# Predicting CME Arrival Time Through Data Integration And Ensemble Learning

# Table of Contents

## Author(s)

- Author1 = {"name": "Khalid A. Alobaid", "affiliation": "Department of Computer Science, New Jersey Institute of Technology", "email": "kaa65@njit.edu (mailto:kaa65@njit.edu)"}
- Author2 = {"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)
- Author3 = {"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)
- Author4 = {"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)

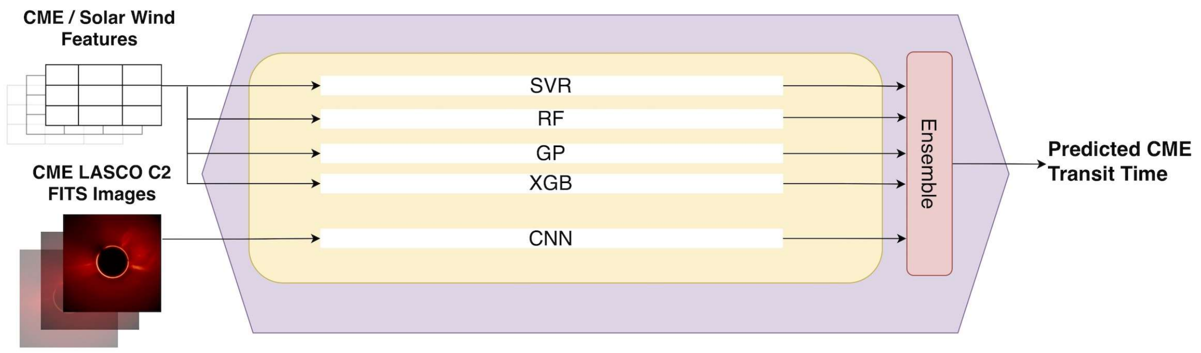
## Purpose

The Sun constantly releases radiation and plasma into the heliosphere. Sporadically, the Sun launches solar eruptions such as flares and coronal mass ejections (CMEs). CMEs carry away a huge amount of mass and magnetic flux with them. An Earth-directed CME can cause serious consequences to the human system. It can destroy power grids/pipelines, satellites, and communications. Therefore, accurately monitoring and predicting CMEs is important to minimize damages to the human system. In this study we propose an ensemble learning approach, named CMETNet, for predicting the arrival time of CMEs from the Sun to the Earth.

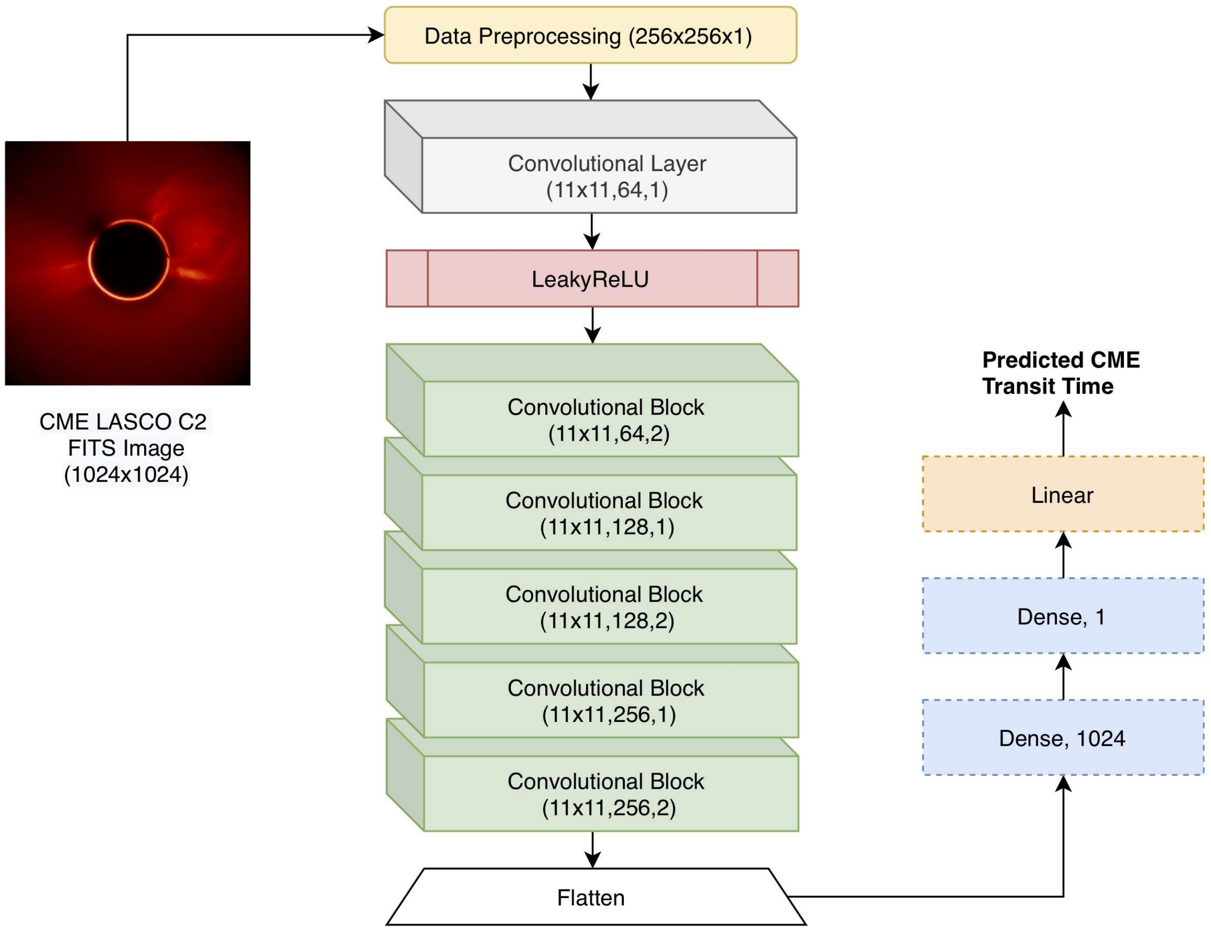
In this notebook we provide an overview of the CMETNet system to demonstrate how to predict the arrival time of CMEs using machine learning (ML) and data include CME features, solar wind parameters and CME images obtained from the SOHO/LASCO C2 coronagraph.

## Methodology

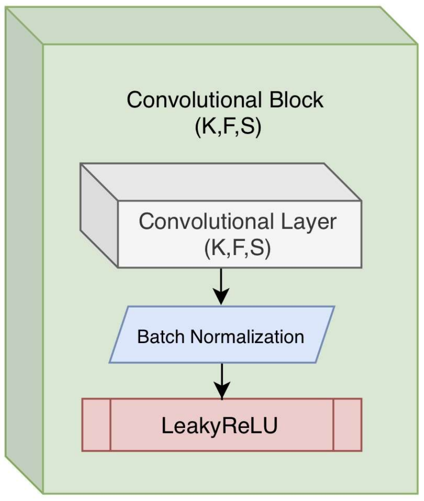
Our ensemble learning framework, named CMETNet and illustrated in Figure 1, comprises five base learners: (i) a support vector regression (SVR) algorithm , (ii) a random forest (RF) algorithm , (iii) a Gaussian process (GP) algorithm , (iv) an XGBoost (XGB) algorithm , and (v) a convolutional neural network (CNN) illustrated in Figure 2. The first four learners are regression algorithms used for analyzing the numeral/tabular data related to the CME features and solar wind parameters considered in the study. These four regression algorithms are commonly used in heliophysics and space weather research. The fifth learner is a CNN model used for CME image processing. More detials can be found in the paper.



(Figure 1)


Data Preprocessing (256x256x1)
Convolutional Layer (11x11,64,1)
LeakyReLU
Convolutional Block (11x11,64,2)
Convolutional Block (11x11,128,1)
Convolutional Block (11x11,128,2)
Convolutional Block (11x11,256,1)
Convolutional Block (11x11,256,2)
Flatten
CME LASCO C2 FITS Image (1024x1024)
Predicted CME Transit Time
Linear
Dense, 1
Dense, 1024

(a)



(b)

(Figure 2)

## Funding

This work was supported by U.S. NSF grants AGS-1927578, AGS-1954737 and AGS-2149748.key

### Keywords

keywords=["heliophysics", "space weather", "coronal mass ejections", "interplanetary shocks", "machine learning"]

### Citation

To cite this notebook: Khalid A. Alobaid, Yasser Abduallah, Jason T. L. Wang, & Haimin Wang. Predicting CME Arrival Time through Data Integration and Ensemble Learning, available at: https://github.com/ccsc-tools/CMETNet/blob/main/KA_01_PredictingCMEArrivalTimeThroughDataIntegrationAndEnsembleL (https://github.com/ccsc-tools/CMETNet/blob/main/KA_01_PredictingCMEArrivalTimeThroughDataIntegrationAndEnsembleL

### Acknowledgements

The solar wind data is provided by NASA’s Goddard Space Physics Data Facility. The SOHO LASCO CME Catalog is created and maintained by NASA’s CDAW Data Center and the Catholic University of America in cooperation with the Naval Research Laboratory. SOHO is a project of international cooperation between ESA and NASA.

## Setup

**Installation on Local Machine**
To install the required packages, you may use Python package manager "pip" as follow:

1. Copy the text file "requirements.txt" into the working directory.
2. Execute the command:

Type:

```
pip install -r requirements.txt
```

Note: There is a requirements file already created for you to use that includes all packages with their versions. The files are located in the root directory of the CMETNet. Note: Python packages and libraries are sensitive to versions. Please make sure you are using the correct packages and libraries versions as specified above.

Cuda Installation Package: You may download and install Cuda v 10.1 from https://developer.nvidia.com/cuda-10.1-download-archive-base (https://developer.nvidia.com/cuda-10.1-download-archive-base)

## Data Processing and Analysis

We stored 351 LASCO C2 FITS data samples in the data folder of this repo. The "ICMEs_C2_fits_list.csv" file represnt a list of the LASCO C2 FITS images along with the transit time which is the label. The transit time is in hours. These labeled images are used to train/test the

CNN model. For the other models in CMETNet, "ICME_list.csv" file represnt the CME events concidered in this study along with the CME and SW parameters. These events are used to train/test the COMB model (SVR, RF, GP, XGP) of CMETNet.

For this notebook, we use sample data sets for training and testing.

## 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/CMETNet/HEAD?labpath=KA_01_PredictingCMEArrivalTimeThroughDataIntegrationAndEnsembleLearning.ipynb)

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

Please also note that the execution time in Binder varies based on the availability of resources. The average time to run the notebook is 10-15 minutes, but it could be more.

## Plotting the Pretrained Models Results

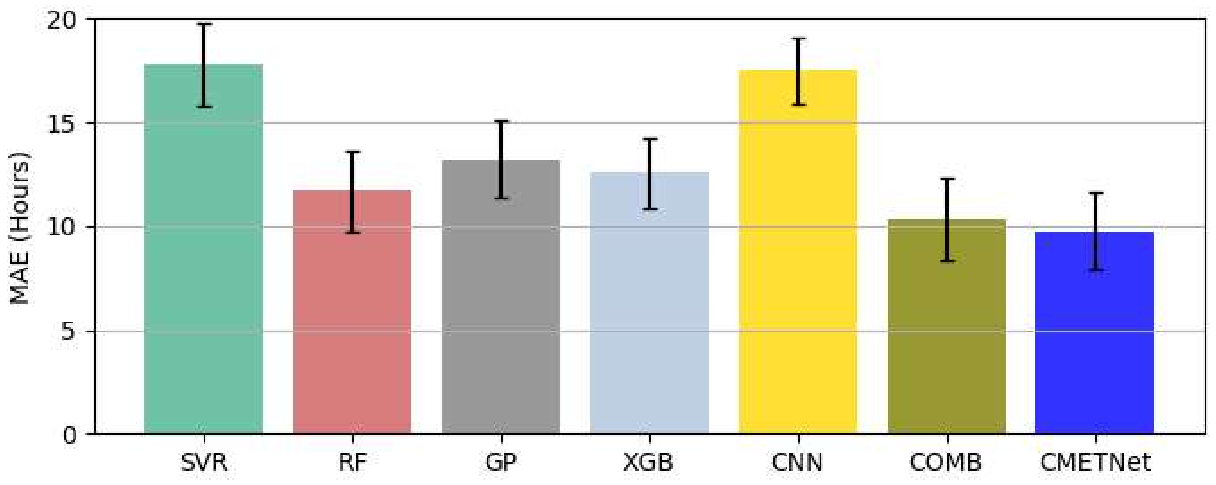
The prediction result can be plotted as shown in the following example. The result shows the performance metrics: PPMCC and MAE for each model.

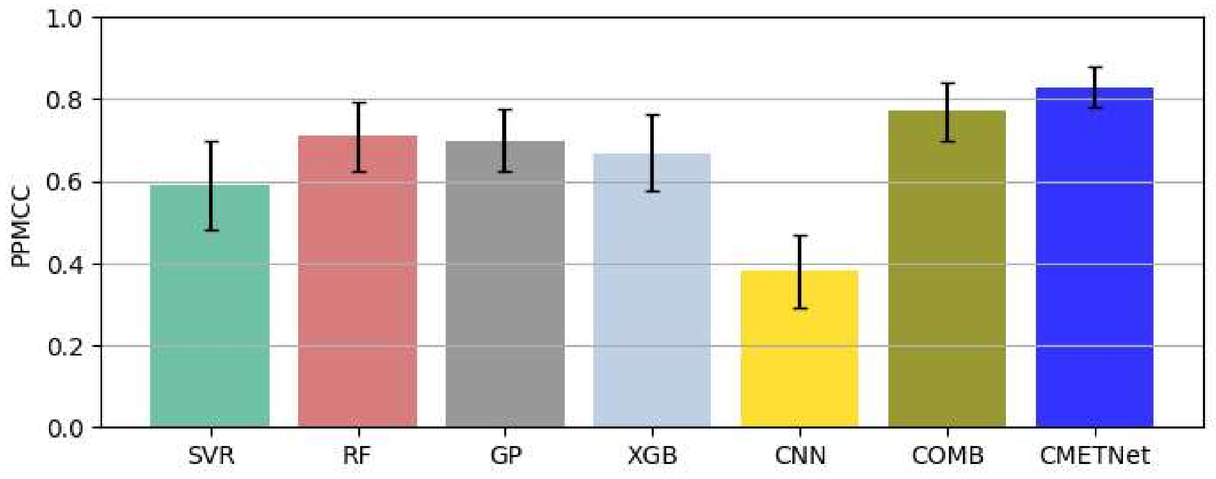
In [1]:
```
import warnings
warnings.filterwarnings('ignore')

import os
print('Packages imported')
#make sure the scripts are executed in the correct pacakage location.
if os.path.exists('CMETNet_Package'):
    print('Changing working directory to CMETNet_Package..')
    os.chdir('CMETNet_Package')
import sys
sys.path.append('.')

from pretrained_models_results import plot_results
#Plotting the results
plot_results()
```

```
Packages imported
Changing working directory to CMETNet_Package..
```





## CMETNet Training and Testing

### CNN Model Training and Testing

You may train the model with your own data or train the model with the default data.

Here, we show how to train the CMETNet model with default data. To train the model with your own data:

1. You should first upload your file to the data directory (in the left hand side file list).
2. Edit the "ICMEs_C2_fits_list.csv" file in the data directory by adding the C2 FITS images file name and the transit time.

In [2]:
```
print('Running the CMETNet_CNN_train.py file...')
print('Starting with data preparation, then training the CNN model of CMETNe

%run CMETNet_CNN_train.py
```

```
Running the CMETNet_CNN_train.py file...
Starting with data preparation, then training the CNN model of CMETNet
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 256, 256, 1)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 256, 256, 64)      7808
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 256, 256, 64)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 128, 64)      495680
_________________________________________________________________
batch_normalization (BatchNo (None, 128, 128, 64)      256
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 128, 128, 64)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 128, 128, 128)     991360
_________________________________________________________________
batch_normalization_1 (Batch (None, 128, 128, 128)     512
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 128, 128, 128)     0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 128)       1982592
_________________________________________________________________
batch_normalization_2 (Batch (None, 64, 64, 128)       512
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 64, 64, 256)       3965184
_________________________________________________________________
batch_normalization_3 (Batch (None, 64, 64, 256)       1024
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 64, 64, 256)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 32, 32, 256)       7930112
_________________________________________________________________
batch_normalization_4 (Batch (None, 32, 32, 256)       1024
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 32, 32, 256)       0
_________________________________________________________________
flatten (Flatten)            (None, 262144)            0
_________________________________________________________________
dense (Dense)                (None, 1024)              268436480
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 1025
_________________________________________________________________
activation (Activation)      (None, 1)                 0
=================================================================
Total params: 283,813,569
Trainable params: 283,811,905
```

```
Non-trainable params: 1,664
_________________________________________________________________
Epoch 1/10
4/4 [==============================] - 110s 26s/step - loss: 681.5511 - val_los
s: 5468.3667
Epoch 2/10
4/4 [==============================] - 99s 24s/step - loss: 548.7322 - val_los
s: 1006.2743
Epoch 3/10
4/4 [==============================] - 99s 24s/step - loss: 725.4680 - val_los
s: 1457.2574
Epoch 4/10
4/4 [==============================] - 98s 24s/step - loss: 719.3953 - val_los
s: 1382.3331
Epoch 5/10
4/4 [==============================] - 96s 23s/step - loss: 726.6939 - val_los
s: 606.1398
Epoch 6/10
4/4 [==============================] - 97s 24s/step - loss: 424.7784 - val_los
s: 387.8782
Epoch 7/10
4/4 [==============================] - 98s 23s/step - loss: 270.4783 - val_los
s: 333.1194
Epoch 8/10
4/4 [==============================] - 96s 23s/step - loss: 151.9844 - val_los
s: 223.7884
Epoch 9/10
4/4 [==============================] - 97s 24s/step - loss: 104.8791 - val_los
s: 42.8680
Epoch 10/10
4/4 [==============================] - 97s 23s/step - loss: 71.7454 - val_loss:
29.5061
Done ------------------------------------
```

## COMB Model Training and Testing

In this step, the COMB models (SVR, RF, GP, XGP) will be trained with a sample data. The trained models then will be used to predict test data.

In [3]:
```
print('Running the CMETNet_COMB_train.py file...')
print('Starting with data preparation, then training the COMB models of CMET

%run CMETNet_COMB_train.py
```

```
Running the CMETNet_COMB_train.py file...
Starting with data preparation, then training the COMB models of CMETNet
Epoch  01 /10
Epoch  02 /10
Epoch  03 /10
Epoch  04 /10
Epoch  05 /10
Epoch  06 /10
Epoch  07 /10
Epoch  08 /10
Epoch  09 /10
Epoch  10 /10
Done ------------------------------------
```

## Testing the Ensemble Learning method

#### Combining CNN and COMB models to show CMETNet prediction results

In [4]:
```
print('Running the results_ensemble.py file...')
print('Final results also will be stored in the results directory')
%run results_ensemble.py
```

```
Running the results_ensemble.py file...
Final results also will be stored in the results directory

       Observed Data                           Prediction
CME Appearance Time    CME Arrival Time            CMETNet Difference in Hours
2014-01-04 21:25:00 2014-01-07 14:25:00 2014-01-07 07:14:12                  7
2014-01-07 18:24:00 2014-01-09 19:24:00 2014-01-09 22:19:48                 -2
2014-01-14 10:24:00 2014-01-18 17:24:00 2014-01-17 04:26:24                 36
2014-01-30 16:24:00 2014-02-02 22:24:00 2014-02-02 03:10:12                 19
2014-02-04 01:25:00 2014-02-07 15:25:00 2014-02-06 16:55:36                 22
2014-02-12 06:12:00 2014-02-15 13:12:00 2014-02-15 07:46:48                  5
2014-02-18 01:25:00 2014-02-20 02:25:00 2014-02-20 14:52:36                -12
2014-02-19 16:00:00 2014-02-23 06:00:00 2014-02-22 17:20:24                 12
2014-02-25 01:25:00 2014-02-27 20:25:00 2014-02-27 03:20:48                 17
2014-03-23 03:48:05 2014-03-26 04:48:05 2014-03-25 14:19:17                 14
2014-04-01 17:00:00 2014-04-05 09:00:00 2014-04-04 10:39:36                 22
2014-04-02 14:00:00 2014-04-05 22:00:00 2014-04-05 00:38:24                 21
2014-04-08 23:12:00 2014-04-11 08:12:00 2014-04-11 14:50:24                 -6
2014-04-18 13:25:51 2014-04-21 06:25:51 2014-04-20 20:59:27                  9
2014-06-25 12:54:00 2014-06-29 17:54:00 2014-06-28 07:07:12                 34
2014-08-15 18:12:00 2014-08-19 06:12:00 2014-08-18 12:00:00                 18
2014-09-09 00:16:00 2014-09-12 09:46:00 2014-09-11 15:29:48                 18
2014-09-10 18:24:00 2014-09-12 15:24:00 2014-09-13 01:37:48                -10
2014-11-01 05:12:00 2014-11-03 22:12:00 2014-11-03 06:07:48                 16
2014-11-07 18:08:00 2014-11-10 01:08:00 2014-11-10 08:20:00                 -7
2014-12-17 05:00:00 2014-12-21 11:30:00 2014-12-20 04:14:24                 31

CMENet PPMCC:    0.62
CMENet MAE:      16.51
Done ------------------------------------
```

## Conclusions

In this work we presented an ensemble framework (CMETNet) for predicting the arrival time of CME events. Each event contains 5 CME features (angular width, main position angle, linear speed, 2nd-order speed at final height, and mass), 7 solar wind parameters (Bx, Bz, alpha to proton ratio, flow longitude, plasma pressure, flow speed, and proton temperature) and 0-17 CME images. The CMETNet framework is composed of five machine learning models including support vector regression (SVR), random forest (RF), Gaussian process (GP), XGBoost (XGB), and a convolutional neural network (CNN). Our experimental results demonstrated that using all the CME features, solar wind parameters and CME images together yields better performance than using part of them.

## CMETNet paper

Predicting CME arrival time through data integration and ensemble learning
Khalid A. Alobaid, Yasser Abduallah, Jason T. L. Wang, Haimin Wang, Haodi Jiang, Yan Xu, Vasyl Yurchyshyn, Hongyang Zhang, Huseyin Cavus and Ju Jing
https://www.frontiersin.org/articles/10.3389/fspas.2022.1013345/full
(https://www.frontiersin.org/articles/10.3389/fspas.2022.1013345/full)

```
In [ ]:
```