pretrained Models Results

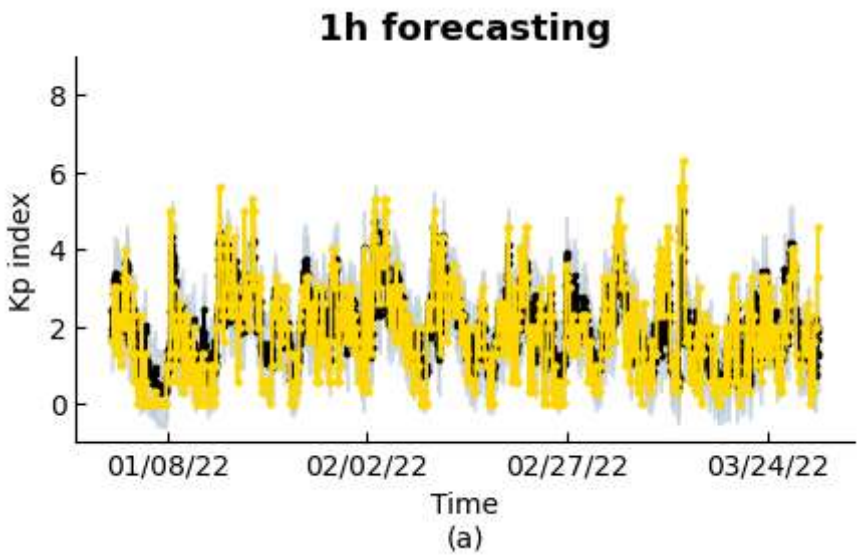
The prediction result can be plotted using the function plot_figures. It uses the results produced by the model from the "default_results" directory.

In [3]:
```
from KpNet_plot_results_figures import plot_figures

figures_dir='default_figures'
results_dir='default_results'
print('Plotting figures for default models results for Kp forecasting for 1-
start_hour=1
end_hour=9

plot_figures(start_hour, end_hour+1,show_figures=True,figures_dir = figures_

```



1h forecasting

KpNet Model Training and Testing Example

KpNet Model Training with Sample Data

Here, we show how to train the model with sample data example. In this exmaple, we show how to train the model for tiwe window h = 1 to 9 hour ahead.

In [ ]:
```
#Training for F 4 hour ahead on sample data.
print('Loading the train_model function...')
from KpNet_train import train_model
print('Train custom model for h=4')
start_hour=4
end_hour=4
#set the number of epochs=10
epochs=100
train_model(start_hour,end_hour+1,epochs=epochs)

```

Predicting with Your Trained KpNet Model

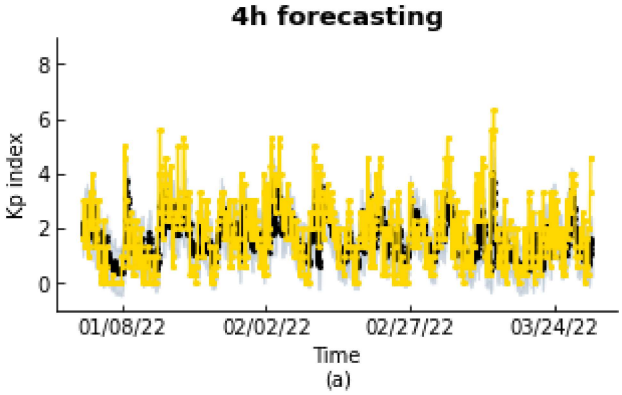
To predict the testing data using the model you trained above, make sure the models_directory variable is set to models:
models_directory='models'

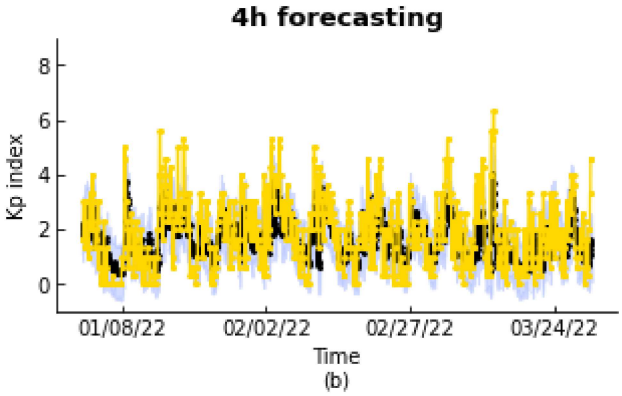
Note: this is training job is only an example that uses less, training proceses, epochs, therefore the results and performance metrics are not comparable to fully developed pretrained and default models.

In [2]:
```
#Test custom model for 4-hour ahread.
from KpNet_test import test

models_directory='models'
print('Test your trained models for Kp forecasting for 6-hour ahead.')
start_hour=4
end_hour=4
test(start_hour,end_hour+1,models_directory=models_directory)

```





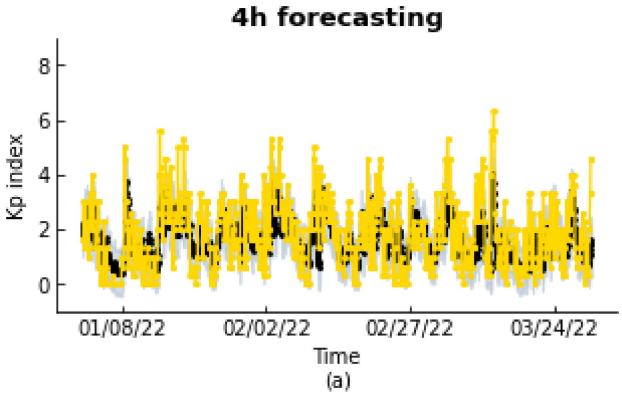
Plotting the Results for Your Trained Model

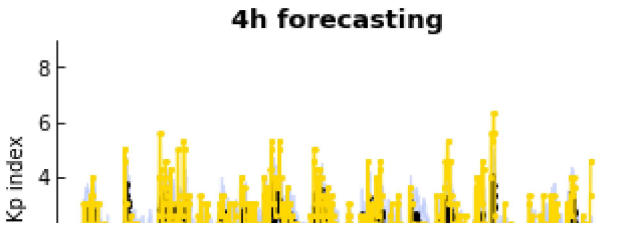
The prediction result can be plotted using the function plot_figures as shown in the following example. The example plots your trained model results for h=4 hours ahead.

In [3]:
```
from KpNet_plot_results_figures import plot_figures

figures_dir='figures'
results_dir='results'
print('Plotting figures for default models results for KP forecasting for 4
start_hour=4
end_hour=4

plot_figures(start_hour, end_hour+1,show_figures=True,figures_dir = figures_
```





Timing

Please note that the execution time in mybinder varies based on the availability of resources. The average time to run the notebook is 30-40 minutes, but it could be more.

Conclusions

We presented a novel deep learning model, named KpNet, to perform short-term, 1-9 hour ahead, forecasting of the Kp index based on the solar wind parameters taken from the NASA Space Science Data Coordinated Archive. KpNet combines transformer encoder blocks with Bayesian inference, which is capable of quantifying both aleatoric uncertainty and epistemic uncertainty when making Kp predictions. Our experimental results demonstrated the good performance of KpNet and its superiority over related machine learning methods. These results were based on the data collected in the period between January 1, 2010 and March 31, 2022. The training set contained hourly records from January 1, 2010 to December 31, 2021. The test set contained hourly records from January 1, 2022 to March 31, 2022. To avoid bias in our findings, we performed additional experiments using 10-fold cross validation where the data was divided into 10 approximately equal partitions or folds. The sequential order of the data in each fold was maintained. In each run, one fold was used for testing and the other nine folds together were used for training. There were 10 folds and hence 10 runs where the average performance metric values

over the 10 runs were calculated. Results from the 10-fold cross validation were consistent with those reported in the paper. Thus we conclude that the proposed KpNet is a feasible machine learning method for short-term, 1-9 hour, ahead predictions of the Kp index. In the future we plan to extend KpNet to perform long-term Kp forecasting using other data sources such as solar images in addition to solar wind parameters.

References

1. A Transformer-Based Framework for Geomagnetic Activity Prediction
   Yasser Abduallah, Jason T. L. Wang, Chunhui Xu, and Haimin Wang
   https://link.springer.com/chapter/10.1007/978-3-031-16564-1_31 (https://link.springer.com/chapter/10.1007/978-3-031-16564-1_31)
2. Forecasting the Disturbance Storm Time Index with Bayesian Deep Learning
   Yasser Abduallah, Jason T. L. Wang, Prianka Bose, Genwei Zhang, Firas Gerges, and Haimin Wang
   https://iopscience.iop.org/article/10.3847/1538-4365/ac5f56 (https://iopscience.iop.org/article/10.3847/1538-4365/ac5f56)
3. DeepSun: Machine-Learning-as-a-Service for Solar Flare Prediction
   Yasser Abduallah, Jason T. L. Wang and Haimin Wang
   https://doi.org/10.32473/flairs.v35i.130564 (https://doi.org/10.32473/flairs.v35i.130564)
4. Tracing Hα Fibrils through Bayesian Deep Learning
   Haodi Jiang, Ju Jing, Jiasheng Wang, Chang Liu, Qin Li, Yan Xu, Jason T. L. Wang, and Haimin Wang
   https://doi.org/10.3847/1538-4365/ac14b7 (https://doi.org/10.3847/1538-4365/ac14b7)
5. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning
   Yarin Gal and Zoubin Ghahramani
   https://dl.acm.org/doi/10.5555/3045390.3045502 (https://dl.acm.org/doi/10.5555/3045390.3045502)

In [ ]:
```
1
```