# Predicting Solar Flares Using a Long Short-term Memory Network

**Hao Liu, Chang Liu, Jason T. L. Wang and Haimin Wang**

## 1. Introduction

Solar flares, the largest explosive events in our solar system, are intense bursts of radiation that occur in the Sun's atmosphere and release massive amounts of energy into space. They last from minutes to hours and are often seen as bright chromospheric ribbons and hot coronal loops on the Sun. Some flares are small and innocent while others can be tremendous and violent. Powerful flares and the often accompanied coronal mass ejections (CMEs) can cause severe influences on the near-Earth environment, resulting in potentially life-threatening consequences (Daglis et al. 2004). Therefore, substantial efforts are being invested on solar-flare research including forecast and mitigation plans.

In this notebook, we attempt to use SDO/HMI vector magnetic field data together with flaring history to predict solar flares that would occur in an AR within 24 hr of a given time point, with a deep-learning method, named long short-term memory (LSTM; Hochreiter & Schmidhuber 1997).

## 2. LSTM Workflow

### 2.1 Data Prepration & Loading

The data folder includes three sub-directories: LSTM_C_sample_run, LSTM_M_sample_run, and LSTM_M5_sample_run.

- The LSTM_C_sample_run includes a CSV training data file that is used to train the model and a CSV test data file that is used to predict the C category and higher flares.
- The LSTM_M_sample_run includes a CSV training data file that is used to train the model and a CSV test data file that is used to predict the M category and higher flares.
- The LSTM_M5_sample_run includes a CSV training data file that is used to train the model and a CSV test data file that is used to predict the M5 category and higher flares.

The files are loaded and used during the testing and training process.

### 2.2 C Flare Model Training and Testing

You may train the model with your own data or train the model with the default data (see Sections 2.2.1 and 2.2.2).

#### 2.2.1 C Flare Model Training with Default Data

Here, we show how to train the 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 path to the training file:
   'train_data_file':'data/LSTM_C_sample_run/normalized_training.csv'
   and replace the value 'data/LSTM_C_sample_run/normalized_training.csv' with your new file name.

In [1]:
```
print('Loading the train_model function...')
from tensorflow.python.keras import regularizers
from flarepredict_train import train_model
args = {'train_data_file':'data/LSTM_C_sample_run/normalized_training.csv',
        'flare': 'C',
       'modelid': 'custom_model_id'
      }
train_model(args)
```

```
Loading the train_model function...
Starting training with a model with id: custom_model_id training data file: data/LSTM_C_sample_run/normalized_training.csv
Loading data set...
(84577, 10, 14)
Done loading data...
Training is in progress, please wait until it is done...

Finished training the C flare model, you may use the flarepredict_test.py program to make prediction.
```

#### 2.2.2 Predicting with Your C Flare Model

To predict the testing data using the model you trained above, make sure the modelid value in the args variable in the following code is set exactly as the one used in the training, for example: 'custom_model_id'.

In [2]:
```
from flarepredict_test import test_model
args =  {'test_data_file': 'data/LSTM_C_sample_run/normalized_testing.csv',
         'flare': 'C',
         'modelid': 'custom_model_id'}
result_file_name = test_model(args)
```

```
Starting testing with a model with id: custom_model_id testing data file: data/LSTM_C_sample_run/normalized_testing.csv
Loading data set...
(185, 10, 14)
Done loading data...
Formatting and mapping the data...
Prediction is in progress, please wait until it is done...
Finished the prediction task..
```

#### 2.2.3 Reading the Results

The prediction result can be shown by passing the result variable to the pandas function read_csv as shown in the following example. The result shows the label your model predicted for each case.

In [3]:
```
import pandas as pd
df = pd.read_csv(result_file_name)
df
```

Out[3]:

| | Predicted Label | label | flare | timestamp | NOAA | HARP | TOTUSJH | Cdec | TOTUSJZ | Chis1d | USFLUX | TOTBSQ | R_VALUE | TOTPOT | Chis | S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Negative | Negative | B2.8 | 2017-03-25T01:22:36.70Z | 12644 | 6972 | -0.805765 | 0.000000 | -0.509193 | 0.000000 | -0.925044 | -0.599012 | -1.136922 | -0.855732 | 0.000000 | |
| **1** | Negative | Negative | B2.8 | 2017-03-25T02:22:36.70Z | 12644 | 6972 | -0.769690 | 0.000000 | -0.473642 | 0.000000 | -0.900514 | -0.592492 | -1.180087 | -0.822543 | 0.000000 | |
| **2** | Negative | Negative | B2.8 | 2017-03-25T04:22:36.80Z | 12644 | 6972 | -0.755696 | 0.000000 | -0.441264 | 0.000000 | -0.900614 | -0.590719 | -1.414541 | -0.819419 | 0.000000 | |
| **3** | Negative | Negative | B2.8 | 2017-03-25T15:22:36.60Z | 12644 | 6972 | -0.713604 | 0.000000 | -0.505940 | 0.000000 | -0.871843 | -0.568816 | -0.503298 | -0.755192 | 0.000000 | |
| **4** | Negative | Negative | B2.8 | 2017-03-25T16:22:36.60Z | 12644 | 6972 | -0.681474 | 0.000000 | -0.348887 | 0.000000 | -0.853421 | -0.556959 | -0.601962 | -0.720032 | 0.000000 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **180** | Positive | Positive | M5.3 | 2017-04-02T07:22:37.90Z | 12644 | 6972 | 1.735370 | 0.126305 | 1.325002 | 0.111111 | 0.833936 | 1.003125 | 0.600823 | 1.487252 | 0.045977 | |
| **181** | Positive | Positive | M2.3 | 2017-04-02T08:22:37.90Z | 12644 | 6972 | 1.810731 | 0.116207 | 1.202598 | 0.111111 | 0.854023 | 1.118890 | 0.716461 | 1.526074 | 0.045977 | |
| **182** | Positive | Positive | M2.3 | 2017-04-02T09:22:37.90Z | 12644 | 6972 | 1.799658 | 0.106915 | 1.588251 | 0.111111 | 0.897038 | 1.193614 | 0.740491 | 1.437500 | 0.045977 | |
| **183** | Positive | Positive | M2.3 | 2017-04-02T10:22:37.90Z | 12644 | 6972 | 1.819823 | 0.098367 | 1.659994 | 0.111111 | 0.943544 | 1.256551 | 0.717097 | 1.448933 | 0.045977 | |
| **184** | Positive | Positive | M2.3 | 2017-04-02T11:22:37.90Z | 12644 | 6972 | 1.968782 | 0.090502 | 1.791367 | 0.111111 | 1.001174 | 1.383610 | 0.689125 | 1.500635 | 0.045977 | |

185 rows × 20 columns

## 2.3 M Flare Model Training and Testing

You may train the model with your own data or train the model with the default data (see Sections 2.3.1 and 2.3.2).

#### 2.3.1 M Flare Model Training with Default Data

Here, we show how to train the 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 path to the training file:
   'train_data_file':'data/LSTM_M_sample_run/normalized_training.csv'
   and replace the value 'data/LSTM_M_sample_run/normalized_training.csv' with your new file name.

In [4]:
```
print('Loading the train_model function...')
from flarepredict_train import train_model
args = {'train_data_file':'data/LSTM_M_sample_run/normalized_training.csv',
        'flare': 'M',
       'modelid': 'custom_model_id'
      }
train_model(args)
```

```
Loading the train_model function...
Starting training with a model with id: custom_model_id training data file: data/LSTM_M_sample_run/normalized_training.csv
Loading data set...
(84742, 10, 22)
Done loading data...
Training is in progress, please wait until it is done...

Finished training the M flare model, you may use the flarepredict_test.py program to make prediction.
```

#### 2.3.2 Predicting with Your M Flare Model

To predict the testing data using the model you trained above, make sure the modelid value in the args variable in the following code is set exactly as the one used in the training, for example: 'custom_model_id'.

In [5]:
```
from flarepredict_test import test_model
args =  {'test_data_file': 'data/LSTM_M_sample_run/normalized_testing.csv',
         'flare': 'M',
         'modelid': 'custom_model_id'}
result_file_name = test_model(args)
```

```
Starting testing with a model with id: custom_model_id testing data file: data/LSTM_M_sample_run/normalized_testing.csv
Loading data set...
(212, 10, 22)
Done loading data...
Formatting and mapping the data...
Prediction is in progress, please wait until it is done...
Finished the prediction task..
```

#### 2.3.3 Reading the Results

The prediction result can be shown by passing the result variable to the pandas function read_csv as shown in the following example. The result shows the label your model predicted for each case.

In [6]:
```
import pandas as pd
df = pd.read_csv(result_file_name)
df
```

Out[6]:

| | Predicted Label | Label | flare | timestamp | NOAA | HARP | TOTUSJH | TOTUSJZ | TOTPOT | TOTBSQ | ... | Xmax1d | Mhis | R_VALUE | Mdec | MEANPC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Negative | Negative | C1.0 | 2015-04-04T05:34:17.60Z | 12320 | 5415 | -0.104048 | -0.301538 | -0.137916 | -0.364581 | ... | 0.000000 | 0.000000 | -0.841946 | 0.000000 | 0.68745 |
| **1** | Negative | Negative | C1.0 | 2015-04-04T06:34:17.60Z | 12320 | 5415 | -0.078954 | -0.175091 | -0.094344 | -0.356320 | ... | 0.000000 | 0.000000 | -0.792015 | 0.000000 | 0.43214 |
| **2** | Negative | Negative | C1.0 | 2015-04-04T07:34:17.60Z | 12320 | 5415 | -0.043782 | -0.338201 | -0.018729 | -0.349030 | ... | 0.000000 | 0.000000 | -0.798981 | 0.000000 | 0.32862 |
| **3** | Negative | Negative | C1.0 | 2015-04-04T08:34:17.60Z | 12320 | 5415 | 0.018644 | -0.306688 | 0.096848 | -0.326073 | ... | 0.000000 | 0.000000 | -0.690922 | 0.000000 | 0.66244 |
| **4** | Negative | Negative | C1.0 | 2015-04-04T09:34:17.60Z | 12320 | 5415 | 0.055247 | -0.372511 | 0.116056 | -0.300240 | ... | 0.000000 | 0.000000 | -0.600492 | 0.000000 | 0.73223 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **207** | Negative | Negative | C2.9 | 2015-04-13T15:34:18.70Z | 12320 | 5415 | 0.947701 | -0.229454 | 0.790858 | 0.109802 | ... | 0.006882 | 0.038462 | -0.572136 | 0.000007 | 1.05675 |
| **208** | Negative | Negative | C2.9 | 2015-04-13T16:34:18.70Z | 12320 | 5415 | 0.945490 | -0.256896 | 0.851387 | 0.127448 | ... | 0.006882 | 0.038462 | -0.496255 | 0.000007 | 1.08873 |
| **209** | Negative | Negative | C2.9 | 2015-04-13T17:34:18.70Z | 12320 | 5415 | 0.826441 | -0.314707 | 0.766035 | 0.110474 | ... | 0.006882 | 0.038462 | -0.503343 | 0.000006 | 1.08815 |
| **210** | Negative | Negative | C2.9 | 2015-04-13T18:34:18.70Z | 12320 | 5415 | 0.944913 | -0.175675 | 0.811451 | 0.161015 | ... | 0.006882 | 0.038462 | -0.487871 | 0.000006 | 1.05209 |
| **211** | Negative | Negative | C2.9 | 2015-04-13T19:34:18.80Z | 12320 | 5415 | 0.959147 | -0.067101 | 0.905028 | 0.203846 | ... | 0.006882 | 0.038462 | -0.399229 | 0.000005 | 1.03174 |

212 rows × 28 columns

## 2.4 M5 Flare Model Training and Testing

You may train the model with your own data or train the model with the default data (see Sections 2.4.1 and 2.4.2).

### 2.4.1 M5 Flare Model Training with Default Data

Here, we show how to train the 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 path to the training file:
   'train_data_file':'data/LSTM_M5_sample_run/normalized_training.csv'
   and replace the value 'data/LSTM_M5_sample_run/normalized_training.csv' with your new file name.

In [7]:
```
print('Loading the train_model function...')
from flarepredict_train import train_model
args = {'train_data_file':'data/LSTM_M5_sample_run/normalized_training.csv',
        'flare': 'M5',
       'modelid': 'custom_model_id'
      }
train_model(args)
```

```
Loading the train_model function...
Starting training with a model with id: custom_model_id training data file: data/LSTM_M5_sample_run/normalized_training.csv
Loading data set...
(84798, 10, 20)
Done loading data...
Training is in progress, please wait until it is done...

Finished training the M5 flare model, you may use the flarepredict_test.py program to make prediction.
```

### 2.4.2 Predicting with Your M5 Flare Model

To predict the testing data using the model you trained above, make sure the modelid value in the args variable in the following code is set exactly as the one used in the training, for example: 'custom_model_id'.

In [8]:
```
from flarepredict_test import test_model
args =  {'test_data_file': 'data/LSTM_M5_sample_run/normalized_testing.csv',
         'flare': 'M5',
         'modelid': 'custom_model_id'}
result_file_name = test_model(args)
```

```
Starting testing with a model with id: custom_model_id testing data file: data/LSTM_M5_sample_run/normalized_testing.csv
Loading data set...
(189, 10, 20)
Done loading data...
Formatting and mapping the data...
Prediction is in progress, please wait until it is done...
Finished the prediction task..
```

### 2.4.3 Reading the Results

The prediction result can be shown by passing the result variable to the pandas function read_csv as shown in the following example. The result shows the label your model predicted for each case.

In [9]:
```
import pandas as pd
df = pd.read_csv(result_file_name)
df
```

Out[9]:

| | Predicted Label | label | flare | timestamp | NOAA | HARP | TOTUSJH | SAVNCPP | ABSNJZH | TOTPOT | ... | Edec | Mhis | Xmax1d | Mdec | AREA_A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Positive | Negative | N | 2017-08-30T08:58:42.90Z | 12673 | 7115 | -0.498770 | -0.503428 | -0.374758 | -0.578207 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 2.485 |
| **1** | Positive | Negative | N | 2017-08-30T09:58:42.90Z | 12673 | 7115 | -0.503880 | -0.510741 | -0.418607 | -0.588074 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 2.535 |
| **2** | Positive | Negative | N | 2017-08-30T10:58:42.90Z | 12673 | 7115 | -0.488118 | -0.504173 | -0.449640 | -0.571011 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 2.289 |
| **3** | Positive | Negative | N | 2017-08-30T11:58:42.80Z | 12673 | 7115 | -0.481332 | -0.500172 | -0.444952 | -0.574907 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 2.316 |
| **4** | Positive | Negative | N | 2017-08-30T12:58:42.80Z | 12673 | 7115 | -0.473880 | -0.501714 | -0.419139 | -0.567325 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 2.338 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **184** | Negative | Negative | M1.3 | 2017-09-08T01:58:41.80Z | 12673 | 7115 | 4.957265 | 3.600708 | 5.360394 | 3.316161 | ... | 0.173517 | 0.692308 | 0.139785 | 0.306003 | 2.061 |
| **185** | Negative | Negative | M1.2 | 2017-09-08T02:58:41.80Z | 12673 | 7115 | 4.998832 | 3.638885 | 5.449238 | 3.350028 | ... | 0.172529 | 0.730769 | 0.139785 | 0.449894 | 2.045 |
| **186** | Negative | Positive | M8.1 | 2017-09-08T03:58:41.80Z | 12673 | 7115 | 4.875287 | 3.552344 | 5.284544 | 3.316022 | ... | 0.171174 | 0.769231 | 0.139785 | 0.586064 | 2.201 |
| **187** | Negative | Positive | M8.1 | 2017-09-08T04:58:41.80Z | 12673 | 7115 | 4.843132 | 3.583137 | 5.129436 | 3.522198 | ... | 0.157488 | 0.769231 | 0.139785 | 0.539205 | 2.217 |
| **188** | Negative | Positive | M8.1 | 2017-09-08T05:58:41.80Z | 12673 | 7115 | 4.762235 | 3.627277 | 5.090359 | 3.258578 | ... | 0.152696 | 0.769231 | 0.139785 | 0.496092 | 1.978 |

189 rows × 26 columns

## 3. Acknowledgment

We thank the team of SDO/HMI for producing vector magnetic data products. The flare catalogs were prepared by and made available through NOAA NCEI. This work was supported by U.S. NSF grants AGS-1927578 and AGS-1954737.

## 4. References

DeepSun: Predicting Solar Flares Using a Long Short-term Memory Network

Hao Liu, Chang Liu, Jason T. L. Wang and Haimin Wang

https://iopscience.iop.org/article/10.3847/1538-4357/ab1b3c (https://iopscience.iop.org/article/10.3847/1538-4357/ab1b3c)

https://github.com/deepsuncode/LSTM-flare-prediction (https://github.com/deepsuncode/LSTM-flare-prediction)