NASA Space Apps challenge

Carrington

Predicting a catastrophe.

By:

- Abdallah Khairy Werby *Abdallhwerby@gmail.com*
- Abdulrahman Morsi Ali *abdalr7manalsisi@gmail*
- Yousef Hegazi *yousefhegazi74@gmail.com*
- Thabet Elmahdy *thabet7474@gmail.com*
- Osama Said *Osamasaid2009@hotmail.com*

Abstract

Solar flares are large eruptions of electromagnetic radiation from the Sun lasting from minutes to hours. The sudden outburst of electromagnetic energy travels at the speed of light, therefore any effect upon the sunlit side of Earth's exposed outer atmosphere occurs at the same time the event is observed.

Solar flares occur when a large magnetic energy builds up in the solar atmosphere and is released suddenly. These outbursts are intrinsically linked to the solar cycle — an approximately 11-year cycle of solar activity driven by the sun's magnetic field. These solar flares are accompanied by Coronal Mass Ejections (CMEs), they are large expulsions of plasma and magnetic fields from the Sun's corona. They can eject billions of tons of coronal material and carry an embedded magnetic field (frozen in flux) that is stronger than the background solar wind interplanetary magnetic field (IMF) strength.

Solar flares can be the reason for the end of all human technology for more than 10 years, 10 years of total blackout, with the potential of causing radiation hazards to astronauts. So it is crucial to build a model that predicts solar flares and CMEs, and in our thesis we discuss the journey of avoiding a global catastrophe by a time-series data consisting of active solar region magnetic field parameters acquired from the NOAA observatory satellite DSCOVR. Our approach to predict the Carrington event is building a LSTM model that predicts the value of the DST(disturbance storm index) feature.

Table of Contents

Acronyms

MVTS	Multivariate Time Series
AR	Active Region
SDO	Solar Dynamics Laboratory
NASA	National Aeronautics and Space Administration
CME	Coronal Mass Ejections
NOAA	National Oceanic and Atmospheric Administration
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
ANN	Artificial Neural Network
CNN	Convolutional Neural Network

1. Introduction

Part of the Intelligent design of the universe, is the fine tuned parameters of it, a fine tuned universe suggests that the occurrence of life in the universe is very sensitive to the values of certain fundamental physical constants and that the observed values are, for some reason, improbable. Important example of it is the distance between our star; the sun and earth, approximately 149.82 million km. The Sun's temprature is exactly 5,778 K (15 million degrees Celsius), and this fine-tuned sun has many changes, disturbances, and phenomena that keep occurring in and around the Sun's atmosphere, generally referred to as Solar activity. It has several variables that fluctuate even on a fraction of a millisecond.

This solar activity happens when the energy stored in 'twisted' magnetic fields (usually above sunspots) is suddenly released. These outbursts are called "Solar flares". Solar flares are one of the solar activities of the sun. There are four primary forms of solar activity: solar flares, coronal mass ejections, high-speed solar winds, and solar energetic particles.

A powerful Class-X solar flare accompanied by a Coronal Mass Ejection (CME) – a cascade of highly energetic particles accelerated from the sun's corona by magnetic field collapse – could cause catastrophic damage to Earth's ground electronic and orbital satellite infrastructures. These giant eruptions have the capability to wreak havoc on GPS and other satellites, airplane communications, power grids, copper wiring in transformers, and even hand-held modern devices like smartphones, and a carrington event can occur.

What is the Carrington event?

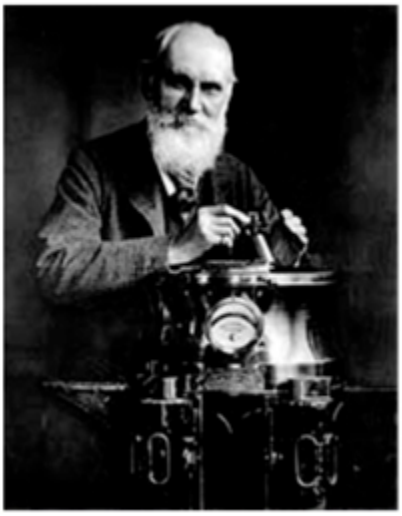
The Carrington Event was a large solar storm that took place at the beginning of September 1859.

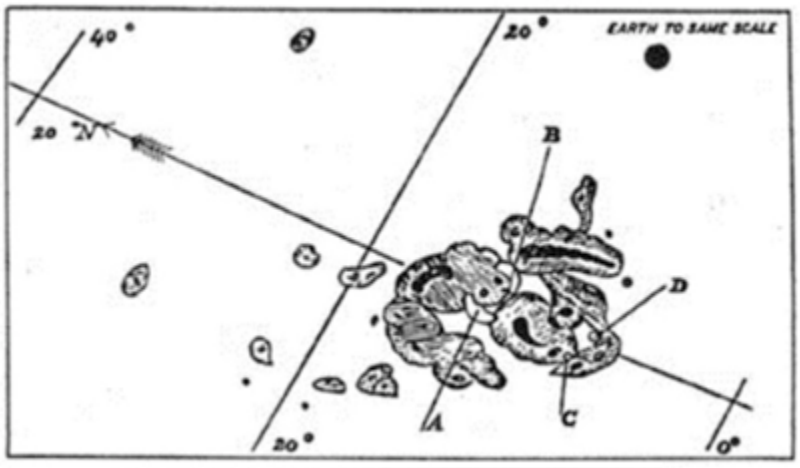
World was in a contrasting situation where people in the northern countries were amazed by the beauty of the aurora lights(Northern lights) and across the earth there was a terrifying storm.

Northern lights : Is one visible effect of solar radiation. Earth's magnetic field, which protects us from most of the dangers of space radiation, directs the charged particles to the poles, where they enter our atmosphere and cause beautiful light displays. in the sky,

In August 1859, astronomers around the world watched with fascination as the number of sunspots on the solar disk grew. Among them was Richard Carrington, an amateur skywatcher in a small town called Redhill, near London in England.

On Sep. 1, as Carrington was sketching the sunspots, he was blinded by a sudden flash of light. Carrington described it as a "white light flare" according to NASA spaceflight. The whole event lasted about five minutes.



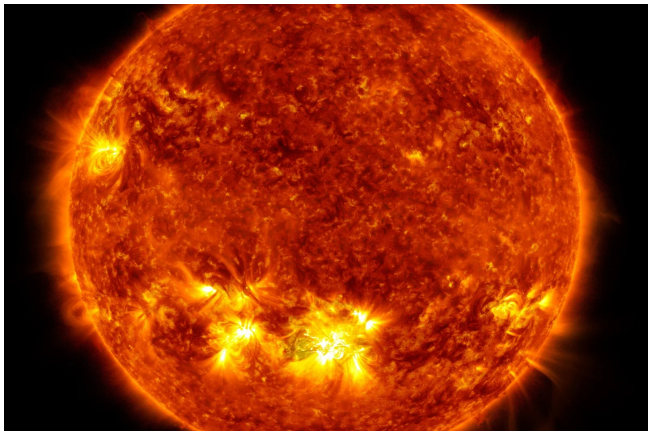


Solar sketch, September 1, 1859, by R. C. Carrington

What are solar flares?

Solar flares are short-term outbursts on the sun, caused by the sudden release of energy stored in twisted magnetic fields in the solar atmosphere. Flares are more contained than coronal mass ejections but still release up to 1025 joules of energy—the energy equivalent of ten million volcanic eruptions. They can last just a few minutes or up to several hours.

Flares occur in active regions, areas on the sun where the magnetic fields are very strong. In this ultraviolet light image of the sun on the right, the active regions stand out as bright spots or clusters of spots. (In visible light images of the sun, the dark spots you see—called sunspots—are active regions.) By the way, the green color isn't real: It's the color chosen to represent ultraviolet light, which is invisible to our eyes.

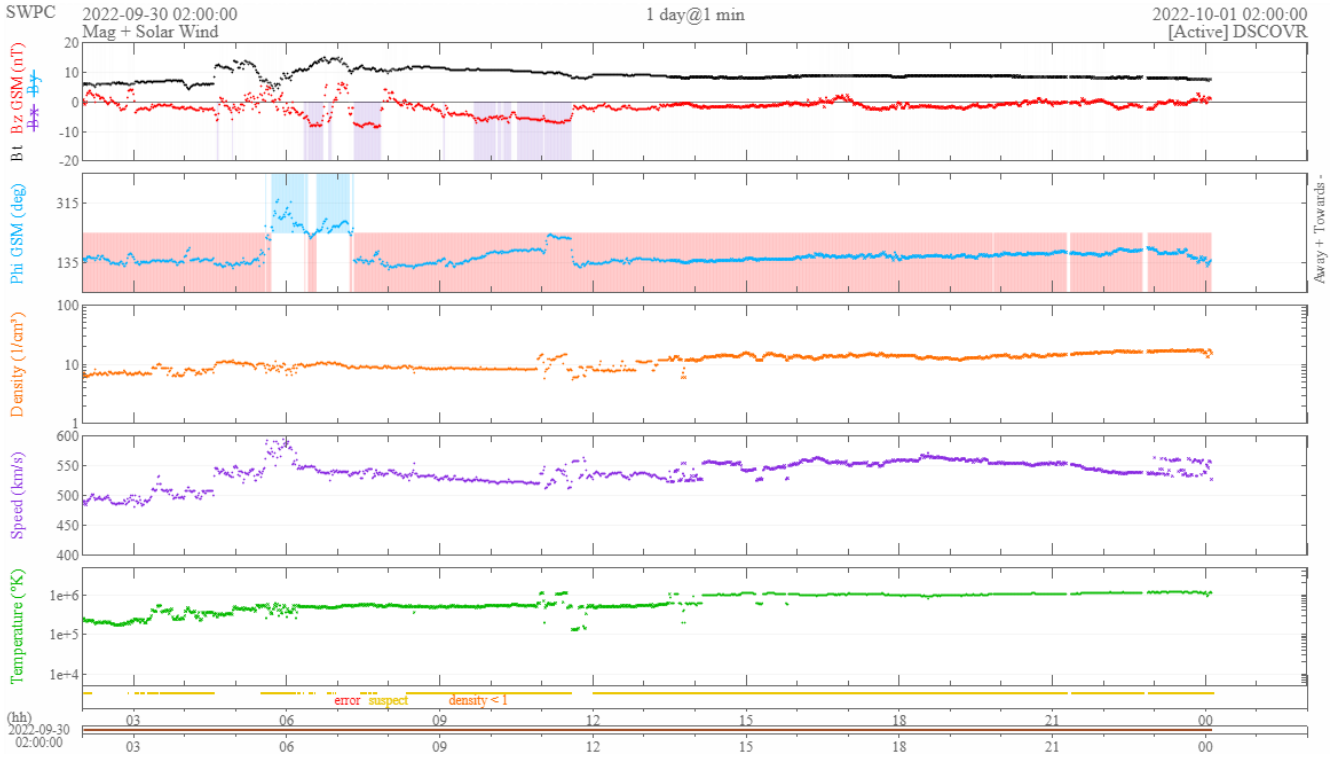


What are solar winds:

The solar wind is a continual stream of protons and electrons from the sun's outermost atmosphere — the corona.

These charged particles breeze through the solar system at speeds ranging from around 250 miles (400 kilometers) per second to 500 miles (800 km) per second, in a plasma state, according to the National Oceanic and Administration Space Weather Prediction Center(opens in new tab) (SWPC).

REAL TIME SOLAR WIND today 1/10/22:



Prediction of solar activity:

Accurate predictions of solar activity are essential for satellite tracking and constellation management, orbit lifetime studies, and mission planning for long-term orbiting platforms, such as the proposed space station Freedom.

But in our case, Intense solar flares and CMEs can be a cause for total technological failure, with a huge risk for lives like airline passengers and crew, astronauts in the space with a failure for space infrastructure, and satellites, that could cause up to $2.6 trillion nowadays in damage. That is why humans after the last Carrington event in 1859 tried to predict the solar activities.

In the previous attempts to predict solar activities such as solar flare, It was based on observations of sunspots, estimating flare sizes based on sunspot and magnetic field properties. In a paper published in 2019, researchers from Nagoya University and the National Astronomical Observatory of Japan describe "a physics-based method that can predict future large solar flares."

The method is based on the discovery that relatively small reconfigurations of the Sun's magnetic field lines can cause significant instabilities, just as small cracks in mountain snowpacks can cause avalanches.

Solar Flare Occurrence Process in Physics-Based Prediction Method. (a) Current flows along magnetic field lines that cross the reversal lines on the Sun's surface where the magnetic field changes polarity. (b) The field lines reconnect and form a double arc loop moving away from the surface due to magnetohydrodynamic instability. (c) The upward motion of the double arc loop induces another magnetic reconnection. Solar flares begin to erupt at the base of the reconnected magnetic field lines. (d) Furthermore, magnetic reconnection amplifies the instability and expands the solar flare.

"If the torsional flux density is strong near the polarity reversal line, small-scale reconnections can trigger eruptions," says Kusano, lead author of the study and director of the Institute for

Space-Earth Environmental Research at Nagoya University. Hiroya says. "Twisted magnetic flux density is the source of instability that can cause flare.

But the physical solutions and methods of predicting solar flares have a big error rate starting from the instrumental errors to the applications errors, that makes them neither enough nor reliable. So we decided to use the power of data science to predict the Carrington Event based on collected data since 1995.

Our model is a LSTM model that predicts the Dst, which is the measure of the severity of the geomagnetic storm.

Data Analysis

Introduction to the Dataset

The data is made up of solar wind measurements acquired by two satellites: NASA's Advanced Composition Explorer (ACE) and NOAA's Deep Space Climate Observatory (DSCOVR).

Datasets

Geomagnetic storms form Due to the transfer of the energy from solar wind to the earth's magnetic field. The magnetic navigation errors are increased by the resulting magnetic field changes. The geomagnetic storm's intensity is measured by the disturbance-storm-time index, or Dst.

Solar Wind Data

Solar wind.csv contains the major feature data. They are made up of data from the ACE and DSCOVR satellites on solar wind:

bx_gse	Interplanetary-magnetic-field (IMF) X-component in geocentric solar ecliptic (GSE) coordinate (nanotesla (nT))
by_gse	Interplanetary-magnetic-field Y-component in GSE coordinate (nT)
bZ_gse	Interplanetary-magnetic-field Z-component in GSE coordinate (nT)
theta_gse	Interplanetary-magnetic-field latitude in GSM coordinates (degrees)
phi_gse	Interplanetary-magnetic-field longitude in GSM coordinates (degrees)
bx_gsm	Interplanetary-magnetic-field component magnitude (nT)
by_gsm	Interplanetary-magnetic-field Y-component in GSM coordinate (nT)
bz_gsm	Interplanetary-magnetic-field Z-component in (GSM) coordinate (nT)
theta_gsm	Interplanetary-magnetic-field latitude in GSM coordinates (degrees)
phi_gsm	Interplanetary-magnetic-field longitude in GSM coordinates (degrees)

bt	Interplanetary-magnetic-field component magnitude (nT)
density	Solar wind proton density (N/cm^3)
speed	Solar wind bulk speed (km/s)
temperature	Solar wind ion temperature (Kelvin)
source	Starting in 2016, the solar wind data for any given point in time can be sourced from either DSCOVR or ACE satellites depending on the quality. "ac" denotes it was sourced from ACE, and "ds" from DSCOVR.

Exploration of Solar_wind data set:

We have nearly 140,000 observations of hourly dst data, representing over 15 years.

period	timedelta	bx_gse	by_gse	bz_gse	theta_gse	phi_gse	bx_gsm	by_gsm	bz_gsm	theta_gsm	phi_gsm	bt	density	speed	temperature	source
train_a	0 days 00:00:00	-5.55	3.00	1.25	11.09	153.37	-5.55	3.00	1.25	11.09	153.37	6.80	1.53	383.92	110237.0	ac
	0 days 00:01:00	-5.58	3.16	1.17	10.10	151.91	-5.58	3.16	1.17	10.10	151.91	6.83	1.69	381.79	123825.0	ac
	0 days 00:02:00	-5.15	3.66	0.85	7.87	146.04	-5.15	3.66	0.85	7.87	146.04	6.77	1.97	389.11	82548.0	ac
	0 days 00:03:00	-5.20	3.68	0.68	6.17	146.17	-5.20	3.68	0.68	6.17	146.17	6.74	1.97	389.11	82548.0	ac
	0 days 00:04:00	-5.12	3.68	0.49	4.62	145.72	-5.12	3.68	0.49	4.62	145.72	6.65	1.77	384.26	94269.0	ac

There are almost twice as many observations in either the train_b or train_c periods than there are in train_a. It also seems train_a represents a more intense period, given that it has a lower mean and higher standard deviation. Also note that most of the values are negative.

	bx_gse								by_gse		...	speed					
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std	
period																	
train_a	1575012.0	-1.781301	4.339212	-54.63	-4.96	-2.01	1.55	41.13	1575012.0	0.406389	...	484.13	1011.50	1396624.0	105917.432742	100098.319077	10
train_b	3084130.0	-0.308879	3.627830	-29.37	-3.07	-0.45	2.42	25.07	3084130.0	0.159068	...	481.95	1198.49	3036895.0	136424.705063	149788.503256	1
train_c	3407290.0	-0.461908	3.245485	-45.46	-2.80	-0.42	1.87	55.55	3407290.0	-0.071924	...	490.60	1064.00	3147033.0	98588.925913	89556.600276	

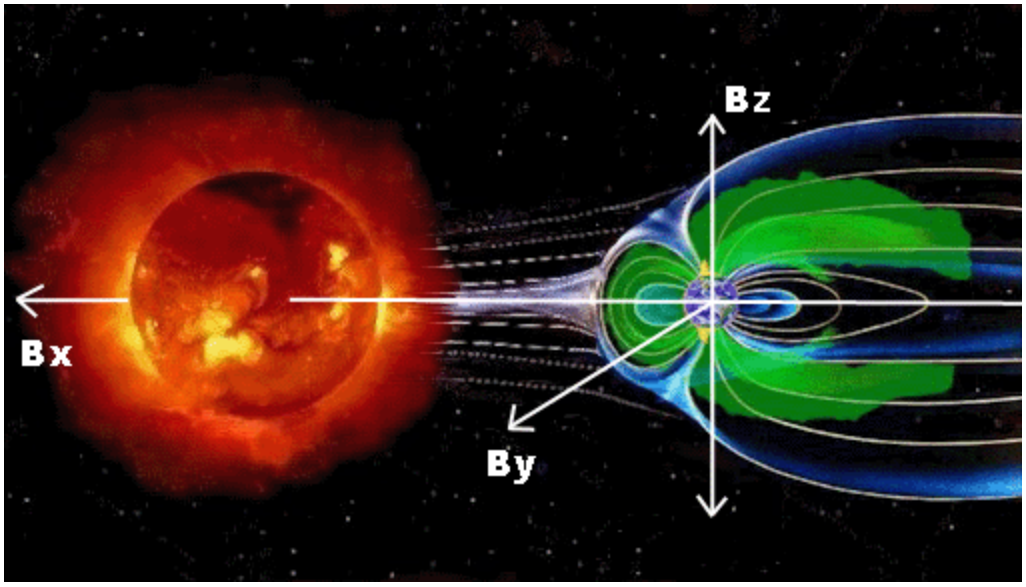
A very strong magnetic field disturbance has a large Dst value, measured in nano-Teslas (nT). Because these disturbances are usually flowing towards the Earth, the values are negative.

		dst
period	timedelta	
train_a	0 days 00:00:00	-7
	0 days 01:00:00	-10
	0 days 02:00:00	-10
	0 days 03:00:00	-6
	0 days 04:00:00	-2

Sometimes Dst can be highly positive. During calm conditions, Dst values are situated at or just below 0.

	dst							
	count	mean	std	min	25%	50%	75%	max
period								
train_a	28824.0	-16.576707	26.083191	-387.0	-26.0	-12.0	-1.0	65.0
train_b	52584.0	-9.695154	16.443049	-223.0	-17.0	-7.0	1.0	59.0
train_c	58464.0	-9.556325	16.506404	-374.0	-16.0	-7.0	0.0	67.0

The Interplanetary Magnetic Field (IMF) components:



Satellite Data

The ACE and DSCOVR satellites move around. They effectively move in an orbit around the L1 point, maintaining a largely constant distance from the Earth as it circles around the sun.

For projections in the XY, XZ, and YZ planes, satellite pos.csv keeps track of the daily locations of the DSCOVR and ACE satellites in geocentric solar ecliptic (GSE) coordinates. The suffixes _ace or _dscovr designate the columns for each spacecraft.

gse_x	Position of the satellite in the X direction of GSE coordinates (km)
gse_y	Position of the satellite in the Y direction of GSE coordinates (km)
gse_z	Position of the satellite in the Z direction of GSE coordinates (km)

Sunspots Data

The solar cycle, which lasts around 11 years, describes the Sun's well-known, cyclical change in the number of spots on its surface. Large geomagnetic storms tend to happen more frequently when these cycles are at their strongest. Sunspot counts could make it possible to calibrate models to the solar cycle.Sunspots are indexed in labels.csv according to the first corresponding day.

Sunspots in the sunspots data that is read in a timedelta of 13 days per reading:

		smoothed_ssn
period	timedelta	
	0 days	65.4
	13 days	72.0
train_a	44 days	76.9
	74 days	80.8
	105 days	85.4

Count, Mean and std for sunspots.

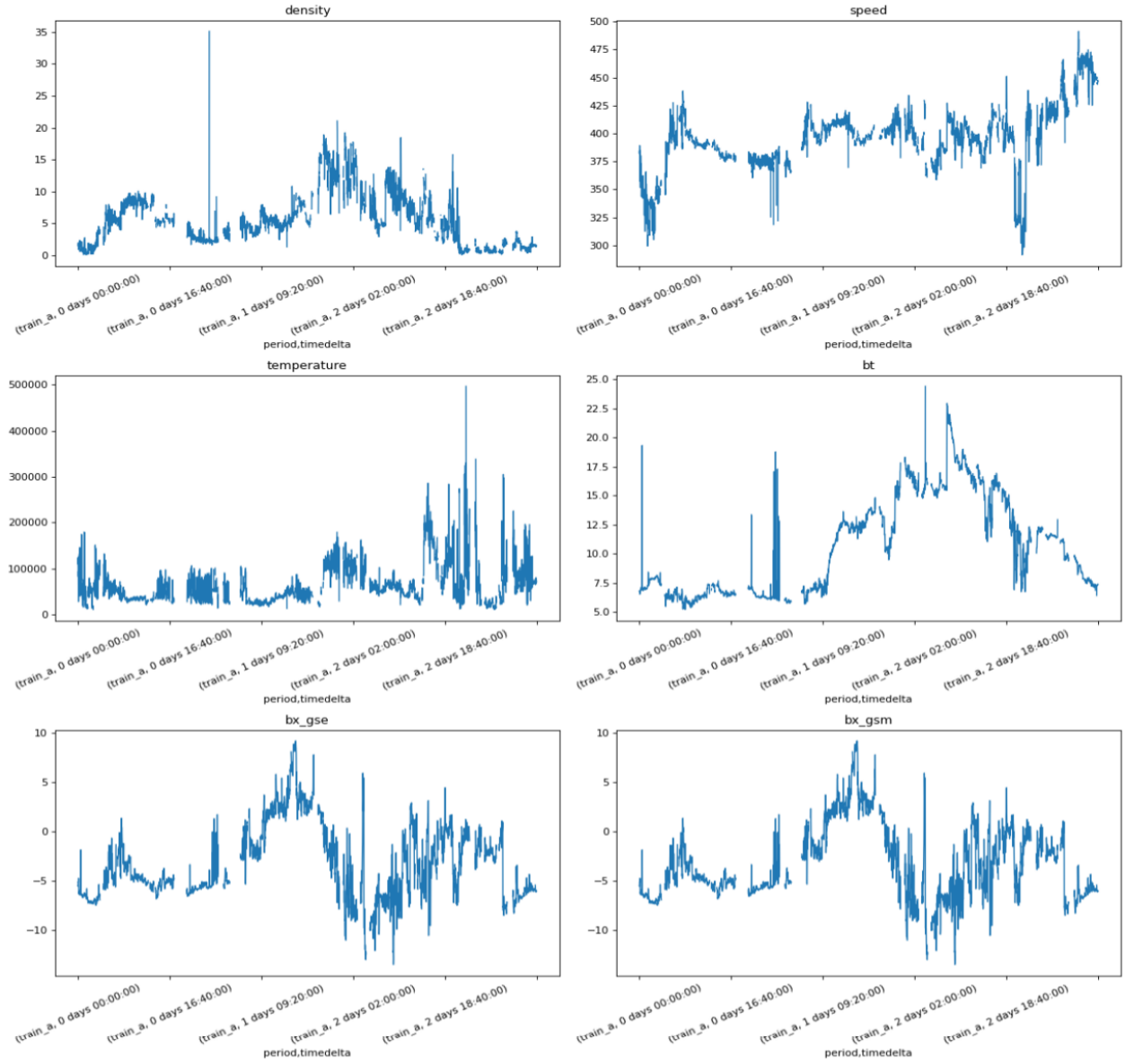
							smoothed_ssn	
	count	mean	std	min	25%	50%	75%	max
period								
train_a	40.0	136.90250	34.563168	65.4	108.375	151.50	164.400	175.2
train_b	72.0	51.85000	39.200266	3.9	15.325	43.15	91.225	116.4
train_c	80.0	24.31375	19.020414	2.2	7.775	20.50	38.525	69.5

Labels

The labels are indexed hourly Dst values with the using of a timedelta multi-index and the same period. It is intended to forecast both the current timestep (t0) and the subsequent timestep (t+1).

EDA

Plotting some features of the solar wind data like density, temperature, magnetic field:



As we can see, the temperature gets very high, but on the other hand magnetic fields go low to the negative values. And in a deep learning model, to avoid any issues in the model, it's best to scale these features, to be easily modeled, as we will discuss later on.

Distribution of Null values among features:

```
bx_gse          325888
by_gse          325888
bz_gse          325888
theta_gse       325888
phi_gse         326388
bx_gsm          325888
by_gsm          325888
bz_gsm          325888
theta_gsm       325888
phi_gsm         326388
bt              325888
density         684890
speed           689555
temperature     811768
source          316816
dtype: int64
```

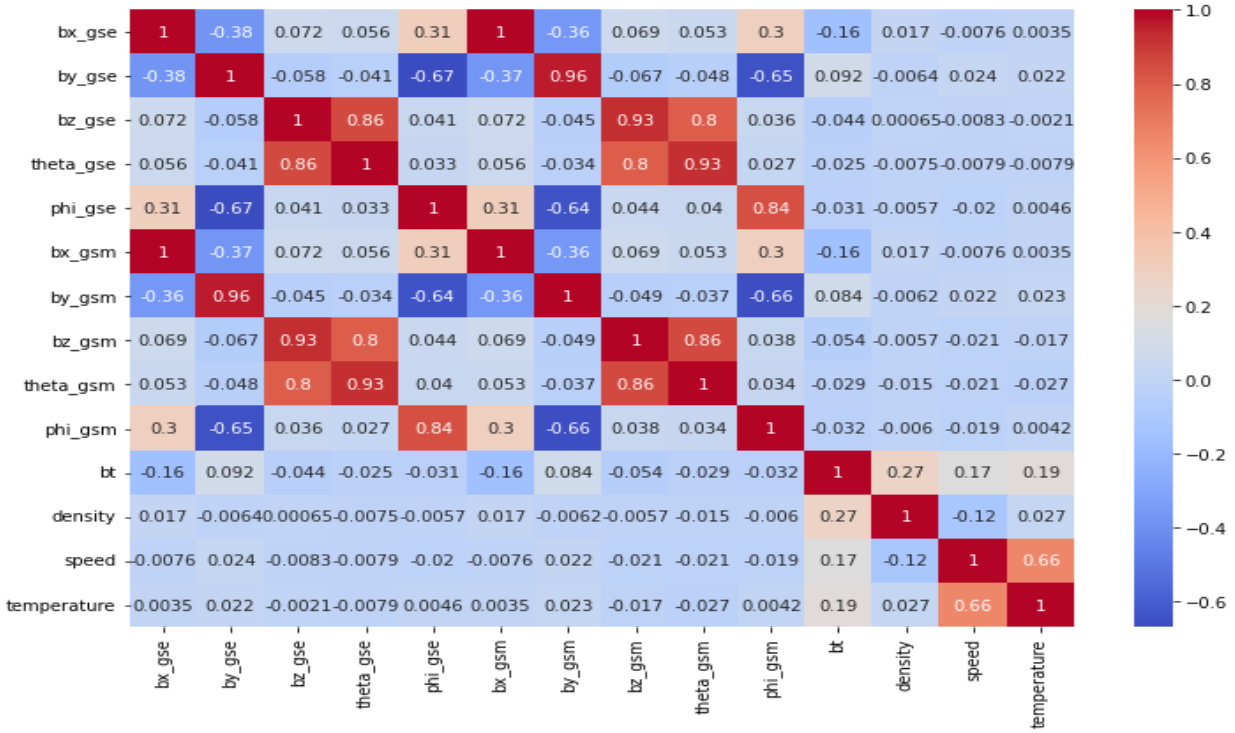
Just like any device error, DSCOVR makes some errors in its readings, the first one we noticed is the null reading, and as we can see there is a lot of missing data, and we will solve this later on in our thesis.

Correlations between features:

We plot a heatmap that draws the colinearity and correlations between features.

To understand the correlation heatmap: it creates a relation between 2 variables and checks if it's proportional(higher than 0.5) or inversely proportional (less than -0.5) , or no relation (close to zero)

Understanding the correlation between columns helps you make a better model as the insertion of lesser important columns will actually cause bias or corrupt the model.



We found this:

- IMF magnetic field components features bx_gsm and bx_gse are positively correlated.
- bz_gse and theta_gse are positively correlated.
- bz_gse and bz_gsm are positively correlated.
- Phi_gsm and phi_gse are positively correlated.

Data preprocessing

Feature selection:

After plotting the heatmap and finding the highly correlated features, we only select one of every two features with high multicollinearity and drop the other, this is done by VIF or normally by heatmap, and because our features aren't too many, it is okay to use the heatmap method.

We selected "bt" , "bx_gse" , "by_gse" , "bz_gse" , "temperature" , "speed" , "density" features as a subset for solar wind features. After selecting them we calculate "Mean, Std, sunspot numbers" for each one in a new dataframe called XCOLS.

Flooring: Aggregating the two datasets:

First of all, Aggregating the two datasets we will convert the solar wind data set of minutes to the sunspot hourly data. Utilizing the mean and standard deviation, aggregates features to the hourly floor. For instance, "10:00:00" will be the total of all values from "10:00:00" to "10:59:00."

Imputing missing values:

For missing values, we will Imputes data using the following methods:

- For Solar_wind : Interpolation between missing solar wind values.
- For Smoothed_ssn : Forward fill for the rest of the month.

Scaling

After plotting the features of solar wind data we found the huge scale difference between some features, in order to solve this we will use StandardScaler.

StandardScaler follows Standard Normal Distribution (SND). Therefore, it makes mean = 0 and scales the data to unit variance.

Feature engineering:

First feature:

To predict current time (present) and upcoming time(future), we created a new column for future called (future) that shifts the upcoming hour to a new column called Future.

Second feature:

A new time series feature that iterates through time periods and groups them by period.

period	timedelta	bt_mean	bt_std	bx_gse_mean	bx_gse_std	by_gse_mean	by_gse_std	bz_gse_mean	bz_gse_std	temperature_mean	temperature_std	s
train_a	0 days 00:00:00	0.499705	2.443614	-1.599207	-0.381502	0.419516	0.031658	0.300358	-0.651645	-0.375267	0.383941	
	0 days 01:00:00	0.547177	-0.224580	-1.757995	-0.867747	0.179257	-0.272971	0.446103	-0.517913	-0.479430	0.953178	
	0 days 02:00:00	0.739905	-0.770240	-1.912116	-1.114317	0.183266	-0.822786	0.770174	-0.876490	-0.574831	-0.192518	
	0 days 03:00:00	0.699098	-0.278783	-1.809045	-0.783042	-0.378111	0.341156	0.621194	-0.290211	-0.324709	0.325491	
	0 days 04:00:00	0.223933	-0.225168	-1.338802	-0.484910	0.072745	1.023019	0.467629	-0.478080	-0.313432	0.201600	

Prediction:

Make predictions for times t and t+1 using all of the data up to time t-1.

Parameters:

inputs :

1- solar_wind_7d: pd.DataFrame

The last 7 days of satellite data up until (t - 1) minutes [exclusive of t]

2- latest_sunspot_number: float

The latest monthly sunspot number (SSN) to be available

Returns :

predictions : Tuple[float, float]

A tuple of two forecasts for (now) and (future), respectively.

Process:

1) Re-formatting the sunspots data into.

2) Processing our features and grab last 32 (timesteps) hours.

Now our final feature set is composed of 15 features - the mean and standard deviation of seven solar_wind features, along with smoothed_ssn. We've also saved our scaler object. We'll serialize this later along with our model so that it can be used to preprocess features during prediction.

Before we start modeling, we also have to reshape our label, DST. we have to predict both t0, the current timestep, and t+1, an hour ahead. We'll train our model to do multi-step prediction by providing it with both steps. To do that, we just add another column called t1 that is dst shifted by 1. We'll also rename our DST column to t0 for consistency.

period	timedelta	now	future
train_a	0 days 00:00:00	-7	-10.0
	0 days 01:00:00	-10	-10.0
	0 days 02:00:00	-10	-6.0
	0 days 03:00:00	-6	-2.0
	0 days 04:00:00	-2	3.0

We join this output with our dataframe:

period	timedelta	now	future	bt_mean	bt_std	bx_gse_mean	bx_gse_std	by_gse_mean	by_gse_std	bz_gse_mean	bz_gse_std	temperature_mean	temp
train_a	0 days 00:00:00	-7	-10.0	0.499705	2.443614	-1.599207	-0.381502	0.419516	0.031658	0.300358	-0.651645	-0.375267	
	0 days 01:00:00	-10	-10.0	0.547177	-0.224580	-1.757995	-0.867747	0.179257	-0.272971	0.446103	-0.517913	-0.479430	
	0 days 02:00:00	-10	-6.0	0.739905	-0.770240	-1.912116	-1.114317	0.183266	-0.822786	0.770174	-0.876490	-0.574831	
	0 days 03:00:00	-6	-2.0	0.699098	-0.278783	-1.809045	-0.783042	-0.378111	0.341156	0.621194	-0.290211	-0.324709	
	0 days 04:00:00	-2	3.0	0.223933	-0.225168	-1.338802	-0.484910	0.072745	1.023019	0.467629	-0.478080	-0.313432	

Modeling :

Splitting data:

We want to split our data into three datasets. As you might have guessed, the train set will be used for training. We'll also pass a validation set to keras as it's training to monitor the modeling fitting over epochs. Finally, we'll use a test set to evaluate our model for over or under fitting before we submit to the competition.

We have two atypical concerns to consider when splitting this dataset:

We're dealing with time series data. Observations in a time series are not independent, so we cannot randomly assign observations across our datasets. We also don't want to "cheat" by leaking future information into our training data. In the real-world, we will never be able to train on data from the future, so we should emulate those same constraints here.

We have three non-contiguous periods, meaning we have gaps in our data. We don't know how long each gap is or in what order each period occurred. We also know that the three periods are differently distributed. That suggests that observations from each period should be included in our train set, instead of reserving one wholesale as our test or validation set.

To solve these problems, we'll hold out the last 6,000 rows from each period for our test set, and reserve the last 3,000 before that for our validation set. The remaining rows will be used in the training set.

Sequences and batches:

The first thing we have to do is separate our data into sequences and batches for modeling. We have to decide on:

timesteps: this determines the sequence length, ie. how many timesteps in the past to use to predict each step at t0 and t1. Our data is aggregated hourly, so timesteps is equal to the number of hours we want to use for each prediction. batch_size: this determines the number of samples to work through before a model's parameters are updated. For this tutorial, we'll choose fairly standard numbers of 32 timesteps per sequence and 32 sequences per batch. These numbers will likely have a large impact on your model, so feel free to experiment.

Now we need to use these numbers to separate our training data and labels into batches of sequences that will be fed into the model. Luckily, keras has just the tool with their timeseries_dataset_from_array function. According to the documentation:

If targets were passed, the dataset yields tuples (batch_of_sequences, batch_of_targets). If not, the dataset yields only batch_of_sequences.

So We can easily specify timesteps (referred to as sequence_length in the documentation) and batch size to get a feature generator and a target generator that we can pass to model.fit(). For your implementation, you can experiment with different sequence_lengths along with sequence_stride (how many observations to skip between sequences) and sampling_rate (how many observations to sample per sequence).

There's one hiccup - we need to make sure that our sequences don't span across periods. To get around that, we'll iterate through our periods and generate a time series dataset for each one. Then we'll concatenate them at the end to rejoin our training set and validation set. And let's not

forget - since we're only allowed to use feature data up until t-1, we'll need to realign our features and labels. We'll do that during our loop as well.

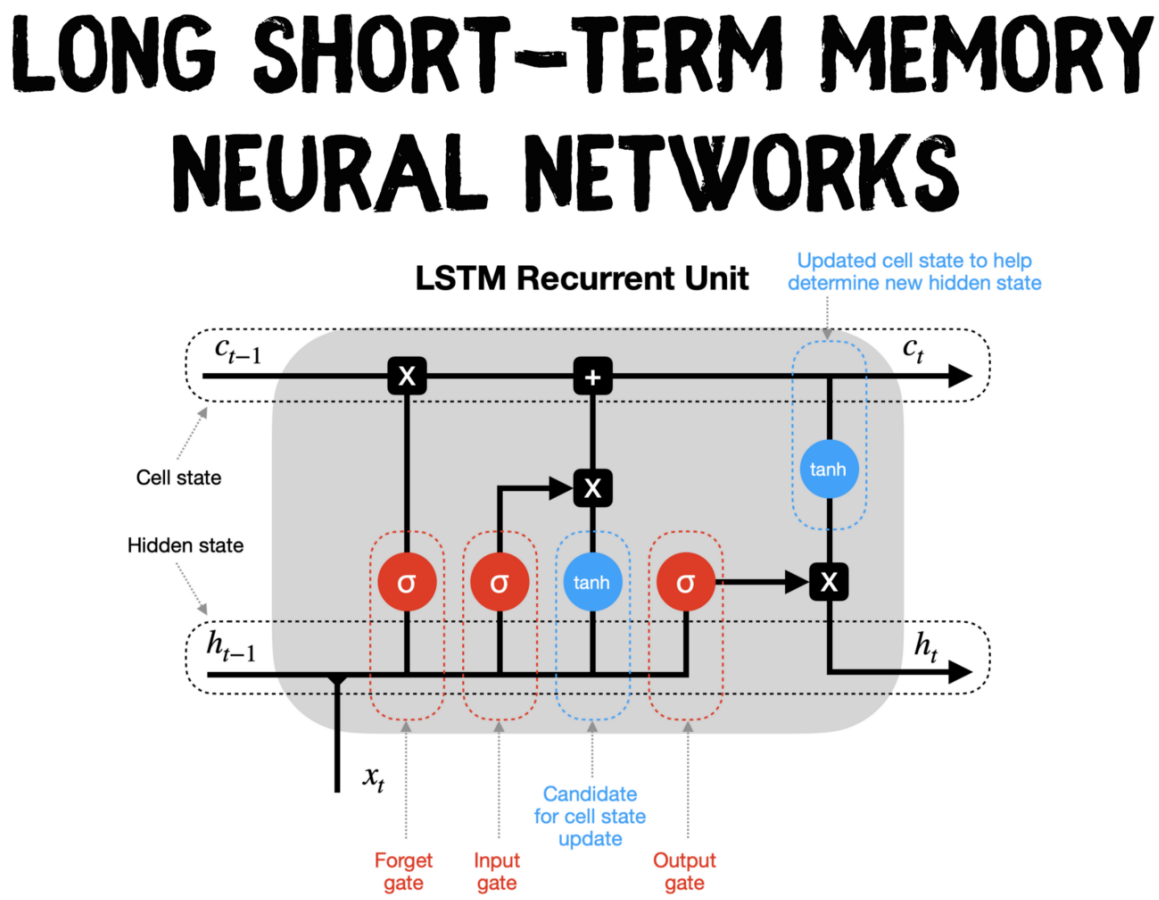
Output:

Number of train batches: 971

Number of val batches: 54

Our models:

For this thesis we used three models for three different algorithms of Neural network based algorithm (UNI and BI LSTM) along with GRU:



LSTM model:

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.

This is a behavior required in complex problem domains like machine translation, speech recognition, and more.

LSTMs are a complex area of deep learning. It can be hard to get your hands around what LSTMs are, and how terms like bidirectional and sequence-to-sequence relate to the field.

To design our LSTM network, We're going to build a simple sequential model with one hidden LSTM layer and one output later with 2 output values (t0 and t1). You can experiment by making your model deep (many layers) or wide (adding more neurons).

There are many hyperparameters that we can tune. We'll concentrate on a few here:

n_epochs: This determines the number of complete passes your model takes through the training data. For this tutorial, we'll choose 20 epochs. You'll want to monitor your rate of convergence to tune this number.

n_neurons: The number of hidden nodes. Usually increases by powers of 2. We'll use 512. dropout: This regularizes by randomly "ignoring" a dropout fraction of a layer's neurons during each pass through the network during training, so that no particular neuron overfits its input. We'll start with a value of 0.4. stateful: This determines whether the model keeps track of the historical data that it sees within each batch. Since each sample within a batch encodes the entire sequence we care about, we can set this to False.

Tuning these values will impact how fast your model learns, whether it converges, and affect over/under fitting. Play around to see what works best.

After instantiating the model with these hyperparameters, we'll compile it with mean_squared_error as our loss function and adam as our optimizer. We'll have to remember to take the square root of our loss to get our competition metric, root_mean_squared_error.

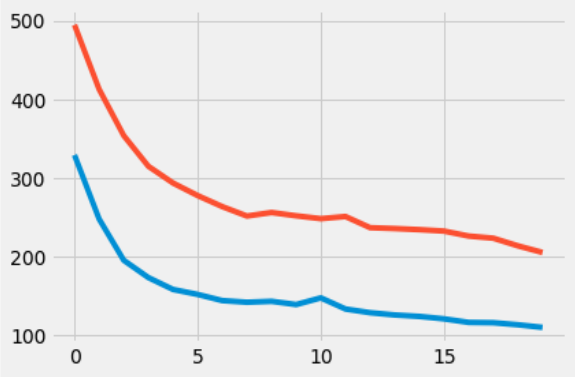
Specs:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 512)               1081344

 dense (Dense)               (None, 2)                 1026

=================================================================
Total params: 1,082,370
Trainable params: 1,082,370
Non-trainable params: 0
_________________________________________________________________
```

1) **Loss**

```
[23] 3804/3804 [==============================] - 36s 9ms/step - loss: 126.1761 - val_loss: 236.0191
     Epoch 15/20
     3804/3804 [==============================] - 36s 9ms/step - loss: 124.3523 - val_loss: 234.6222
     Epoch 16/20
     3804/3804 [==============================] - 35s 9ms/step - loss: 121.2506 - val_loss: 232.7896
     Epoch 17/20
     3804/3804 [==============================] - 36s 9ms/step - loss: 116.7236 - val_loss: 226.4891
     Epoch 18/20
     3804/3804 [==============================] - 36s 10ms/step - loss: 116.3416 - val_loss: 223.7539
     Epoch 19/20
     3804/3804 [==============================] - 36s 9ms/step - loss: 113.7280 - val_loss: 214.0392
     Epoch 20/20
     3804/3804 [==============================] - 36s 9ms/step - loss: 110.3776 - val_loss: 205.6184
```

2) **Model's loss output plot.**



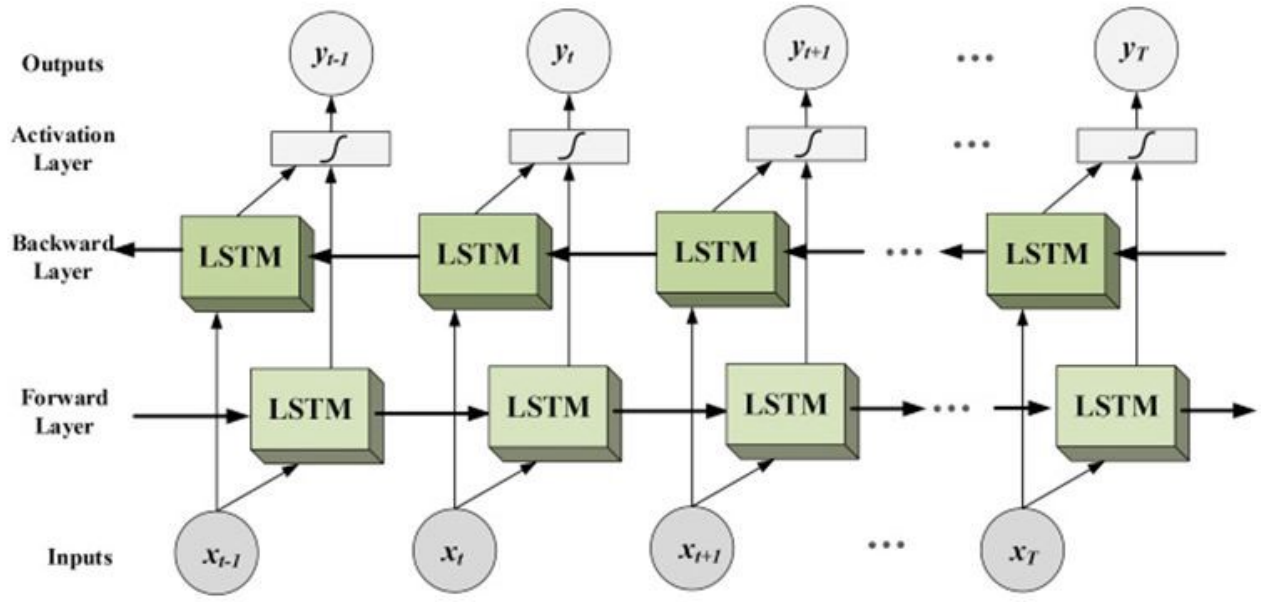
3) Error

```
558/558 [==============================] - 4s 7ms/step - loss: 180.4832
Test RMSE: 13.43
```

Bi-directional LSTM model:

Bidirectional long-short term memory(bi-lstm) is the process of making any neural network to have the sequence information in both directions backwards (future to past) or forward(past to future).

In bidirectional, our input flows in two directions, making a bi-lstm different from the regular LSTM. With the regular LSTM, we can make input flow in one direction, either backwards or forward. However, in bi-directional, we can make the input flow in both directions to preserve the future and the past information.


Outputs
Activation Layer
Backward Layer
Forward Layer
Inputs
LSTM
y_{t-1}
y_t
y_{t+1}
y_T
x_{t-1}
x_t
x_{t+1}
x_T

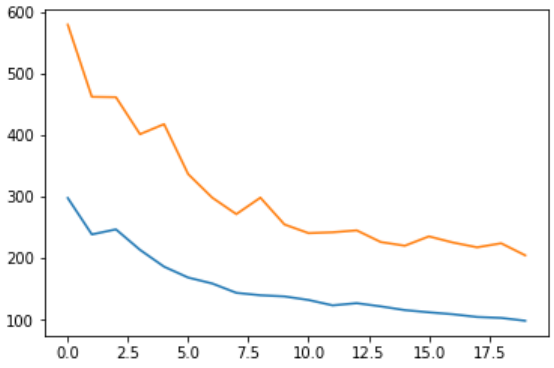
Specs:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional (Bidirectiona  (None, 34, 500)          532000
 l)

 bidirectional_1 (Bidirectio  (None, 34, 256)          644096
 nal)

 bidirectional_2 (Bidirectio  (None, 34, 256)          394240
 nal)

 bidirectional_3 (Bidirectio  (None, 256)              394240
 nal)

 dense (Dense)               (None, 8)                 2056

 dense_1 (Dense)             (None, 2)                 18

=================================================================
Total params: 1,966,650
Trainable params: 1,966,650
Non-trainable params: 0
_________________________________________________________________
```

1) Loss:

```
✓ [24] 3581/3581 [==============================] - 110s 31ms/step - loss: 115.9753 - val_loss: 220.6525
15m    Epoch 16/20
       3581/3581 [==============================] - 111s 31ms/step - loss: 112.3986 - val_loss: 235.8097
       Epoch 17/20
       3581/3581 [==============================] - 112s 31ms/step - loss: 109.2256 - val_loss: 225.8107
       Epoch 18/20
       3581/3581 [==============================] - 111s 31ms/step - loss: 104.8251 - val_loss: 218.0297
       Epoch 19/20
       3581/3581 [==============================] - 109s 31ms/step - loss: 103.2729 - val_loss: 224.5892
       Epoch 20/20
       3581/3581 [==============================] - 110s 31ms/step - loss: 98.6594 - val_loss: 204.9773
```

2) **Loss output plot**:



3) Error:

```
525/525 [==============================] - 8s 14ms/step - loss: 161.9143
Test RMSE: 12.72
```

GRU model:

The workflow of the Gated Recurrent Unit, in short GRU, is the same as the RNN but the difference is in the operation and gates associated with each GRU unit. To solve the problem faced by standard RNN, GRU incorporates the two gate operating mechanisms called Update gate and Reset gate.

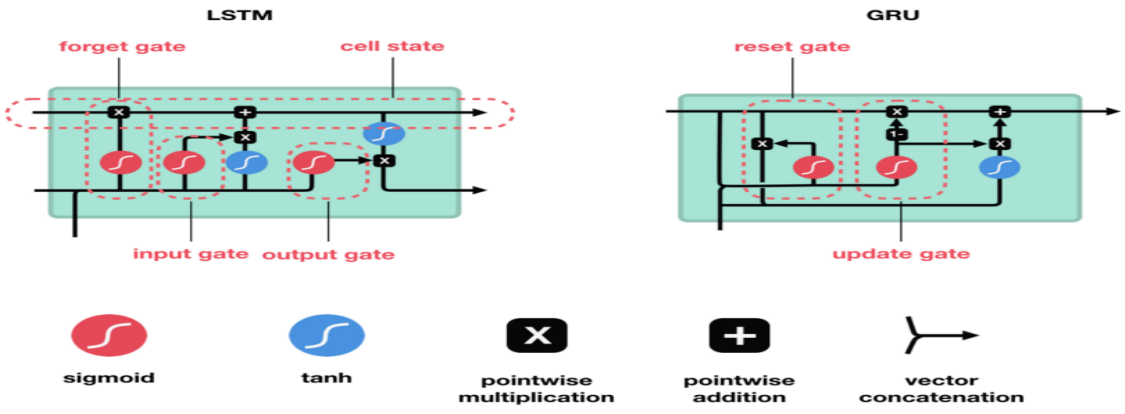
Update gate

The update gate is responsible for determining the amount of previous information that needs to pass along the next state. This is really powerful because the model can decide to copy all the information from the past and eliminate the risk of vanishing gradient.

Reset gate

The reset gate is used from the model to decide how much of the past information is needed to neglect; in short, it decides whether the previous cell state is important or not.

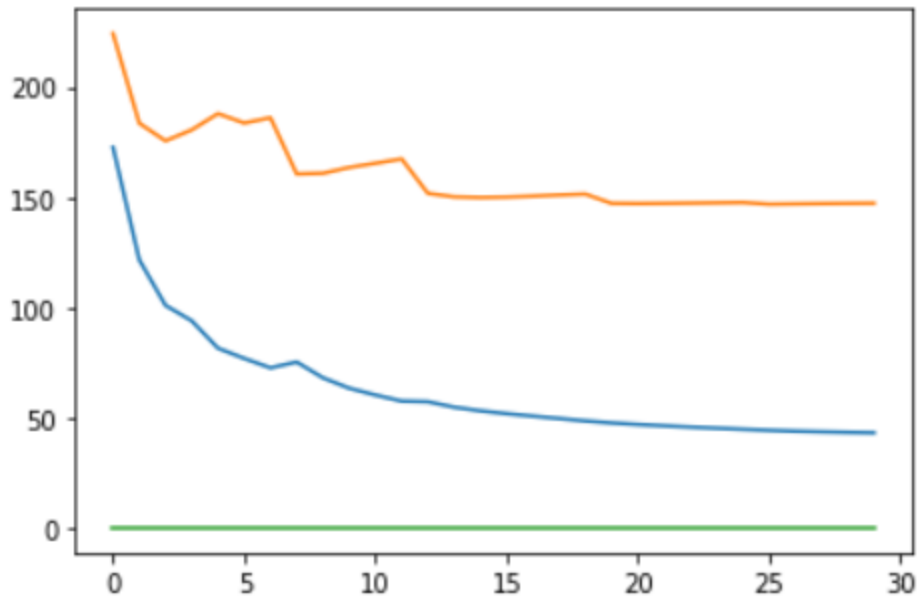
First, the reset gate comes into action it stores relevant information from the past time step into new memory content. Then it multiplies the input vector and hidden state with their weights. Next, it calculates element-wise multiplication between the reset gate and previously hidden state multiple. After summing up the above steps the non-linear activation function is applied and the next sequence is generated.



Specs:

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
x1 (InputLayer)              [(None, 128, 29)]         0
_________________________________________________________________
bidirectional_2 (Bidirection (None, 128, 768)          1271808
_________________________________________________________________
bidirectional_3 (Bidirection (None, 128, 2304)         13284864
_________________________________________________________________
flatten_1 (Flatten)          (None, 294912)            0
_________________________________________________________________
dense_4 (Dense)              (None, 96)                28311648
_________________________________________________________________
dense_5 (Dense)              (None, 128)               12416
_________________________________________________________________
dense_6 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_7 (Dense)              (None, 2)                 130
=================================================================
Total params: 42,889,122
Trainable params: 42,889,122
Non-trainable params: 0
```

Loss:



RESULTS AND DISCUSSION:

Comparing the results obtained from the three experimented models we conclude:

-	LSTM	GRU	BI-LSTM
Advantages	1- Shortest runtime. 2- Simpler model.	1- Lower loss	1- Lowest loss 2- Medium runtime
Disadvantages	1- Highest loss. 2- Unreliable.	1- Complex 2- Longest runtime	1- More complex than Uni.

The BI-LSTM has a longer run time compared to the LSTM, but it provides more reliable results and lower loss**.** GRU on the other hand provides similar results to the BI-LSTM in terms of loss but the downside is that it has a much larger runtime and it is a very complex model.

Due to the reasons mentioned above we decided to go with the BI-LSTM model to predict the DST and give the warning of solar flares we are looking for**.**

This warning is set before an **hour** of the solar flare activity, giving scientists time to prepare for a new carrington event.

REFERENCES:

1) Main resources:

https://ngdc.noaa.gov/dscovr/portal/index.html#/

https://www.ngdc.noaa.gov/geomag/data/geomag/magnet/

2) Rest of references:

https://www.geeksforgeeks.org/standardscaler-minmaxscaler-and-robustscaler-techniques-ml/

https://analyticsindiamag.com/complete-guide-to-bidirectional-lstm-with-python-codes/#:~:text=Bidirectional%20long%2Dshort%20term%20memory(bi%2Dlstm)%20is,different%20from%20the%20regular%20LSTM.

https://analyticsindiamag.com/lstm-vs-gru-in-recurrent-neural-network-a-comparative-study/

https://www.space.com/22215-solar-wind.html

https://www.exploratorium.edu/spaceweather/flares.html

http://www.pas.rochester.edu/~blackman/ast104/wind.html

https://www.esa.int/Science_Exploration/Space_Science/What_are_solar_flares

https://theconversation.com/why-we-need-to-get-better-at-predicting-space-weather-157630

https://www.space.com/the-carrington-event

Inspiration:

https://www.kaggle.com/code/arashnic/eda-prep-and-keras-lstm/notebook