IIT INDORE
ONLINE RESEARCH INTERNSHIP REPORT

on

AI/ML in Space Weather

From July 15, 2021 to August 15, 2021

Submitted by

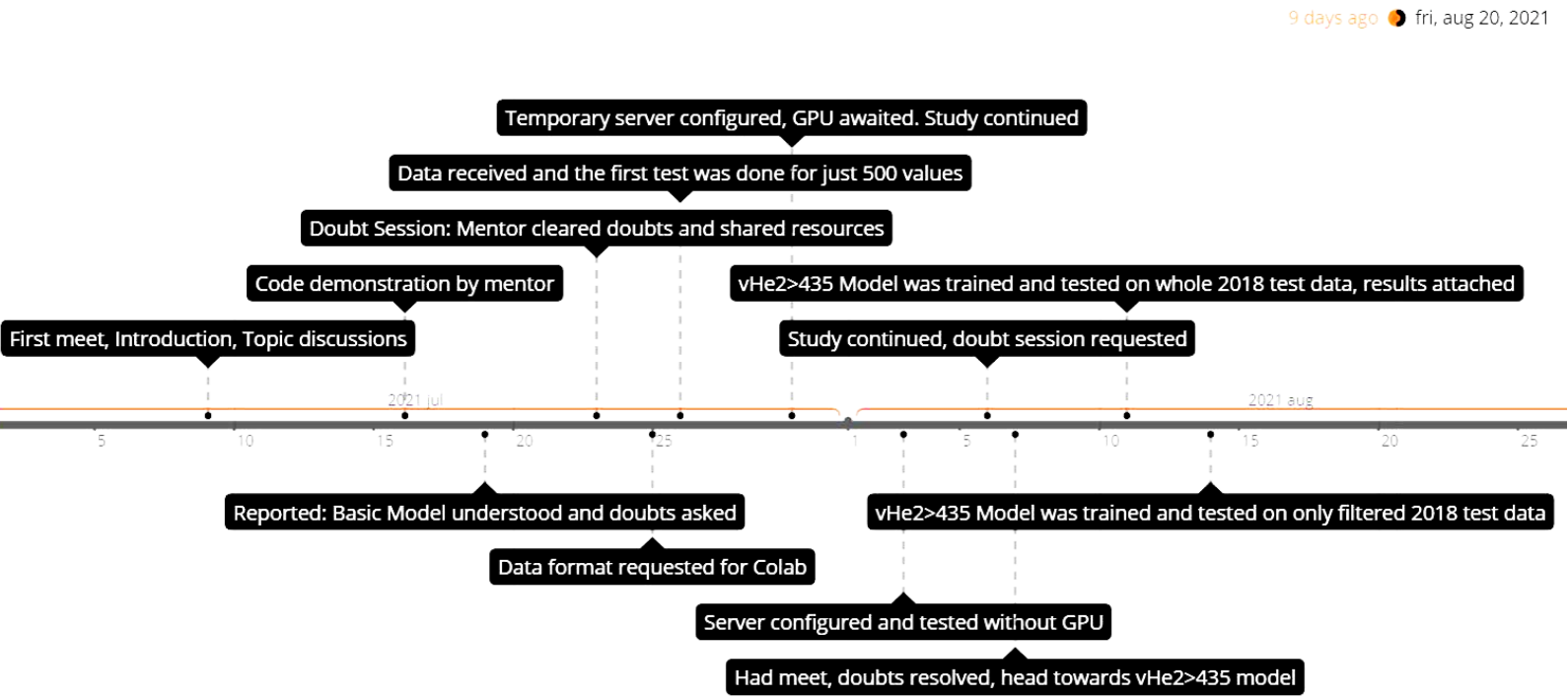
Bhavya Tyagi
Bachelor of Engineering
Computer Science and Engineering (expected 2023)
Thapar Institute of Engineering & Technology
Patiala, Punjab-147 004, India

Under the mentorship of

Dr. Saurabh Das
Assistant Professor
Department of Astronomy, Astrophysics and Space Engineering
Indian Institute of Technology Indore
Khandwa Road, Simrol Indore-453 552, India

TABLE OF CONTENTS

TIMELINE



9 July: First meet, Introduction, Topic discussions
16 July: Code demonstration by mentor
19 July: Reported: Basic Model understood and doubts asked
23 July: Doubt Session: Mentor cleared doubts and shared resources
25 July: Data format requested for Google Colab
26 July: Data received, and the first test was done for first 500 values
30 July: Temporary server configured, GPU awaited. Study continued
3 August: Server configured and tested without GPU
6 August: Study continued, doubt session requested
7 August: Had meet, doubts resolved, head towards vHe2>435 model
11 August: **vHe2>435 (mean) Model was trained and tested on whole 2018 test data, results attached**
14 August: vHe2>435 (mean) Model was trained and tested on only filtered 2018 test data
16 August - 1 Sept: Implemented different architectures

ABSTRACT

Solar wind that emerges out of the Sun's corona fills the interplanetary medium with a magnetized stream of charged particles. The interaction of Solar wind with the Earth's magnetosphere has space weather consequences such as geomagnetic storms. They can also disrupt the communication and navigation that happen here on earth. Predicting the solar wind through measurements of the evolving conditions both spatially and temporally in the solar atmosphere accurately is essential. Still, the prediction of high-speed solar winds efficiently and accurately remains a problem in heliophysics and space weather research.

This work uses Deep Learning techniques to predict solar wind properties. We use extreme ultraviolet images of the solar corona from space-based observations to predict the Solar Wind speed from the National Aeronautics and Space Administration (NASA), ACE data set, measured at Lagrangian Point 1. We evaluate our model against the preexisting model and alter the architectures resulting in root mean square errors ranging from 76-85 km/s.

This report briefs about the work of altering the architectures that were aimed to improve the pre-existing model's accuracy in predicting the high-speed events efficiently.

We also split data into two streams consisting of high-speed events and low-speed events and evaluated the model against it. In this approach, one can improve the results of an underpredicting model as the model can focus on the different speed groups. Later, these models can be combined using the techniques available. Such an approach opens up a different way of thinking and a variety of conclusions.

OBJECTIVE

Currently, I am pursuing a Bachelor of Engineering in Computer Science and Engineering From Thapar Institute of Engineering & Technology Patiala. The major objective of the online internship was to test my interest in the said field and develop the skills alongside while working on an industry-level project. The goal is to strengthen teamwork, technical skills, communication skills and build a more robust career ahead.

The objective of doing research on such underestimated topic, Prediction of Solar Wind & use of AI/ML in space weather is to predict the High-Speed Events of Solar Wind in which plasma emerges out of the sun that can disrupt communications, navigation systems, and satellites through data already available with us and our learnings from the past events. The ultimate goal is to predict some kind of catastrophe or loss of resources that could be avoided or prevented or to reduce the impact if possible.

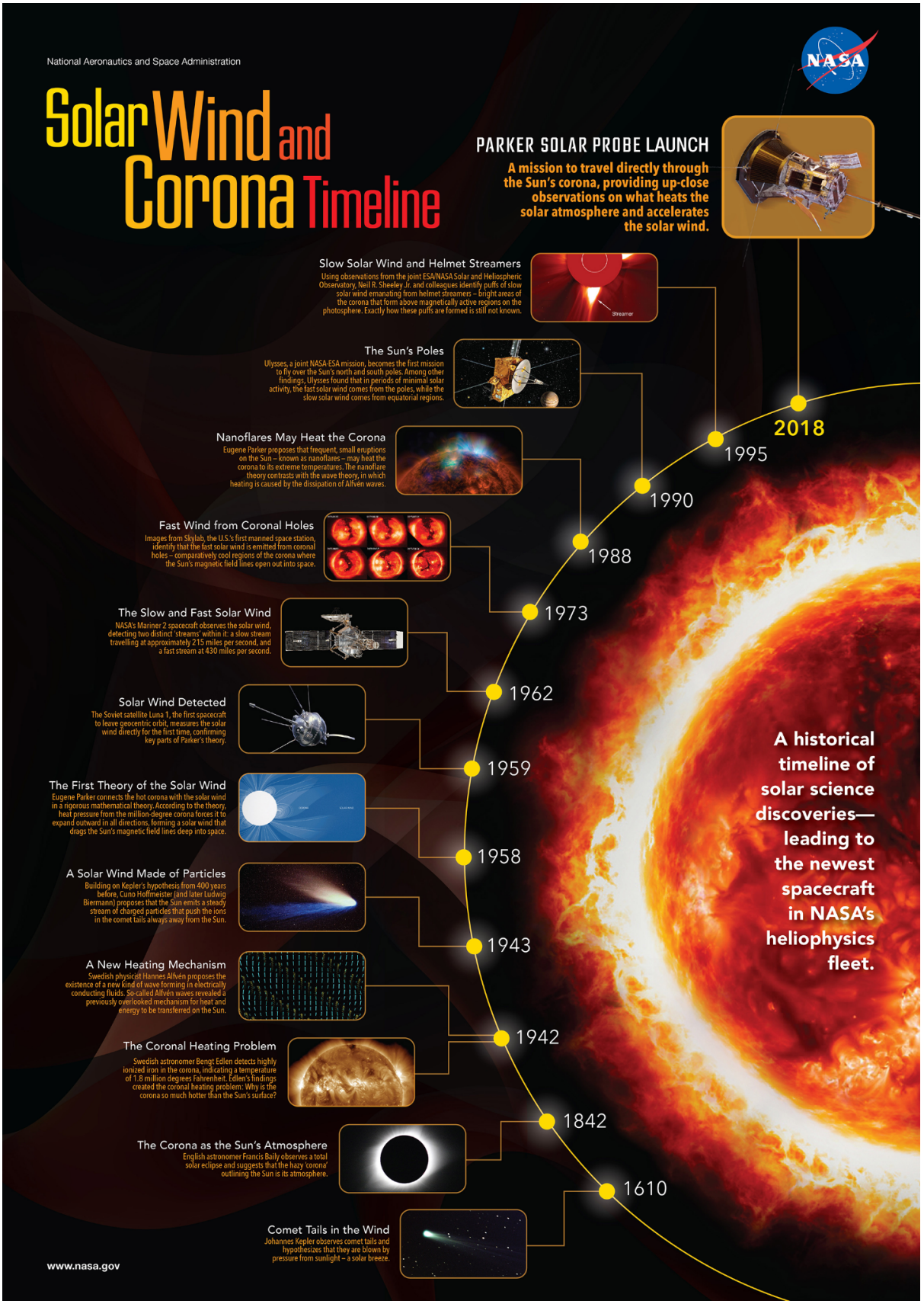
This report will elaborate on my internship experiences on solar wind prediction and my experiments around it. Thank you to IIT Indore for the opportunity and the resources required. Thank you to my mentors, Dr. Saurabh Das from the Department of Astronomy, Astrophysics and Space Engineering, IIT Indore and Ms. Hemapriya, PhD Scholar, IIT Indore who helped me a lot in my journey to achieve my sub-objectives from time to time.

1. INTRODUCTION

1.1 Solar Wind:

The solar wind streams out plasma and particles from the Sun into space. Since the solar wind properties are not constant and evolving continuously throughout 1AU hence it is essential to understand solar wind in order to predict it. This plasma mainly consists of electrons, protons and alpha particles with kinetic energy between 0.5 and 10 keV. The composition of the solar wind plasma also includes trace amounts of heavy ions and various atomic nuclei. Superposed with the solar wind plasma is the interplanetary magnetic field. The solar wind varies in density, temperature, and speed over time and solar latitude and longitude. Solar plasma particles can escape the Sun's gravity because of their high energy resulting from the high temperature of the corona, which will travel along the magnetic field from the Sun.

At a distance of more than a few solar radii from the Sun, the solar wind reaches speeds of 250–750 km/s and is supersonic, meaning it moves faster than the speed of the fast magnetosonic wave. If the material carried by the solar wind reached a planet's surface, its radiation would do severe damage to any life that might exist in its way. Earth's magnetic field serves as a shield, redirecting the material around the planet so that it streams beyond it. The force of the wind stretches out the magnetic field so that it is smooshed inward on the sun side and stretched out on the night side.


National Aeronautics and Space Administration
Solar Wind and Corona Timeline
PARKER SOLAR PROBE LAUNCH
A mission to travel directly through the Sun's corona, providing up-close observations on what heats the solar atmosphere and accelerates the solar wind.
Slow Solar Wind and Helmet Streamers
Using observations from the joint ESA/NASA Solar and Heliospheric Observatory, Neil R. Sheeley Jr. and colleagues identify puffs of slow solar wind emanating from helmet streamers – bright areas of the corona that form above magnetically active regions on the photosphere. Exactly how these puffs are formed is still not known.
Streamer
The Sun's Poles
Ulysses, a joint NASA-ESA mission, becomes the first mission to fly over the Sun's north and south poles. Among other findings, Ulysses found that in periods of minimal solar activity, the fast solar wind comes from the poles, while the slow solar wind comes from equatorial regions.
Nanoflares May Heat the Corona
Eugene Parker proposes that frequent, small eruptions on the Sun – known as nanoflares – may heat the corona to its extreme temperatures. The nanoflare theory contrasts with the wave theory, in which heating is caused by the dissipation of Alfvén waves.
Fast Wind from Coronal Holes
Images from Skylab, the U.S.'s first manned space station, identify that the fast solar wind is emitted from coronal holes – comparatively cool regions of the corona where the Sun's magnetic field lines open out into space.
The Slow and Fast Solar Wind
NASA's Mariner 2 spacecraft observes the solar wind, detecting two distinct 'streams' within it: a slow stream travelling at approximately 215 miles per second, and a fast stream at 430 miles per second.
Solar Wind Detected
The Soviet satellite Luna 1, the first spacecraft to leave geocentric orbit, measures the solar wind directly for the first time, confirming key parts of Parker's theory.
The First Theory of the Solar Wind
Eugene Parker connects the hot corona with the solar wind in a rigorous mathematical theory. According to the theory, heat pressure from the million-degree corona forces it to expand outward in all directions, forming a solar wind that drags the Sun's magnetic field lines deep into space.
A Solar Wind Made of Particles
Building on Kepler's hypothesis from 400 years before, Cuno Hoffmeister (and later Ludwig Biermann) proposes that the Sun emits a steady stream of charged particles that push the ions in the comet tails always away from the Sun.
A New Heating Mechanism
Swedish physicist Hannes Alfvén proposes the existence of a new kind of wave forming in electrically conducting fluids. So-called Alfvén waves revealed a previously overlooked mechanism for heat and energy to be transferred on the Sun.
The Coronal Heating Problem
Swedish astronomer Bengt Edlen detects highly ionized iron in the corona, indicating a temperature of 1.8 million degrees Fahrenheit. Edlen's findings created the coronal heating problem: Why is the corona so much hotter than the Sun's surface?
The Corona as the Sun's Atmosphere
English astronomer Francis Baily observes a total solar eclipse and suggests that the hazy 'corona' outlining the Sun is its atmosphere.
Comet Tails in the Wind
Johannes Kepler observes comet tails and hypothesizes that they are blown by pressure from sunlight – a solar breeze.
2018
1995
1990
1988
1973
1962
1959
1958
1943
1942
1842
1610
A historical timeline of solar science discoveries—leading to the newest spacecraft in NASA's heliophysics fleet.
www.nasa.gov

Source: NASA

1.1 Solar Wind Properties:

1.1.1 Fast and slow Solar Wind:

The solar wind exists in two fundamental states, termed the slow and fast solar wind, though their differences are vast and not limited to their speeds. In near-Earth space, the slow solar wind is observed to have a velocity of 300–500 km/s, a temperature of about 100 MK, although the exact composition is not completely understood yet. Whereas the fast solar wind has a typical velocity of 750 km/s, a temperature of 800 MK. Fast solar winds are majorly sourced from the coronal holes. The slow solar wind is twice as dense and more variable than the fast solar wind.

The slow solar wind appears to originate from a region around the Sun's equatorial belt known as the "streamer belt", where coronal streamers are produced by magnetic flux open to the heliosphere draping over closed magnetic loops. Although, it’s debatable what are the exact coronal structures involved in slow solar wind formation and the method by which the material is released. Observations of the Sun between 1996 and 2001 showed that emission of the slow solar wind occurred at latitudes up to 30–35° during the solar minimum (the period of lowest solar activity), then expanded toward the poles as the solar cycle approached maximum. At solar maximum, the poles were also emitting a slow solar wind.

The Coronal holes originate fast solar wind, funnel-like regions of open field lines in the Sun's magnetic field. Such open lines are particularly prevalent around the Sun's magnetic poles. The plasma source is small magnetic fields created by convection cells in the solar atmosphere. These fields confine the plasma and transport it into the narrow necks of the coronal funnels, which are located only 20,000 km above the photosphere.

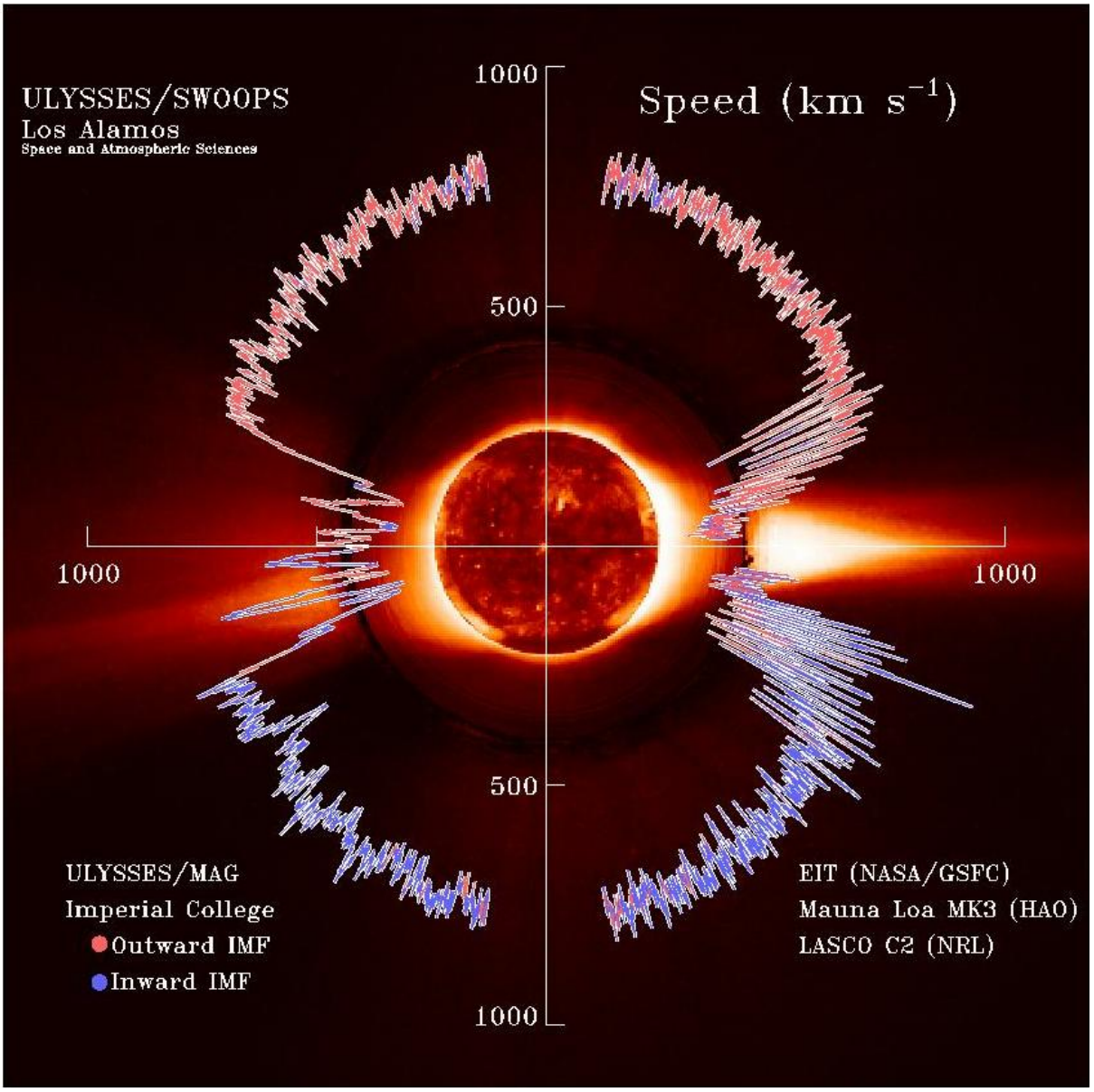
1.1.2 Coronal Mass Ejection(CME):

Both types of winds i.e. slow and high winds can be interrupted by large, fast-moving bursts of plasma called coronal mass ejections, or CMEs. CMEs are caused by a release of magnetic energy at the Sun. CMEs are often called "solar storms" or "space storms" in the popular media. They are sometimes, but not always, associated with solar flares, which are another manifestation of magnetic energy release at the Sun. CMEs cause shock waves in the thin plasma of the heliosphere, launching electromagnetic waves and accelerating particles (mostly protons and electrons) to form showers of ionizing radiation that precede the CME.

When a CME impacts the Earth's magnetosphere, it temporarily deforms the Earth's magnetic field, changing the direction of compass needles and inducing large electrical ground currents in Earth itself. This is called a geomagnetic storm and it is a global phenomenon. CME impacts can induce magnetic reconnection in Earth's magnetotail (the midnight side of the magnetosphere); this launches protons and electrons downward toward Earth's atmosphere, where they form the aurora.

1.2 Solar Wind Classification:

- ❖ To make a statistical study of solar wind parameters, interpret the results by dividing the solar wind measurements according to the wind type.
- ❖ The occurrence rates at Earth of the various types of solar wind plasma systematically change through the different solar cycle phases.

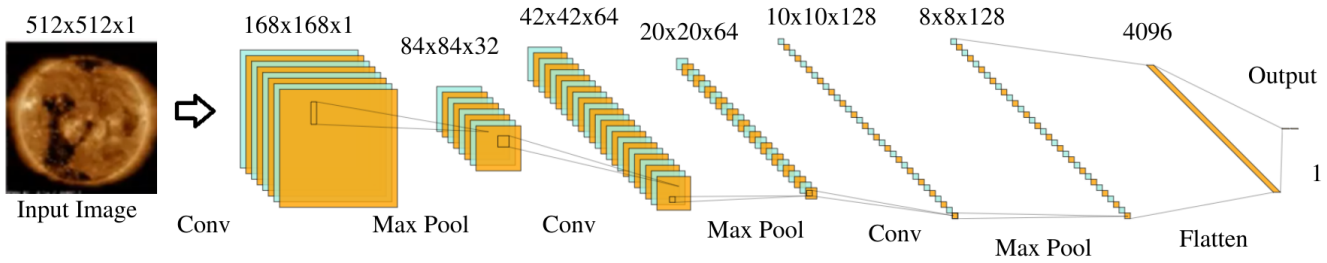


Source: Wikipedia

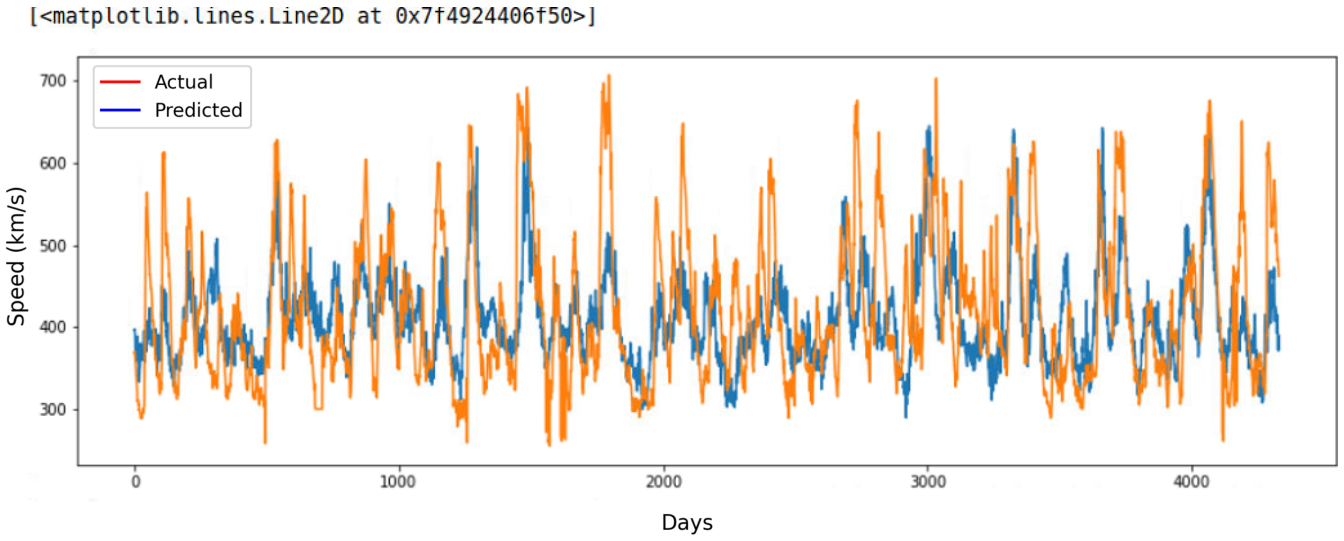
2. CNN MODEL

2.1 Base Architecture:

This architecture is the one that was already implemented at the start of the internship period.



This is the predicted v/s actual plot the base architecture was having. The primary goal is to predict those high-speed events(high peaks) efficiently.



2.2 Approaches:

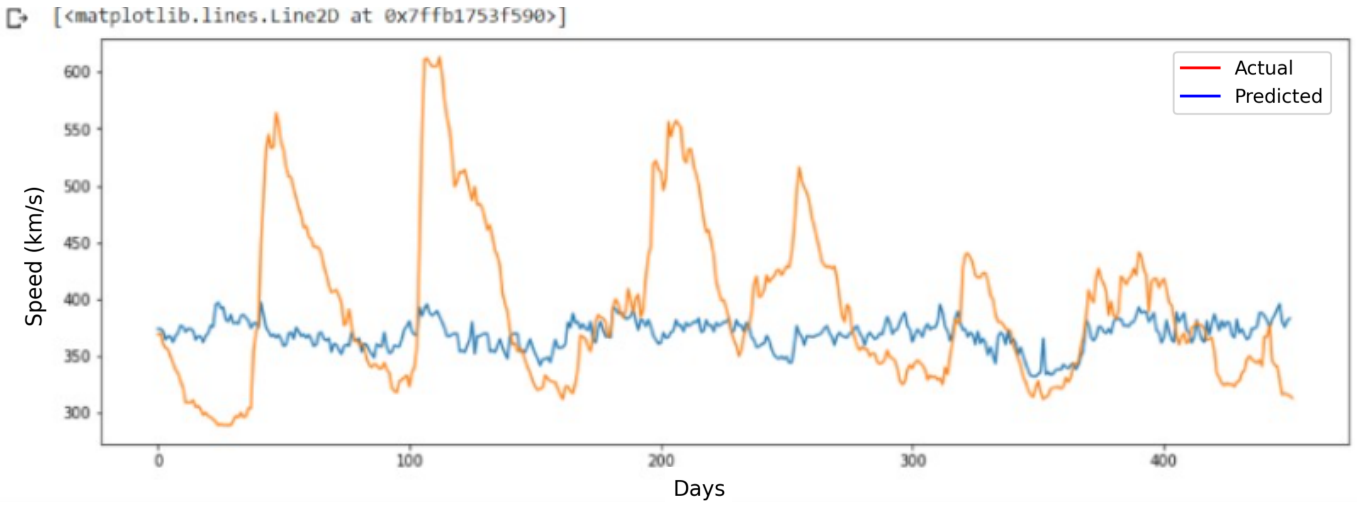
These were the initial tests that were done when the ubuntu server was getting ready and we understood the CNN code on Google Colab. So to test whatever we learnt so far we tried an approach of only testing on some filtered values so that the model can better focus on those values only and the accuracy in predicting the high-speed solar wind can increase.

As we tried to filter the training data first, later we decided to split the test data as well. The intuition was that if the model(being used to predict the high-speed events[HSE] only) gives good results, we can later merge both models made for HSE and LSE using the available techniques. Click here to navigate to the code implementation directly.

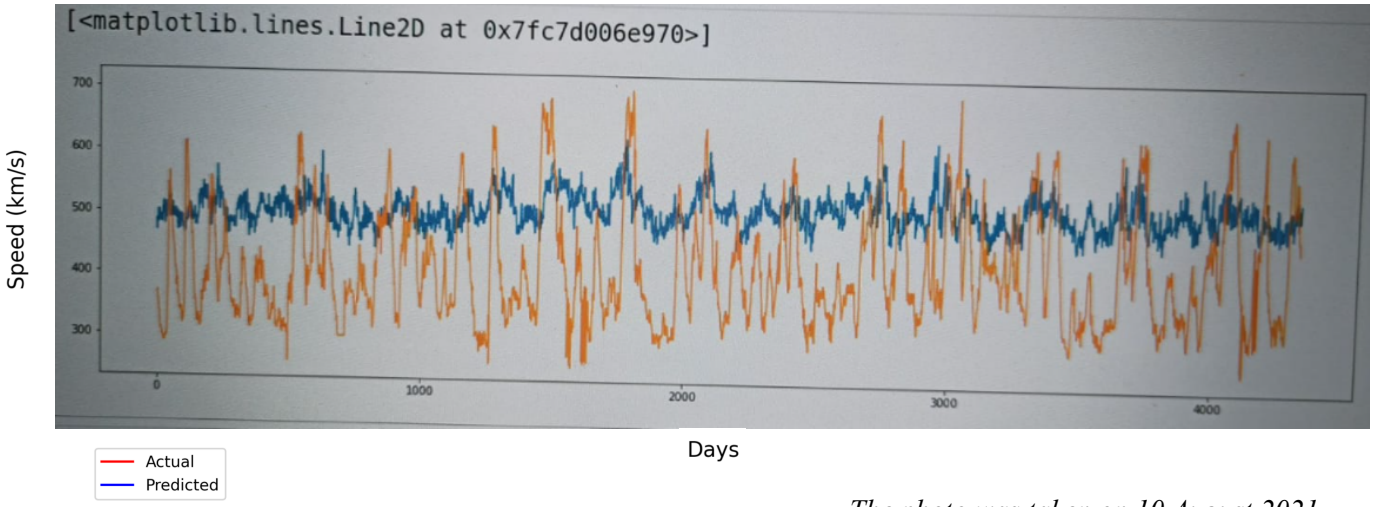
Approach 1: Filtering **Training Data** at some preset threshold (vHe2>435) and Keeping **Testing Data** as whole

Results:

For the first 500 values:



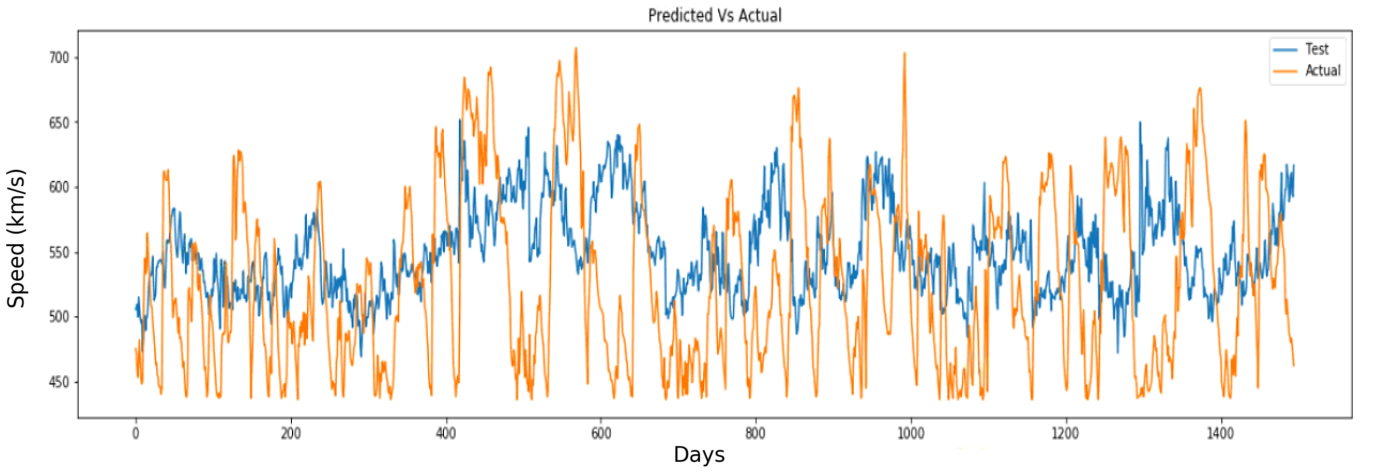
For all filtered Data:



The photo was taken on 10 August 2021

<u>Approach 2:</u> Filtering **Training** & **Testing Data** both at some preset threshold (vHe2>435)

<u>Results:</u>



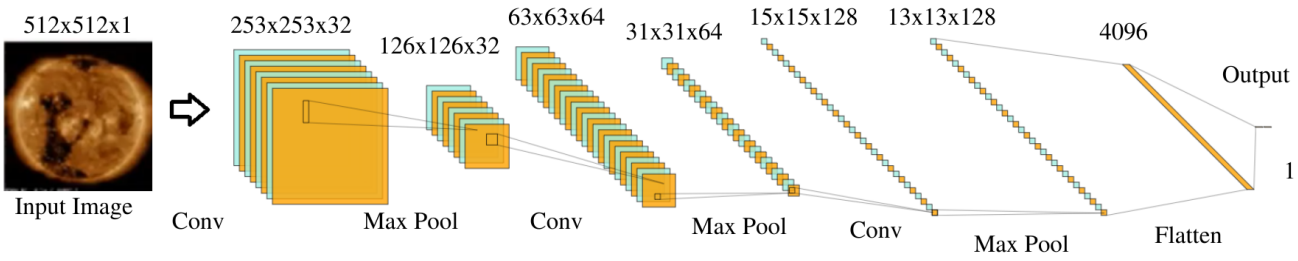
The photo was taken on 26 August 2021

Although the results might not be appealing as the tests were being done on only a few 500 values, this gave us a clear idea about whether we are understanding the model and the CNN correct or not. This gave us further motivation when we actually tried altering the architecture so that we can predict those high-speed events better.

3. ALTERNATE ARCHITECTURES

3.1 Architecture 1:

In this, we have changed the value of stride from 3,3 to 2,2 and in the Conv2D() method, the later half(kernel or convolutional matrix) is changed to (8,8). This gave good results and root mean square error of not more than 81.



conv2d_11 (Conv2D)	(None, 253, 253, 32)	2080
activation_11 (Activation)	(None, 253, 253, 32)	0
max_pooling2d_11 (MaxPooling	(None, 126, 126, 32)	0
conv2d_12 (Conv2D)	(None, 63, 63, 64)	8256
activation_12 (Activation)	(None, 63, 63, 64)	0
max_pooling2d_12 (MaxPooling	(None, 31, 31, 64)	0
conv2d_13 (Conv2D)	(None, 15, 15, 128)	32896
batch_normalization_4 (Batch	(None, 15, 15, 128)	512
activation_13 (Activation)	(None, 15, 15, 128)	0
max_pooling2d_13 (MaxPooling	(None, 13, 13, 128)	0
flatten_4 (Flatten)	(None, 21632)	0
dense_8 (Dense)	(None, 4096)	88608768
dropout_4 (Dropout)	(None, 4096)	0
dense_9 (Dense)	(None, 1)	4097

Total params: 88,656,609
Trainable params: 88,656,353
Non-trainable params: 256

None

This is being done as changing strides can affect the performance of CNN. The further apart two pixels are from each other, the less correlated. Therefore, a big stride in the pooling layer leads to high information loss; hence the intuition was to decrease the stride wherever possible.

3.2 Errors:

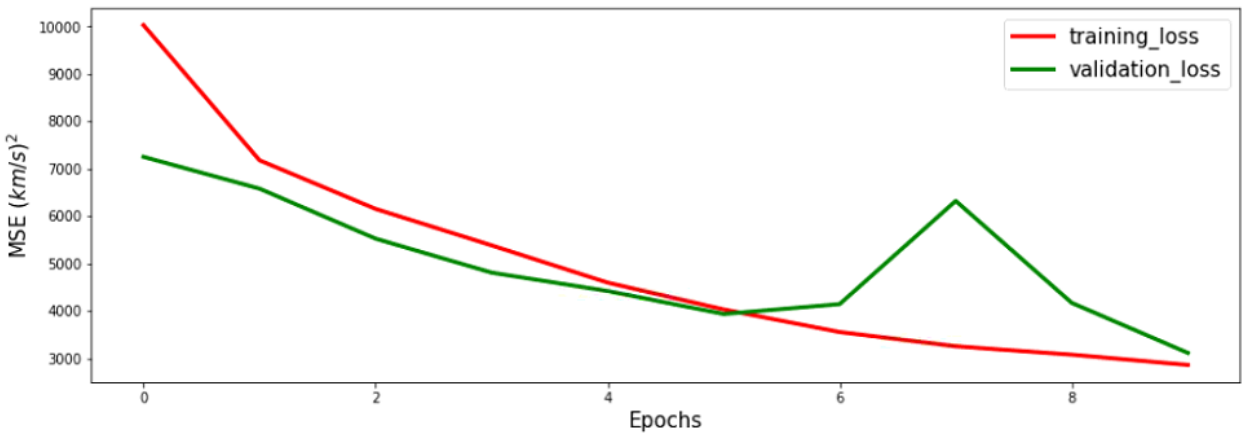
Root Mean Square Error: 90.09142252113286 km/s
Correlation Coefficient: 0.45406035

3.3 Plots:

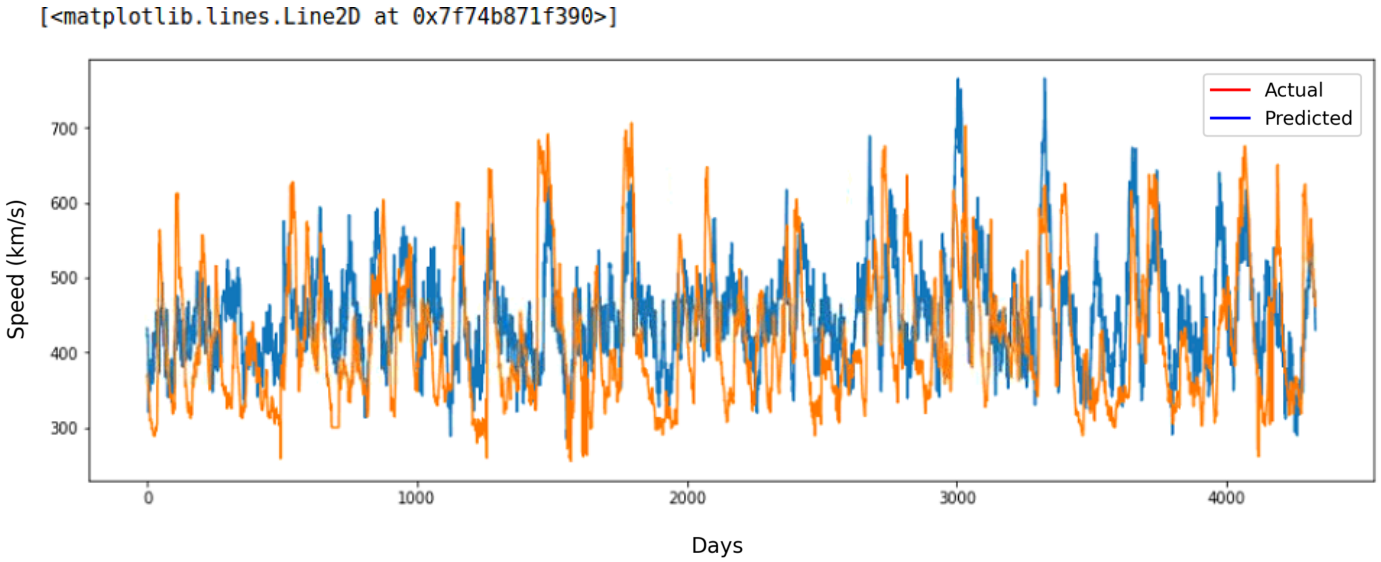
3.3.1 Validation Loss vs Training Loss:

```
plt.plot(y_test1)
plt.plot(data_speed_2018)
```

Plot of y_test1 and data_speed_2018

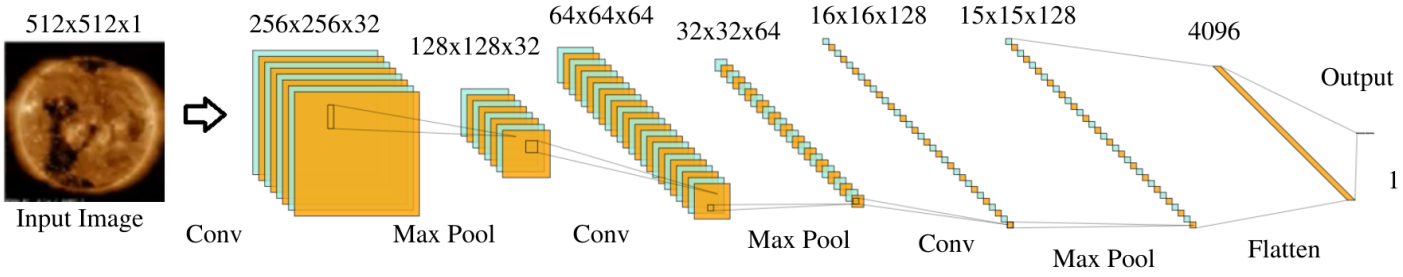


3.3.2 Prediction vs Actual Plot:



3.1 Architecture 2:

In this, to further improve the last architecture, we further changed the strides to (2,2) instead of (3,3) at the leftover places.



We also changed the second argument of the Conv2D function, a kernel that describes a filter that we are going to pass over an input image. To make it simple, the kernel will move over the whole image, from left to right, from top to bottom by applying a convolution product. The output of this operation is called a filtered image.

Changing it to (2,2) ensures that the filter of (2,2) is going through the image of size 512x512. This significantly affects trainable parameters as the trainable elements are the values that compose the kernels.

So the number of parameters increases linearly with the number of convolution kernels. Hence linearly with the number of desired output channels.

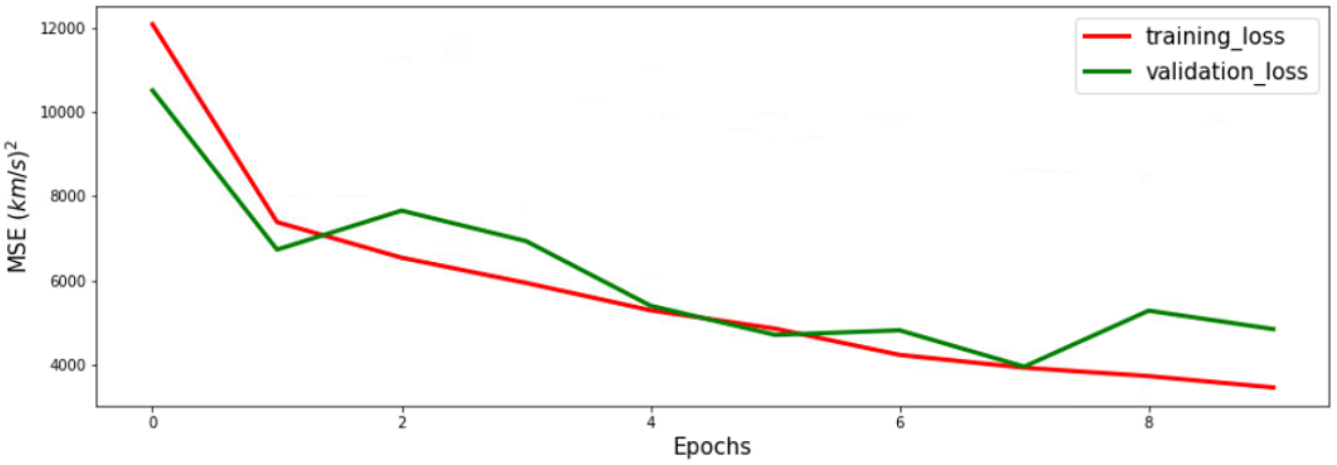
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 256, 256, 32)      160
_________________________________________________________________
activation_3 (Activation)    (None, 256, 256, 32)      0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 64, 64, 64)        8256
_________________________________________________________________
activation_4 (Activation)    (None, 64, 64, 64)        0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 16, 16, 128)       32896
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
activation_5 (Activation)    (None, 16, 16, 128)       0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 15, 15, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 28800)             0
_________________________________________________________________
dense_2 (Dense)              (None, 4096)              117968896
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 4097
=================================================================
Total params: 118,014,817
Trainable params: 118,014,561
Non-trainable params: 256
_________________________________________________________________
None
```

3.2 Errors:

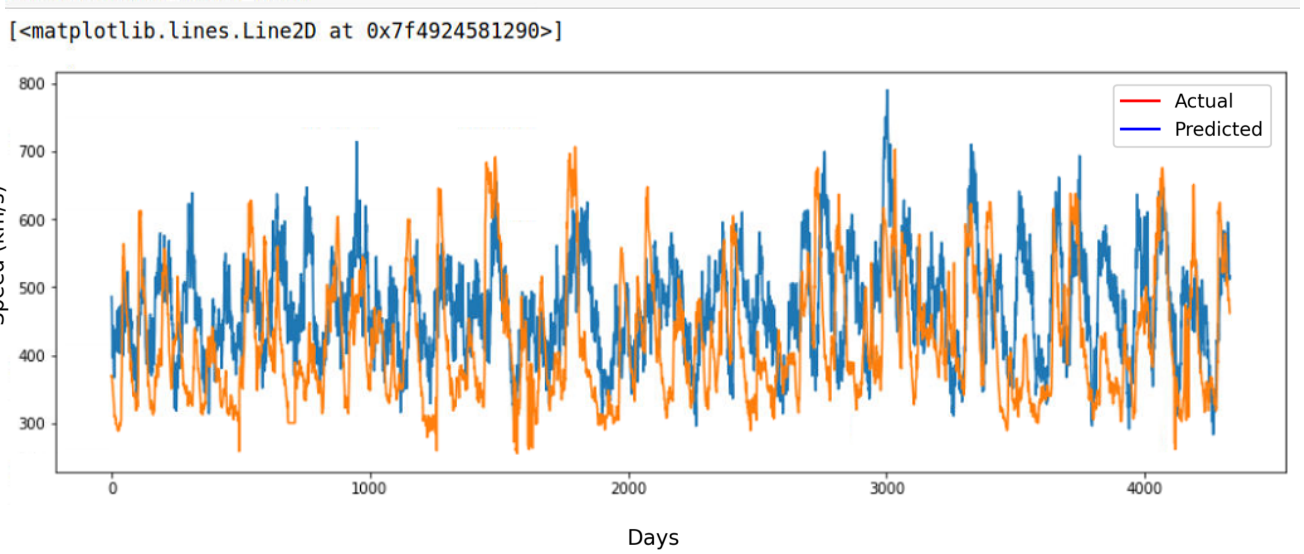
Root Mean Square Error: 106.40551541502319 km/s
Correlation Coefficient: 0.37324723

3.3 Plots:

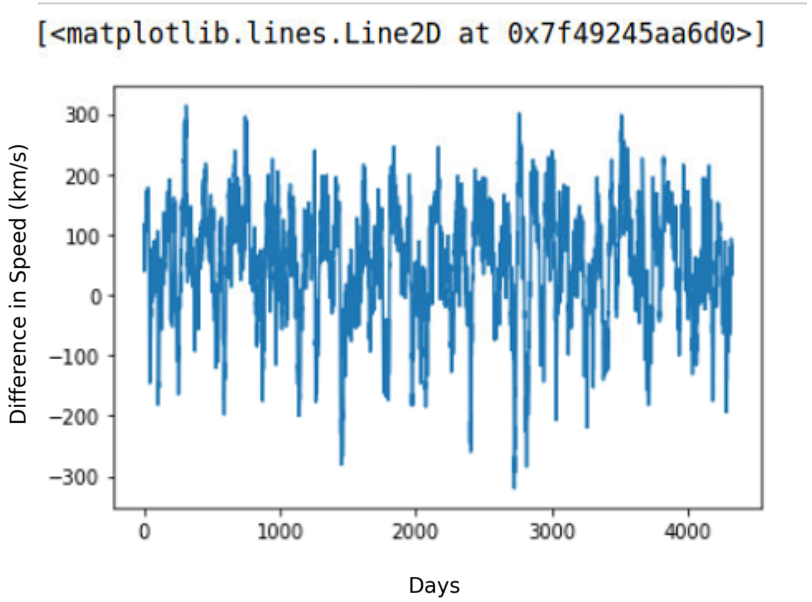
3.3.1 Validation Loss vs Training Loss:



3.3.2 Prediction vs Actual Plot:



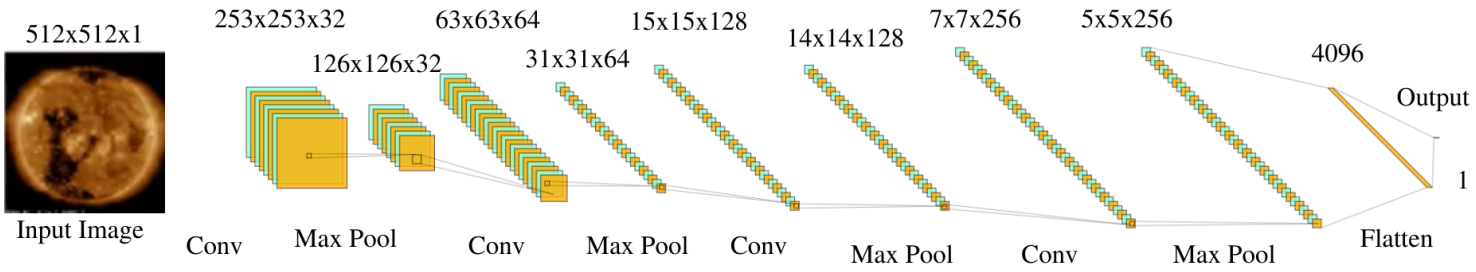
3.3.3 Error Plot:



3.1 Architecture 3:

In this, instead of having just three layers, I tried having four layers of CNN. The number of layers in a model is referred to as its depth.

Increasing the depth increases the capacity of the model. Training deep models, e.g. those with many hidden layers, can be computationally more efficient than training a single-layer network



with a vast number of nodes.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_56 (Conv2D)           (None, 253, 253, 32)      2080
_________________________________________________________________
activation_56 (Activation)   (None, 253, 253, 32)      0
_________________________________________________________________
max_pooling2d_56 (MaxPooling (None, 126, 126, 32)      0
_________________________________________________________________
conv2d_57 (Conv2D)           (None, 63, 63, 64)        8256
_________________________________________________________________
activation_57 (Activation)   (None, 63, 63, 64)        0
_________________________________________________________________
max_pooling2d_57 (MaxPooling (None, 31, 31, 64)        0
_________________________________________________________________
conv2d_58 (Conv2D)           (None, 15, 15, 128)       32896
_________________________________________________________________
activation_58 (Activation)   (None, 15, 15, 128)       0
_________________________________________________________________
max_pooling2d_58 (MaxPooling (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_59 (Conv2D)           (None, 7, 7, 256)         131328
_________________________________________________________________
batch_normalization_23 (Batc (None, 7, 7, 256)         1024
_________________________________________________________________
activation_59 (Activation)   (None, 7, 7, 256)         0
_________________________________________________________________
max_pooling2d_59 (MaxPooling (None, 5, 5, 256)         0
_________________________________________________________________
flatten_16 (Flatten)         (None, 6400)              0
_________________________________________________________________
dense_30 (Dense)             (None, 4096)              26218496
_________________________________________________________________
dropout_14 (Dropout)         (None, 4096)              0
_________________________________________________________________
dense_31 (Dense)             (None, 1)                 4097
=================================================================
Total params: 26,398,177
Trainable params: 26,397,665
Non-trainable params: 512
_________________________________________________________________
None
```

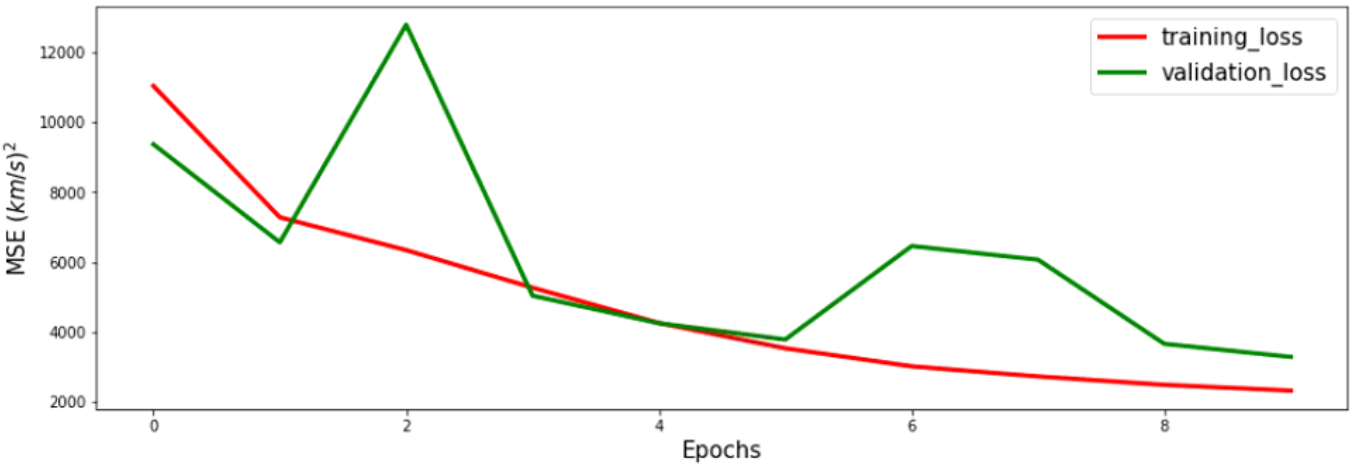
3.2 Errors:

Root Mean Square Error: 85.12724209992173 km/s

Correlation Coefficient: 0.51605934

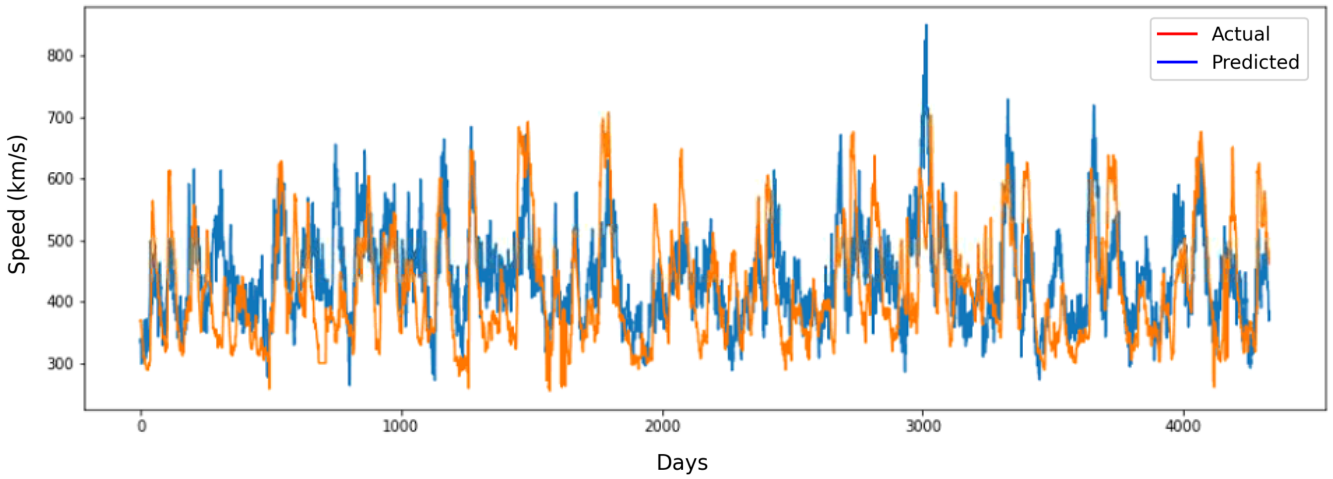
3.3 Plots:

3.3.1 Validation Loss vs Training Loss:

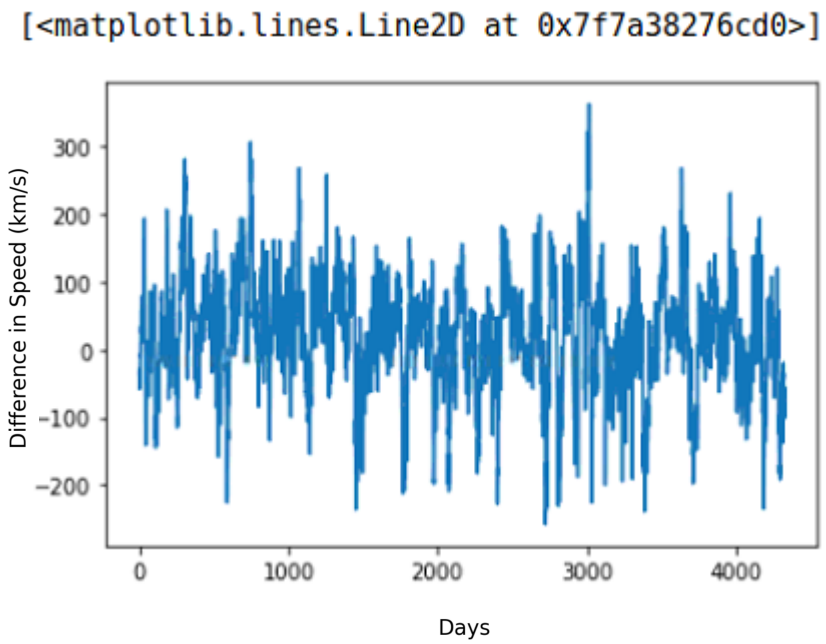


3.3.2 Prediction vs Actual Plot:

```
[<matplotlib.lines.Line2D at 0x7f74dc636bd0>]
```



3.3.3 Error Plot:



Adding a new layer and making the architecture 4-layered gave promising results but adding layers **increases** the number of weights in the network, ergo the model complexity. Without an extensive training set, an increasingly large network is likely to overfit and, in turn, reduce **accuracy** on the test data.

All of the related python code for CCN implementation of each architecture is available in the Appendix of this report. Click here to navigate directly.

4. CONCLUSIONS

This research aimed to predict the Solar Wind Speed using imagery from NASA's Solar Dynamics Observatory (SDO). Deep Neural Networks, to be specific, CNN, were used for predictions. Initially, we were provided with a CNN model on which our primary goal was to increase its accuracy in predicting high-speed events of solar wind. We first selected only those images as our input that had vHe2 greater than a threshold value which was the mean of the training data vHe2 column. We can conclude that training the architecture using only high values of vHe2 somehow could predict the peak values which standard architectures were underpredicting. This made us understand the basic meaning of the pre-existing model and motivated us for our future experiments.

Further, as there is no fixed thumb rule for selecting an ideal architecture in Deep Learning, we implemented several different architectures with varying layers and parameters like kernels, pooling size, strides, padding, etc. Those different architectures concluded with different results and were also somewhat capable of predicting high-speed events. The root mean square error ranged from 75 to 100 km/s, and the other errors and plots are also concluded in the report.

Deep Learning Models using CNN have proved to be beneficial for the immediate prediction of solar wind without waiting for 2-3 days for the reading. These models can prove to be a gamechanger in the future to protect our assets both on earth and in space. For instance, satellites can change their geometric orientation to protect sensitive equipment in the high solar wind by using the predictions.

5. REFERENCES

- https://link.springer.com/article/10.1007/s11207-021-01874-6
- https://arxiv.org/pdf/2006.05825.pdf
- https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote18.html
- https://en.wikipedia.org/wiki/Solar_wind
- https://agupubs.onlinelibrary.wiley.com/doi/full/10.1029/2020SW002478
- https://www.nasa.gov/image-feature/goddard/2018/solar-wind-and-corona-timeline/
- https://www.canva.com/design/DAEp6qLZCvA/Y1gDkQUpFobp2TrNY-093A/edit
- http://alexlenail.me/NN-SVG/LeNet.html
- https://keras.io/api/layers/convolution_layers/convolution2d/

6. APPENDIX

❖ Code for Naive Approaches and understanding:

1. Approach 1 & Approach 2:

```
# Filtering Test Data

test_2018_filtered = data_2018[(data_2018['vHe2']>435)]
indexes_filtered_2018 = list(test_2018_filtered.index)
speed_filtered_2018 = test_2018_filtered.iloc[:,3].values // contains values of vHE2 > 435 only

# for training data

# Calculate Mean
vHe2 = train_y3['vHe2']
average = vHe2.mean()
print(average)

# selecting values above the threshold

new_train = train_y3[(train_y3['vHe2']>435)]
trainy_speed=new_train.iloc[:,3].values
trainy_speed.shape  // contains filtered Training data
```

❖ Code for Alternate Architectures::

1. Code for Alternate Architecture 1:

```
#importing keras libraries
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
#from tensorflow.python.keras import convolutional
#from tensorflow.keras.layers import pooling
#from tensorflow.keras.layers import core
from tensorflow.keras import optimizers
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.regularizers import l2
from tensorflow.keras.models import load_model
#import tensorflow.keras.backend as K

#keras.backend.clear_session()
model_8 = Sequential()
model_8.add(Conv2D(32, (8,8),strides=(2, 2), input_shape=(512,512,1)))
model_8.add(Activation('relu'))
model_8.add(MaxPooling2D(pool_size=(2, 2),strides=(2, 2)))
model_8.add(Conv2D(64, (2,2),strides=(2, 2)))
model_8.add(Activation('relu'))
model_8.add(MaxPooling2D(pool_size=(2, 2),strides=(2, 2)))
model_8.add(Conv2D(128, (2,2),strides=(2, 2)))
model_8.add(BatchNormalization())
model_8.add(Activation('relu'))
model_8.add(MaxPooling2D(pool_size=(3, 3),strides=(1, 1)))
model_8.add(Flatten())
model_8.add(Dense(4096,activation='relu'))
model_8.add(Dropout(.3))
model_8.add(Dense(1,activation='linear'))
model_8.compile(optimizer=optimizers.Adam(lr=1e-04), loss='mse',metrics=['mse'])
print(model_8.summary())
```

2. Code for Alternate Architecture 2:

The only change is this part of the code:

```
#keras.backend.clear_session()
model_8 = Sequential()
model_8.add(Conv2D(32, (2,2),strides=(2, 2), input_shape=(512,512,1)))
model_8.add(Activation('relu'))
model_8.add(MaxPooling2D(pool_size=(2, 2),strides=(2, 2)))
model_8.add(Conv2D(64, (2,2),strides=(2, 2)))
model_8.add(Activation('relu'))
model_8.add(MaxPooling2D(pool_size=(2, 2),strides=(2, 2)))
model_8.add(Conv2D(128, (2,2),strides=(2, 2)))
model_8.add(BatchNormalization())
model_8.add(Activation('relu'))
model_8.add(MaxPooling2D(pool_size=(2, 2),strides=(1, 1)))
```

3. Code for Alternate Architecture 3:

```
#CNN architecture


#keras.backend.clear_session()
model_8 = Sequential()
model_8.add(Conv2D(32, (8,8),strides=(2, 2), input_shape=(512,512,1)))
model_8.add(Activation('relu'))
model_8.add(MaxPooling2D(pool_size=(2, 2),strides=(2, 2)))
model_8.add(Conv2D(64, (2,2),strides=(2, 2)))
model_8.add(Activation('relu'))
model_8.add(MaxPooling2D(pool_size=(2, 2),strides=(2, 2)))
model_8.add(Conv2D(128, (2,2),strides=(2, 2)))
model_8.add(BatchNormalization())
model_8.add(Activation('relu'))
model_8.add(MaxPooling2D(pool_size=(2, 2),strides=(1, 1)))

### 4th Layer
model_8.add(Conv2D(256, (2,2),strides=(2, 2)))
model_8.add(BatchNormalization())
model_8.add(Activation('relu'))
model_8.add(MaxPooling2D(pool_size=(3, 3),strides=(1, 1)))
###

model_8.add(Flatten())
model_8.add(Dense(4096,activation='relu'))
model_8.add(Dropout(.3))
model_8.add(Dense(1,activation='linear'))
model_8.compile(optimizer=optimizers.Adam(lr=1e-04), loss='mse',metrics=['mse'])
print(model_8.summary())
```