Identifying and Tracking Solar Magnetic Flux Elements with Deep Learning

Haodi Jiang, Jiasheng Wang, Chang Liu, Ju Jing, Hao Liu, Jason T. L. Wang and Haimin Wang

Institute for Space Weather Sciences, New Jersey Institute of Technology

1. Introduction

Deep learning has drawn significant interest in recent years due to its effectiveness in processing big and complex observational data gathered from diverse instruments. Here we propose a new deep learning method, called SolarUnet, to identify and track solar magnetic flux elements or features in observed vector magnetograms. SolarUnet is applied to data from the 1.6 meter Goode Solar Telescope at the Big Bear Solar Observatory.

In this notebook, we provide an overview of the SolarUnet tool, detailing how it can be used to identify and track solar magnetic flux elements.

2. Workflow of SolarUnet

2.1 Data Preparation

Import the pre_processing() function from the solarunet module.

Convert the SWAMIS 3-class masks to 2-class masks for model training. You may put your data into this directory.

In [1]:

```
from solarunet import pre_processing
input_path = 'data/exmaple_data_preprocess/3_class/'
output_path = 'data/exmaple_data_preprocess/2_class/'
print('Pre-processing example data...')
pre_processing(input_path, output_path)
print('Finished the pre-processing.')
```

```
Pre-processing example data...
intgr_180607_161752.fts
intgr_180607_161904.fts
Finished the pre-processing.
```

2.2 Model Training and Testing

You may train the model with your own data (see Section 2.2.1) or directly use the pretrained model (see Section 2.2.2) for prediction and feature tracking.

2.2.1 Training and Predicting

Import the model_training() and model_predicting() functions from the solarunet module.

The model is trained with the data prepared in Section 2.1 and tested on the given magnetograms. Please make sure your input data is in the given directory or you may create your own directory and modify the path. The predicted results will be saved in the given path.

Note: The model is not trained in this Jupyter notebook because it requires a large amount of memory. Therefore, the testing is done using the pre-trained model.

To train the model in Colab, visit [here (https://colab.research.google.com/drive/1zOwNwkepiAokuAFWTwn9bx7Pf84c1vWT?usp=sharing)](https://colab.research.google.com/drive/1zOwNwkepiAokuAFWTwn9bx7Pf84c1vWT?usp=sharing).

2.2.2 Predicting with the Pretrained Model

Import the model_predicting() function from the solarunet module.

Predict the binary masks of the given magnetograms by using the pretrained model. We set the 3rd argument of model_predicting() as True. The predicted results will be saved in the given path.

In [2]:

```
from solarunet import model_predicting
input_path ='data/magnetic/'
output_path ='results/predicted_mask/'
model_predicting(input_path, output_path, pretrain=True)
```

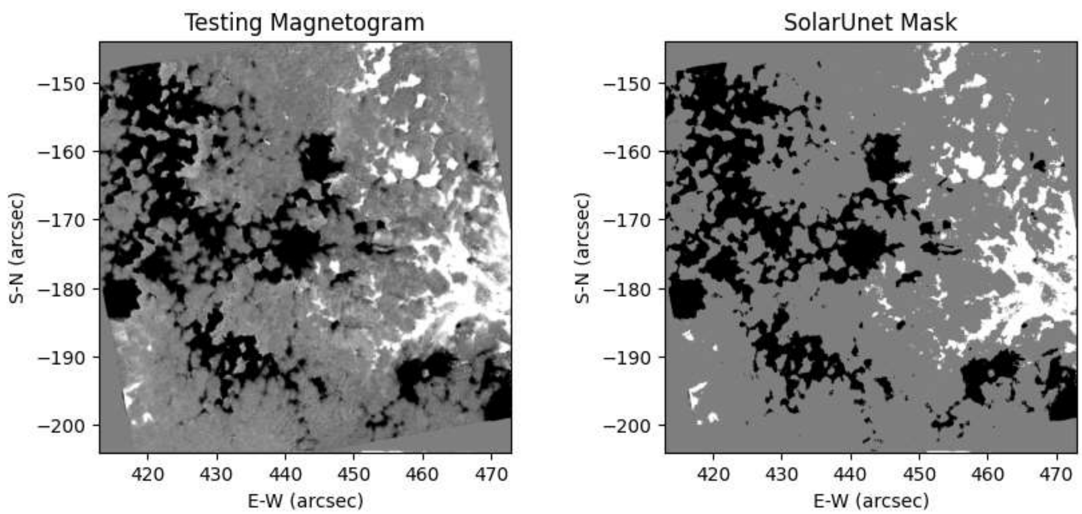
```
3/3 [==============================] - 2s 568ms/step
Prediction on the given data done
```

2.3 Postprocessing Data

Import the post_processing() and plot_mask() functions from the solarunet module.

Convert the predicted binary masks to 3-class masks and use the plot_mask() function to draw the SolarUnet masks of the testing magnetograms.

In [3]:
```
from solarunet import post_processing
from solarunet import plot_mask
%matplotlib inline
post_processing()
plot_mask()
```



2.4 Magnetic Tracking

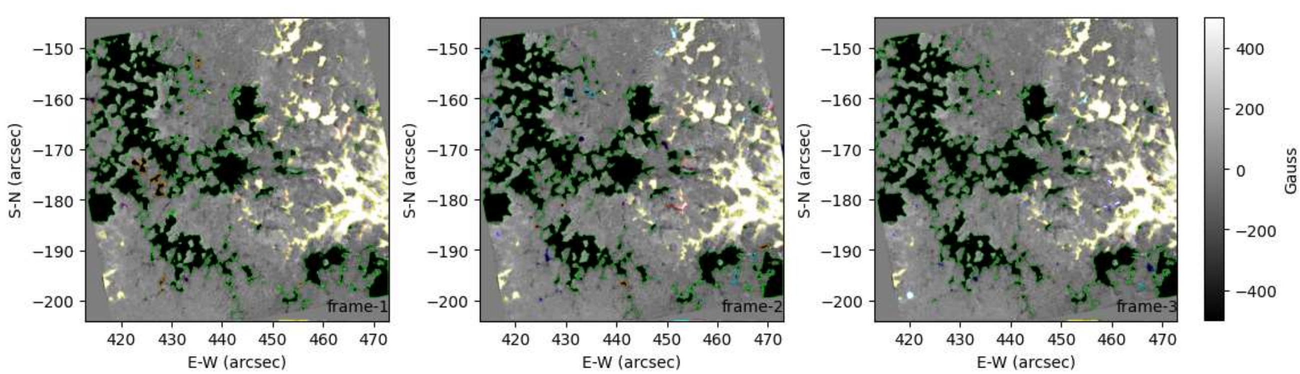
Import the magnetic_tracking() function from the magnetic_tracking module.

Magnetic tracking algorithms are performed on the three consecutive testing magnetograms. An option of saving feature lifetime is provided through the 3rd argument of the magnetic_tracking() function. The tracking results will be saved in the given path.

In [4]:
```
from magnetic_tracking import magnetic_tracking
input_path = 'results/processed_data_for_tracking/'
output_path = 'results/tracking_results/'
magnetic_tracking(input_path, output_path)
# lifetime_path = 'data/statistics_analysis/lifetime'
# magnetic_tracking(input_path, output_path, lifetime_path)
```

```
============magnetic tracking start============
-----------------process frame 1------------------
-----------------process frame 2------------------
-----------------process frame 3------------------
-----------------Done------------------
```

In [5]:
```
from solarunet import plot_tracking_results
%matplotlib inline
plot_tracking_results()
```

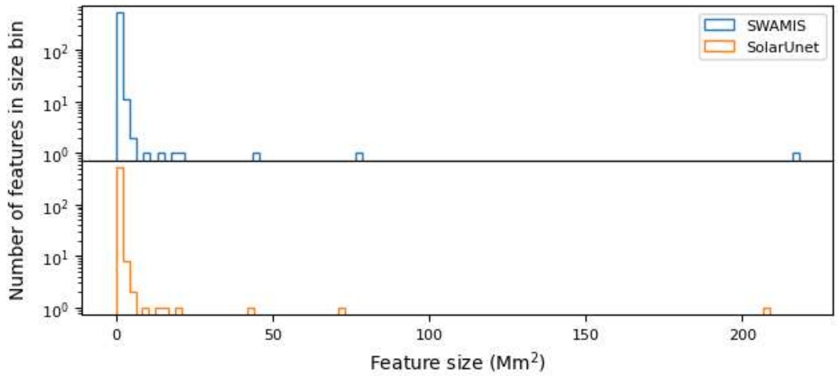


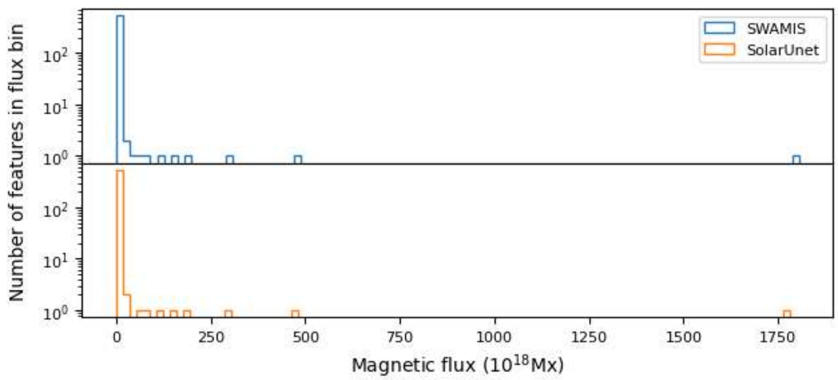
2.5 Statistics Analysis

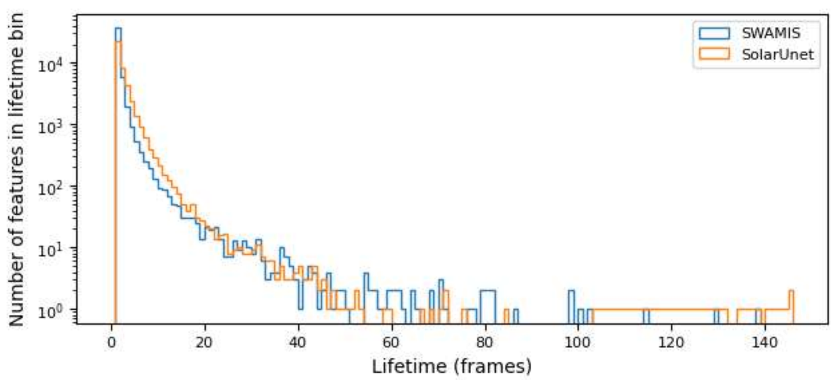
Import the analysis() function from the statistics_analysis module.

The statistics analysis in this work is demoed as follows:

In [6]:
```
from statistics_analysis import analysis
%matplotlib inline
analysis()
```







3. Conclusion

We develop a deep learning method, SolarUnet, for tracking signed magnetic flux elements (features) and detecting magnetic events in observed vector magnetograms. We apply the SolarUnet tool to data from the 1.6 meter Goode Solar Telescope (GST) at the Big Bear Solar Observatory (BBSO). The tool is able to identify the magnetic features and detect three types of events, namely disappearance, merging and cancellation, in the death category and three types of events, namely appearance, splitting and emergence, in the birth category. We use the BBSO/GST images to illustrate how our tool works on feature identification and event detection, and compares with the widely used SWAMIS tool.

Acknowledgment

We thank the BBSO/GST team for providing the data used in this study. This work was supported by US NSF grants AGS-1927578 and AGS-1954737. C.L. and H.W. acknowledge the support of NASA under grants NNX16AF72G, 80NSSC18K0673, and 80NSSC18K1705.