The Astrophysical Journal Supplement Series, 250:5 (13pp), 2020 September https://doi.org/10.3847/1538-4365/aba4aa

Identifying and Tracking Solar Magnetic Flux Elements with Deep Learning

Haodi Jiang[1,2], Jiasheng Wang[1,3,4], Chang Liu[1,3,4], Ju Jing[1,3,4], Hao Liu[1,2], Jason T. L. Wang[1,2], and Haimin Wang[1,3,4]
[1] Institute for Space Weather Sciences, New Jersey Institute of Technology, University Heights, Newark, NJ 07102-1982, USA; hj78@njit.edu, jw438@njit.edu, chang.liu@njit.edu, ju.jing@njit.edu, hl422@njit.edu, wangj@njit.edu, haimin.wang@njit.edu
[2] Department of Computer Science, New Jersey Institute of Technology, University Heights, Newark, NJ 07102-1982, USA
[3] Center for Solar-Terrestrial Research, New Jersey Institute of Technology, University Heights, Newark, NJ 07102-1982, USA
[4] Big Bear Solar Observatory, New Jersey Institute of Technology, 40386 North Shore Lane, Big Bear City, CA 92314-9672, USA
Received 2019 November 2; revised 2020 July 3; accepted 2020 July 8; published 2020 August 26

Abstract

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 based on the Southwest Automatic Magnetic Identification Suite (SWAMIS). Our method consists of a data preprocessing component that prepares training data from the SWAMIS tool, a deep learning model implemented as a U-shaped convolutional neural network for fast and accurate image segmentation, and a postprocessing component that prepares tracking results. SolarUnet is applied to data from the 1.6 m Goode Solar Telescope at the Big Bear Solar Observatory. When compared to the widely used SWAMIS tool, SolarUnet is faster while agreeing mostly with SWAMIS on feature size and flux distributions and complementing SWAMIS in tracking long-lifetime features. Thus, the proposed physics-guided deep learning-based tool can be considered as an alternative method for solar magnetic tracking.

Unified Astronomy Thesaurus concepts: Solar magnetic fields (1503); Solar photosphere (1518); Convolutional neural networks (1938)

1. Introduction

Tracking magnetic flux elements is an important subject in heliophysics research (DeForest et al. 2007; Leenaarts et al. 2015; Wang et al. 2018).[5] Identifying and tracking the surface magnetic elements is useful in deriving statistical parameters of the local and global solar dynamo, allowing for sophisticated analyses of solar activity (DeForest et al. 2007). It not only helps scientists understand the distribution of magnetic fluxes (Parnell 2002), but also helps estimate the amount of energy in acoustic waves, which plays an important part in the heating of the solar chromosphere and corona (Fossum & Carlsson 2006). In addition, magnetic tracking is useful in deriving boundary conditions of magnetohydrodynamic modeling of the solar corona and solar wind. In the past, many researchers have studied the behaviors and patterns of magnetic flux elements. For instance, Chen et al. (2015) developed a technique to detect and classify small-scale magnetic flux cancellations and link them to chromospheric rapid blueshifted excursions. Giannattasio et al. (2018) investigated the occurrence and persistence of magnetic elements in the quiet Sun to understand the scales of organization at which turbulent convection operates. Moreno-Insertis et al. (2018) reported findings related to small-scale magnetic flux emergence in the quiet Sun.

In magnetic tracking, features are defined as a visually identifiable part of an image, such as a clump of magnetic flux or a blob in a magnetogram. One of the most popular software tools for magnetic feature tracking across multiple images/frames is the Southwest Automatic Magnetic Identification Suite (SWAMIS; DeForest et al. 2007). SWAMIS takes five steps to track magnetic flux elements: (1) feature discrimination for each frame, (2) feature identification within a frame, (3) feature association across frames, (4) occasional noise filtering, and (5) event detection (Chen et al. 2015). Magnetic events are broadly classified into two categories: death and birth (Lamb et al. 2008); the former refers to the end of a magnetic feature's existence while the latter refers to the start of a magnetic feature's existence.

In this paper, we present a new tool, called SolarUnet, to track magnetic flux elements. Our tool is built using deep learning (LeCun et al. 2015). The tool can detect three different types of events in each category, namely (i) disappearance and appearance, (ii) merging and splitting, and (iii) cancellation and emergence. The event "disappearance" is defined as the end of a single unipolar magnetic feature that "fades away" to nothing in the absence of nearby features across two frames; the opposite event "appearance" is defined as the origin of a single unipolar feature where the unipolar feature does not exist in the previous frame. The event "merging" is defined as the combination of two or more like-sign features into a single magnetic feature; the opposite event "splitting" is defined as the breakup of a single magnetic feature into at least two like-sign features, where the total flux of all child features is roughly the same as that of the parent feature. The event "cancellation" is defined as the demise of a magnetic feature that collides with one or more opposite-sign features, resulting in the demise of these features or an alive feature carrying the remaining flux; the event "emergence" is defined as the appearance of opposite-sign features with approximately the same magnitude or a new feature adjacent to previously existing opposite-sign features in a nearly flux-conserving manner.

Deep learning, which is a subfield of machine learning, has drawn significant interest in recent years (LeCun et al. 2015). Inspired by its success in computer vision, speech recognition, and natural language processing, researchers have started to use deep learning in astronomy and astrophysics

[5] In the study presented here, we focus on tracking signed, including positive and negative, magnetic flux elements.

Table 1
Numbers of Images Used in Our Study

Number of Training Images	Number of Testing Images
196 (from the second collection)	147 (from the first collection)

(Huertas-Company et al. 2018; Leung & Bovy 2018; Kim et al. 2019; Lieu et al. 2019; Liu et al. 2019; Wu & Boada 2019). In contrast to the existing methods for magnetic tracking (Lamb et al. 2010, 2013; Chen et al. 2015), our SolarUnet tool is built using deep learning. Compared to the most closely related magnetic tracking tool, SWAMIS, which uses hysteresis as the discrimination scheme and a gradient-based "downhill" method to identify features in a frame, SolarUnet runs faster while producing similar or complementary results.

The rest of this paper is organized as follows. Section 2 describes observations and data used in this study. Section 3 presents details of SolarUnet and tracking algorithms used by the tool. Section 4 reports experimental results. Section 5 concludes the paper.

2. Observations and Data Preparation

We adopted two collections of observations in this study. The first collection was conducted by the Near InfraRed Imaging Spectropolarimeter (NIRIS; Cao et al. 2012) of the 1.6 m Goode Solar Telescope (GST) at the Big Bear Solar Observatory (BBSO; Cao et al. 2010; Goode et al. 2010; Goode & Cao 2012; Varsik et al. 2014). This collection contained observations of the magnetic polarity inversion region in National Oceanic and Atmospheric Administration Active Region (NOAA AR) 12665 (431″, −131″) during ∼20:16–22:42 UT on 2017 July 13. The obtained data included spectro-polarimetric observations of a full set of Stokes measurements at the Fe I 1564.8 nm line (0.25 Å bandpass) by NIRIS with a field of view (FOV) of 80″ at 0.″24 resolution and 56 s cadence. Vector magnetic field products in local coordinates were constructed after removing azimuth ambiguity (Leka et al. 2009).

The second collection of observations was conducted with a clear seeing condition; BBSO/GST achieved diffraction-limited imaging during ∼16:17–22:17 UT on 2018 June 7. The obtained multiwavelength observations revealed detailed structural and evolutionary properties of small-scale magnetic polarities in quiescent solar regions north of the disk center (−32″, 294″). The essential data included in this collection were the images taken by the GST's NIRIS using a 2048 × 2048 pixels Teledyne camera with a ∼80″ FOV. The spatial resolution (at a diffraction limit of $\theta = \lambda/D$) of the NIRIS images was 0.″2, and the temporal cadence was 56 s. The magnetograms were then aligned based on sunspot and plage features, with an alignment accuracy within 0.″3, which was the best accuracy by using interpolation.

We prepared our training and testing sets by using the magnetograms taken from the two collections of observations described above. Because the magnetograms taken on 2018 June 7 had higher quality than the observations conducted on 2017 July 13, we used the higher-quality magnetograms to prepare our training data so as to obtain a better magnetic tracking model. Specifically, we gathered all 202 frames from the second collection of observations and excluded 6 images with poor quality (these excluded images were very noisy). The remaining 196 frames were used as training data for magnetic tracking. The testing set contained all 147 magnetograms from the first collection of observations. Table 1 summarizes the numbers of training and testing images used in this study.

3. Methodology

3.1. Overview of SolarUnet

Figure 1 explains how SolarUnet works. Training magnetograms are pre-processed in steps 1 and 2, and then used to train the deep learning model for image segmentation (step 3). The trained model takes a testing magnetogram (step 4) and produces a predicted mask (step 5). Through postprocessing of the predicted mask, SolarUnet produces magnetic tracking results (step 6).

Specifically, in step 1, we apply SWAMIS with the downhill option to the 196 training magnetograms to get 196 masks. These images, including the magnetograms and masks, are converted to 8 bit grayscale images of 720 × 720 pixels, which are suitable for our deep learning model. Pixels in the masks belong to three classes represented by three colors/labels respectively: positive magnetic flux with a label of 1 (white), negative magnetic flux with a label of −1 (black), and nonsignificant flux with a label of 0 (gray). During preprocessing, we convert the 196 three-class masks obtained from SWAMIS to 196 two-class (binary) masks by (i) changing the label of the nonsignificant flux regions from 0 to 1, and (ii) changing both the positive magnetic flux regions and negative magnetic flux regions to significant flux regions with label −1 (step 2).

The 196 magnetograms (images) and two-class (binary) masks are then used to train the deep learning model, implemented in TensorFlow (Abadi et al. 2016) and Keras (Chollet 2018), for image segmentation (step 3). Because our deep learning model needs a large amount of data in order to train successfully, the model invokes the ImageDataGenerator[6] in Keras to perform data augmentation, expanding the training set by shifting, rotating, flipping and scaling the training images during the model training process. Shifting an image is to move all pixels of the image horizontally or vertically while keeping the dimensions of the image the same. Rotating an image is to rotate the image clockwise by a given number of degrees from 0 to 360. Flipping an image is to reverse the rows or columns of pixels in the image. Scaling an image is to randomly zoom in on the image and either add new pixel values around the image or interpolate pixel values in the image. We train the deep learning model using 1 epoch with 10,000 iterations/epoch. In each iteration, the model randomly selects one of the 196 training magnetograms and its binary mask, feeds them to the ImageDataGenerator to generate a synthetic magnetogram and binary mask, and uses the synthetic magnetogram and binary mask to train the model. There are 10,000 iterations and hence 10,000 synthetic magnetograms and binary masks are generated through the data augmentation process, where the 10,000 generated magnetograms and binary masks are used for model training.[7] We have chosen to use data

[6] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

[7] Notice that SWAMIS is applied only to the 196 training magnetograms mentioned in Table 1; SWAMIS is never run on the 10,000 generated (synthetic) magnetograms.

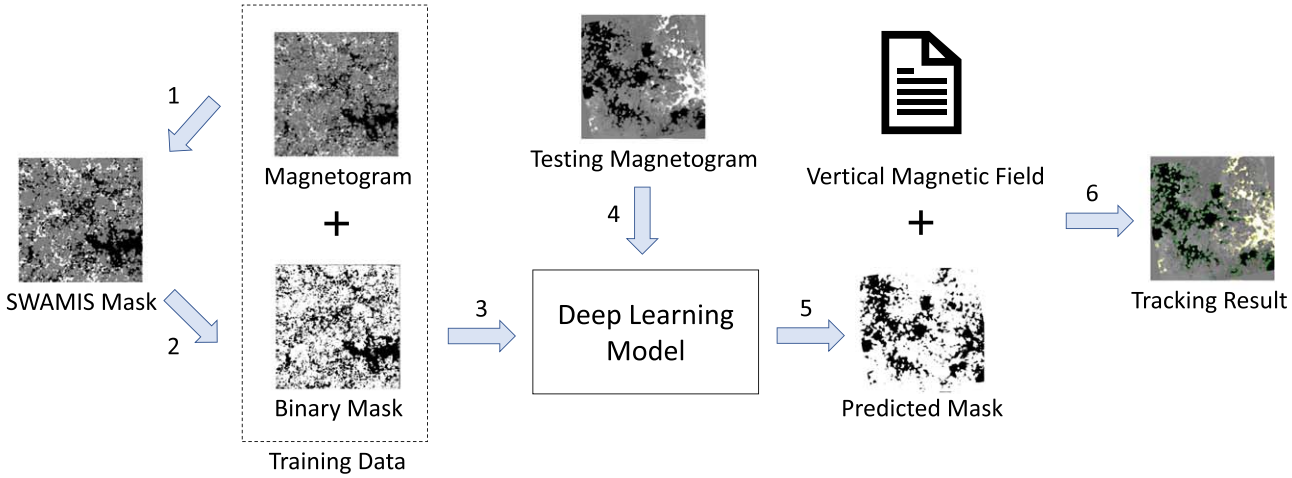

Figure 1. Illustration of the proposed method (SolarUnet) for identifying and tracking solar magnetic flux elements. SolarUnet employs a deep learning model for image segmentation. The training data used to train the deep learning model are highlighted in the dashed box.

augmentation as opposed to acquiring more training data because the quality of ground-based observations is subject to many factors such as seeing conditions and observing time limits. Obtaining large volumes of high-quality training data requires further observations. Nevertheless, using the synthetic training images produces reasonably good results as shown in Section 4.

When a testing magnetogram is submitted, it is converted to a 8 bit grayscale image of 720 × 720 pixels and fed to the trained deep learning model (step 4). The trained deep learning model predicts a two-class (binary) mask, containing non-significant flux regions with label 1 and significant flux regions with label −1 (step 5). We convert the predicted two-class (binary) mask back to a three-class mask via postprocessing as follows. For the nonsignificant flux regions with label 1, we change their label from 1 to 0. For the significant flux regions with label −1, we use the information of radial components in the vertical magnetic field image shown in Figure 1, where the radial components are perpendicular to the plane of the Sun, to reconstruct positive and negative magnetic flux regions. Specifically, for each pixel x in the significant flux regions in the predicted binary mask, we check the magnetic strength of the pixel, y, at x's corresponding location in the vertical magnetic field image. If y's magnetic strength is greater than 150 G, we set x as a positive magnetic flux and change the label of this pixel from −1 to 1. If y's magnetic strength is smaller than −150 G, we set x as a negative magnetic flux and the label of this pixel remains −1. If y's magnetic strength is between −150 and 150 G, we set x as a nonsignificant flux and change the label of this pixel from −1 to 0. This yields a three-class mask with the polarity information.

Finally, we apply our magnetic tracking algorithms described in Section 3.3 to the testing magnetogram and masks to get tracking results (step 6). Magnetic tracking is often involved with more than one testing magnetogram, and we output the tracking results in all of the testing magnetograms.

3.2. Implementation of the Deep Learning Model in SolarUnet

Figure 2 illustrates the deep learning model used in SolarUnet, which is a U-shaped convolutional neural network. We adapt U-Net (Falk et al. 2019) to our work, enhancing it to obtain our model. The model has an encoder, a bottleneck, and a decoder, followed by a pixel-wise binary classification layer.[8] The encoder consists of four blocks: E1, E2, E3, and E4. Each block has two 3 × 3 convolution layers, represented by blue arrows, followed by a 2 × 2 max pooling layer, represented by a red arrow. In each convolution layer, we adopt batch normalization (BN; Ioffe & Szegedy 2015) after convolution, followed by a rectified linear unit (ReLU) activation function. Furthermore, we add a dropout layer (Srivastava et al. 2014) after each max pooling layer. The four encoder blocks E1, E2, E3, and E4 have 32, 64, 128, and 256 kernels, respectively.

The bottleneck, denoted Bot, mediates between the encoder and the decoder. It uses two 3 × 3 convolution layers followed by a 2 × 2 up-convolution layer, represented by a green arrow. The bottleneck has 512 kernels. Similar to the encoder, the decoder consists of four blocks: D1, D2, D3, and D4. Each block has two 3 × 3 convolution layers followed by a 2 × 2 up-convolution layer. The four decoder blocks D1, D2, D3, and D4 have 256, 128, 64, and 32 kernels, respectively. The input of each decoder block is concatenated by the output of the corresponding encoder block where the concatenation is represented by a gray arrow. A dropout layer is added after each concatenation. Finally, a 1 × 1 convolution layer, represented by a turquoise arrow, with two kernels followed by a softmax activation function, is used to produce a segmentation mask. During testing, the deep learning model takes as input a testing magnetogram and produces a two-class mask as an output.

The input resolution of the encoder block E1 is set to 720 × 720 pixels to match the size of the testing magnetogram. Each max pooling layer reduces the size by a factor of 2. Hence, the input resolution of the encoder block E2 (E3 and E4, respectively) is 360 × 360 (180 × 180 and 90 × 90, respectively) pixels. The input resolution of the bottleneck, Bot, is 45 × 45 pixels. Each up-convolution layer increases the size by a factor of 2. Thus, the input resolution of the decoder block D1 (D2, D3, and D4, respectively) is 90 × 90 (180 × 180, 360 × 360, and 720 × 720, respectively) pixels.

[8] Please see the Appendix for more detailed descriptions of the technical terms used here.

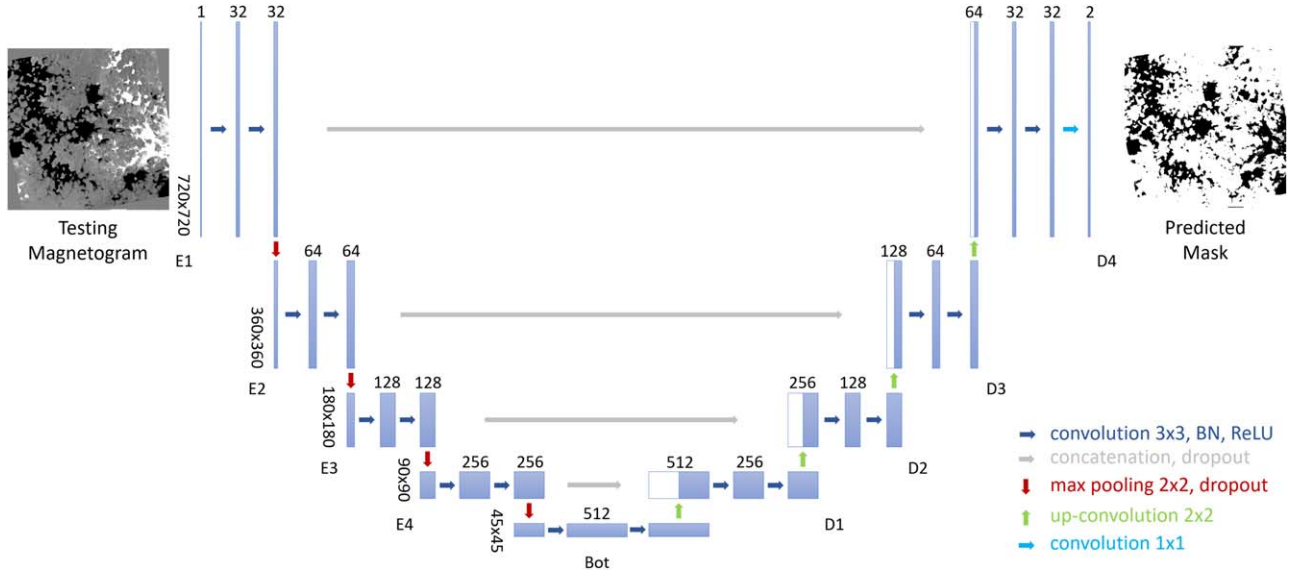

Figure 2. Illustration of the deep learning model used in SolarUnet. This model is a U-shaped convolutional neural network, consisting of an encoder, a bottleneck, a decoder, followed by a pixel-wise binary classification layer. The encoder is comprised of four blocks: E1, E2, E3, and E4. Each block has two 3 × 3 convolution layers, represented by blue arrows, followed by a 2 × 2 max pooling layer, represented by a red arrow. The decoder is also comprised of four blocks: D1, D2, D3, and D4. Each block has two 3 × 3 convolution layers followed by a 2 × 2 up-convolution layer, represented by a green arrow. The bottleneck, denoted Bot, mediates between the encoder and the decoder. It uses two 3 × 3 convolution layers followed by a 2 × 2 up-convolution layer.

The loss function, L, used by the deep learning model is the binary cross-entropy function defined below:

$$L = -\sum_{x} \log y_c(x, W). \tag{1}$$

Here, W is the parameters of the convolutional neural network, $y_c(x, W)$ is the output of the softmax layer of the convolutional neural network, and c is the class label (1 versus −1) of each pixel x.

In training the deep learning model, we adopt adaptive moment estimation (Adam) to find the optimal parameters of the model. The learning rate of Adam is set to 0.0001. Adam combines the advantages of two popular methods: AdaGrad and RMSProp (Goodfellow et al. 2016). In most cases, Adam achieves better performance than other stochastic optimization methods including the stochastic gradient descent (SGD) with momentum employed by U-Net (Goodfellow et al. 2016).

Although both our deep learning model and U-Net (Falk et al. 2019) have the same U-shaped architecture, they differ in several ways. First, U-Net used SGD with a momentum of 0.99 to train and optimize its model. By contrast, we choose Adam because it achieves better performance in our case where the training process would be trapped in a poor local minimum if SGD were used. Second, U-Net focused on imbalanced data sets and used a weighted cross-entropy loss function to tackle the imbalanced classification problem. By contrast, because our training set is relatively balanced in the sense that nonsignificant flux regions roughly have the same number of pixels as significant flux regions, we use the binary cross-entropy loss function as defined in Equation (1). Third, we adopt BN and dropout layers, which were not used by U-Net. BN improves model learning, stabilizes the learning process, reduces the learning (training) time, and improves prediction accuracy (Ioffe & Szegedy 2015). Dropout prevents neural networks from overfitting (Srivastava et al. 2014), where overfitting means that a trained model fits training data too well and cannot generalize to make predictions on unseen testing data. Finally, we reduce the numbers of kernels of the encoder, bottleneck, and decoder blocks by a factor of 2 compared to U-Net to speed up the training process and reduce GPU memory usage.

3.3. Algorithms for Magnetic Tracking and Event Detection

After describing the deep learning model used in SolarUnet, we now turn to the magnetic tracking algorithms employed by SolarUnet. Based on the positive magnetic flux regions and negative magnetic flux regions found in Section 3.1, we identify signed magnetic flux elements or features in a magnetogram (image/frame) by utilizing a connected-component labeling algorithm (He et al. 2009) to group all adjacent segments in the positive magnetic flux regions and negative magnetic flux regions, respectively, if their pixels in edges or corners touch each other. We filter out those magnetic features whose sizes are smaller than a user-determined threshold. The features eliminated from consideration are treated as noise. Then, we assign each of the remaining features a label number and highlight the features with different bordering colors. Finally, we consider the association of features (magnetic flux elements) across different frames to perform event detection.

Based on the observational data and instruments used, we calculate the moving distance D (number of pixels) of a magnetic flux element X as follows:

$$D = \frac{C \times \text{cadence}}{725\ \text{km arcsec}^{-1} \times \Delta s}, \tag{2}$$

where C is the transverse speed (km s^{-1}) on the photosphere according to the observational environment and Sun's activity, and Δs is the pixel scale. In this study, C is set to $4\ \text{km s}^{-1}$. For the NIRIS magnetograms used here, $\Delta s = 0\farcs083\ \text{pixel}^{-1}$. We then calculate the radius of the region of interest (ROI) with respect to the location or position of the magnetic flux element

X, denoted $\mathrm{ROI}_{p(X)}$, as follows:

$$\mathrm{radius}(\mathrm{ROI}_{p(X)}) = 2 \times D + r, \tag{3}$$

where r is the radius of the smallest region that covers the magnetic flux element. The $\mathrm{ROI}_{p(X)}$ defines the region that the magnetic flux element X cannot move beyond between two contiguous frames.

The magnetic flux $\Phi(X)$ is calculated by the surface integral of the normal component of magnetic field passing through X, as follows:

$$\Phi(X) = \int_S B_z \, dS, \tag{4}$$

where B_z is the magnitude of the magnetic field from the vertical magnetic field image of the testing magnetogram and S is the area of the surface of X.

For any two features or magnetic flux elements X and Y in a frame, we define the distance between X and Y, denoted Dist(X, Y), as follows:

$$\mathrm{Dist}(X, Y) = \min_{x \in X, y \in Y} d(x, y), \tag{5}$$

where x and y are pixels in X and Y, respectively, and $d(x, y)$ is the Euclidean distance between x and y. For a given magnetic flux element X in a frame, the adjacent features of X are defined as the k-nearest neighboring features of X in the frame. (In the study presented here, k is set to 10.)

Let X_i be a magnetic flux element in the current frame F_1. Let Y_i be a magnetic flux element in the next frame F_2 where Y_i occurs in the $\mathrm{ROI}_{p(X_i)}$ in F_2. We say X_i is approximately equal to Y_i, denoted $X_i \approx Y_i$, if X_i and Y_i have the same sign, and

$$\left| \frac{\Phi(X_i) - \Phi(Y_i)}{\Phi(X_i)} \right| \leqslant \epsilon_1, \tag{6}$$

where ϵ_1 is a user-determined threshold based on the observation setting and tracking task requirement. (In the study presented here, ϵ_1 is set to 0.33.)

With the above terms and definitions, we are now ready to describe our algorithms for magnetic tracking and event detection. For each magnetic flux element or feature X_i in the current frame F_1, the algorithms below determine and indicate whether X_i exists in the next frame F_2, or X_i is involved in a merging or cancellation event, or X_i disappears.

(Main Algorithm)

(i) If there exists a magnetic feature Y_i in the $\mathrm{ROI}_{p(X_i)}$ in the next frame F_2 such that $X_i \approx Y_i$, then indicate X_i exists in F_2 (more precisely, X_i becomes Y_i in F_2) and go to (ii); otherwise go to (iii).

(ii) Check the sign of X_i, highlighting X_i by yellow bordering if X_i is positive and by green bordering if X_i is negative. Exit the Main Algorithm.

(iii) Find all magnetic features in the $\mathrm{ROI}_{p(X_i)}$ in the current frame F_1. Group those features in the $\mathrm{ROI}_{p(X_i)}$ whose signs are the same as the sign of X_i into $G_{\mathrm{same}}(X_i)$ and group those features in the $\mathrm{ROI}_{p(X_i)}$ whose signs are opposite to the sign of X_i into $G_{\mathrm{opposite}}(X_i)$.

(iv) Go to the Merging Algorithm to check whether X_i and the features in $G_{\mathrm{same}}(X_i)$ meet the merging criterion. If yes, perform the merging using the Merging Algorithm and then exit the Main Algorithm; otherwise indicate X_i is not involved in a merging event.

(v) Go to the Cancellation Algorithm to check whether X_i and the features in $G_{\mathrm{opposite}}(X_i)$ meet the cancellation criterion. If yes, perform the cancellation using the Cancellation Algorithm and then exit the Main Algorithm; otherwise indicate X_i is not involved in a cancellation event.

(vi) If X_i is not involved in a merging event according to (iv) and X_i is not involved in a cancellation event based on (v), indicate X_i disappears and highlight X_i by purple bordering.[9] Exit the Main Algorithm.

(Merging Algorithm)

(i) For each feature X_j in $G_{\mathrm{same}}(X_i)$, check whether there exists a feature Y_j in the $\mathrm{ROI}_{p(X_j)}$ in the next frame F_2 such that $X_j \approx Y_j$, and if yes, delete X_j from $G_{\mathrm{same}}(X_i)$. Call the remaining set, $G'_{\mathrm{same}}(X_i)$. If there are too many features in $G'_{\mathrm{same}}(X_i)$, only keep those adjacent features of X_i in $G'_{\mathrm{same}}(X_i)$.

(ii) If there exist a combination C_s of features in $G'_{\mathrm{same}}(X_i)$ and a magnetic feature Y_i in the $\mathrm{ROI}_{p(X_i)}$ in the next frame F_2 where Y_i and X_i have the same sign, such that Equation (7) below is satisfied, then we say X_i and the features in C_s are merged into Y_i:

$$\left| \frac{\left(\Phi(X_i) + \sum_{X \in C_s} \Phi(X)\right) - \Phi(Y_i)}{\Phi(Y_i)} \right| \leqslant \epsilon_2, \tag{7}$$

where $\Phi(X_i)$ and $\Phi(X)$ have the same sign and ϵ_2 is a user-determined threshold (which is set to 0.5). Indicate X_i and the features in C_s are merged into Y_i by highlighting X_i and the features in C_s using amber bordering. Exit the Merging Algorithm.

(iii) If there does not exist Y_i or a combination of features satisfying Equation (7) (i.e., the condition in (ii) is not satisfied), indicate X_i and the features in $G_{\mathrm{same}}(X_i)$ do not meet the merging criterion. Exit the Merging Algorithm.

(Cancellation Algorithm)

(i) For each feature X_j in $G_{\mathrm{opposite}}(X_i)$, check whether there exists a feature Y_j in the $\mathrm{ROI}_{p(X_j)}$ in the next frame F_2 such that $X_j \approx Y_j$, and if yes, delete X_j from $G_{\mathrm{opposite}}(X_i)$. Call the remaining set, $G'_{\mathrm{opposite}}(X_i)$. If there are too many features in $G'_{\mathrm{opposite}}(X_i)$, only keep those adjacent features of X_i in $G'_{\mathrm{opposite}}(X_i)$.

(ii) If there exists a combination C_o of features in $G'_{\mathrm{opposite}}(X_i)$ such that Equation (8) below is satisfied, then we say X_i and the features in C_o cancel each other (referred to as balanced cancellation in DeForest et al. 2007):

$$\left| \frac{\Phi(X_i) + \sum_{X \in C_o} \Phi(X)}{\Phi(X_i)} \right| \leqslant \epsilon_2, \tag{8}$$

where $\Phi(X_i)$ and $\Phi(X)$ have opposite signs. Indicate X_i and the features in C_o cancel each other by highlighting

[9] For the events belonging to the death category, namely disappearance, merging, and cancellation, magnetic features involved in the events are highlighted by different bordering colors (purple for disappearance, amber for merging, and pink for cancellation) in the current frame F_1. For the events belonging to the birth category, namely appearance, splitting, and emergence, magnetic features involved in the events are highlighted by different bordering colors (blue for appearance, aqua for splitting, and red for emergence) in the next frame F_2.

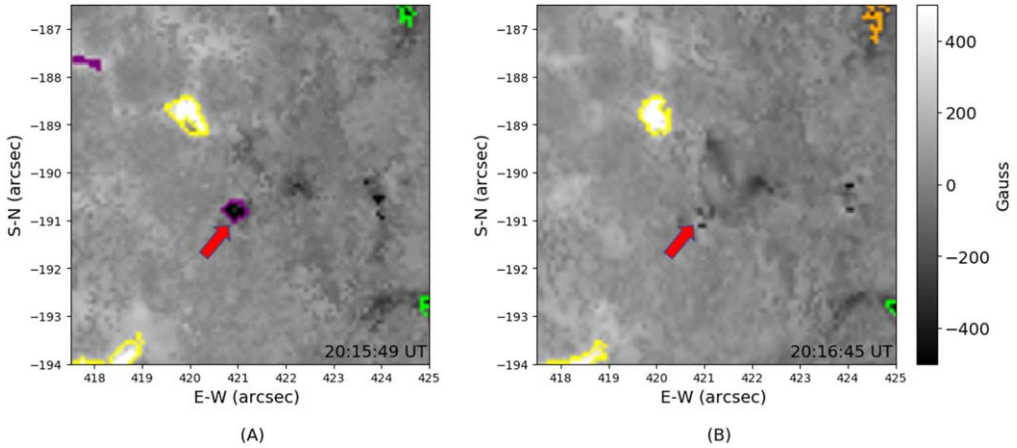

Figure 3. Example of BBSO/GST images of a disappearance event. The negative magnetic flux element highlighted by purple bordering in (A) disappears in (B). Time in UT is at the bottom right of each image.

X_i and the features in C_o using pink bordering. Exit the Cancellation Algorithm.

(iii) If there exist a combination C_o of features in $G'_{\text{opposite}}(X_i)$ and a magnetic feature Y_i in the $\text{ROI}_{p(X_i)}$ in the next frame F_2, such that Equation (9) below is satisfied, then we say X_i and the features in C_o are canceled to yield a feature, Y_i, carrying the remaining flux (referred to as unbalanced cancellation in DeForest et al. 2007):

$$\frac{\left||\Phi(X_i) + \sum_{X\in C_o}\Phi(X)| - |\Phi(Y_i)|\right|}{|\Phi(Y_i)|} \leqslant \epsilon_2, \tag{9}$$

where $\Phi(X_i)$ and $\Phi(X)$ have opposite signs. Indicate X_i and the features in C_o are canceled by highlighting X_i and the features in C_o using pink bordering. Exit the Cancellation Algorithm.

(iv) If neither the condition in (ii) nor the condition in (iii) is satisfied, indicate X_i and the features in $G_{\text{opposite}}(X_i)$ do not meet the cancellation criterion. Exit the Cancellation Algorithm.

To determine whether the magnetic feature X_i appears or is involved in a splitting or emergence event, we use the same algorithms as described above except that we treat the next frame F_2 as the current frame and the current frame F_1 as the next frame.

4. Results

SolarUnet is implemented in Python.[10] Our deep learning model is coded with TensorFlow[11] (Abadi et al. 2016) and Keras[12] (Chollet 2018) libraries. The data processed by SolarUnet, with the aid of Astropy[13] (Astropy Collaboration et al. 2013), include FITS files containing vector magnetic fields, PNG images of the observational data described in Section 2, and image masks obtained from the SWAMIS tool presented in DeForest et al. (2007). SWAMIS, written in Perl Data Language (PDL)[14] and available via SolarSoft[15] (Freeland & Handy 1998), was run with the downhill option. Figures in this section were produced with the aid of matplotlib[16] (Hunter 2007). Statistical tests were performed by SciPy[17] (Virtanen et al. 2020). All experiments were conducted on a Dell PC with i7-8700k CPU, 32 GB RAM, and a NVIDIA GeForce RTX 2080 GPU for training and testing the deep learning model.

4.1. Magnetic Tracking and Event Detection Results

In this series of experiments, we used the 196 magnetograms mentioned in Table 1 and the corresponding masks obtained from SWAMIS to train SolarUnet as described in Section 3.1, and then we performed testing on the set of 147 magnetograms mentioned in Table 1. The testing set contained observations in NOAA AR 12665 (431″, −131″) during ∼20:16–22:42 UT on 2017 July 13. The filter size threshold was fixed at 10 pixels. Thus, in the experiments we considered features or magnetic flux elements having at least 10 pixels.

We present figures to illustrate the six events studied here. Frames taken at 20:15:49 UT and 20:16:45 UT are used to illustrate a disappearance event. Frames taken at 21:00:48 UT and 21:01:45 UT are used to illustrate an appearance event. Frames taken at 20:18:38 UT and 20:19:34 UT are used to illustrate a merging event. Frames taken at 20:19:34 UT and 20:20:30 UT are used to illustrate a splitting event and a cancellation event. Frames taken at 20:17:41 UT and 20:18:38 UT are used to illustrate an emergence event. We present enlarged FOV results in these figures with a FOV of 7.″5 where each figure has two axes: E-W (x-axis) and S-N (y-axis).

Figure 3 shows a disappearance event. In Figure 3(A), a magnetic feature highlighted by purple bordering exists at E-W = 421″ and S-N = −191″, which is pointed to by a red arrow in the frame from 20:15:49 UT. This feature disappears in the next frame from 20:16:45 UT as shown in Figure 3(B). Figure 4 illustrates an appearance event. In Figure 4(A), there exists no feature at E-W = 420″ and S-N = −192″ in the frame from 21:00:48 UT. However, a new feature appears in the next frame from 21:01:45 UT, which is highlighted by blue bordering and pointed to by a red arrow as shown in Figure 4(B).

[10] https://www.python.org/
[11] https://www.tensorflow.org/
[12] https://keras.io/
[13] https://www.astropy.org/
[14] http://pdl.perl.org/
[15] https://sohowww.nascom.nasa.gov/solarsoft/
[16] https://matplotlib.org/
[17] https://www.scipy.org/

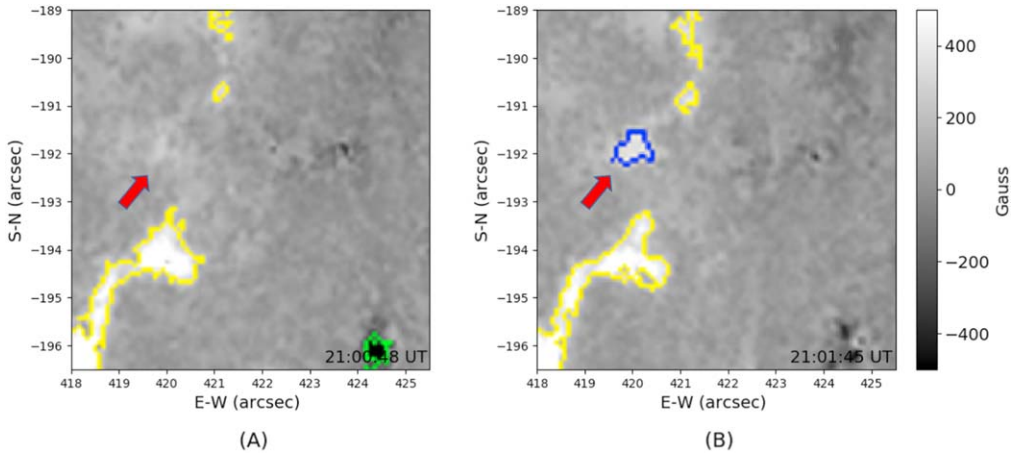

Figure 4. Example of BBSO/GST images of an appearance event. The positive magnetic flux element highlighted in blue bordering in (B) does not exist in (A), and hence an appearance event is detected. Time in UT is at the bottom right of each image.

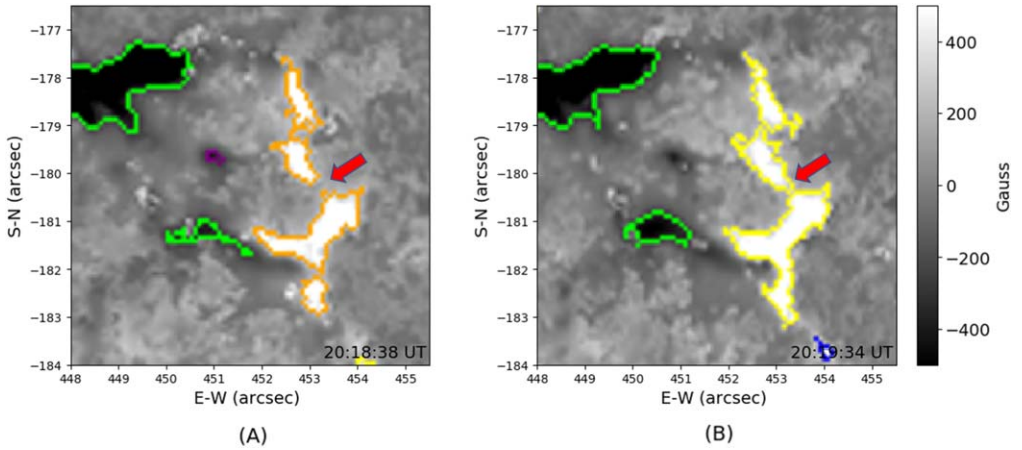

Figure 5. Example of BBSO/GST images of a merging event. Two positive magnetic flux elements highlighted by amber bordering in (A) are merged into a single positive magnetic flux element in (B). Time in UT is at the bottom right of each image.

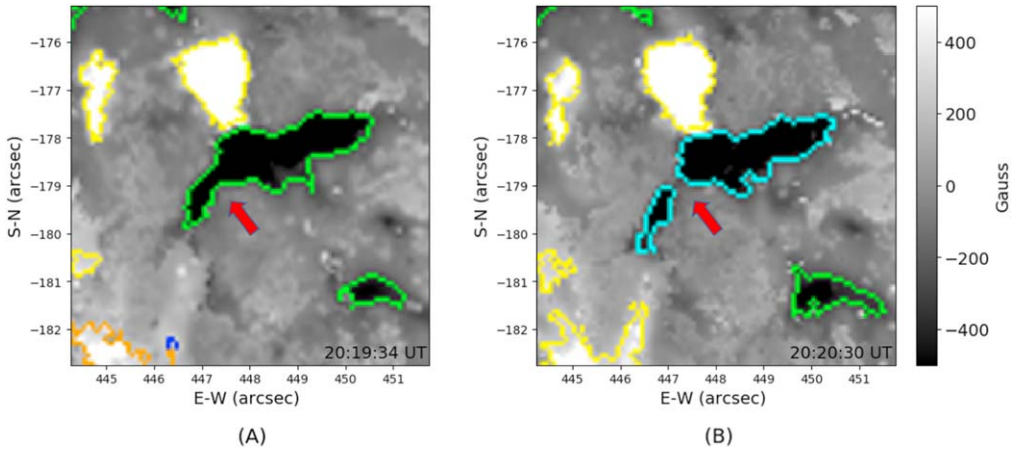

Figure 6. Example of BBSO/GST images of a splitting event. A negative magnetic flux element in (A) is split into two negative magnetic flux elements highlighted by aqua bordering in (B). Time in UT is at the bottom right of each image.

Figure 5 shows a merging event. In Figure 5(A), two separate positive polarity magnetic features highlighted by amber bordering and pointed to by a red arrow in the frame from 20:18:38 UT are merged into a single positive polarity feature at E-W $= 453''$ and S-N $= -180''$ in the frame from 20:19:34 UT as shown in Figure 5(B). Figure 6 illustrates a splitting event. In Figure 6(A), a negative polarity feature exists at E-W $= 448''$ and S-N $= -179''$ in the frame from 20:19:34 UT. This negative polarity feature is split into two negative polarity features highlighted by aqua bordering and

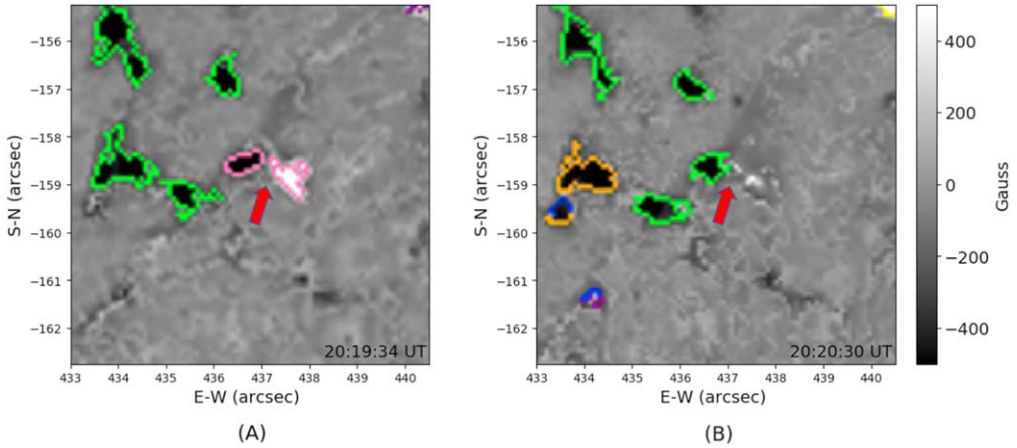

Figure 7. Example of BBSO/GST images of an unbalanced cancellation event. A positive magnetic flux element and a negative magnetic flux element, both of which are highlighted by pink bordering in (A), are canceled to yield a negative magnetic flux element carrying the remaining flux, which is pointed to by a red arrow in (B). Time in UT is at the bottom right of each image.

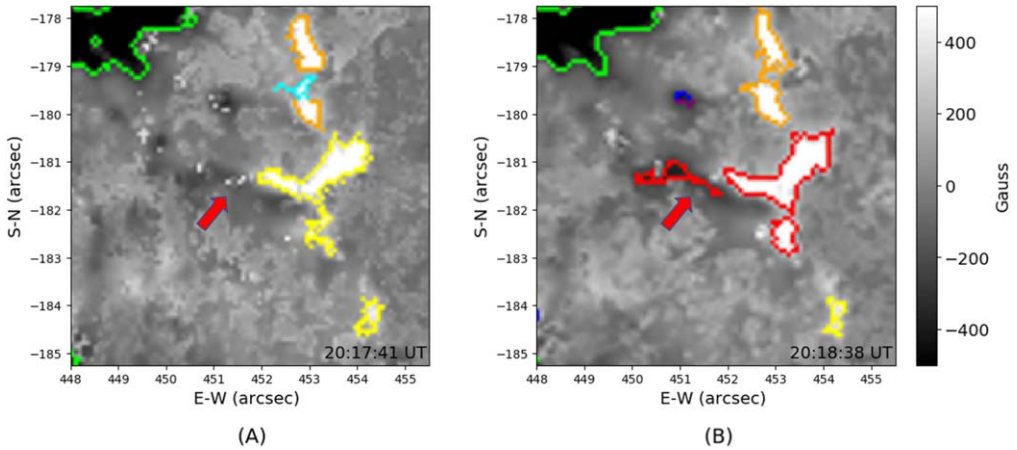

Figure 8. Example of BBSO/GST images of an unbalanced emergence event. A new negative magnetic flux element emerges, next to a pre-existing positive magnetic flux element in (A), in a nearly flux-conserving manner where the two magnetic flux elements with opposite signs are highlighted by red bordering in (B). Time in UT is at the bottom right of each image.

pointed to by a red arrow in the frame from 20:20:30 UT as shown in Figure 6(B).

Figure 7 shows an unbalanced cancellation event. In Figure 7(A), there exist two magnetic features with opposite signs around E-W = 437″ and S-N = −158.5″ in the frame from 20:19:34 UT. The two magnetic features with opposite signs, highlighted by pink bordering, are canceled to yield a negative polarity magnetic feature carrying the remaining flux, pointed to by a red arrow, in the frame from 20:20:30 UT as shown in Figure 7(B). Figure 8 illustrates an unbalanced emergence event. A new negative polarity feature emerges, next to a pre-existing positive polarity feature, in the frame from 20:18:38 UT as shown in Figure 8(B). The flux of the positive polarity feature pointed to by a red arrow in Figure 8(A) is approximately equal to the total flux of the two features with opposite signs, highlighted by red bordering and pointed to by a red arrow in Figure 8(B).

4.2. Comparison with SWAMIS

While both SolarUnet and SWAMIS (DeForest et al. 2007) aim to track magnetic features and detect magnetic events, they differ in two ways.

1. Their feature discrimination and identification algorithms are different. SWAMIS used hysteresis and a threshold-based method to separate nonsignificant flux regions, positive magnetic flux regions, and negative magnetic flux regions. Then it used direct clumping and a gradient-based ("downhill") method to identify magnetic features in these regions. By contrast, SolarUnet employs a U-shaped convolutional neural network to gain knowledge from training data and then predicts a binary (two-class) mask containing nonsignificant flux regions and significant flux regions. Next, SolarUnet separates the significant flux regions into positive magnetic flux regions and negative magnetic flux regions through postprocessing of the binary mask. Finally, SolarUnet uses a connected-component labeling algorithm (He et al. 2009) to group all adjacent segments in the positive magnetic flux regions and negative magnetic flux regions respectively if their pixels in edges or corners touch each other to identify positive and negative magnetic flux elements.
2. Their feature tracking and event detection algorithms are different. SWAMIS used a dual-maximum-overlap criterion to find persistent features across frames. In contrast, SolarUnet defines the ROI of a magnetic feature and traces the flux changes of the magnetic features in the ROI to find the association of features across frames.

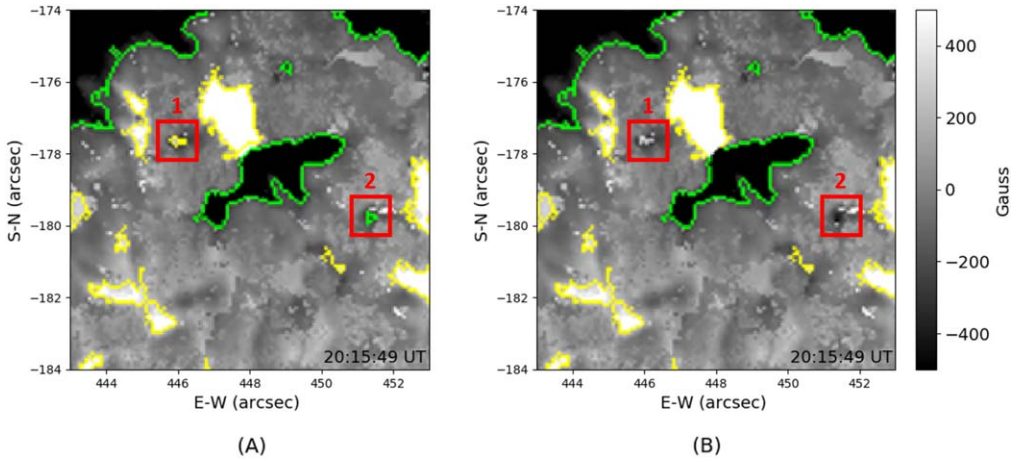

Figure 9. Illustration of the magnetic flux elements detected by SolarUnet but not found by SWAMIS on the testing magnetogram from AR 12665 collected on 2017 July 13 20:15:49 UT. (A) SolarUnet identifies a positive feature highlighted by yellow bordering and a negative feature highlighted by green bordering where the two highlighted features are enclosed by red square boxes numbered by 1 and 2, respectively. (B) SWAMIS does not find the two features as no bordering is shown inside the red square boxes numbered by 1 and 2, respectively. Time in UT is at the bottom right of each image.

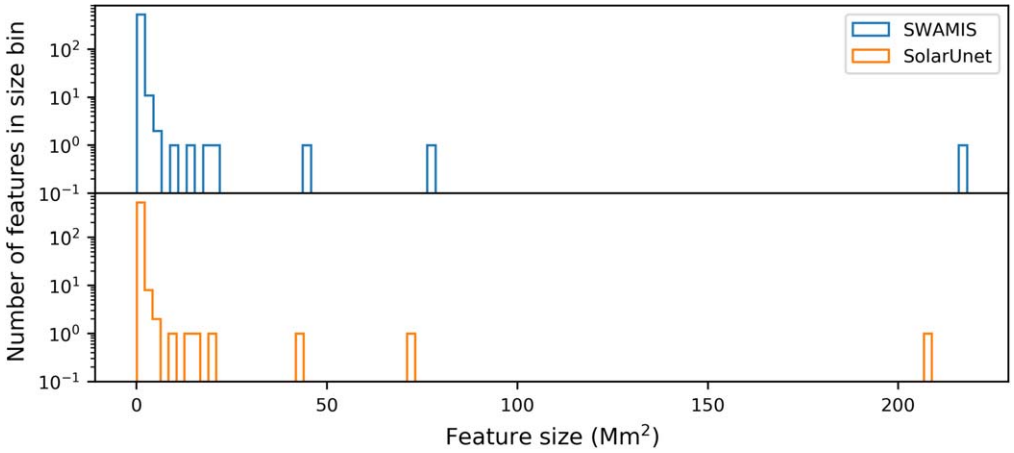

Figure 10. Magnetic feature size distributions as derived by SWAMIS (represented by blue) and SolarUnet (represented by orange) on the testing magnetogram from AR 12665 collected on 2017 July 13 20:15:49 UT. SolarUnet agrees mostly with SWAMIS on the feature size distributions.

It should be pointed out that, although the U-shaped network (i.e., the deep learning model) in SolarUnet gains knowledge from the training data prepared by SWAMIS, the model is able to generalize learned features to more generic forms. In our work, the model gains knowledge from the training images in quiescent solar regions collected on 2018 June 7 and uses the acquired knowledge to make predictions on unseen testing images from an active region (NOAA AR 12665) collected on 2017 July 13. With the generalization and inference capability, the model may discover new magnetic flux elements not found by the SWAMIS method. For example, with the filter size threshold of SolarUnet fixed at 10 pixels, SolarUnet detected two opposite-sign features not found by SWAMIS on the testing image (magnetogram) from AR 12665 collected on 2017 July 13 20:15:49 UT. Figure 9(A) highlights these two features; Figure 9(B) shows that the two features were not found by SWAMIS.

Figure 10 compares the feature size distributions of SWAMIS and SolarUnet on the testing image (magnetogram) where the features had at least 2 pixels (0.007242 Mm^2).[18] The feature sizes of SWAMIS are represented by blue and those of SolarUnet are represented by orange. Figure 10 shows that SolarUnet agrees mostly with SWAMIS on the feature size distributions. To quantify this finding, we conducted the Epps-Singleton two-sample test (Epps & Singleton 1986; Goerg & Kaiser 2009; Gibbons & Chakraborti 2011). According to the test, the results of SolarUnet and SWAMIS have a significant difference when $p \leqslant 0.05$. In our case $p = 0.858 > 0.05$, and hence we conclude that the results of the two tools are similar. Table 2 shows the minimum, maximum, median, mean, and standard deviation (SD) of the feature sizes found by SWAMIS and SolarUnet, respectively. SWAMIS detected 548 features while SolarUnet identified 543 features. The largest magnetic feature, which was a negative feature, found by SWAMIS had 60,213 pixels (218.03 Mm^2). This feature was also detected by SolarUnet, with a smaller size of 57,662 pixels (208.80 Mm^2). This size difference occurs due to the different feature identification and tracking algorithms used by the two tools.

Next, for each feature detected by the tools, we calculated its flux using the formula in Equation (4). Figure 11 compares the feature flux distributions of SWAMIS and SolarUnet. The results in Figure 11 are consistent with those in Figure 10; SolarUnet agrees mostly with SWAMIS on the feature flux

[18] In this and subsequent experiments, features with 1 pixel were considered as noise and excluded.

Table 2
Summary Statistics of Feature Size and Flux Distributions as Derived by SWAMIS and SolarUnet[a]

	Method	Minimum	Maximum	Median	Mean	SD
Feature size (Mm^2)	SWAMIS[b]	0.007242	218.03	0.029	0.94	10.14
	SolarUnet[c]	0.007242	208.80	0.022	0.89	9.72
Feature flux (10^{18} Mx)	SWAMIS[b]	0.009507	1805.13	0.048	6.70	81.59
	SolarUnet[c]	0.011051	1780.96	0.048	6.61	80.89

Notes.
[a] The data presented in this table are based on the testing magnetogram from AR 12665 collected on 2017 July 13 20:15:49 UT.
[b] SWAMIS detected 548 features in the testing magnetogram.
[c] SolarUnet identified 543 features in the testing magnetogram.

distributions. According to the Epps-Singleton two-sample test, the feature flux distributions of SolarUnet and SWAMIS have a significant difference when $p \leqslant 0.05$. In our case, $p = 0.983 > 0.05$, and consequently we conclude that the feature flux distributions of SolarUnet and SWAMIS are similar. Table 2 shows the minimum, maximum, median, mean, and SD of the feature fluxes found by SWAMIS and SolarUnet, respectively. The feature fluxes detected by SWAMIS ranged from 0.009507×10^{18} to 1805.13×10^{18} Mx. The feature fluxes detected by SolarUnet ranged from 0.011051×10^{18} to 1780.96×10^{18} Mx. Some of the small fluxes could be noise whiles others might be involved in small-scale magnetic flux emergence (Moreno-Insertis et al. 2018) or small-scale magnetic flux cancellation (Chen et al. 2015). Similar results on feature size and flux distributions were obtained from the other magnetograms in the testing set.

To further understand the behavior of SolarUnet and compare it with SWAMIS, we performed additional experiments to examine the lifetimes of the features identified and tracked by the two tools. We applied SolarUnet and SWAMIS to all of the 147 testing magnetograms mentioned in Table 1. The lifetime of a feature X is defined as X's disappearance time minus X's appearance time. More precisely, assuming X appears in the mth frame and disappears after the nth frame (i.e., X is not shown in the $(n+1)$th frame), the lifetime of X is defined to be $n - m + 1$ frames. Feature lifetime is strongly dependent on the feature identification and tracking algorithms employed by a tool (DeForest et al. 2007) and can be used to measure flux turnover rate (Hagenaar et al. 2003).

Figure 12 compares the lifetimes of features found by SWAMIS and SolarUnet. SWAMIS tracked 48,145 features across the 147 testing magnetograms while SolarUnet tracked 42,470 features. The lifetimes of features found by SWAMIS ranged from 1 frame (56 s) to 138 frames (128.8 minutes). The lifetimes of features detected by SolarUnet ranged from 1 frame to 147 frames (137.2 minutes). SWAMIS tracked more short-lifetime features than SolarUnet while SolarUnet tracked more long-lifetime features than SWAMIS. Specifically, among the 48,145 features tracked by SWAMIS, 37,110 features had a lifetime of one frame while SolarUnet only identified and tracked 22,657 such features. On the other hand, SolarUnet tracked 19,813 features whose lifetimes lasted more than one frame while SWAMIS only identified and tracked 11,035 such features. SolarUnet complements SWAMIS in tracking long-lifetime features. We note that the training data of SolarUnet are from SWAMIS. For those features with short lifetime in the training images, our deep learning model may not acquire enough knowledge about them and hence may miss similar features in the testing images. This may explain why SolarUnet detects fewer short-lifetime features than SWAMIS.

5. Discussion and Conclusions

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 m GST at the 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 (DeForest et al. 2007).

Our main results are summarized as follows:

1. For the testing data considered, SolarUnet agrees mostly with SWAMIS on feature size (area) and flux distributions and complements SWAMIS in tracking long-lifetime features. It is worth noting that because SolarUnet performs magnetic tracking through making predictions, it is faster than the current version of SWAMIS. In general, SolarUnet runs in seconds on a testing magnetogram while the current version of SWAMIS runs in minutes on the same testing magnetogram.
2. SolarUnet is a physics-guided tool in the sense that it incorporates physics knowledge into its model and algorithms in several ways. First, the training data of SolarUnet are from the physics-based SWAMIS tool. Second, when designing the loss function for the deep learning model used by SolarUnet, based on the observation that nonsignificant flux regions roughly have the same number of pixels as significant flux regions in the training set, we adopt a binary cross-entropy loss function as defined in Equation (1) instead of the weighted cross-entropy loss function used by the related U-Net model (Falk et al. 2019). Third, when converting the binary (two-class) mask predicted by our deep learning model for a testing magnetogram to a three-class mask with polarity information, we use the information of radial components in the vertical magnetic field image of the testing magnetogram to reconstruct positive and negative magnetic flux regions in the predicted mask. Lastly, by exploiting physics knowledge and based on the observational data and instruments used, we introduce the moving distance as defined in

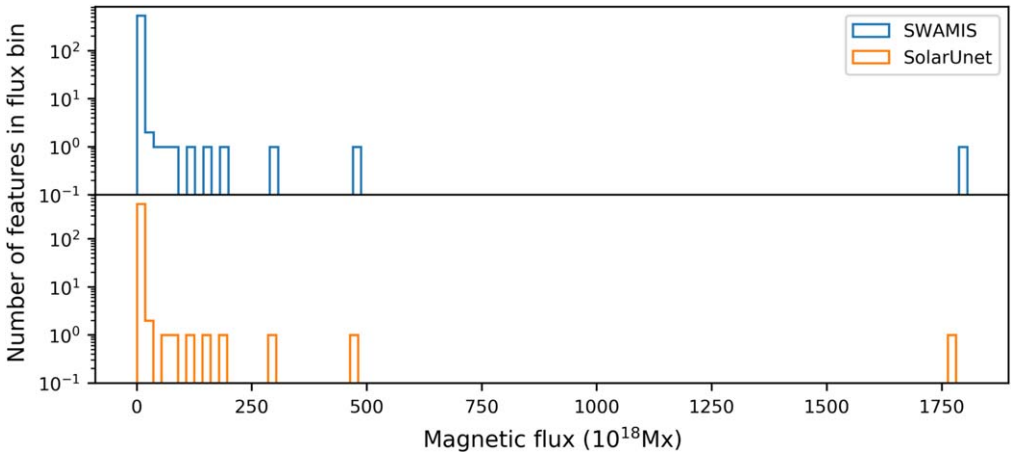

Figure 11. Magnetic feature flux distributions as derived by SWAMIS (represented by blue) and SolarUnet (represented by orange) on the testing magnetogram from AR 12665 collected on 2017 July 13 20:15:49 UT. SolarUnet agrees mostly with SWAMIS on the feature flux distributions.

Equation (2) and ROI as defined in Equation (3) of a magnetic flux element to find the association of features across frames so as to track these features.

3. Although SolarUnet gets training data from SWAMIS, our tool may discover new features not found by the SWAMIS method. For example, refer to Figure 9. SolarUnet may detect smaller opposite-polarity features, as shown and highlighted in Figure 9(A), near larger magnetic flux elements. Small-scale energy release phenomena, ranging from coronal jets down to spicules, may be responsible for providing the upward flux of energy and momentum for the observed heatings and flows in the corona and may plausibly drive the small transients in the solar wind recently discovered by the Parker Solar Probe (Parker 2020). There is mounting evidence that these events are generated via small-scale magnetic reconnection (e.g., Samanta et al. 2019), the photospheric signature of which is flux cancellation involving opposite magnetic polarities (Zwaan 1987). The ability of SolarUnet in detecting smaller opposite-polarity features near larger magnetic flux elements in a faster manner can result in an improved determination of magnetic reconnection rate, thus contributing to the understanding of the mechanisms of solar coronal heating and the acceleration of the solar wind.
4. The deep learning model in SolarUnet performs binary (two-class) classification, i.e., predicting a two-class mask, rather than three-class classification, i.e., predicting a three-class mask, during image segmentation. SolarUnet produces a three-class mask through postprocessing of the predicted two-class mask as described in item 2 above and in Section 3.1. As indicated in the machine-learning literature, multiclass classification including three-class classification often adds more noise to the loss function (see, e.g., the Abstract of Gupta et al. 2014), and it is easier to devise algorithms for binary classification (see, e.g., the Introduction in Allwein et al. 2001). We conducted additional experiments to compare SolarUnet with a three-class classification method. This method trained its deep learning model using the three-class masks obtained directly from SWAMIS and predicted three-class masks. Its model was the same as SolarUnet's model except that (i) its loss function was changed from the binary cross-entropy function defined in Equation (1) to a categorical cross-entropy loss with three-class labels (1, 0, −1), and (ii) its softmax activation function was modified to output three-class masks. The three-class classification method used the same tracking algorithms as described in Section 3.3 for magnetic tracking and event detection. The results of the three-class classification method were not as good as those of SolarUnet. For example, the feature size distribution obtained from the three-class classification method was significantly different from the feature size distribution obtained from SWAMIS with $p = 0.025 \leqslant 0.05$ according to the Epps-Singleton two-sample test on the testing magnetogram from AR 12665 collected on 2017 July 13 20:15:49 UT.

Based on our experimental results, we conclude that the proposed SolarUnet should be considered a novel and alternative method for identifying and tracking magnetic flux elements. More testing of the method, using different training and test data, should be performed. With the advent of big and complex observational data gathered from diverse instruments such as BBSO/GST and the upcoming Daniel K. Inouye Solar Telescope (DKIST), it is expected that the physics-guided deep learning-based SolarUnet tool will be a useful utility for processing and analyzing the data.

We thank the referee for very helpful and thoughtful comments. We also thank the BBSO/GST team for providing the data used in this study. The BBSO operation is supported by the New Jersey Institute of Technology and US NSF grant AGS-1821294. The GST operation is partly supported by the Korea Astronomy and Space Science Institute, the Seoul National University, and the Key Laboratory of Solar Activities of the Chinese Academy of Sciences (CAS) and the Operation, Maintenance and Upgrading Fund of CAS for Astronomical Telescopes and Facility Instruments. Portions of the analysis presented here made use of the Perl Data Language (PDL), which has been developed by K. Glazebrook, J. Brinchmann, J. Cerney, C. DeForest, D. Hunt, T. Jenness, T. Lukka, R. Schwebel, and C. Soeller and can be obtained from http://pdl.perl.org. 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.

Facility: Big Bear Solar Observatory.

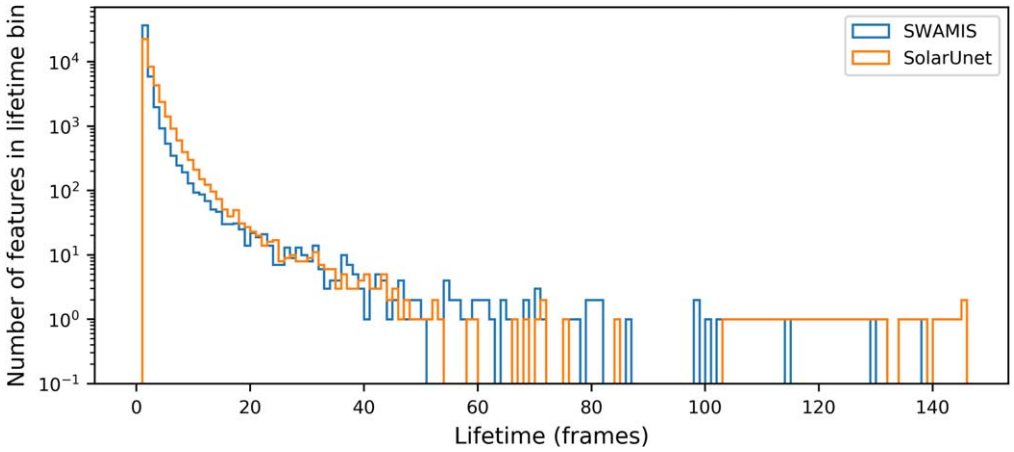

Figure 12. Feature lifetime histograms derived from SWAMIS and SolarUnet based on the 147 testing magnetograms (frames) from AR 12665 collected on 2017 July 13. SWAMIS tracks 48,145 features, among which 37,110 features have a lifetime of one frame. SolarUnet tracks 42,470 features, among which 22,657 features have a lifetime of 1 frame. On the other hand, SolarUnet tracks 19,813 features whose lifetimes last more than one frame while SWAMIS only tracks 11,035 such features. SolarUnet complements SWAMIS in tracking long-lifetime features.

Appendix

Here we explain the technical terms used in describing our deep learning model (i.e., the U-shaped convolutional neural network).

Encoder is a neural network, which takes an input image and generates a high-dimensional vector that is an abstract representation of the image (see Chapter 8.5.2 in Aggarwal 2018). By using the encoder, our model can better understand the content and context of the image.

Decoder is a neural network, which takes a high-dimensional vector and generates a segmentation mask (see Chapter 8.5.2 in Aggarwal 2018). By using the decoder, our model can recover the spatial information in the input image.

Bottleneck, also known as the "compressed code" (see Chapter 8.5.2 in Aggarwal 2018), is a layer with less neurons than the layer below or above it (Gehring et al. 2013). In general, it can be used to obtain a representation of the input with reduced size (dimensionality). In our model, bottleneck mediates between the encoder and the decoder.

Convolution layer contains multiple kernels where a kernel is a matrix whose elements (weights) need to be learned from training data (see Chapter 9 in Goodfellow et al. 2016). Each kernel is multiplied with an image vector X (via element-wise multiplications) to produce a new image vector that contains only the important information in X (see Chapter 8 in Aggarwal 2018).

Max pooling layer reduces the size of an image vector X while retaining only the important information in X (see Chapter 8.2 in Aggarwal 2018).

Up-convolution layer, containing learnable parameters (weights), increases the size of an image vector X. This layer, also called an upsampling (Shelhamer et al. 2017) or deconvolution layer, can recover the spatial information in X (see Chapter 8.5.2 in Aggarwal 2018).

Softmax activation function converts a vector of k real values to a vector of k real values that sum to 1 (see page 14 in Aggarwal 2018). Softmax is useful because it converts the scores in the vector to a normalized probability distribution, which can be displayed to a user. In our model, softmax is used to output the class label (1 versus −1 or nonsignificant flux versus significant flux) of each pixel.

ReLU employs an activation function $f(x)$, defined as $f(x) = \max(0, x)$, where x is the input to a neuron, $f(x) = x$ if $x \geqslant 0$ and $f(x) = 0$ otherwise (see Chapter 1.2 in Aggarwal 2018). It is easy to train a model that uses ReLUs, which often achieves good performance.

ORCID iDs

Jiasheng Wang https://orcid.org/0000-0001-5099-8209
Chang Liu https://orcid.org/0000-0002-6178-7471
Ju Jing https://orcid.org/0000-0002-8179-3625
Jason T. L. Wang https://orcid.org/0000-0002-2486-1097
Haimin Wang https://orcid.org/0000-0002-5233-565X

References

Abadi, M., Barham, P., & Chen, J. 2016, in Proc. 12th USENIX Conf. Operating Systems and Design (Berkeley, CA: USENIX), 265, doi:10.5555/3026877.3026899
Aggarwal, C. C. 2018, Neural Networks and Deep Learning (Berlin: Springer), doi:10.1007/978-3-319-94463-0
Allwein, E. L., Schapire, R. E., & Singer, Y. 2001, J. Mach. Learn. Res., 1, 113
Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33
Cao, W., Goode, P. R., Ahn, K., et al. 2012, in ASP Conf. Ser. 463, Second ATST-EAST Meeting: Magnetic Fields from the Photosphere to the Corona (San Francisco, CA: ASP), 291
Cao, W., Gorceix, N., Coulter, R., et al. 2010, AN, 331, 636
Chen, X., Deng, N., Lamb, D. A., et al. 2015, RAA, 15, 1012
Chollet, F. 2018, Keras, Astrophysics Source Code Library, ascl:1806.022
DeForest, C. E., Hagenaar, H. J., Lamb, D. A., Parnell, C. E., & Welsch, B. T. 2007, ApJ, 666, 576
Epps, T. W., & Singleton, K. J. 1986, J. Stat. Comput. Simul., 26, 177
Falk, T., Mai, D., Bensch, R., et al. 2019, Nature Methods, 16, 67
Fossum, A., & Carlsson, M. 2006, ApJ, 646, 579
Freeland, S. L., & Handy, B. N. 1998, SoPh, 182, 497
Gehring, J., Miao, Y., Metze, F., & Waibel, A. 2013, in 2013 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (Piscataway, NJ: IEEE), 3377, doi:10.1109/ICASSP.2013.6638284
Giannattasio, F., Berrilli, F., Consolini, G., et al. 2018, A&A, 611, A56
Gibbons, J. D., & Chakraborti, S. 2011, in International Encyclopedia of Statistical Science, ed. M. Lovric (Berlin: Springer), 977, doi:10.1007/978-3-642-04898-2_420
Goerg, S. J., & Kaiser, J. 2009, Stata J., 9, 454
Goode, P. R., & Cao, W. 2012, Proc. SPIE, 8444, 844403
Goode, P. R., Yurchyshyn, V., Cao, W., et al. 2010, ApJL, 714, L31
Goodfellow, I., Bengio, Y., & Courville, A. 2016, Deep Learning (Cambridge, MA: MIT Press), http://www.deeplearningbook.org
Gupta, M. R., Bengio, S., & Weston, J. 2014, J. Mach. Learn. Res., 15, 1532

Hagenaar, H. J., Schrijver, C. J., & Title, A. M. 2003, ApJ, 584, 1107
He, L., Chao, Y., Suzuki, K., & Wu, K. 2009, Pattern Recogn., 42, 1977
Huertas-Company, M., Primack, J. R., Dekel, A., et al. 2018, ApJ, 858, 114
Hunter, J. D. 2007, CSE, 9, 90
Ioffe, S., & Szegedy, C. 2015, in Proc. 32nd Int. Conf. on Machine Learning 37 (New York: ACM), 448, doi:10.5555/3045118.3045167
Kim, T., Park, E., Lee, H., et al. 2019, NatAs, 3, 397
Lamb, D. A., DeForest, C. E., Hagenaar, H. J., Parnell, C. E., & Welsch, B. T. 2008, ApJ, 674, 520
Lamb, D. A., DeForest, C. E., Hagenaar, H. J., Parnell, C. E., & Welsch, B. T. 2010, ApJ, 720, 1405
Lamb, D. A., Howard, T. A., DeForest, C. E., Parnell, C. E., & Welsch, B. T. 2013, ApJ, 774, 127
LeCun, Y., Bengio, Y., & Hinton, G. 2015, Natur, 521, 436
Leenaarts, J., Carlsson, M., & Rouppe van der Voort, L. 2015, ApJ, 802, 136
Leka, K. D., Barnes, G., & Crouch, A. 2009, in ASP Conf. Ser. 415, The Second Hinode Science Meeting: Beyond Discovery-Toward Understanding, ed. B. Lites (San Francisco, CA: ASP), 365
Leung, H. W., & Bovy, J. 2018, MNRAS, 483, 3255
Lieu, M., Conversi, L., Altieri, B., & Carry, B. 2019, MNRAS, 485, 5831
Liu, H., Liu, C., Wang, J. T. L., & Wang, H. 2019, ApJ, 877, 121
Moreno-Insertis, F., Martinez-Sykora, J., Hansteen, V. H., & Muñoz, D. 2018, ApJL, 859, L26
Parker, E. N. 2020, NatAs, 4, 19
Parnell, C. 2002, MNRAS, 335, 389
Samanta, T., Tian, H., Yurchyshyn, V., et al. 2019, Sci, 366, 890
Shelhamer, E., Long, J., & Darrell, T. 2017, ITPAM, 39, 640
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. 2014, J. Mach. Learn. Res., 15, 1929
Varsik, J., Plymate, C., Goode, P., et al. 2014, Proc. SPIE, 9147, 91475D
Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, Nature Methods, 17, 261
Wang, J., Liu, C., Deng, N., & Wang, H. 2018, ApJ, 853, 143
Wu, J. F., & Boada, S. 2019, MNRAS, 484, 4683
Zwaan, C. 1987, ARA&A, 25, 83